TCNT*n* and the OCR*nx*. Note that when working with fixed TOP values, the unused bits are masked to zero when any of the OCR*nx* Registers are written. As the third period shown in Figure 18-8 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an asymmetrical output. The reason for this can be found in the update time of the OCR*nx* Register. Since the OCR*nx* update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values are not equal the two slopes of the period will differ in length. The difference in length gives the asymmetrical result of the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generating PWM waveforms on the OC*nx* pins. Setting the COM*nx*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM*nx*1:0 to 3 (see Table 18-4 on page 256). The actual OC*nx* value will only be visible on the port pin if the data direction of the port pin is set to output (DDR_OC*nx*). The PWM waveform is generated by setting (or clearing) the OC*nx* Register at the compare match between OCR*nx* and TCNT*n* when the counter increments, and by clearing (or setting) the OC*nx* Register at compare match between OCR*nx* and TCNT*n* when the counter decrements. The PWM frequency of the output $f_{OCnxPCPWM}$ when using phase-correct PWM can be calculated with the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR*nx* Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR*nx* is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

18.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation (PWM) mode (WGM*n*3:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC*nx*) is cleared on the compare match between TCNT*n* and OCR*nx* while up-counting, and set on the compare match while down-counting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However these modes are preferred for motor control applications due to the symmetric feature of the dual-slope PWM modes.

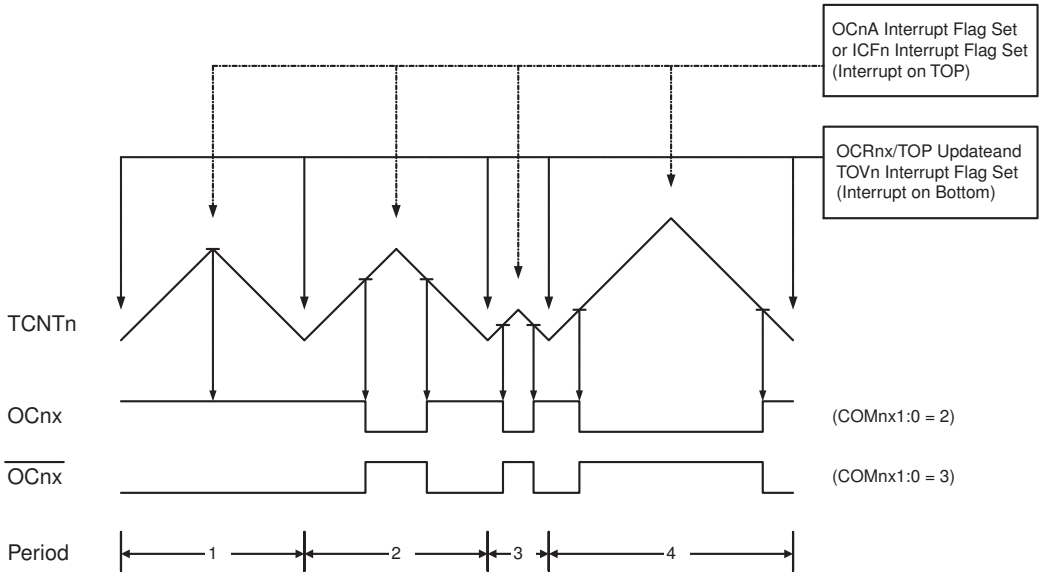
The main difference between the phase correct and the phase and frequency correct PWM mode is the time the OCR*nx* Register is updated by the OCR*nx* Buffer Register, (see Figure 18-8 on page 261 and Figure 18-9 on page 263).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR*n* or OCR*n*A. The minimum resolution allowed is 2 bit (ICR*n* or OCR*n*A set to 0x0003), and the maximum resolution is 16 bit (ICR*n* or OCR*n*A set to MAX). The PWM resolution R_{PFCPWM} in bits can be calculated with the following equation:

$$R_{PFCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR*n* (WGM*n*3:0 = 8), or the value in OCR*n*A (WGM*n*3:0 = 9). The counter has then reached TOP and changes the count direction. The TCNT*n* value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown in Figure 18-9 below. The figure shows phase and frequency correct PWM mode when OCR*n*A or ICR*n* is used to define TOP. The TCNT*n* value is shown in the timing diagram as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT*n* slopes represent compare matches between OCR*nx* and TCNT*n*. The OC*nx* Interrupt Flag will be set when a compare match occurs.

Figure 18-9. Phase and Frequency Correct PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV*n*) is set at the timer clock cycle when the OCR*nx* Registers are updated with the double-buffered value (at BOTTOM). The OC*n*A or ICF*n* Flag is set after TCNT*n* has reached TOP when either OCR*n*A or ICR*n* is used for defining the TOP value. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT*n* and the OCR*nx*.

As Figure 18-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR*nx* Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

The definition of TOP with the ICR*n* Register works well when using fixed TOP values. Combined with ICR*n* the OCR*n*A Register is available for generating a PWM output on OC*n*A. However, if the base PWM frequency is actively changed by modifying the TOP value, using the OCR*n*A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generating PWM waveforms on the OC*nx* pins. Setting the COM*nx*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM*nx*1:0 to 3 (see Table 18-4 on page 256). The actual OC*nx* value will only be visible at the port pin if the data direction of the port pin is set to output (DDR_OC*nx*). The PWM waveform is generated by setting (or clearing) the OC*nx* Register at the compare match between OCR*nx* and TCNT*n* when the counter increments, and by clearing (or setting) the OC*nx* Register at compare match between OCR*nx* and TCNT*n* when the counter decrements. The PWM frequency of the output $f_{OCnxPFCPWM}$ when using phase and frequency correct PWM can be calculated with the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR*nx* Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR*nx* is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

18.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set and when the OCR*nx* Register is updated with the OCR*nx* buffer value (only for modes utilizing double buffering). Figure 18-10 shows a timing diagram for the setting of OCF*nx*.

Figure 18-10. Timer/Counter Timing Diagram, Setting of OCF*nx*, no Prescaling

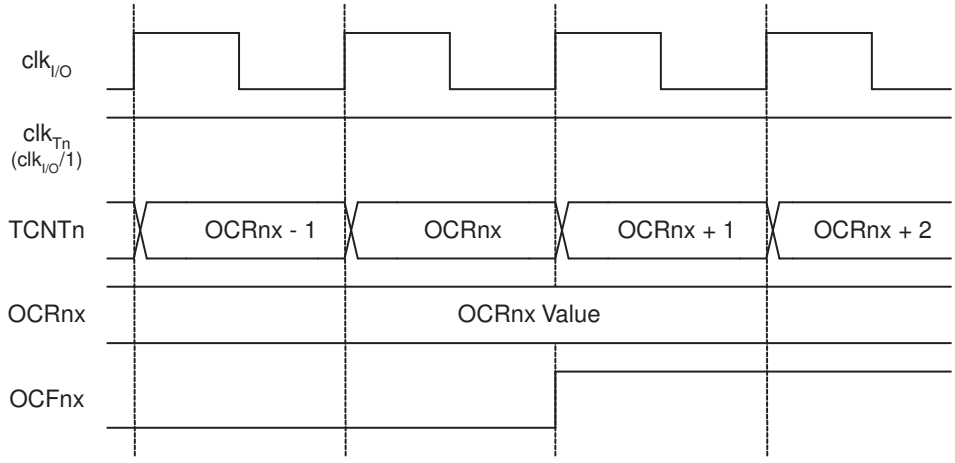


Figure 18-11 shows the same timing data, but with the prescaler enabled.

Figure 18-11. Timer/Counter Timing Diagram, Setting of OCF*nx* with Prescaler ($f_{clk\_I/O}/8$)

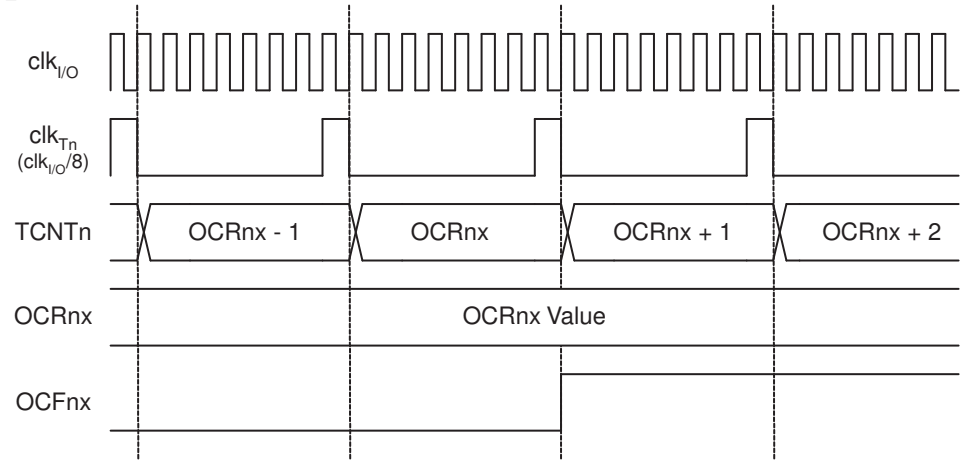


Figure 18-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR*nx* Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV*n* Flag at BOTTOM.

Figure 18-12. Timer/Counter Timing Diagram, no Prescaling

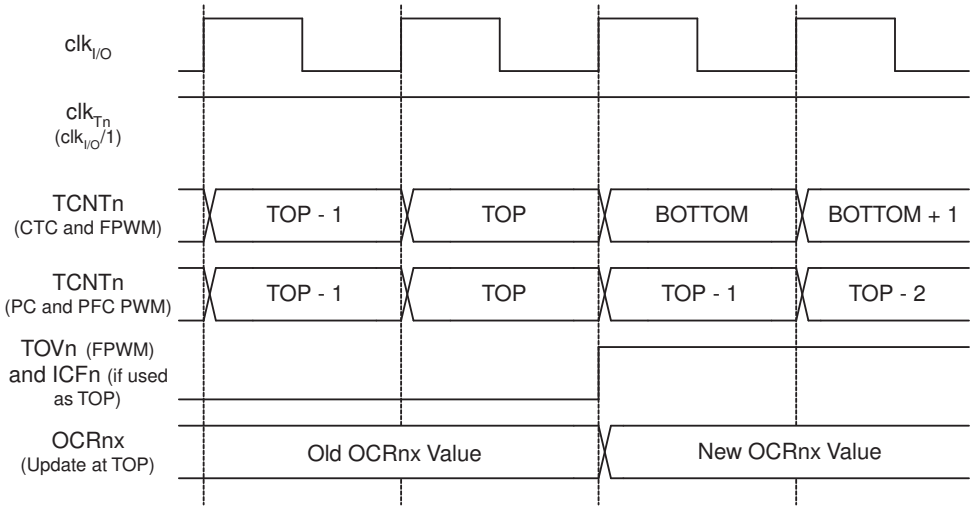
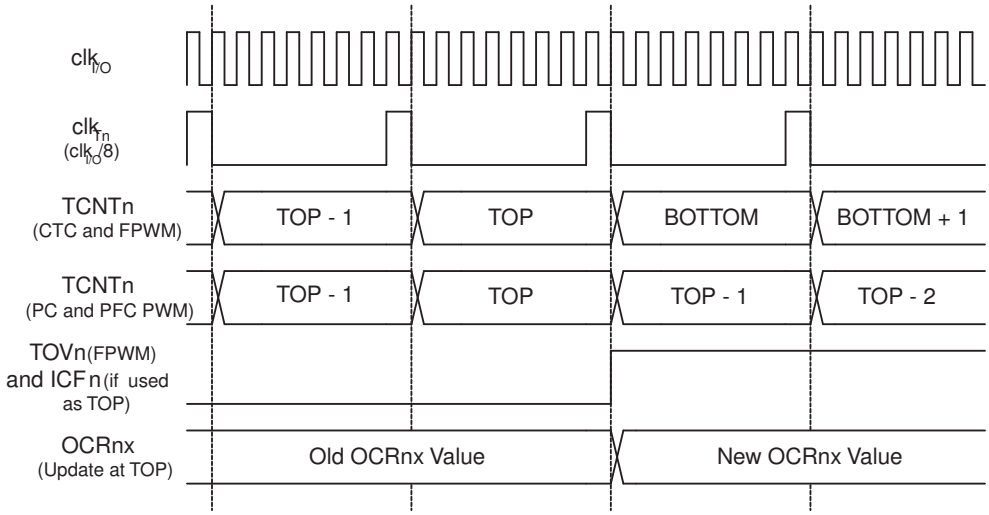


Figure 18-13 shows the same timing data, but with the prescaler enabled.

Figure 18-13. Timer/Counter Timing Diagram with Prescaler ($f_{clk\_I/O}/8$)



18.11 Register Description

18.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($80)	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM1A1:0 - Compare Output Mode for Channel A**

The COM1A1:0 bits control the output compare behavior of pin OC1A. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1A pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1A1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1A1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-6 COM1A Register Bits

Register Bits	Value	Description
COM1A1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 5:4 – COM1B1:0 - Compare Output Mode for Channel B**

The COM1B1:0 bits control the output compare behavior of pin OC1B. If one or both of the COM1B1:0 bits are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1B pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1B1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1B1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-7 COM1B Register Bits

Register Bits	Value	Description
COM1B1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 3:2 – COM1C1:0 - Compare Output Mode for Channel C**

The COM1C1:0 bits control the output compare behavior of pin OC1C. If one or both of the COM1C1:0 bits are written to one, the OC1C output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1C pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1C1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1C1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-8 COM1C Register Bits

Register Bits	Value	Description
COM1C1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 1:0 – WGM11:10 - Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B 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. 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. For more information on the different modes see section "Modes of Operation".

Table 18-9 WGM1 Register Bits

Register Bits	Value	Description
WGM11:10	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit

Register Bits	Value	Description
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($81)	ICNC1	ICES1	Res	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1 - Input Capture 1 Noise Canceller**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The input capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES1 - Input Capture 1 Edge Select**

This bit selects which edge on the Input Capture Pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger. When the ICES1 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1). This can be used to cause an Input Capture Interrupt, if this interrupt is enabled. When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM11:10 - Waveform Generation Mode**

Combined with the WGM11:0 bits found in the TCCR1A 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. 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. For more information on the different modes see section "Modes of Operation".

Table 18-10 WGM1 Register Bits

Register Bits	Value	Description
WGM11:10	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS12:10 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter1 according to the following table. If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 18-11 CS1 Register Bits

Register Bits	Value	Description
CS12:10	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	External clock source on Tn pin, clock on falling edge
	0x07	External clock source on Tn pin, clock on rising edge

18.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($82)	FOC1A	FOC1B	FOC1C	Res4	Res3	Res2	Res1	Res0	TCCR1C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A - Force Output Compare for Channel A**

The FOC1A bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1A bit, an immediate compare match is forced on the waveform generation unit. The OC1A output is changed according to its COM1A1:0 bits setting. Note that the FOC1A bits are implemented as strobes. Therefore it is the value present in the COM1A1:0 bits that determine the effect of the forced compare. A FOC1A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1A as TOP. The FOC1A bits are always read as zero.

- **Bit 6 – FOC1B - Force Output Compare for Channel B**

The FOC1B bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1B output is changed according to its COM1B1:0 bits setting. Note that the FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1B1:0 bits that determine the effect of the forced compare. A FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1B as TOP. The FOC1B bits are always read as zero.

- **Bit 5 – FOC1C - Force Output Compare for Channel C**

The FOC1C bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1C bit, an immediate compare match is forced on the waveform generation unit. The OC1C output is changed according to its COM1C1:0 bits setting. Note that the FOC1C bits are implemented as strobes. Therefore it is the value present in the COM1C1:0 bits that determine the effect of the forced compare. A FOC1C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1C as TOP. The FOC1C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.4 TCNT1H – Timer/Counter1 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($85)	TCNT1H7:0								TCNT1H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other

16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers. Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT1H7:0 - Timer/Counter1 High Byte**

18.11.5 TCNT1L – Timer/Counter1 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($84)	TCNT1L7:0								TCNT1L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers. Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT1L7:0 - Timer/Counter1 Low Byte**

18.11.6 OCR1AH – Timer/Counter1 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($89)	OCR1AH7:0								OCR1AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1AH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.7 OCR1AL – Timer/Counter1 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($88)	OCR1AL7:0								OCR1AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1AL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.8 OCR1BH – Timer/Counter1 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8B)	OCR1BH7:0								OCR1BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1BH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.9 OCR1BL – Timer/Counter1 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8A)	OCR1BL7:0								OCR1BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1BL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.10 OCR1CH – Timer/Counter1 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8D)	OCR1CH7:0								OCR1CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1CH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.11 OCR1CL – Timer/Counter1 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8C)	OCR1CL7:0								OCR1CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1CL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.12 ICR1H – Timer/Counter1 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($87)	ICR1H7:0								ICR1H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin or on the Analog Comparator output. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR1H7:0 - Timer/Counter1 Input Capture Register High Byte**

18.11.13 ICR1L – Timer/Counter1 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($86)	ICR1L7:0								ICR1L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin or on the Analog Comparator output. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR1L7:0 - Timer/Counter1 Input Capture Register Low Byte**

18.11.14 TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($6F)	Res1	Res0	ICIE1	Res	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE1 - Timer/Counter1 Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF1 Flag, located in TIFR1, is set.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE1C - Timer/Counter1 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1C Flag, located in TIFR1, is set.

- **Bit 2 – OCIE1B - Timer/Counter1 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1B Flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A - Timer/Counter1 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled.

The corresponding Interrupt Vector is executed when the OCF1A Flag, located in TIFR1, is set.

- **Bit 0 – TOIE1 - Timer/Counter1 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV1 Flag, located in TIFR1, is set.

18.11.15 TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	Res1	Res0	ICF1	Res	OCF1C	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF1 - Timer/Counter1 Input Capture Flag**

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value. ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF1C - Timer/Counter1 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register C (OCR1C). Note that a Forced Output Compare (FOC1C) strobe will not set the OCF1C Flag. OCF1C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF1C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF1B - Timer/Counter1 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B). Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B Flag. OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF1A - Timer/Counter1 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A). Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag. OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV1 - Timer/Counter1 Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting of the Timer/Counter1 Control Register. In Normal and CTC modes, the TOV1 Flag is set when the timer

overflows. TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

18.11.16 TCCR3A – Timer/Counter3 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($90)	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM3A1:0 - Compare Output Mode for Channel A**

The COM3A1:0 bits control the output compare behavior of pin OC3A. If one or both of the COM3A1:0 bits are written to one, the OC3A output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3A pin must be set in order to enable the output driver. When the OC3A is connected to the pin, the function of the COM3A1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3A1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-12 COM3A Register Bits

Register Bits	Value	Description
COM3A1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 5:4 – COM3B1:0 - Compare Output Mode for Channel B**

The COM3B1:0 bits control the output compare behavior of pin OC3B. If one or both of the COM3B1:0 bits are written to one, the OC3B output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3B pin must be set in order to enable the output driver. When the OC3B is connected to the pin, the function of the COM3B1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3B1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-13 COM3B Register Bits

Register Bits	Value	Description
COM3B1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match

Register Bits	Value	Description
		(set output to high level).

- **Bit 3:2 – COM3C1:0 - Compare Output Mode for Channel C**

The COM3C1:0 bits control the output compare behavior of pin OC3C. If one or both of the COM3C1:0 bits are written to one, the OC3C output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3C pin must be set in order to enable the output driver. When the OC3C is connected to the pin, the function of the COM3C1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3C1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-14 COM3C Register Bits

Register Bits	Value	Description
COM3C1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 1:0 – WGM31:30 - Waveform Generation Mode**

Combined with the WGM33:2 bits found in the TCCR3B 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. 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. For more information on the different modes see section "Modes of Operation".

Table 18-15 WGM3 Register Bits

Register Bits	Value	Description
WGM31:30	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved

Register Bits	Value	Description
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.17 TCCR3B – Timer/Counter3 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($91)	ICNC3	ICES3	Res	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC3 - Input Capture 3 Noise Canceller**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP3) is filtered. The filter function requires four successive equal valued samples of the ICP3 pin for changing its output. The input capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES3 - Input Capture 3 Edge Select**

This bit selects which edge on the Input Capture Pin (ICP3) that is used to trigger a capture event. When the ICES3 bit is written to zero, a falling (negative) edge is used as trigger. When the ICES3 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES3 setting, the counter value is copied into the Input Capture Register (ICR3). The event will also set the Input Capture Flag (ICF3). This can be used to cause an Input Capture Interrupt, if this interrupt is enabled. When the ICR3 is used as TOP value (see description of the WGM33:0 bits located in the TCCR3A and the TCCR3B Register), the ICP3 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM31:30 - Waveform Generation Mode**

Combined with the WGM31:0 bits found in the TCCR3A 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. 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. For more information on the different modes see section "Modes of Operation".

Table 18-16 WGM3 Register Bits

Register Bits	Value	Description
WGM31:30	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit

Register Bits	Value	Description
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS32:30 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter3 according to the following table. If external pin modes are used for the Timer/Counter3, transitions on the T3 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 18-17 CS3 Register Bits

Register Bits	Value	Description
CS32:30	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	External clock source on Tn pin, clock on falling edge
	0x07	External clock source on Tn pin, clock on rising edge

18.11.18 TCCR3C – Timer/Counter3 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($92)	FOC3A	FOC3B	FOC3C	Res4	Res3	Res2	Res1	Res0	TCCR3C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC3A - Force Output Compare for Channel A**

The FOC3A bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3A bit, an immediate compare match is forced on the waveform generation unit. The OC3A output is changed according to its COM3A1:0 bits setting. Note that the FOC3A bits are implemented as strobes. Therefore it is the value present in the COM3A1:0 bits that determine the effect of the forced compare. A FOC3A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR3A as TOP. The FOC3A bits are always read as zero.

- **Bit 6 – FOC3B - Force Output Compare for Channel B**

The FOC3B bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3B bit, an immediate compare match is forced on the waveform generation unit. The OC3B output is changed according to its COM3B1:0 bits setting. Note that the FOC3B bits are implemented as strobes. Therefore it is the value present in the COM3B1:0 bits that determine the effect of the forced compare. A FOC3B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1B as TOP. The FOC3B bits are always read as zero.

- **Bit 5 – FOC3C - Force Output Compare for Channel C**

The FOC3C bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3C bit, an immediate compare match is forced on the waveform generation unit. The OC3C output is changed according to its COM3C1:0 bits setting. Note that the FOC3C bits are implemented as strobes. Therefore it is the value present in the COM3C1:0 bits that determine the effect of the forced compare. A FOC3C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR3C as TOP. The FOC3C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.19 TCNT3H – Timer/Counter3 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($95)	TCNT3H7:0								TCNT3H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT3H and TCNT3L, combined TCNT3) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCR3x Registers. Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT3H7:0 - Timer/Counter3 High Byte**

18.11.20 TCNT3L – Timer/Counter3 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($94)	TCNT3L7:0								TCNT3L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT3H and TCNT3L, combined TCNT3) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCR3x Registers. Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT3L7:0 - Timer/Counter3 Low Byte**

18.11.21 OCR3AH – Timer/Counter3 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($99)	OCR3AH7:0								OCR3AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3AH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.22 OCR3AL – Timer/Counter3 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($98)	OCR3AL7:0								OCR3AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3AL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.23 OCR3BH – Timer/Counter3 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9B)	OCR3BH7:0								OCR3BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3BH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.24 OCR3BL – Timer/Counter3 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9A)	OCR3BL7:0								OCR3BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3BL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.25 OCR3CH – Timer/Counter3 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9D)	OCR3CH7:0								OCR3CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3CH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.26 OCR3CL – Timer/Counter3 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9C)	OCR3CL7:0								OCR3CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3CL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.27 ICR3H – Timer/Counter3 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($97)	ICR3H7:0								ICR3H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT3) value each time an event occurs on the ICP3 pin. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR3H7:0 - Timer/Counter3 Input Capture Register High Byte**

18.11.28 ICR3L – Timer/Counter3 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($96)	ICR3L7:0								ICR3L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT3) value each time an event occurs on the ICP3 pin. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR3L7:0 - Timer/Counter3 Input Capture Register Low Byte**

18.11.29 TIMSK3 – Timer/Counter3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($71)	Res1	Res0	ICIE3	Res	OCIE3C	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE3 - Timer/Counter3 Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF3 Flag, located in TIFR3, is set.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE3C - Timer/Counter3 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3C Flag, located in TIFR3, is set.

- **Bit 2 – OCIE3B - Timer/Counter3 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3B Flag, located in TIFR3, is set.

- **Bit 1 – OCIE3A - Timer/Counter3 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3A Flag, located in TIFR3, is set.

- **Bit 0 – TOIE3 - Timer/Counter3 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV3 Flag, located in TIFR3, is set.

18.11.30 TIFR3 – Timer/Counter3 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	Res1	Res0	ICF3	Res	OCF3C	OCF3B	OCF3A	TOV3	TIFR3
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF3 - Timer/Counter3 Input Capture Flag**

This flag is set when a capture event occurs on the ICP3 pin. When the Input Capture Register (ICR3) is set by the WGM33:0 to be used as the TOP value, the ICF3 Flag is set when the counter reaches the TOP value. ICF3 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF3 can be cleared by writing a logic one to its bit location.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF3C - Timer/Counter3 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register C (OCR3C). Note that a Forced Output Compare (FOC3C) strobe will not set the OCF3C Flag. OCF3C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF3C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF3B - Timer/Counter3 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register B (OCR3B). Note that a Forced Output Compare (FOC3B) strobe will not set the OCF3B Flag. OCF3B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF3B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF3A - Timer/Counter3 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register A (OCR3A). Note that a Forced Output Compare (FOC3A) strobe will not set the OCF3A Flag. OCF3A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF3A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV3 - Timer/Counter3 Overflow Flag**

The setting of this flag is dependent of the WGM33:0 bits setting of the Timer/Counter3 Control Register. In Normal and CTC modes, the TOV3 Flag is set when the timer overflows. TOV3 is automatically cleared when the Timer/Counter3 Overflow Interrupt Vector is executed. Alternatively, TOV3 can be cleared by writing a logic one to its bit location.

18.11.31 TCCR4A – Timer/Counter4 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($A0)	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	TCCR4A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM4A1:0 - Compare Output Mode for Channel A**

The Timer/Counter4 has only limited functionality. Therefore the COM4A1:0 bits do not control the output compare behavior of any pin. The following table shows the

COM4A1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-18 COM4A Register Bits

Register Bits	Value	Description
COM4A1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 5:4 – COM4B1:0 - Compare Output Mode for Channel B**

The Timer/Counter4 has only limited functionality. Therefore the COM4B1:0 bits do not control the output compare behavior of any pin. The following table shows the COM4B1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-19 COM4B Register Bits

Register Bits	Value	Description
COM4B1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 3:2 – COM4C1:0 - Compare Output Mode for Channel C**

The Timer/Counter4 has only limited functionality. Therefore the COM4C1:0 bits do not control the output compare behavior of any pin. The following table shows the COM4C1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-20 COM4C Register Bits

Register Bits	Value	Description
COM4C1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 1:0 – WGM41:40 - Waveform Generation Mode**

Combined with the WGM43:2 bits found in the TCCR4B 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. 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. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter4 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-21 WGM4 Register Bits

Register Bits	Value	Description
WGM41:40	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA

Register Bits	Value	Description
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.32 TCCR4B – Timer/Counter4 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($A1)	ICNC4	ICES4	Res	WGM43	WGM42	CS42	CS41	CS40	TCCR4B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC4 - Input Capture 4 Noise Canceller**

Timer/Counter4 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 6 – ICES4 - Input Capture 4 Edge Select**

Timer/Counter4 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM41:40 - Waveform Generation Mode**

Combined with the WGM41:0 bits found in the TCCR4A 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. 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. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter4 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-22 WGM4 Register Bits

Register Bits	Value	Description
WGM41:40	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit

Register Bits	Value	Description
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS42:40 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter4 according to the following table. External pin modes cannot be used for the Timer/Counter4.

Table 18-23 CS4 Register Bits

Register Bits	Value	Description
CS42:40	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	Reserved
	0x07	Reserved

18.11.33 TCCR4C – Timer/Counter4 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($A2)	FOC4A	FOC4B	FOC4C	Res4	Res3	Res2	Res1	Res0	TCCR4C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC4A - Force Output Compare for Channel A**

The FOC4A bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4A bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4A bits are implemented as strobes. Therefore it is the value present in the COM4A1:0 bits that determine the effect of the forced compare. A FOC4A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on

Compare Match (CTC) mode using OCR4A as TOP. The FOC4A bits are always read as zero.

- **Bit 6 – FOC4B - Force Output Compare for Channel B**

The FOC4B bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4B bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4B bits are implemented as strobes. Therefore it is the value present in the COM4B1:0 bits that determine the effect of the forced compare. A FOC4B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR4B as TOP. The FOC4B bits are always read as zero.

- **Bit 5 – FOC4C - Force Output Compare for Channel C**

The FOC4C bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4C bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4C bits are implemented as strobes. Therefore it is the value present in the COM4C1:0 bits that determine the effect of the forced compare. A FOC4C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR4C as TOP. The FOC4C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.34 TCNT4H – Timer/Counter4 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A5)	TCNT4H7:0								TCNT4H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT4H and TCNT4L, combined TCNT4) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT4) while the counter is running introduces a risk of missing a compare match between TCNT4 and one of the OCR4x Registers. Writing to the TCNT4 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT4H7:0 - Timer/Counter4 High Byte**

18.11.35 TCNT4L – Timer/Counter4 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A4)	TCNT4L7:0								TCNT4L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT4H and TCNT4L, combined TCNT4) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT4) while the counter is running introduces a risk of missing a compare match between TCNT4 and one of the OCR4x Registers. Writing to the TCNT4 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT4L7:0 - Timer/Counter4 Low Byte**

18.11.36 OCR4AH – Timer/Counter4 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A9)	OCR4AH7:0								OCR4AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4AH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.37 OCR4AL – Timer/Counter4 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A8)	OCR4AL7:0								OCR4AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4AL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.38 OCR4BH – Timer/Counter4 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AB)	OCR4BH7:0								OCR4BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4BH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.39 OCR4BL – Timer/Counter4 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AA)	OCR4BL7:0								OCR4BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4BL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.40 OCR4CH – Timer/Counter4 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AD)	OCR4CH7:0								OCR4CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4CH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.41 OCR4CL – Timer/Counter4 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AC)	OCR4CL7:0								OCR4CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4CL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.42 ICR4H – Timer/Counter4 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A7)	ICR4H7:0								ICR4H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter4 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT4) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR4H7:0 - Timer/Counter4 Input Capture Register High Byte**

18.11.43 ICR4L – Timer/Counter4 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A6)	ICR4L7:0								ICR4L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter4 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT4) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR4L7:0 - Timer/Counter4 Input Capture Register Low Byte**

18.11.44 TIMSK4 – Timer/Counter4 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($72)	Res1	Res0	ICIE4	Res	OCIE4C	OCIE4B	OCIE4A	TOIE4	TIMSK4
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE4 - Timer/Counter4 Input Capture Interrupt Enable**

The Timer/Counter4 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE4C - Timer/Counter4 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4C Flag, located in TIFR4, is set.

- **Bit 2 – OCIE4B - Timer/Counter4 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4B Flag, located in TIFR4, is set.

- **Bit 1 – OCIE4A - Timer/Counter4 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4A Flag, located in TIFR4, is set.

- **Bit 0 – TOIE4 - Timer/Counter4 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV4 Flag, located in TIFR4, is set.

18.11.45 TIFR4 – Timer/Counter4 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	Res1	Res0	ICF4	Res	OCF4C	OCF4B	OCF4A	TOV4	TIFR4
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF4 - Timer/Counter4 Input Capture Flag**

The Timer/Counter4 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF4C - Timer/Counter4 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register C (OCR4C). Note that a Forced Output Compare (FOC4C) strobe will not set the OCF4C Flag. OCF4C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF4C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF4B - Timer/Counter4 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register B (OCR4B). Note that a Forced Output Compare (FOC4B) strobe will not set the OCF4B Flag. OCF4B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF4B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF4A - Timer/Counter4 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register A (OCR4A). Note that a Forced Output Compare (FOC4A) strobe will not set the OCF4A Flag. OCF4A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF4A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV4 - Timer/Counter4 Overflow Flag**

The setting of this flag is dependent of the WGM43:0 bits setting of the Timer/Counter4 Control Register. In Normal and CTC modes, the TOV4 Flag is set when the timer overflows. TOV4 is automatically cleared when the Timer/Counter4 Overflow Interrupt Vector is executed. Alternatively, TOV4 can be cleared by writing a logic one to its bit location.

18.11.46 TCCR5A – Timer/Counter5 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($120)	COM5A1	COM5A0	COM5B1	COM5B0	COM5C1	COM5C0	WGM51	WGM50	TCCR5A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM5A1:0 - Compare Output Mode for Channel A**

The Timer/Counter5 has only limited functionality. Therefore the COM5A1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5A1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-24 COM5A Register Bits

Register Bits	Value	Description
COM5A1:0	0	Normal operation

Register Bits	Value	Description
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 5:4 – COM5B1:0 - Compare Output Mode for Channel B**

The Timer/Counter5 has only limited functionality. Therefore the COM5B1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5B1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-25 COM5B Register Bits

Register Bits	Value	Description
COM5B1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 3:2 – COM5C1:0 - Compare Output Mode for Channel C**

The Timer/Counter5 has only limited functionality. Therefore the COM5C1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5C1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-26 COM5C Register Bits

Register Bits	Value	Description
COM5C1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 1:0 – WGM51:50 - Waveform Generation Mode**

Combined with the WGM53:2 bits found in the TCCR5B 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. 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. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter5 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-27 WGM5 Register Bits

Register Bits	Value	Description
WGM51:50	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit

Register Bits	Value	Description
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.47 TCCR5B – Timer/Counter5 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($121)	ICNC5	ICES5	Res	WGM53	WGM52	CS52	CS51	CS50	TCCR5B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC5 - Input Capture 5 Noise Canceller**

Timer/Counter5 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 6 – ICES5 - Input Capture 5 Edge Select**

Timer/Counter5 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM51:50 - Waveform Generation Mode**

Combined with the WGM51:0 bits found in the TCCR5A 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. 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. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter5 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-28 WGM5 Register Bits

Register Bits	Value	Description
WGM51:50	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit

Register Bits	Value	Description
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS52:50 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter5 according to the following table. External pin modes cannot be used for the Timer/Counter5.

Table 18-29 CS5 Register Bits

Register Bits	Value	Description
CS52:50	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	Reserved
	0x07	Reserved

18.11.48 TCCR5C – Timer/Counter5 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($122)	FOC5A	FOC5B	FOC5C	Res4	Res3	Res2	Res1	Res0	TCCR5C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC5A - Force Output Compare for Channel A**

The FOC5A bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5A bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5A bits are implemented as strobes. Therefore it is the value present in the COM5A1:0 bits that determine the effect of the forced compare. A FOC5A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5A as TOP. The FOC5A bits are always read as zero.

- **Bit 6 – FOC5B - Force Output Compare for Channel B**

The FOC5B bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5B bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5B bits are implemented as strobes. Therefore it is the value present in the COM5B1:0 bits that determine the effect of the forced compare. A FOC5B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5B as TOP. The FOC5B bits are always read as zero.

- **Bit 5 – FOC5C - Force Output Compare for Channel C**

The FOC5C bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5C bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5C bits are implemented as strobes. Therefore it is the value present in the COM5C1:0 bits that determine the effect of the forced compare. A FOC5C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5C as TOP. The FOC5C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.49 TCNT5H – Timer/Counter5 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($125)	TCNT5H7:0								TCNT5H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT5H and TCNT5L, combined TCNT5) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT5) while the counter is running introduces a risk of missing a compare match between TCNT5 and one of the OCR5x Registers. Writing to the TCNT5 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT5H7:0 - Timer/Counter5 High Byte**

18.11.50 TCNT5L – Timer/Counter5 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($124)	TCNT5L7:0								TCNT5L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT5H and TCNT5L, combined TCNT5) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit

counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT5) while the counter is running introduces a risk of missing a compare match between TCNT5 and one of the OCR5x Registers. Writing to the TCNT5 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT5L7:0 - Timer/Counter5 Low Byte**

18.11.51 OCR5AH – Timer/Counter5 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($129)	OCR5AH7:0								OCR5AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5AH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.52 OCR5AL – Timer/Counter5 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($128)	OCR5AL7:0								OCR5AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5AL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.53 OCR5BH – Timer/Counter5 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12B)	OCR5BH7:0								OCR5BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5BH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.54 OCR5BL – Timer/Counter5 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12A)	OCR5BL7:0								OCR5BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5BL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.55 OCR5CH – Timer/Counter5 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12D)	OCR5CH7:0								OCR5CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5CH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.56 OCR5CL – Timer/Counter5 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12C)	OCR5CL7:0								OCR5CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5CL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.57 ICR5H – Timer/Counter5 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($127)	ICR5H7:0								ICR5H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter5 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT5) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR5H7:0 - Timer/Counter5 Input Capture Register High Byte**

18.11.58 ICR5L – Timer/Counter5 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($126)	ICR5L7:0								ICR5L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter5 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT5) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR5L7:0 - Timer/Counter5 Input Capture Register Low Byte**

18.11.59 TIMSK5 – Timer/Counter5 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($73)	Res1	Res0	ICIE5	Res	OCIE5C	OCIE5B	OCIE5A	TOIE5	TIMSK5
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE5 - Timer/Counter5 Input Capture Interrupt Enable**

The Timer/Counter5 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE5C - Timer/Counter5 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5C Flag, located in TIFR5, is set.

- **Bit 2 – OCIE5B - Timer/Counter5 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5B Flag, located in TIFR5, is set.

- **Bit 1 – OCIE5A - Timer/Counter5 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5A Flag, located in TIFR5, is set.

- **Bit 0 – TOIE5 - Timer/Counter5 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV5 Flag, located in TIFR5, is set.

18.11.60 TIFR5 – Timer/Counter5 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	Res1	Res0	ICF5	Res	OCF5C	OCF5B	OCF5A	TOV5	TIFR5
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF5 - Timer/Counter5 Input Capture Flag**

The Timer/Counter5 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF5C - Timer/Counter5 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register C (OCR5C). Note that a Forced Output Compare (FOC5C) strobe will not set the OCF5C Flag. OCF5C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF5C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF5B - Timer/Counter5 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register B (OCR5B). Note that a Forced Output Compare (FOC5B) strobe will not set the OCF5B Flag. OCF5B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF5B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF5A - Timer/Counter5 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register A (OCR5A). Note that a Forced Output Compare (FOC5A) strobe will not set the OCF5A Flag. OCF5A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF5A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV5 - Timer/Counter5 Overflow Flag**

The setting of this flag is dependent of the WGM53:0 bits setting of the Timer/Counter5 Control Register. In Normal and CTC modes, the TOV5 Flag is set when the timer overflows. TOV5 is automatically cleared when the Timer/Counter5 Overflow Interrupt Vector is executed. Alternatively, TOV5 can be cleared by writing a logic one to its bit location.

19 Timer/Counter 0, 1, 3, 4, and 5 Prescaler

Timer/Counter 0, 1, 3, 4, and 5 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to all Timer/Counters. T*n* is used as a general name, *n* = 0, 1, 3, 4, or 5.

19.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CS*n*2:0 = 1). This provides the fastest operation with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$ or $f_{CLK\_I/O}/1024$.

19.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by the Timer/Counter T*n*. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CS*n*2:0 > 1). The number of system clock cycles from the moment the timer is enabled until the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

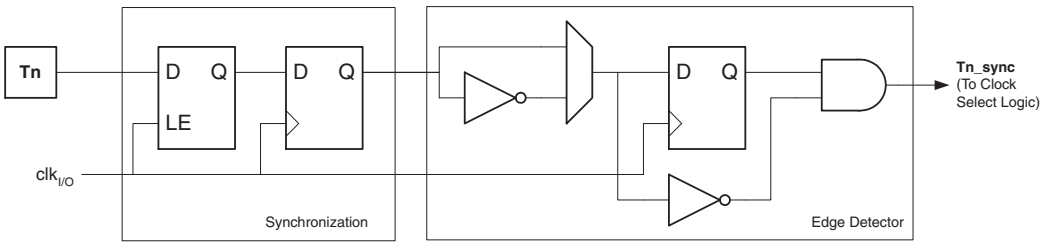
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all connected Timer/Counters.

19.3 External Clock Source

An external clock source applied to the T*n* pin can be used as Timer/Counter clock (clk_{Tn}). The T*n* pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 19-1 shows a functional equivalent block diagram of the T*n* synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{Tn} pulse for each positive (CS*n*2:0 = 7) or negative (CS*n*2:0 = 6) edge it detects.

Figure 19-1. Tn/T0 Pin Sampling

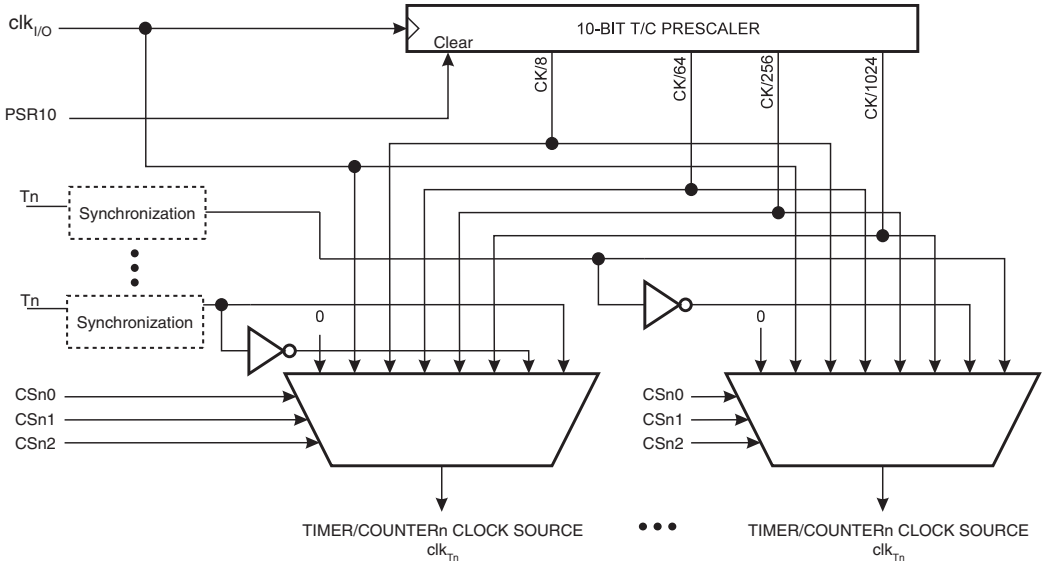


The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge applied to the T*n* pin to the counter being updated.

Enabling and disabling of the clock input must be done when T*n* has been stable for at least one system clock cycle. Otherwise there is a risk of generating a false Timer/Counter clock pulse.

Each half period of the applied, external clock must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of a detectable external clock is half the sampling frequency (Nyquist sampling theorem). However due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator and capacitors) tolerances, it is recommended to limit the maximum frequency of an external clock source to less than $f_{clk\_I/O}/2.5$. An external clock source can not be prescaled.

Figure 19-2. Prescaler for synchronous Timer/Counters



19.4 Register Description

19.4.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	TSM	Res4	Res3	Res2	Res1	Res0	PSRASY	PSRSYNC	GTCCR
Read/Write	RW	R	R	R	R	R	R	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM - Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during the configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware and the Timer/Counters simultaneously start counting.

- **Bit 6:2 – Res4:0 - Reserved**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1 – PSRASY - Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

- **Bit 0 – PSRSYNC - Prescaler Reset for Synchronous Timer/Counters**

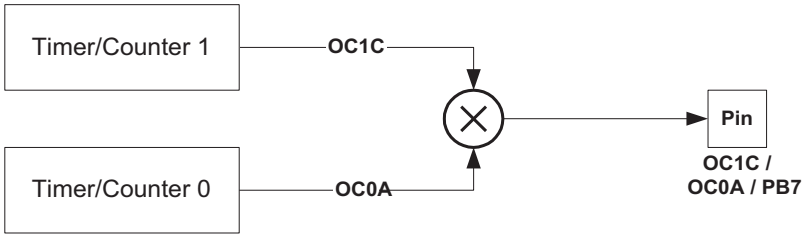
When this bit is one, the Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 share the same prescaler and a reset of this prescaler will affect all timers.

20 Output Compare Modulator (OCM1C0A)

20.1 Overview

The Output Compare Modulator (OCM) allows generation of waveforms modulated with a carrier frequency. The modulator uses the outputs from the Output Compare Unit C of the 16-bit Timer/Counter1 and the Output Compare Unit of the 8-bit Timer/Counter0. For more details about these Timer/Counters see "Timer/Counter 0, 1, 3, 4, and 5 Prescaler" on page 304 and "8-bit Timer/Counter2 with PWM and Asynchronous Operation" on page 309.

Figure 20-1. Output Compare Modulator, Block Diagram



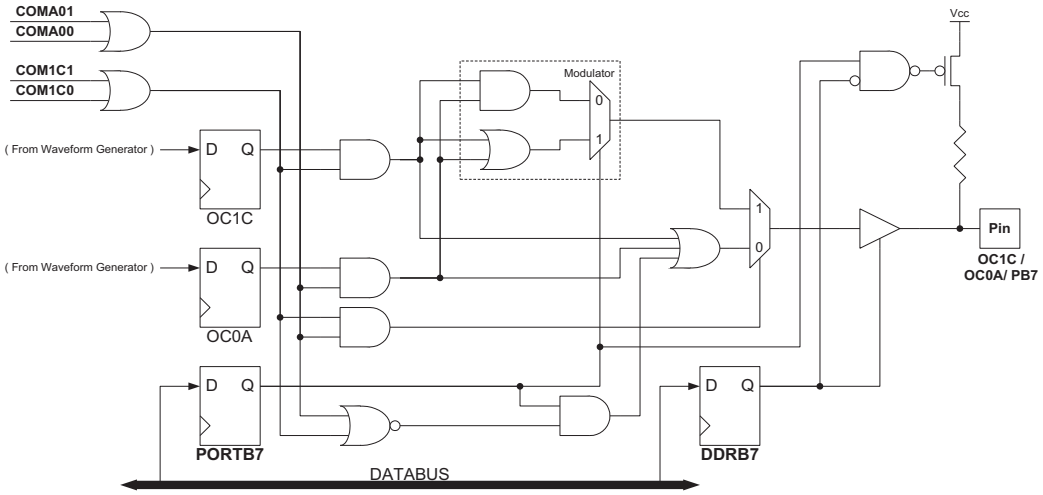
When the modulator is enabled, the two output compare channels are modulated together as shown in the block diagram (Figure 20-1 above).

20.2 Description

The Output Compare unit 1C and Output Compare unit 2 share the PB7 port pin for output. The outputs of the Output Compare units (OC1C and OC0A) override the normal PORTB7 Register when one of them is enabled (i.e., when COMnx1:0 is not equal to zero). When both OC1C and OC0A are enabled at the same time, the modulator is automatically enabled.

The functional equivalent schematic of the modulator is shown on in the following figure. The schematic includes part of the Timer/Counter units and the port B bit 7 output driver circuit.

Figure 20-2. Output Compare Modulator, Schematic

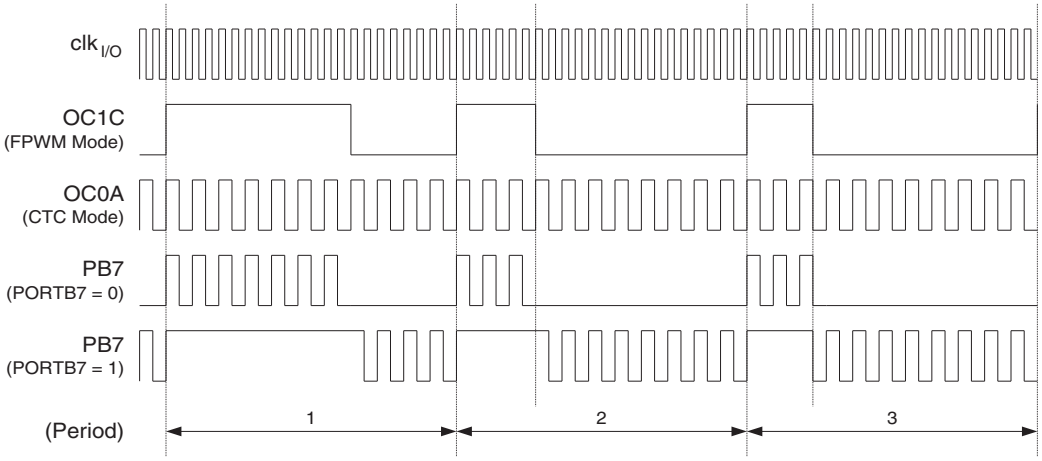


When the modulator is enabled the type of modulation (logical AND or OR) can be selected by the PORTB7 Register. Note that the DDRB7 controls the direction of the port independent of the COMnx1:0 bit setting.

20.3 Timing Example

Figure 20-3 below illustrates the modulator in action. In this example the Timer/Counter1 is set to operate in fast PWM mode (non-inverted) and Timer/Counter0 uses CTC waveform mode with toggle Compare Output mode (COMnx1:0 = 1).

Figure 20-3. Output Compare Modulator, Timing Diagram



In this example Timer/Counter2 provides the carrier while the modulating signal is generated by the Output Compare unit C of the Timer/Counter1.

The resolution of the PWM signal (OC1C) is reduced by the modulation. The reduction factor is equal to the number of system clock cycles of one period of the carrier (OC0A). In this example the resolution is reduced by a factor of two. The reason for the reduction is illustrated in Figure 20-3 above at the second and third period of the PB7 output when PORTB7 equals zero. The period 2 high time is one cycle longer than the period 3 high time, but the result on the PB7 output is equal in both periods.

21 8-bit Timer/Counter2 with PWM and Asynchronous Operation

21.1 Features

Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single channel counter**
- **Clear timer on compare match (auto reload)**
- **Glitch-free, phase-correct pulse-width modulator (PWM)**
- **Frequency generator**
- **10 bit clock prescaler**
- **Overflow and compare match interrupt sources (TOV2, OCF2A and OCF2B)**
- **Able to run with external 32 kHz watch crystal independent of the I/O clock**

21.2 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown on Figure 21-1 on page 310. For the current placement of I/O pins, see chapter "Pin Configurations" on page 2. 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 the "Register Description" on page 323.

The Power Reduction Timer/Counter2 bit PRTIM2 in register PRR0 (see "PRR0 – Power Reduction Register0" on page 167) must be written to zero to enable Timer/Counter2 module.

Note: OC2B is implemented but not routed to a pin and for this reason it can't be used.

21.2.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8 bit registers. Interrupt request (abbreviated to Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, asynchronously clocked from the TOSC1/2 pins or alternatively from the Automated Meter Reading (AMR) pin as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).

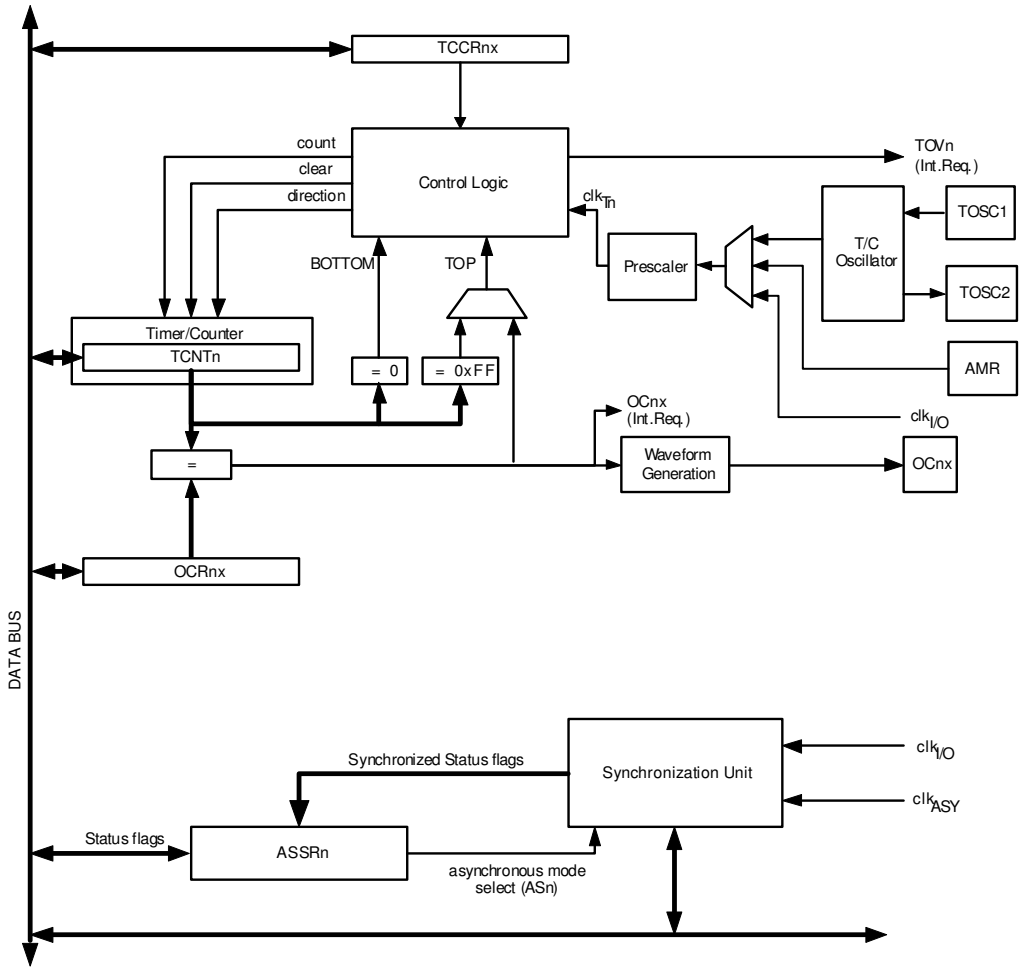
The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See chapter "Output Compare Unit" on page 316 for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

21.2.2 Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the

register or bit defines in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value and so on.

Figure 21-1. 8-bit Timer/Counter Block Diagram



The definitions in Table Table 21-1 below are also used extensively throughout the section.

Table 21-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

21.3 Timer/Counter Clock Sources

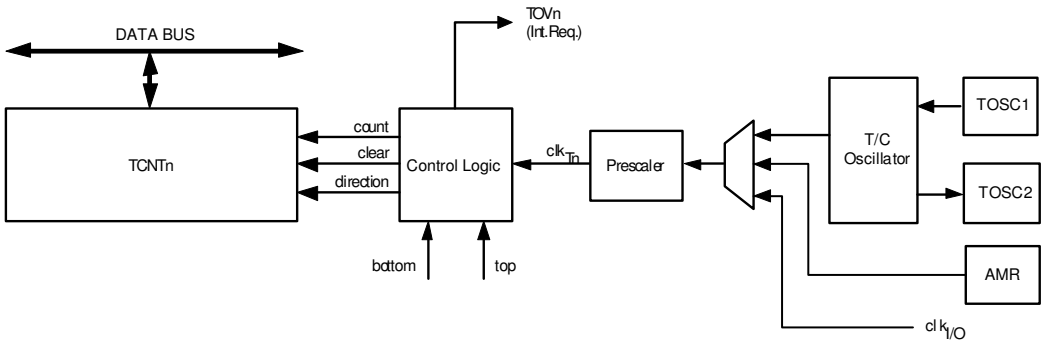
The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is either taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2 or from the AMR pin. For details on asynchronous operation, see section

"Asynchronous Operation of Timer/Counter2" on page 320. For details on clock sources and prescaler, see section "Timer/Counter Prescaler" on page 322.

21.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 21-2 below shows a block diagram of the counter and its surrounding environment.

Figure 21-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2A) and the WGM22 located in the Timer/Counter Control Register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see chapter "Modes of Operation" below.

The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

21.5 Modes of Operation

The mode of operation, i.e., the behaviour of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM22:0) and Compare Output mode (COM2x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM2x1:0 bits control whether the PWM output generated should be inverted or

not (inverted or non-inverted PWM). For non-PWM modes the COM2x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (see chapter "Compare Match Output Unit" on page 317).

For detailed timing information refer to chapter "Timer/Counter Timing Diagrams" on page 319.

The following table shows the function of the WGM22:0 bits of registers TCCR2A and TCCR2B. 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.

Table 21-2. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	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	BOTTOM	TOP

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

21.5.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM22:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8 bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However combined with the timer overflow interrupt that automatically clears the TOV2 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

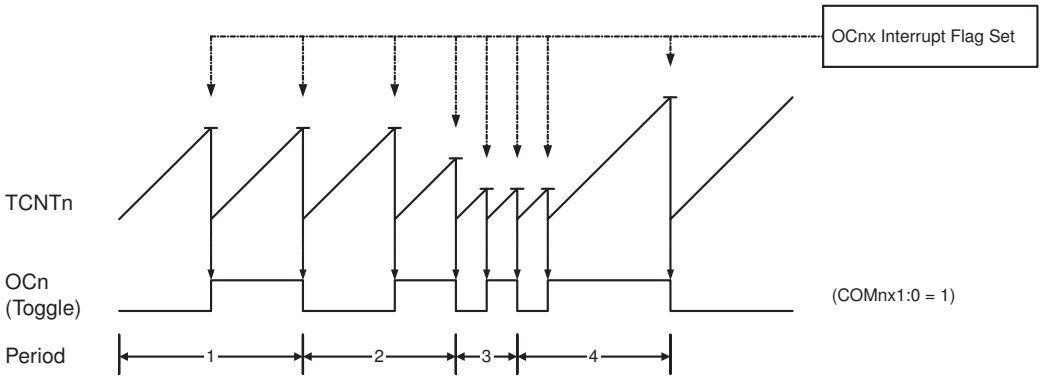
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

21.5.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM22:0 = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Table 20-3. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.

Figure 21-3. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.

For generating a waveform output in CTC mode, the OC2A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM2A1:0 = 1). The OC2A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC2A} = f_{clk\_I/O}/2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

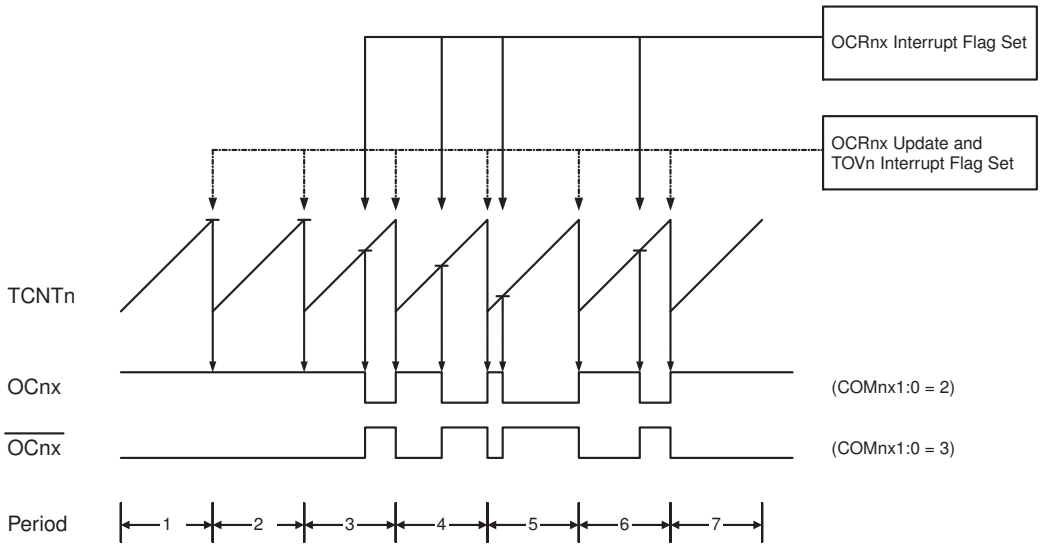
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

21.5.3 Fast PWM Mode

The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM22:0 = 3, and OCR2A when WGM22:0 = 7 (see section "Register Description" on page 323 for register TCCR2A). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2x Register at the compare match between OCR2x and TCNT2, and clearing (or setting) the OC2x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

Figure 21-4. Fast PWM Mode, Timing Diagram



The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM2A1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x1:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk\_I/O}/2$ when OCR2A is set to zero. This feature is similar to the OC2A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

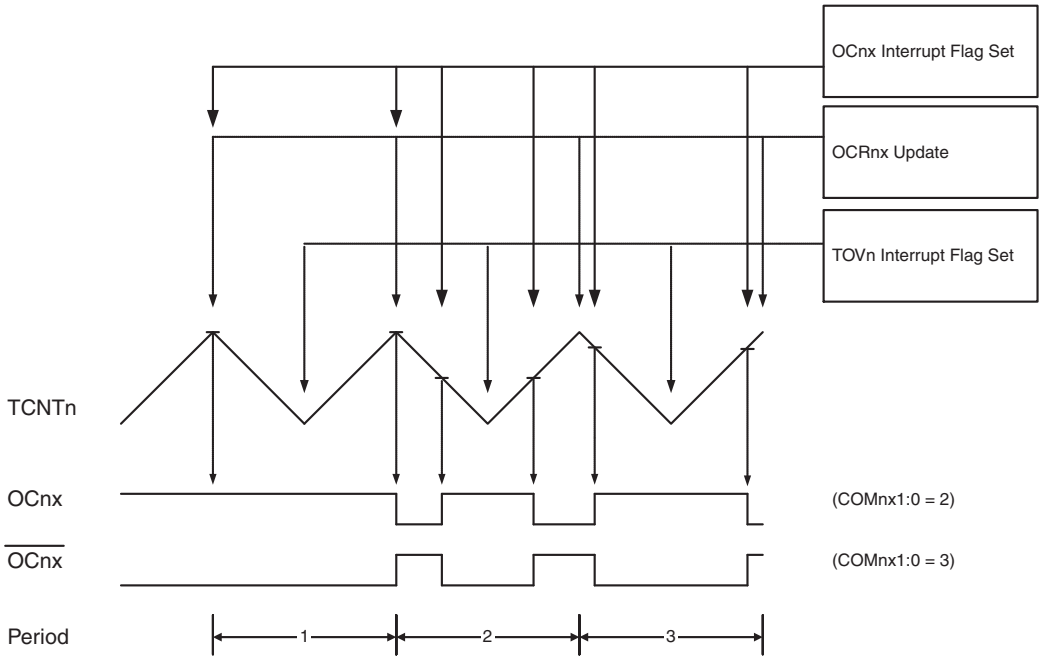
21.5.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM22:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM22:0 = 1, and OCR2A when WGM22:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x while up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT2 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 21-5 on page 315. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram

includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 21-5**.** Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM22:0 = 3, and OCR2A when WGM22:0 = 7 (see section "Register Description" on page 323 for register TCCR2A). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2x Register at the compare match between OCR2x and TCNT2 when the counter increments, and setting (or clearing) the OC2x Register at compare match between OCR2x and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 21-5 above OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

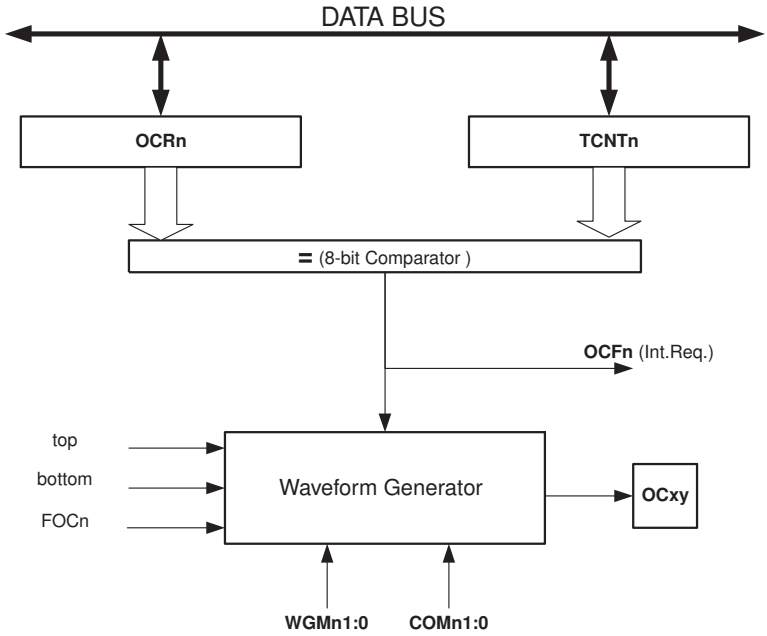
- OCR2A changes its value from MAX, like in Figure 21-5 on page 315. When the OCR2A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

21.6 Output Compare Unit

The 8 bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the Output Compare Flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the Output Compare 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 WGM22:0 bits and Compare Output mode (COM2x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (chapter "Modes of Operation" on page 311).

Figure 21-6 below shows a block diagram of the Output Compare unit.

Figure 21-6**.** Output Compare Unit, Block Diagram



The OCR2x Register is 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 OCR2x 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 OCR2x Register access may seem complex, but this is not the case. When the double buffering is enabled, the CPU has access to the OCR2x Buffer Register, and if double buffering is disabled the CPU will access the OCR2x directly.

21.6.1 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 (FOC2x) bit. Forcing compare match will not set the OCF2x Flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x1:0 bits settings define whether the OC2x pin is set, cleared or toggled).

21.6.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

21.6.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is down-counting.

The setup of the OC2x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2x value is to use the Force Output Compare (FOC2x) strobe bit in Normal mode. The OC2x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM2x1:0 bits are not double buffered together with the compare value. A change of the COM2x1:0 bits will take effect immediately.

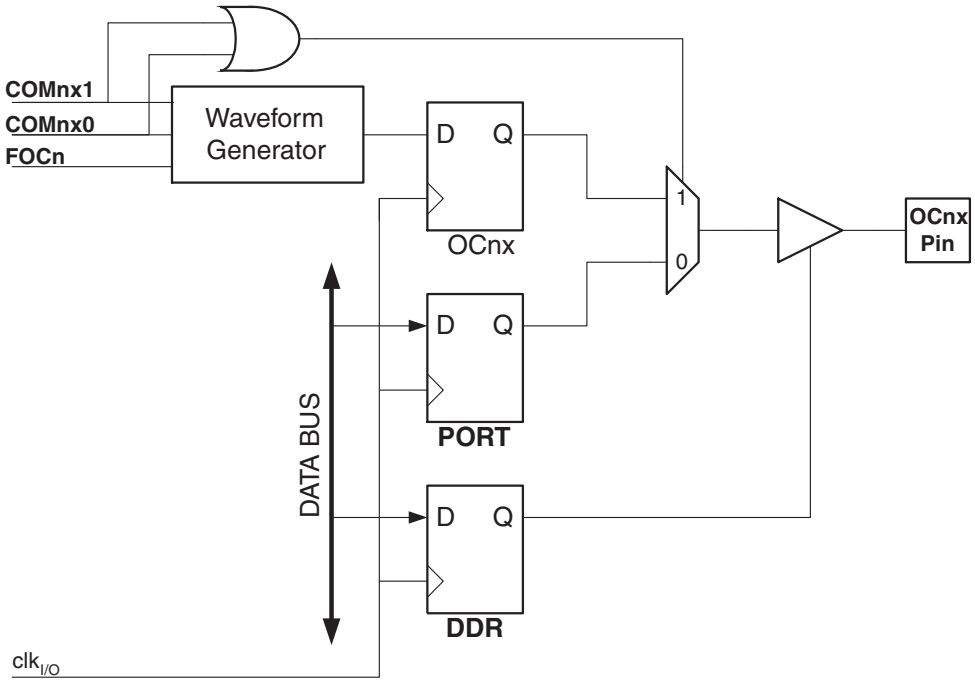
21.7 Compare Match Output Unit

The Compare Output mode (COM2x1:0) bits have two functions. The Waveform Generator uses the COM2x1:0 bits for defining the Output Compare (OC2x) state at the next compare match. Also, the COM2x1:0 bits control the OC2x pin output source. Figure 20-7 shows a simplified schematic of the logic affected by the COM2x1:0 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 COM2x1:0 bits are shown. When referring to the OC2x state, the reference is for the internal OC2x Register, not the OC2x pin.

The general I/O port function is overridden by the Output Compare (OC2x) from the Waveform Generator if either of the COM2x1:0 bits are set. However, the OC2x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2x pin (DDR_OC2x) must be set as output before the OC2x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC2x state before the output is enabled. Note that some COM2x1:0 bit settings are reserved for certain modes of operation. See section "Register Description" on page 323 for details.

Figure 21-7. Compare Match Output Unit, Schematic



21.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM2x1:0 bits differently in normal, CTC, and PWM modes. Setting the COM2x1:0 = 0 for all modes tells the Waveform Generator that no action on the OC2x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes for fast PWM mode and for phase correct PWM refer to section "Register Description" on page 323 for register TCCR2A**.** A change of the COM2x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2x strobe bits.

The following table shows the COM2*x*1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 21-3. Compare Output Mode, non-PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected;
0	1	Toggle OC2*x* on Compare Match;
1	0	Clear OC2*x* on Compare Match;
1	1	Set OC2*x* on Compare Match;

Table 17-3 shows the COM2*x*1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 21-4. Compare Output Mode, Fast PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match. OC2B: not applicable, reserved function;

COM2x1	COM2x0	Description
1	0	Clear OC2*x* on Compare Match, set OC2*x* at BOTTOM, (non-inverting mode).
1	1	Set OC2*x* on Compare Match, clear OC2*x* at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR2*x* equals TOP and COM2*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 313.

Table 17-4 shows the COM2*x*1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 21-5. Compare Output Mode, Phase Correct PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match. OC2B: not applicable, reserved function;
1	0	Clear OC2*x* on Compare Match when up-counting. Set OC2*x* on Compare Match when down-counting.
1	1	Set OC2*x* on Compare Match when up-counting. Clear OC2*x* on Compare Match when down-counting.

Note: 1. A special case occurs when OCR2*x* equals TOP and COM2*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 314 for more details.

21.8 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 21-8 below contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 21-8**.** Timer/Counter Timing Diagram, no Prescaling

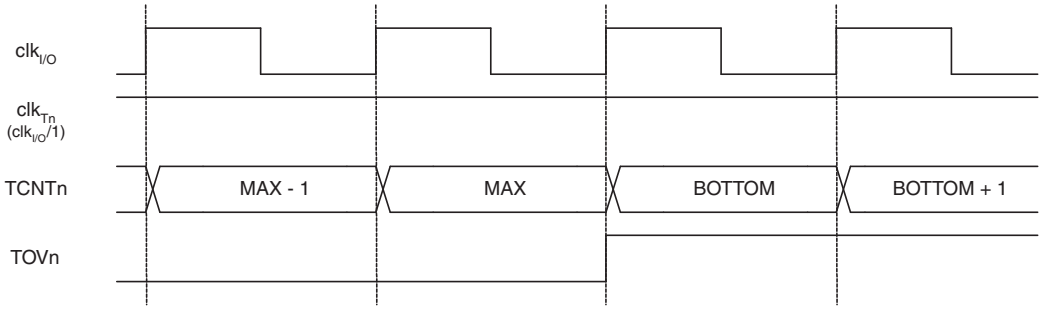


Figure 21-9 below shows the same timing data, but with the prescaler enabled.

Figure 21-9**.** Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

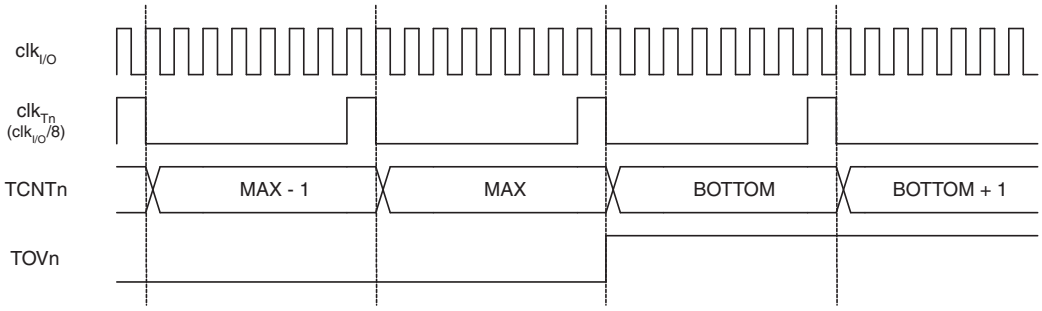


Figure 21-10 below shows the setting of OCF2A in all modes except CTC mode.

Figure 21-10**.** Timer/Counter Timing Diagram, Setting of OCF2A, with Prescaler ($f_{clk\_I/O}/8$)

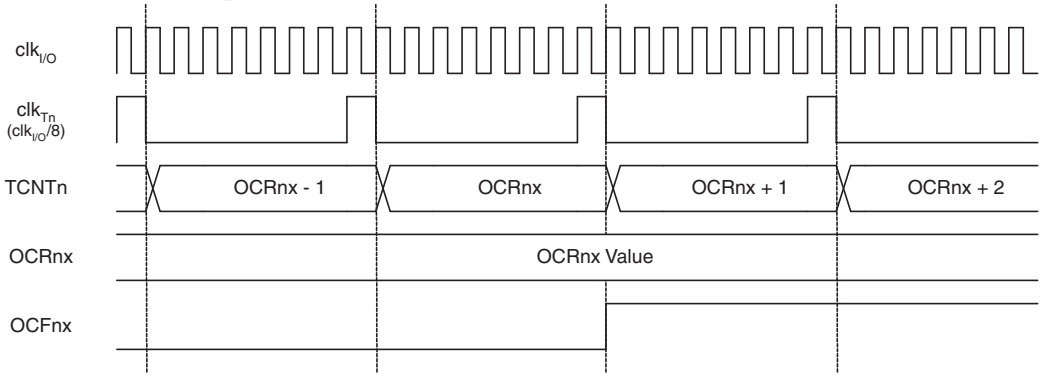
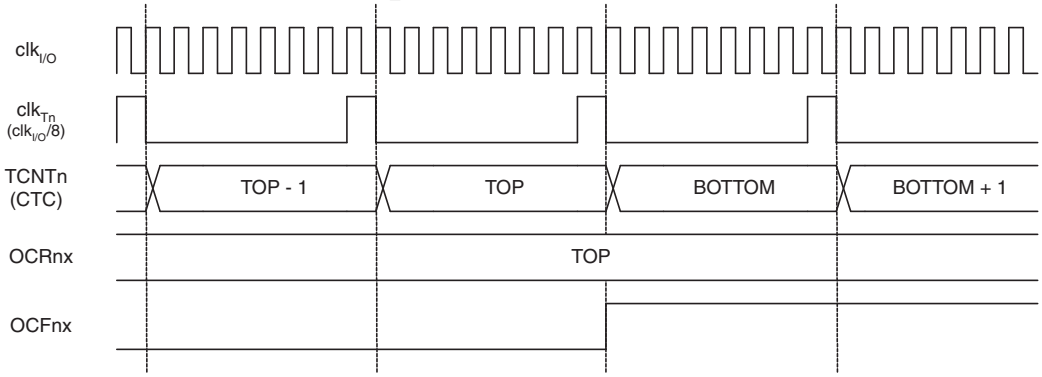


Figure 21-11 below shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 21-11**.** Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)



21.9 Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCIE2x and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2x, and TCCR2x.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2xUB, and TCR2xUB.
  5. Clear the Timer/Counter2 Interrupt Flags.
  6. Enable interrupts, if needed.
- The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2x, or TCCR2x, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the five mentioned registers have their individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2x write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save or ADC Noise Reduction mode after having written to TCNT2, OCR2x, or TCCR2x, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if any of the Output Compare2 interrupt is used to wake up the device, since the Output Compare function is disabled during writing to OCR2x or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the corresponding OCR2xUB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or ADC Noise Reduction mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering Powersave or ADC Noise Reduction mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  1. Write a value to TCCR2x, TCNT2, or OCR2x.
  2. Wait until the corresponding Update Busy Flag in ASSR returns to zero. .
  3. Enter Power-save or ADC Noise Reduction mode.
- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.
- Description of wake up from Power-save or ADC Noise Reduction mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always

advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.

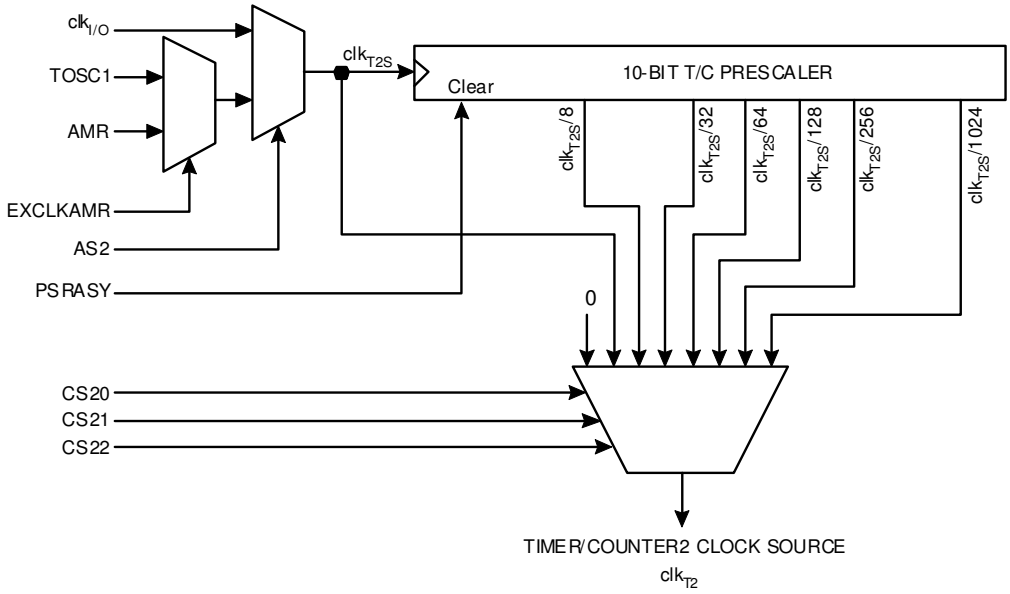
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Powersave mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:

  1. Write any value to either of the registers OCR2x or TCCR2x.
  2. Wait for the corresponding Update Busy Flag to be cleared.
  3. Read TCNT2.

- During asynchronous operation, the synchronization of the Interrupt Flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The Output Compare pin is changed on the timer clock and is not synchronized to the processor clock.
- If the CPU wakes up from asynchronous timer and goes back to sleep again, it may wakeup multiple times or the IRQ is called multiple times. This may be avoided if the CPU waits with the next sleep instruction until the next asynchronous clock arrives.

21.10 Timer/Counter Prescaler

Figure 21-12**.** Prescaler for Timer/Counter2



The register ASSR defines the clock source for the asynchronous Timer/Counter2. The clock source for Timer/Counter2 is named clk_{T2S}. clk_{T2S} is by default connected to the main system I/O clock clk_{IO}. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked either from the TOSC1 or from the AMR pin. This enables the use of Timer/Counter2 as a Real Time Counter (RTC).

The TOSC1 pin is selected by setting the EXCLKAMR bit in the ASSR register to logic zero. Under this condition TOSC1 and TOSC2 are disconnected from Port G and a crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. By setting the EXCLK bit in the ASSR, a 32 kHz external clock can be applied on TOSC1.

Setting the EXCLKAMR bit to logic one selects the AMR pin as the Timer/Counter2 clock source. Thus the 32 kHz oscillator can be used by the MAC symbol counter while the Timer/Counter2 uses pin AMR as clock source, see "MAC Symbol Counter" on page 133.

A complete overview of the implemented asynchronous clock sources can be found in Table 21-6 below. The last column mentions which pins are available for GPIO functionality. For details about the ASSR register refer to section "Register Description" below.

For Timer/Counter2, the possible pre-scaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, clk_{T2S} as well as 0 (stop) may be selected. Setting the PSRASY bit in GTCCR resets the prescaler. This allows the user to operate with a predictable prescaler.

Table 21-6. Asynchronous clock selection for Timer/Counter2 and Symbol-Counter

AS2	EXCLK	EXCLKAMR	Timer/Counter2 clock source	32 kHz crystal Osc. (TOSC1/TOSC2)	PG2, PG3, PG4 as GPIOs
0	0	0	cp2io	off	PG2, PG3, PG4
0	1	0	not defined	not defined	not defined
1	0	0	32 kHz crystal Osc	on	PG2
1	1	0	TOSC1 (PG4)	off	PG2, PG3
0	0	1	cp2io	off	PG2, PG3, PG4
0	1	1	not defined	not defined	not defined
1	0	1	AMR (PG2)	on	
1	1	1	AMR (PG2)	off	PG3, PG4

21.11 Register Description

21.11.1 TIMSK2 – Timer/Counter Interrupt Mask register

Bit	7	6	5	4	3	2	1	0	
NA ($70)	Res4	Res3	Res2	Res1	Res0	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCIE2B - Timer/Counter2 Output Compare Match B Interrupt Enable**

When the OCIE2B bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2B bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

- **Bit 1 – OCIE2A - Timer/Counter2 Output Compare Match A Interrupt Enable**

When the OCIE2A bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2A bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

- **Bit 0 – TOIE2 - Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs i.e., when the TOV2 bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

21.11.2 TIFR2 – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	Res4	Res3	Res2	Res1	Res0	OCF2B	OCF2A	TOV2	TIFR2
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCF2B - Output Compare Flag 2 B**

The OCF2B bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2B Output Compare Register2. OCF2B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2B (Timer/Counter2 Compare Match Interrupt Enable), and OCF2B are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 1 – OCF2A - Output Compare Flag 2 A**

The OCF2A bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2A Output Compare Register2. OCF2A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2A (Timer/Counter2 Compare Match Interrupt Enable), and OCF2A are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 0 – TOV2 - Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE2A (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 changes counting direction at 0x00.

21.11.3 TCCR2A – Timer/Counter2 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($B0)	COM2A1	COM2A0	COM2B1	COM2B0	Res1	Res0	WGM21	WGM20	TCCR2A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM2A1:0 - Compare Match Output A Mode**

These bits control the Output Compare pin (OC2A) behavior. If one or both of the COM2A1:0 bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2A pin must be set in order to enable the output driver. When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM22:0 bit setting. The following table shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM). Refer to section "Compare Match Output Unit" for a description of the functionality in the other modes.

Table 21-7 COM2A Register Bits

Register Bits	Value	Description
COM2A1:0	0	Normal port operation, OC2A disconnected
	1	Toggle OC2A on Compare Match
	2	Clear OC2A on Compare Match
	3	Set OC2A on Compare Match

- **Bit 5:4 – COM2B1:0 - Compare Match Output B Mode**

These bits control the Output Compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver. When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. The following table shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM). Refer to section "Compare Match Output Unit" for a description of the functionality in the other modes.

Table 21-8 COM2B Register Bits

Register Bits	Value	Description
COM2B1:0	0	Normal port operation, OC2B disconnected
	1	Toggle OC2B on Compare Match
	2	Clear OC2B on Compare Match
	3	Set OC2B on Compare Match

- **Bit 3:2 – Res1:0 - Reserved**
- **Bit 1:0 – WGM21:20 - Waveform Generation Mode**

Combined with the WGM22 bit found in the TCCR2B 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. Modes of operation supported by the Timer/Counter2 unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see section "Modes of Operation" for details).

Table 21-9 WGM2 Register Bits

Register Bits	Value	Description
WGM21:20	0x0	Normal mode of operation
	0x1	PWM, phase correct, TOP=0xFF
	0x2	CTC, TOP = OCRA
	0x3	Fast PWM, TOP=0xFF
	0x4	Reserved
	0x5	PWM, Phase correct, TOP = OCRA
	0x6	Reserved
	0x7	Fast PWM, TOP=OCRA

21.11.4 TCCR2B – Timer/Counter2 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($B1)	FOC2A	FOC2B	Res1	Res0	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC2A - Force Output Compare A**

The FOC2A bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written in PWM mode operation. When writing a logical one to the FOC2A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2A output is changed according to its COM2A1:0 bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A1:0 bits that determines the effect of the forced compare. A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP. The FOC2A bit is always read as zero.

- **Bit 6 – FOC2B - Force Output Compare B**

The FOC2B bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written in PWM mode operation. When writing a logical one to the FOC2B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2B output is changed according to its COM2B1:0 bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B1:0 bits that determines the effect of the forced compare. A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP. The FOC2B bit is always read as zero.

- **Bit 5:4 – Res1:0 - Reserved**
- **Bit 3 – WGM22 - Waveform Generation Mode**

Combined with the WGM21:0 bits found in the TCCR2A Register, this bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. See description of "TCCR2A - Timer/Counter2 Control Register A" for details.

- **Bit 2:0 – CS22:20 - Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter2. If external pin modes are used for the Timer/Counter2, transitions on the T2 pin will clock

the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 21-10 CS2 Register Bits

Register Bits	Value	Description
CS22:20	0x00	No clock source (Timer/Counter2 stopped)
	0x01	clk_T2S/1 (no prescaling)
	0x02	clk_T2S/8 (from prescaler)
	0x03	clk_T2S/32 (from prescaler)
	0x04	clk_T2S/64 (from prescaler)
	0x05	clk_T2S/128 (from prescaler)
	0x06	clk_T2S/256 (from prescaler)
	0x07	clk_T2S/1024 (from prescaler)

21.11.5 TCNT2 – Timer/Counter2

Bit	7	6	5	4	3	2	1	0	
NA ($B2)	TCNT27:20								TCNT2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the 8-bit counter unit of the Timer/Counter2. Writing to the TCNT2 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a Compare Match between TCNT2 and the OCR2x Registers.

- **Bit 7:0 – TCNT27:20 - Timer/Counter2 Byte**

21.11.6 OCR2A – Timer/Counter2 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
NA ($B3)	OCR2A7:0								OCR2A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2A pin.

- **Bit 7:0 – OCR2A7:0 - Output Compare Register**

21.11.7 OCR2B – Timer/Counter2 Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
NA ($B4)	OCR2B7:0								OCR2B
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2B pin.

- **Bit 7:0 – OCR2B7:0 - Output Compare Register**

21.11.8 ASSR – Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($B6)	EXCLKAMR	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	ASSR
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial	0	0	0	0	0	0	0	0	

The register ASSR controls the asynchronous clocks for Timer/Counter2 and enables the asynchronous 32kHz clock for the symbol counter. Three bits (AS2,EXCLK,EXCLKAMR) are used to control the clocks. Note, to prevent clock spikes on asynchronous clock wires, every access to ASSR should change only one of the three bits.

- **Bit 7 – EXCLKAMR - Enable External Clock Input for AMR**

The bit EXCLKAMR extends the available clock sources for Timer/Counter2. If this bit is written to one, and asynchronous clock is selected (bit AS2 set), AMR functionality is enabled and Timer/Counter2 is clocked by pin AMR.

- **Bit 6 – EXCLK - Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on Timer Oscillator 1 (TOSC1) pin instead of a 32 kHz crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal Oscillator will only run when this bit is zero.

- **Bit 5 – AS2 - Timer/Counter2 Asynchronous Mode**

When AS2 is written to zero, Timer/Counter2 is clocked from the I/O clock, clkI/O. When AS2 is written to one, Timer/Counter2 is clocked from a crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B might be corrupted.

- **Bit 4 – TCN2UB - Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 3 – OCR2AUB - Timer/Counter2 Output Compare Register A Update Busy**

When Timer/Counter2 operates asynchronously and OCR2A is written, this bit becomes set. When OCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2A is ready to be updated with a new value.

- **Bit 2 – OCR2BUB - Timer/Counter2 Output Compare Register B Update Busy**

When Timer/Counter2 operates asynchronously and OCR2B is written, this bit becomes set. When OCR2B has been updated from the temporary storage register,

this bit is cleared by hardware. A logical zero in this bit indicates that OCR2B is ready to be updated with a new value.

- **Bit 1 – TCR2AUB - Timer/Counter2 Control Register A Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2A is written, this bit becomes set. When TCCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2A is ready to be updated with a new value.

- **Bit 0 – TCR2BUB - Timer/Counter2 Control Register B Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2B is written, this bit becomes set. When TCCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2B is ready to be updated with a new value.

21.11.9 GTCCR – General Timer Counter Control register

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	TSM						PSRASY		GTCCR
Read/Write	RW						RW		
Initial Value	0						0		

- **Bit 7 – TSM - Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during the configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware and the Timer/Counters simultaneously start counting.

- **Bit 1 – PSRASY - Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

22 SPI- Serial Peripheral Interface

22.1 Features

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega128RFA1 and peripheral devices or between several AVR devices.

The ATmega128RFA1 SPI includes the following features:

- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

22.2 Functional Description

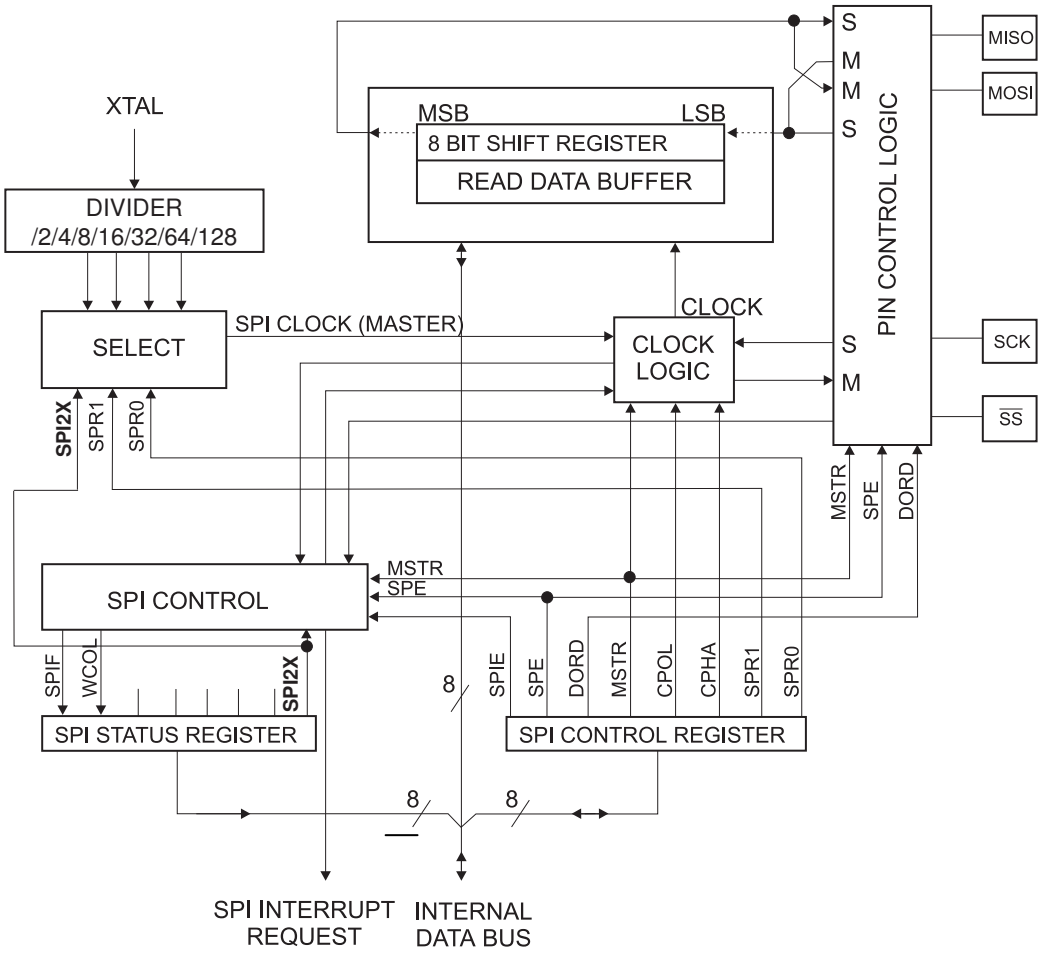
USART can also be used in Master SPI mode, see "USART in SPI Mode" on page 368. The Power Reduction SPI bit, PRSPI, in "PRR0 – Power Reduction Register0" on page 167 must be written to zero to enable SPI module. The block diagram of the SPI interface is shown in Figure 22-1 on page 331.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 22-2 on page 331. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

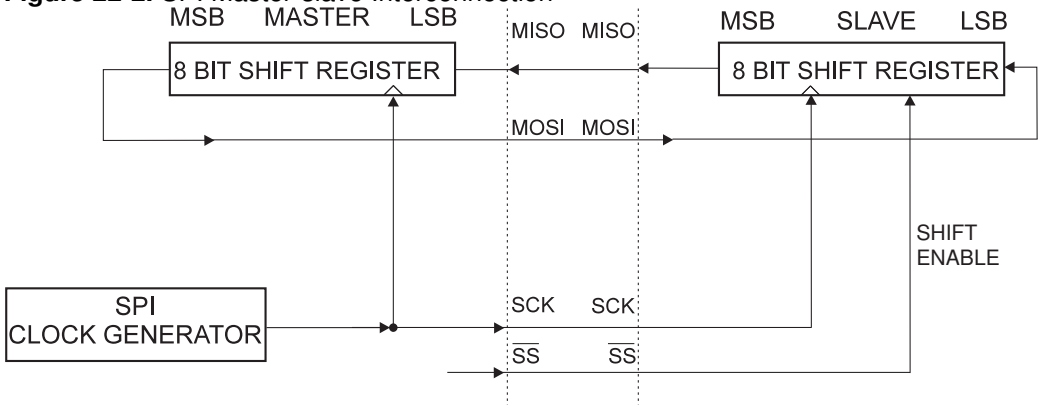
When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the SS pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 22-1. SPI Block Diagram[1]



Note: 1. Refer to Figure 1-1 on page 2 and Table 14-3 on page 193 for SPI pin placement.

Figure 22-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost. In SPI Slave mode, the

control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

Low period: longer than 2 CPU clock cycles
High period: longer than 2 CPU clock cycles

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 21-1. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 191.

Table 22-1. Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See "Alternate Functions of Port B" on page 192 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 and DDR_SPI with DDRB.

Assembly Code Example[1]

```
SPI_MasterInit:
   ; Set MOSI and SCK output, all others input
   ldi r17,(1<<DD_MOSI)|(1<<DD_SCK)
   out DDR_SPI,r17
   ; Enable SPI, Master, set clock rate fck/16
   ldi r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
   out SPCR,r17
   ret
SPI_MasterTransmit:
   ; Start transmission of data (r16)
   out SPDR,r16
Wait_Transmit:
   ; Wait for transmission complete
   sbis SPSR,SPIF
   rjmp Wait_Transmit
   ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
 while(!(SPSR & (1<<SPIF)))
 ;
}
```

Note: 1. See "About Code Examples" on page 7

Assembly Code Example[1]

```
SPI_SlaveInit:
   ; Set MISO output, all others input
   ldi r17,(1<<DD_MISO)
   out DDR_SPI,r17
   ; Enable SPI
   ldi r17,(1<<SPE)
   out SPCR,r17
   ret
SPI_SlaveReceive:
   ; Wait for reception complete
   sbis SPSR,SPIF
   rjmp SPI_SlaveReceive
   ; Read received data and return
   in r16,SPDR
   ret
```

C Code Example[1]

```
void SPI_SlaveInit(void)
{
  /* Set MISO output, all others input */
  DDR_SPI = (1<<DD_MISO);
  /* Enable SPI */
  SPCR = (1<<SPE);
}
  char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)))
  ;
  /* Return Data Register */
  return SPDR;
}
```

Note: 1. See "About Code Examples" on page 7;

22.3 $\overline{SS}$ Pin Functionality

22.3.1 Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high. The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

22.3.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin. If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave. If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master Mode.

22.3.3 Data Mode

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 22-3 below and Figure 22-4 below. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen in the summary of Table 22-2 below:

Table 22-2. CPOL Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)	0
CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)	1
CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)	2
CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)	3

Figure 22-3. SPI Transfer Format with CPHA = 0

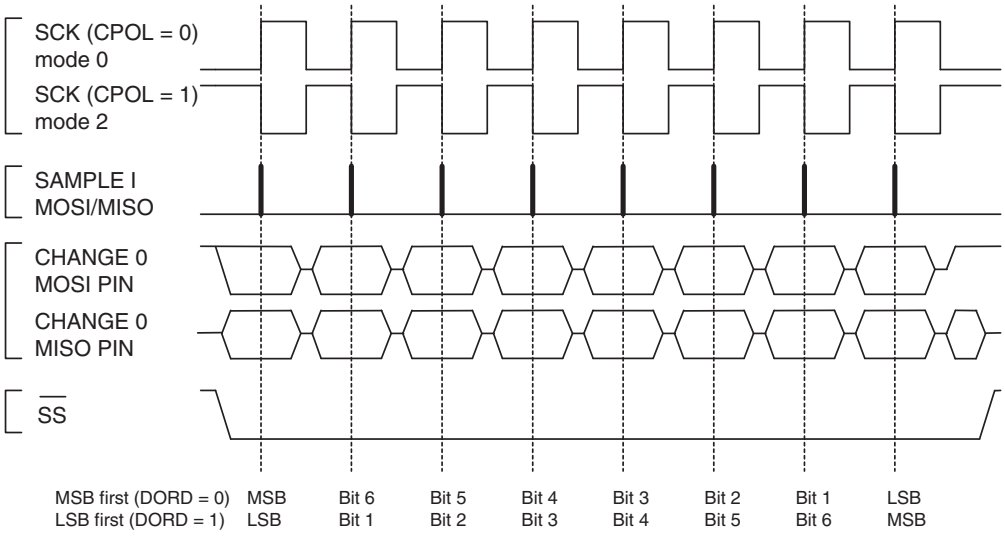
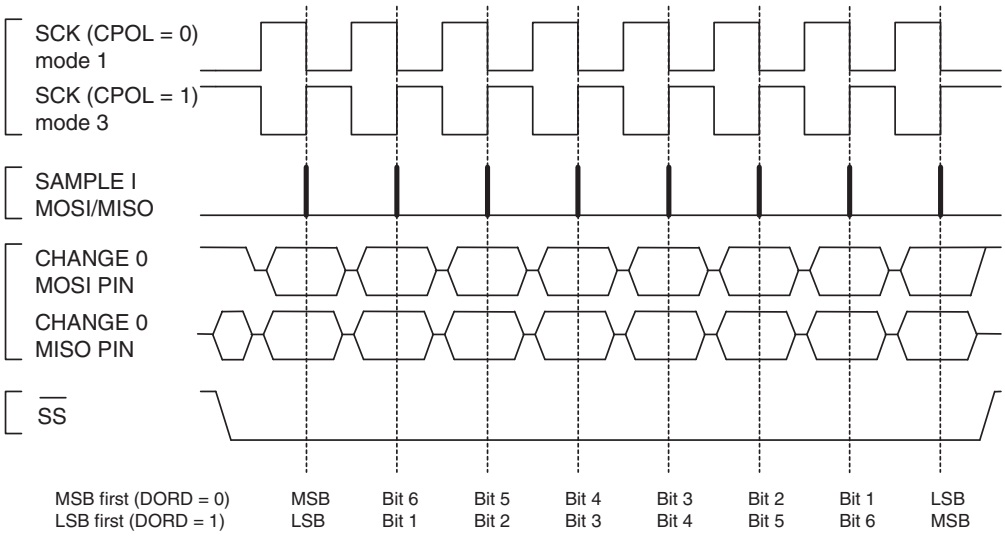


Figure 22-4. SPI Transfer Format with CPHA = 1



22.4 Register Description

22.4.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
$2C ($4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE - SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the Global Interrupt Enable bit in SREG is set.s

- **Bit 6 – SPE - SPI Enable**

When the SPE bit is set (one), the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD - Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first. When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR - Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If the Slave Select pin is configured as an input and is driven low while MSTR is set, MSTR will be cleared and SPIF in SPSR are set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL - Clock polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to the "Data Modes" section for an example. The CPOL functionality is summarized below.

Table 22-3 CPOL Register Bits

Register Bits	Value	Description
CPOL	0	Rising (Leading Edge), Falling (Trailing Edge)
	1	Falling (Leading Egde), Rising (Trailing Edge)

- **Bit 2 – CPHA - Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to the "Data Modes" section for an example. The CPOL functionality is summarized below.

Table 22-4 CPHA Register Bits

Register Bits	Value	Description
CPHA	0	Sample (Leading Edge), Setup (Trailing Edge)
	1	Setup (Leading Edge), Sample (Trailing Edge)

- **Bit 1:0 – SPR1:0 - SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency fosc is shown in the following table.

Table 22-5 SPR Register Bits

Register Bits	Value	Description
SPR1:0	0x00	fosc/4
	0x01	fosc/16
	0x02	fosc/64
	0x03	fosc/128
	0x04	fosc/2
	0x05	fosc/8
	0x06	fosc/32
	0x07	fosc/64

22.4.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
$2D ($4D)	SPIF	WCOL	Res4	Res3	Res2	Res1	Res0	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF - SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. The SPIF Flag is also set if the Slave Select pin is an input and is driven low when the SPI is in Master mode. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set and then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL - Write Collision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set and then accessing the SPI Data Register.

- **Bit 5:1 – Res4:0 - Reserved**
- **Bit 0 – SPI2X - Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode. This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower. The SPI interface on the ATmega128RFA1 is also used for program memory and EEPROM downloading or uploading. See section "Serial Downloading" for serial programming and verification.

22.4.3 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	SPDR7:0								SPDR
Read/Write	RW	RW	RW	RW	RW	RW	R	R	
Initial Value	X	X	X	X	X	X	0	0	

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

- **Bit 7:0 – SPDR7:0 - SPI Data Register**

23 USART

23.1 Features

- **Full duplex operation (independent serial receive and transmit registers)**
- **Asynchronous or synchronous operation**
- **Master or slave clocked synchronous operation**
- **High resolution baud rate generator**
- **Supports serial frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits**
- **Odd or even parity generation and parity check supported by hardware**
- **Data overrun detection**
- **Framing error detection**
- **Noise filtering includes false start bit detection and digital low pass filter**
- **3 separate interrupts on TX complete, TX data register empty and RX complete**
- **Multi-processor communication mode**
- **Double speed, asynchronous communication mode**

23.2 Overview

The Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

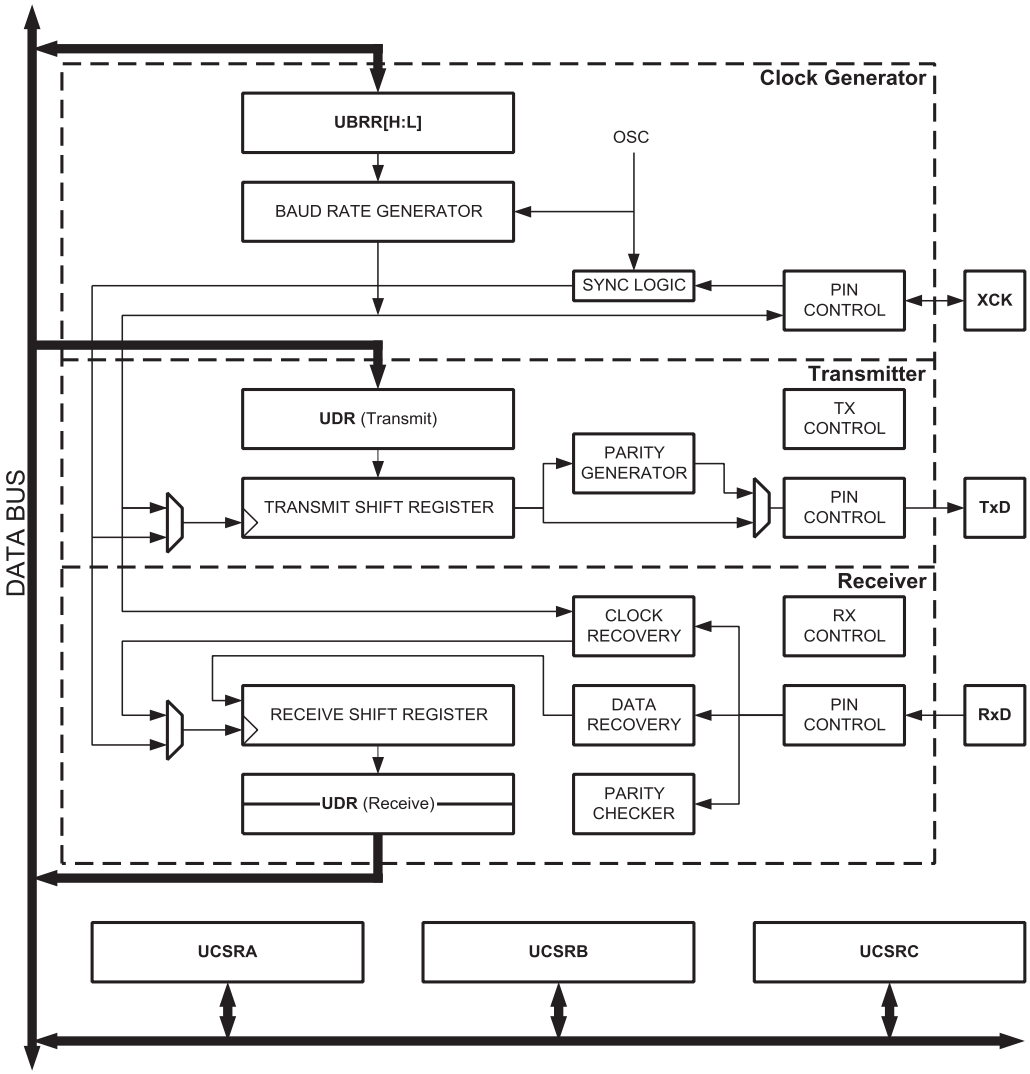
The ATmega128RFA1 has two USART's, USART0 and USART1. The functionality for all two USART's is described below. USART0 and USART1 have different I/O registers as shown in "Register Summary" on page 496.

A simplified block diagram of the USART transmitter is shown in Figure 23-1 on page 340 on page 340. CPU accessible I/O registers and I/O pins are shown in bold.

The Power Reduction USART0 bit, PRUSART0, in "PRR0 – Power Reduction Register0" on page 167 must be disabled by writing a logical zero to it. The Power Reduction USART1 bit, PRUSART1, in "PRR1 – Power Reduction Register 1" on page 168 must be disabled by writing a logical zero to it.

The dashed boxes in the block diagram Figure 23-1 on page 340 separate the three main parts of the USART (listed from the top): clock generator, transmitter and receiver. Control registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCK*n* (transfer clock) pin is only used by synchronous transfer mode. The transmitter consists of a single write buffer, a serial shift register, Parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a shift register and a two level receive buffer (UDR*n*). The receiver supports the same frame formats as the transmitter, and can detect frame, data overrun and parity errors.

Figure 23-1. USART Block Diagram[1]



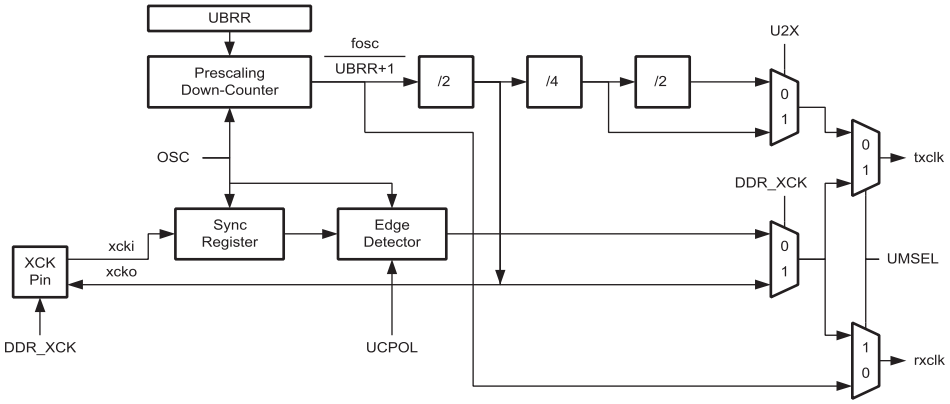
Note: 1. See "Figure 1-1" on page 2, Table 14-6 on page 195and Table 14-9 on page 197Table 14-9 on page 197for USART pin placement.

23.3 Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. The USART supports four modes of clock operation: Normal asynchronous, double speed asynchronous, master synchronous and slave synchronous mode. The UMSEL*n* bit in USART Control and Status Register C (UCSR*n*C) selects between asynchronous and synchronous operation. Double speed (asynchronous mode only) is controlled by the U2X*n* found in the UCSR*n*A register. When using synchronous mode (UMSEL*n* = 1), the data direction register for the XCK*n* pin (DDR_XCK*n*) controls whether the clock source is internal (master mode) or external (slave mode). The XCK*n* pin is only active when using synchronous mode.

Figure 22-2 on page 331 shows a block diagram of the clock generation logic.

Figure 23-2. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (internal signal).

rxclk Receiver base clock (internal signal).

xcki Input from XCK pin (internal signal). Used for synchronous slave operation.

xcko Clock output to XCK pin (internal signal). Used for synchronous master operation.

f_{osc} System clock frequency.

23.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 22-2 on page 331.

The USART Baud Rate Register (UBRR*n*) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{OSC}), is loaded with the UBRR*n* value each time the counter has counted down to zero or when the UBRRL*n* register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= $f_{OSC}/(UBRRn+1)$). The transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL*n*, U2X*n* and DDR_XCK*n* bits.

Table 23-1 below contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR*n* value for each mode of operation using an internally generated clock source.

Table 23-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X*n* = 0)	$BAUD = \frac{f_{osc}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{16\,BAUD} - 1$
Asynchronous Double Speed Mode (U2X*n* = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{8\,BAUD} - 1$

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2\,BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System oscillator clock frequency

UBRR*n* Contents of the UBRRH*n* and UBRRL*n* registers, (0-4095)

Some examples of UBRR*n* values for some system clock frequencies are found in Table 23-14 on page 365.

23.3.2 Double Speed Operation (U2X*n*)

The transfer rate can be doubled by setting the U2X*n* bit in UCSR*n*A. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the transmitter, there are no downsides.

23.3.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 22-2 on page 331 for details.

External clock input from the XCK*n* pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK*n* clock frequency is limited by the following equation:

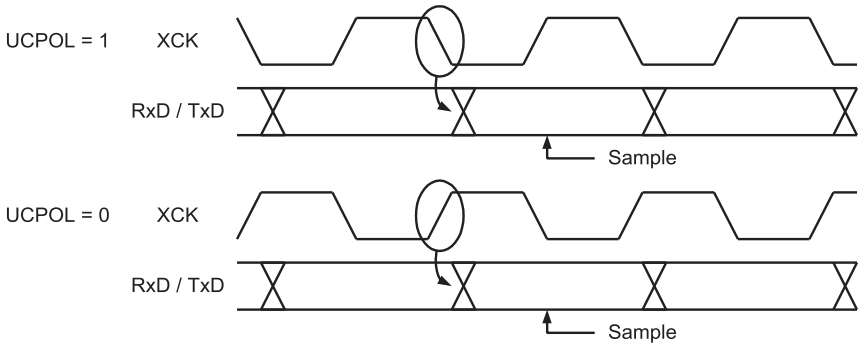
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{OSC} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

23.3.4 Synchronous Clock Operation

When synchronous mode is used (UMSEL*n* = 1), the XCK*n* pin will be used as either clock input (slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD*n*) is sampled at the opposite XCK*n* clock edge of the edge the data output (TxD*n*) is changed.

Figure 23-3. Synchronous Mode XCK*n* Timing



The UCPOL*n* bit UCRSC selects which XCK*n* clock edge is used for data sampling and which is used for data change. As Figure 22-3 on page 335 shows, when UCPOL*n* is zero the data will be changed at rising XCK*n* edge and sampled at falling XCK*n* edge. If UCPOLn is set, the data will be changed at falling XCK*n* edge and sampled at rising XCK*n* edge.

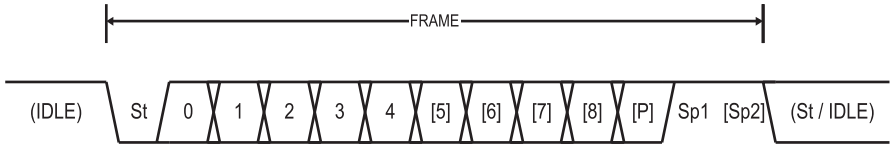
23.4 Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 23-4 below illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 23-4. Frame Formats



St Start bit, always low

(n) Data bits (0 to 8)

P Parity bit - can be odd or even

Sp Stop bit, always high

IDLE No transfers on the communication line (RxD*n* or TxD*n*). An IDLE line must be high

The frame format used by the USART is set by the UCSZ*n*2:0, UPM*n*1:0 and USBS*n* bits in UCSR*n*B and UCSR*n*C. The receiver and transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

The USART Character Size (UCSZ*n*2:0) bits select the number of data bits in the frame. The USART Parity Mode (UPM*n*1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS*n*) bit. The receiver ignores the second stop bit. A frame error will therefore only be detected in cases where the first stop bit is zero.

23.4.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The parity bit is located between the last data bit and first stop bit of a serial frame. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even}	Parity bit using even parity
P_{odd}	Parity bit using odd parity
d_n	Data bit n of the character

23.5 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the transmitter or the receiver depending on the usage. For interrupt driven USART operation, the global interrupt flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC*n* flag can be used to check that the transmitter has completed all transfers, and the RXC flag can be used to check that there are no unread data in the receive buffer. Note that the TXC*n* flag must be cleared before each transmission (before UDR*n* is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

Assembly Code Example[1]

```
USART_Init:
 ; Set baud rate
 out UBRRnH, r17
 out UBRRnL, r16
 ; Enable receiver and transmitter
 ldi r16, (1<<RXENn)|(1<<TXENn)
 out UCSRnB,r16
 ; Set frame format: 8data, 2stop bit
 ldi r16, (1<<USBSn)|(3<<UCSZn0)
 out UCSRnC,r16
 ret
```

C Code Example[1]

```
#define FOSC 8000000// Clock Speed
#define BAUD 9600
#define (MYUBRR FOSC/16/BAUD-1)
void main( void )
{...
USART_Init ( MYUBRR );
...} // main
void USART_Init( unsigned int ubrr){
/* Set baud rate */
UBRRnH = (unsigned char)(ubrr>>8);
UBRRnL = (unsigned char) ubrr;
/* Enable receiver and transmitter */
UCSRnB = (1<<RXEN)|(1<<TXEN);
/* Set frame format: 8data, 2stop bit */
UCSRnC = (1<<USBS)|(3<<UCSZ0);
} // USART_Init
```

Note: 1. See "About Code Examples" on page 7

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

23.6 Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the Transmit Enable (TXEN) bit in the UCSR*n*B register. When the transmitter is enabled, the normal port operation of the TxD*n* pin is overridden by the USART and gives the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK*n* pin will be overridden and used as transmission clock.

23.6.1 Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR*n* I/O location. The buffered data in the transmit buffer will be moved to the shift register when the shift register is ready to send a new frame. The shift register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the shift register is loaded with new data, it will transfer one complete frame at the rate given by the baud rate register, U2Xn bit or by XCK*n* depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty Flag (UDRE*n*). When using frames with less than eight bits, the most significant bits written to the UDR*n* are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register r16.

Assembly Code Example[1]

```
USART_Transmit:
 ; Wait for empty transmit buffer
 sbis UCSRnA,UDREn rjmp USART_Transmit
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn)) );
 /* Put data into buffer, sends the data */
 UDRn = data;
}
```

Note: 1. See "About Code Examples" on page 7

The function simply waits for the transmit buffer to be empty by checking the UDRE*n* flag, before loading it with new data to be transmitted. If the data register empty interrupt is utilized, the interrupt routine writes the data into the buffer.

23.6.2 Sending Frames with 9 Data Bit

If 9 bit characters are used (UCSZ*n*2:0 = 7), the ninth bit must be written to the TXB8 bit in UCSR*n*B before the low byte of the character is written to UDR*n*. The following code examples show a transmit function that handles 9 bit characters. For the assembly code, the data to be sent is assumed to be stored in registers r17:r16.

Assembly Code Example[1][2]

```
USART_Transmit:
 ; Wait for empty transmit buffer
 sbis UCSRnA,UDREn
 rjmp USART_Transmit
 ; Copy 9th bit from r17 to TXB8
 cbi UCSRnB,TXB8
 sbrc r17,0
 sbi UCSRnB,TXB8
 ; Put LSB data (r16) into buffer, sends the data
 out UDRn,r16
 ret
```

C Code Example[1][2]

```
void USART_Transmit( unsigned int data )
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn)) );
 /* Copy 9th bit to TXB8 */
 UCSRnB &= ~(1<<TXB8);
 if ( data & 0x0100 )
   UCSRnB |= (1<<TXB8);
 /* Put data into buffer, sends the data */
 UDRn = data;
}
```

Note:
1. These transmit functions are written to be general functions. They can be optimized if the content of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB register is used after initialization.
2. See "About Code Examples" on page 7

The 9th bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

23.6.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE*n*) and Transmit Complete (TXC*n*). Both flags can be used for generating interrupts.

The Data Register Empty Flag (UDRE*n*) indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always write this bit to zero when writing the UCSR*n*A register.

When the USART Data Register Empty Interrupt Enable (UDRIE*n*) bit in UCSR*n*B is written to one, the USART data register empty interrupt will be executed as long as UDRE*n* is set (provided that global interrupts are enabled). UDRE*n* is cleared by writing UDR*n*. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to UDR*n* in order to clear UDRE*n* or disable the data register empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete Flag (TXC*n*) bit is set one when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXC*n* flag bit is automatically cleared when a transmission complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC*n* flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmission Complete Interrupt Enable (TXCIE*n*) bit in UCSR*n*B is set, the USART transmission complete interrupt will be executed when the TXC*n* flag becomes set (provided that global interrupts are enabled). When the transmission complete interrupt is used, the interrupt handling routine does not have to clear the TXC*n* flag. This is done automatically when the interrupt is executed.

23.6.4 Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM*n*1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

23.6.5 Disabling the Transmitter

The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. The transmitter will no longer override the TxD*n* pin when disabled.

23.7 Data Reception – The USART Receiver

The USART receiver is enabled by writing the Receive Enable (RXEN*n*) bit in the UCSR*n*B register to one. When the receiver is enabled, the normal pin operation of the RxD*n* pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK*n* pin will be used as transfer clock.

23.7.1 Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK*n* clock, and shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, i.e., a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR*n* I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete Flag (RXC*n*). When using frames with less than eight bits the most significant bits of the data read from the UDR*n* will be masked to zero. The USART has to be initialized before the function can be used. The function simply waits for data to be present in the receive buffer by checking the RXC*n* flag before reading the buffer and returning the value.

Assembly Code Example[1]

```
USART_Receive:
 ; Wait for data to be received
 sbis UCSRnA, RXCn
 rjmp USART_Receive
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

C Code Example(1)

```
unsigned char USART_Receive( void )
{
 /* Wait for data to be received */
 while ( !(UCSRnA & (1<<RXCn)) );
 /* Get and return received data from buffer */
 return UDRn;
}
```

Note: 1. See "About Code Examples" on page 7

23.7.2 Receiving Frames with 9 Data Bits

If 9 bit characters are used (UCSZ*n*2:0=7) the 9th bit must be read from the RXB8*n* bit in UCSR*n*B before reading the low bits from the UDR*n* register. This rule applies to the FE*n*, DOR*n* and UPE*n* status flags as well. Read status from UCSR*n*A, then data from UDR*n*. Reading the UDR*n* I/O location will change the state of the receive buffer FIFO and consequently the TXB8*n*, FE*n*, DOR*n* and UPE*n* bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example(1)

```
USART_Receive:
 ; Wait for data to be received
 sbis UCSRnA, RXCn
 rjmp USART_Receive
 ; Get status and 9th bit, then data from buffer
 in r18, UCSRnA
 in r17, UCSRnB
 in r16, UDRn
 ; If error, return -1
 andi r18,(1<<FEn)|(1<<DORn)|(1<<UPEn)
 breq USART_ReceiveNoError
 ldi r17, HIGH(-1)
 ldi r16, LOW(-1)
USART_ReceiveNoError:
 ; Filter the 9th bit, then return
 lsr r17
 andi r17, 0x01
 ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
 unsigned char status, resh, resl;
 /* Wait for data to be received */
 while ( !(UCSRnA & (1<<RXCn)) );
 /* Get status and 9th bit, then data */
 /* from buffer */
 status = UCSRnA;
 resh = UCSRnB;
 resl = UDRn;
 /* If error, return -1 */
 if ( status & (1<<FEn)|(1<<DORn)|(1<<UPEn) )
   return -1;
 /* Filter the 9th bit, then return */
 resh = (resh >> 1) & 0x01;
 return ((resh << 8) | resl);
}
```

Note: 1. See "About Code Examples" on page 7

The receive function example reads all the I/O registers into the register file before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

23.7.3 Receive Complete Flag and Interrupt

The USART receiver has one flag that indicates the receiver state.

The Receive Complete Flag (RXC*n*) indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled (RXEN*n* = 0), the receive buffer will be flushed and consequently the RXC*n* bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE*n*) in UCSR*n*B is set, the USART receive complete interrupt will be executed as long as the RXC*n* flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR*n* in order to clear the RXC*n* flag, otherwise a new interrupt will occur once the interrupt routine terminates.

23.7.4 Receiver Error Flags

The USART receiver has three error flags: Frame Error (FE*n*), Data OverRun (DOR*n*) and Parity Error (UPE*n*). All can be accessed by reading UCSR*n*A. Common for the error flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSR*n*A must be read before the receive buffer (UDR*n*), since reading the UDR*n* I/O location changes the buffer read location. The error flags cannot be altered by the application software doing a write to the flag location. However, all flags must be set to zero when the UCSR*n*A is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The Frame Error Flag (FE*n*) indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE*n* flag is zero when the stop bit was correctly

read (as one), and the FE*n* flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE*n* flag is not affected by the setting of the USBS*n* bit in UCSR*n*C since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSR*n*A.

The Data OverRun Flag (DOR*n*) indicates data loss due to a receiver buffer full condition. A data overrun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. If the DOR*n* flag is set there was one or more serial frame lost between the frame last read from UDR*n*, and the next frame read from UDR*n*. For compatibility with future devices, always write this bit to zero when writing to UCSR*n*A. The DOR*n* flag is cleared when the frame received was successfully moved from the shift register to the receive buffer.

The Parity Error Flag (UPE*n*) indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the UPE*n* bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSR*n*A. For more details see "Parity Bit Calculation" on page 344 and "Parity Checker" below.

23.7.5 Parity Checker

The parity checker is active when the high USART parity mode (UPM*n*1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM*n*0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error Flag (UPE*n*) can then be read by software to check if the frame had a parity error.

The UPE*n* bit is set if the next character that can be read from the receive buffer had a parity error when received .The parity checking was enabled at that point (UPM*n*1 = 1). This bit is valid until the receive buffer (UDR*n*) is read.

23.7.6 Disabling the Receiver

In contrast to the transmitter, disabling of the receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXEN*n* is set to zero) the receiver will no longer override the normal function of the RxD*n* port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost

23.7.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR*n* I/O location until the RXC*n* flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example[1]

```
USART_Flush:
 sbis UCSRnA, RXCn
 ret
 in r16, UDRn
 rjmp USART_Flush
```

C Code Example[1]

```
void USART_Flush( void )
{
 unsigned char dummy;
 while ( UCSRnA & (1<<RXCn) ) dummy = UDRn;
}
```

Note: 1. See "About Code Examples" on page 7

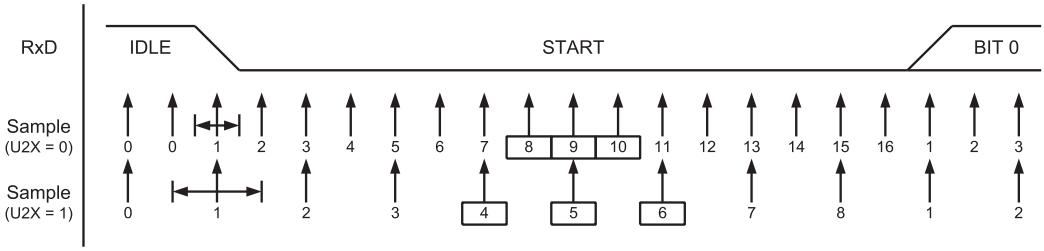
23.8 Asynchronous Data Reception

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

23.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 23-5 below illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and eight times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode (U2X*n* = 1) of operation. Samples denoted zero are samples done when the RxD*n* line is idle (i.e., no communication activity).

Figure 23-5. Start Bit Sampling



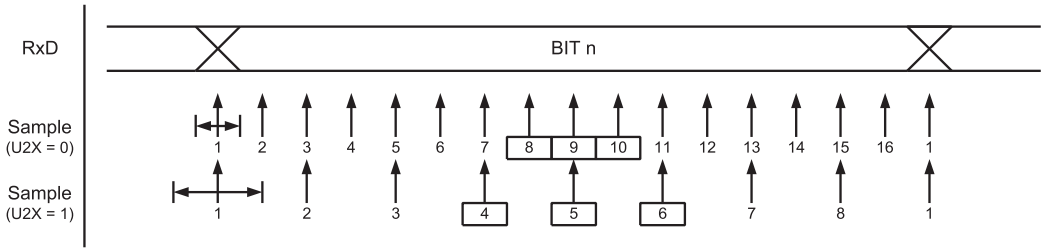
When the clock recovery logic detects a high (idle) to low (start) transition on the RxD*n* line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9 and 10 for Normal mode, and samples 4, 5 and 6 for double speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

23.8.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in double speed mode. Figure 23-6 on page 353

shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Figure 23-6. Sampling of Data and Parity Bit

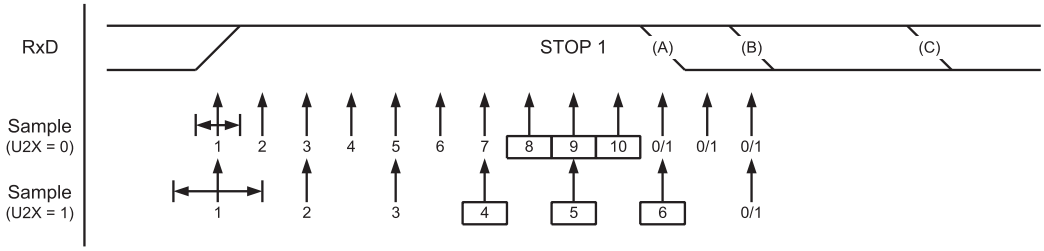


The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the centre of the received bit. The centre samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows:

If two or all three samples have high levels, the received bit is registered to be logic 1. If two or all three samples have low levels, the received bit is registered to be logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD*n* pin. The recovery process is then repeated until a complete frame is received including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 23-7 below shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 23-7. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error Flag (FE*n*) will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For normal speed mode, the first low level sample can be at point marked (A) in Figure 23-7 above. For double speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

23.8.3 Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see Table 23-2 on page 354) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_{MF}}$$

D Sum of character size and parity size (D = 5 to 10 bit)

S Samples per bit. S = 16 for normal speed mode and S = 8 for double speed mode.

S_F First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for double speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for double speed mode.

R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.

R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 23-2 below and Table 23-3 below list the maximum receiver baud rate error that can be tolerated. Note that normal speed mode has higher tolerance of baud rate variations.

Table 23-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2Xn = 0)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 23-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2Xn = 1)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

The recommendations of the maximum receiver baud rate error were made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receiver baud rate error. The receiver's system clock will always have some minor instability over the supply voltage range and the temperature range. When using the radio transceiver crystal oscillator (XOSC) to generate the system clock, this is rarely a problem, but for the internal RC oscillator the system clock may differ more than 2% over the temperature range. The second source for the error is more controllable. The baud rate generator can not always do an exact

division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

23.9 Multi-processor Communication Mode

Setting the Multi-processor Communication Mode (MPCM*n*) bit in UCSR*n*A enables a filtering function of incoming frames received by the USART receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the MCU, in a system with multiple MCUs that communicate via the same serial bus. The transmitter is unaffected by the MPCM*n* setting, but has to be used differently when it is a part of a system utilizing the multi-processor communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8*n*) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The multi-processor communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

23.9.1 Using MPCMn

For an MCU to act as a master MCU, it can use a 9 bit character frame format (UCSZ*n*2:0 = 7). The 9th bit (TXB8*n*) must be set when an address frame (TXB8*n* = 1) or cleared when a data frame (TXB = 0) is being transmitted. The slave MCUs must in this case be set to use a 9 bit character frame format.

The following procedure should be used to exchange data in multi-processor communication mode:

1. All slave MCUs are in multi-processor communication mode (MPCM*n* in UCSR*n*A is set).
2. The master MCU sends an address frame, and all slaves receive and read this frame. In the slave MCUs, the RXC*n* flag in UCSRnA will be set as normal.
3. Each slave MCU reads the UDR*n* register and determines if it has been selected. If so, it clears the MPCM*n* bit in UCSR*n*A, otherwise it waits for the next address byte and keeps the MPCM*n* setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other slave MCUs, which still have the MPCM*n* bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM*n* bit and waits for a new address frame from master. The process then repeats from 2.

Using any of the 5 to 8 bit character frame formats is possible, but impractical since the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver uses the same character size setting. If 5 to 8 bit character frames are used, the transmitter must be set to use two stop bit (USBS*n* = 1) since the first stop bit is used for indicating the frame type.

Do not use read-modify-write instructions (SBI and CBI) to set or clear the MPCM*n* bit. The MPCM*n* bit shares the same I/O location as the TXC*n* flag and this might accidentally be cleared when using SBI or CBI instructions.

23.10 Register Description

23.10.1 UDR0 – USART0 I/O Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($C6)	UDR07:00								UDR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR0. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR0 Register location. Reading the UDR0 Register location will return the contents of the Receive Data Buffer Register (RXB). For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver. The transmit buffer can only be written when the UDRE0 Flag in the UCSR0A Register is set. Data written to UDR0 when the UDRE0 Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD0 pin. The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

- **Bit 7:0 – UDR07:00 - USART I/O Data Register**

23.10.2 UCSR0A – USART0 Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C0)	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/Write	R	RW	R	R	R	R	RW	RW	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC0 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

- **Bit 6 – TXC0 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

- **Bit 5 – UDRE0 - USART Data Register Empty**

The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE0 - Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FE0 bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

- **Bit 3 – DOR0 - Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register and a new start bit is detected. This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 2 – UPE0 - USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 1 – U2X0 - Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation. Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM0 - Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM0 bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM0 setting. For more detailed information see section "Multi-processor Communication Mode".

23.10.3 UCSR0B – USART0 Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C1)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	RW	RW	RW	RW	RW	RW	R	W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCIE0 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

- **Bit 6 – TXCIE0 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

- **Bit 5 – UDRIE0 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

- **Bit 4 – RXEN0 - Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD0 pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE0, DOR0 and UPE0 Flags.

- **Bit 3 – TXEN0 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXEN0 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

- **Bit 2 – UCSZ02 - Character Size**

The UCSZ02 bits combined with the UCSZ01:0 bit in UCSR0C sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

- **Bit 1 – RXB80 - Receive Data Bit 8**

RXB80 is the 9th data bit of the received character when operating with serial frames with nine data bits. The bit must be read before reading the lower 8 bits from UDR0.

- **Bit 0 – TXB80 - Transmit Data Bit 8**

TXB80 is the 9th data bit in the character to be transmitted when operating with serial frames with nine data bits. The bit must be written before writing the lower 8 bits to UDR0.

23.10.4 UCSR0C – USART0 Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($C2)	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	1	0	

- **Bit 7:6 – UMSEL01:00 - USART Mode Select**

These bits select the mode of operation of the USART0 as shown in the following table. See section "USART in SPI Mode" for a full description of the Master SPI Mode (MSPIM) operation.

Table 23-4 UMSEL0 Register Bits

Register Bits	Value	Description
UMSEL01:00	0x00	Asynchronous USART
	0x01	Synchronous USART
	0x02	Reserved
	0x03	Master SPI (MSPIM)

- **Bit 5:4 – UPM01:00 - Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and

compare it to the UPM0 setting. If a mismatch is detected, the UPE0 Flag in UCSR0A will be set.

Table 23-5 UPM0 Register Bits

Register Bits	Value	Description
UPM01:00	0x00	Disabled
	0x01	Reserved
	0x02	Enabled, Even Parity
	0x03	Enabled, Odd Parity

- **Bit 3 – USBS0 - Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 23-6 USBS0 Register Bits

Register Bits	Value	Description
USBS0	0x00	1-bit
	0x01	2-bit

- **Bit 2:1 – UCSZ01:00 - Character Size**

The UCSZ01:0 bits combined with the UCSZ02 bit in UCSR0B sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

Table 23-7 UCSZ0 Register Bits

Register Bits	Value	Description
UCSZ01:00	0	5-bit
	1	6-bit
	2	7-bit
	3	8-bit
	4	Reserved
	5	Reserved
	6	Reserved
	7	9-bit

- **Bit 0 – UCPOL0 - Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL0 bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK0).

Table 23-8 UCPOL0 Register Bits

Register Bits	Value	Description
UCPOL0	0	Rising XCKn Edge (Transmitted Data Changed), Falling XCKn Edge (Received Data Sampled)
	1	Falling XCKn Edge (Transmitted Data Changed), Rising XCKn Edge (Received Data Sampled)

23.10.5 UBRR0H – USART0 Baud Rate Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($C5)	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	UBRR0H
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR0 is a 12-bit register which contains the USART baud rate. The UBRR0H contains the four most significant bits, and the UBRR0L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR0L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:4 – Res3:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3:0 – UBRR11:8 - USART Baud Rate Register**

These bits represent bits [11:8] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.6 UBRR0L – USART0 Baud Rate Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($C4)	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRR0L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR0 is a 12-bit register which contains the USART baud rate. The UBRR0H contains the four most significant bits, and the UBRR0L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR0L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:0 – UBRR7:0 - USART Baud Rate Register**

These bits represent bits [7:0] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.7 UDR1 – USART1 I/O Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($CE)	UDR17:10								UDR1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR1. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR1 Register location. Reading the UDR1 Register location will return the contents of the

Receive Data Buffer Register (RXB). For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver. The transmit buffer can only be written when the UDRE1 Flag in the UCSR1A Register is set. Data written to UDR1 when the UDRE1 Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD1 pin. The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

- **Bit 7:0 – UDR17:10 - USART I/O Data Register**

23.10.8 UCSR1A – USART1 Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C8)	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	UCSR1A
Read/Write	R	RW	R	R	R	R	RW	RW	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC1 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC1 bit will become zero. The RXC1 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE1 bit).

- **Bit 6 – TXC1 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR1). The TXC1 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC1 Flag can generate a Transmit Complete interrupt (see description of the TXCIE1 bit).

- **Bit 5 – UDRE1 - USART Data Register Empty**

The UDRE1 Flag indicates if the transmit buffer (UDR1) is ready to receive new data. If UDRE1 is one, the buffer is empty, and therefore ready to be written. The UDRE1 Flag can generate a Data Register Empty interrupt (see description of the UDRIE1 bit). UDRE1 is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE1 - Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR1) is read. The FE1 bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR1A.

- **Bit 3 – DOR1 - Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register and a new start bit is detected. This bit is valid until the receive buffer (UDR1) is read. Always set this bit to zero when writing to UCSR1A.

- **Bit 2 – UPE1 - USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM11 = 1). This bit is valid until the receive buffer (UDR1) is read. Always set this bit to zero when writing to UCSR1A.

- **Bit 1 – U2X1 - Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation. Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM1 - Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM1 bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM1 setting. For more detailed information see section "Multi-processor Communication Mode".

23.10.9 UCSR1B – USART1 Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C9)	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	UCSR1B
Read/Write	RW	RW	RW	RW	RW	RW	R	W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCIE1 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC1 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC1 bit in UCSR1A is set.

- **Bit 6 – TXCIE1 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC1 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC1 bit in UCSR1A is set.

- **Bit 5 – UDRIE1 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE1 Flag. A Data Register Empty interrupt will be generated only if the UDRIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE1 bit in UCSR1A is set.

- **Bit 4 – RXEN1 - Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD1 pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE1, DOR1 and UPE1 Flags.

- **Bit 3 – TXEN1 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD1 pin when enabled. The disabling of the Transmitter (writing TXEN1 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD1 port.

- **Bit 2 – UCSZ12 - Character Size**

The UCSZ12 bits combined with the UCSZ11:0 bit in UCSR1C sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

- **Bit 1 – RXB81 - Receive Data Bit 8**

RXB81 is the 9th data bit of the received character when operating with serial frames with nine data bits. The bit must be read before reading the lower 8 bits from UDR1.

- **Bit 0 – TXB81 - Transmit Data Bit 8**

TXB81 is the 9th data bit in the character to be transmitted when operating with serial frames with nine data bits. The bit must be written before writing the lower 8 bits to UDR1.

23.10.10 UCSR1C – USART1 Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($CA)	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	UCSR1C
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7:6 – UMSEL11:10 - USART Mode Select**

These bits select the mode of operation of the USART1 as shown in the following table. See section "USART in SPI Mode" for a full description of the Master SPI Mode (MSPIM) operation.

Table 23-9 UMSEL1 Register Bits

Register Bits	Value	Description
UMSEL11:10	0x00	Asynchronous USART
	0x01	Synchronous USART
	0x02	Reserved
	0x03	Master SPI (MSPIM)

- **Bit 5:4 – UPM11:10 - Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM1 setting. If a mismatch is detected, the UPE1 Flag in UCSR1A will be set.

Table 23-10 UPM1 Register Bits

Register Bits	Value	Description
UPM11:10	0x00	Disabled
	0x01	Reserved
	0x02	Enabled, Even Parity
	0x03	Enabled, Odd Parity

- **Bit 3 – USBS1 - Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 23-11 USBS1 Register Bits

Register Bits	Value	Description
USBS1	0x00	1-bit
	0x01	2-bit

- **Bit 2:1 – UCSZ11:10 - Character Size**

The UCSZ11:0 bits combined with the UCSZ12 bit in UCSR1B sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

Table 23-12 UCSZ1 Register Bits

Register Bits	Value	Description
UCSZ11:10	0	5-bit
	1	6-bit
	2	7-bit
	3	8-bit
	4	Reserved
	5	Reserved
	6	Reserved
	7	9-bit

- **Bit 0 – UCPOL1 - Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL1 bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK1).

Table 23-13 UCPOL1 Register Bits

Register Bits	Value	Description
UCPOL1	0	Rising XCKn Edge (Transmitted Data Changed), Falling XCKn Edge (Received Data Sampled)
	1	Falling XCKn Edge (Transmitted Data Changed), Rising XCKn Edge (Received Data Sampled)

23.10.11 UBRR1H – USART1 Baud Rate Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($CD)	**Res3**	**Res2**	**Res1**	**Res0**	**UBRR11**	**UBRR10**	**UBRR9**	**UBRR8**	**UBRR1H**
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR1 is a 12-bit register which contains the USART baud rate. The UBRR1H contains the four most significant bits, and the UBRR1L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR1L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:4 – Res3:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3:0 – UBRR11:8 - USART Baud Rate Register**

These bits represent bits [11:8] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.12 UBRR1L – USART1 Baud Rate Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($CC)	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRR1L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR1 is a 12-bit register which contains the USART baud rate. The UBRR1H contains the four most significant bits, and the UBRR1L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR1L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:0 – UBRR7:0 - USART Baud Rate Register**

These bits represent bits [7:0] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.11 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 23-14 below to Table 23-16 on page 367. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 353). The error values are calculated using the following equation:

$$Error[\%] = \left(\frac{BaudRate_{\,Closest\ Match}}{BaudRate} - 1 \right) \cdot 100\%$$

Table 23-14. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	fosc = 1.8432 MHz				fosc = 2.0000 MHz				fosc = 3.6864 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
4800	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
9600	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
14.4k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
19.2k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
28.8k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
38.4k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%

Baud Rate (bps)	fosc = 1.8432 MHz				fosc = 2.0000 MHz				fosc = 3.6864 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
57.6k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
76.8k	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%	2	0.0%	5	0.0%
115.2k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
230.4k	-	-	0	0.0%	-	-	-	-	0	0.0%	1	0.0%
250k	-	-	-	-	-	-	0	0.0%%	0	-7.8%	1	-7.8%
Max. (1)	115.2 kbps		230.4 kbps		125 kbps		250 kbps		230.4 kbps		460.8 kbps	

Notes: 1. UBRR = 0, Error = 0.0%

Table 23-15. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	fosc = 4.0000 MHz				fosc = 7.3728 MHz				fosc = 8.0000 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	103	0.2%	207	0.2%	191	0.0%	383	0.0%	207	0.2%	416	-0.1%
4800	51	0.2%	103	0.2%	95	0.0%	191	0.0%	103	0.2%	207	0.2%
9600	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
14.4k	16	2.1%	34	-0.8%	31	0.0%	63	0.0%	34	-0.8%	68	0.6%
19.2k	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
28.8k	8	-3.5%	16	2.1%	15	0.0%	31	0.0%	16	2.1%	34	-0.8%
38.4k	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
57.6k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
76.8k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
115.2k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
230.4k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
250k	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%	1	0.0%	3	0.0%
0.5M	-	-	0	0.0%	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%
1M	-	-	-	-	-	-	0	-7.8%	-	-	0	0.0%
Max. (1)	250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps		0.5 Mbps		1 Mbps	

Notes: 1. UBRR = 0, Error = 0.0%

Table 23-16. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	fosc = 11.0592 MHz				fosc = 14.7456 MHz				fosc = 16.0000 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	287	0.0%	575	0.0%	383	0.0%	767	0.0%	416	-0.1%	832	0.0%
4800	143	0.0%	287	0.0%	191	0.0%	383	0.0%	207	0.2%	416	-0.1%
9600	71	0.0%	143	0.0%	95	0.0%	191	0.0%	103	0.2%	207	0.2%
14.4k	47	0.0%	95	0.0%	63	0.0%	127	0.0%	68	0.6%	138	-0.1%
19.2k	35	0.0%	71	0.0%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
28.8k	23	0.0%	47	0.0%	31	0.0%	63	0.0%	34	-0.8%	68	0.6%
38.4k	17	0.0%	35	0.0%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
57.6k	11	0.0%	23	0.0%	15	0.0%	31	0.0%	16	2.1%	34	-0.8%
76.8k	8	0.0%	17	0.0%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
115.2k	5	0.0%	11	0.0%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
230.4k	2	0.0%	5	0.0%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
250k	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%	3	0.0%	7	0.0%
0.5M	-	-	2	-7.8%	1	-7.8%	3	-7.8%	1	0.0%	3	0.0%
1M	-	-	-	-	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%
Max. (1)	691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps		1 Mbps		2 Mbps	

Notes: 1. UBRR = 0, Error = 0.0%

24 USART in SPI Mode

The Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation. The Master SPI Mode (MSPIM) has the following features:

- **Full duplex, three-wire synchronous data transfer**
- **Master operation**
- **Supports all four SPI modes of operation (mode 0, 1, 2, and 3)**
- **LSB first or MSB first data transfer (configurable data order)**
- **Queued operation (double buffered)**
- **High resolution baud rate generator**
- **High speed operation ($f_{XCK,MAX} = f_{CK}/2$)**
- **Flexible interrupt generation**

24.1 Overview

Setting both UMSELn1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

24.2 USART MSPIM vs. SPI

The ATmega128RFA1 USART in MSPIM mode is fully compatible with the ATmega128RFA1 SPI regarding:

- Master mode timing diagram.
- The UCPOLn bit functionality is identical to the SPI CPOL bit.
- The UCPHAn bit functionality is identical to the SPI CPHA bit.
- The UDORDn bit functionality is identical to the SPI DORD bit.

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer.
- The USART in MSPIM mode receiver includes an additional buffer level.
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode.
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly.
- Interrupt timing is not compatible.
- Pin control differs due to the master only operation of the USART in MSPIM mode.

A comparison of the USART in MSPIM mode and the SPI pins is shown in Table 24–3 on page 373.

24.2.1 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRRn setting can therefore be calculated using the same equations, see Table 24-1 below:

Table 24-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Synchronous Master mode	$BAUD = \frac{f_{osc}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{2\,BAUD} - 1$

Note: The Baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System Oscillator clock frequency

UBRRn Contents of the UBRRHn and UBRRLn Registers, (0-4095)

24.3 SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAn and UCPOLn. The data transfer timing diagrams are shown in Figure 24-1 below. Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAn functionality is summarized in Table 24-2 below. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

Figure 24-1. UCPHAn and UCPOLn data transfer timing diagrams

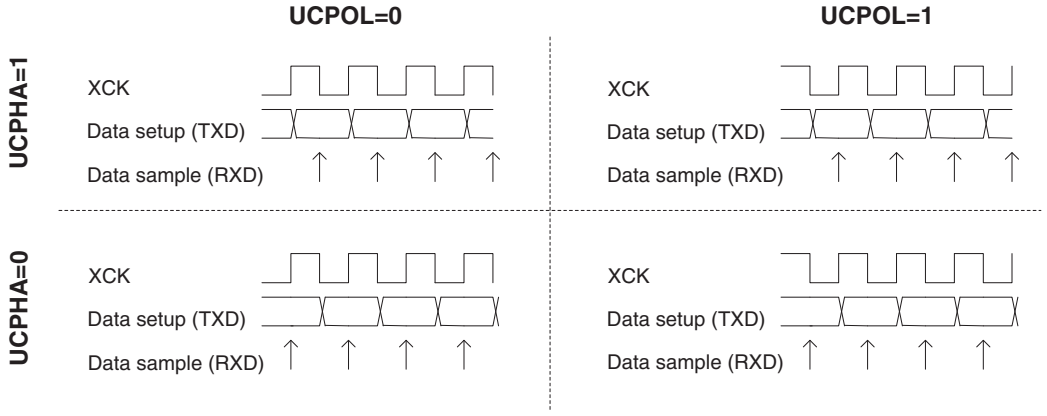


Table 24-2. UCPOLn and UCPHAn Functionality

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
1	1	3	Setup (Falling)	Sample (Rising)

24.4 Frame Formats

A serial frame for the MSPIM is defined to be one character of 8 data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

24.4.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXCn Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example[1]

```
USART_Init:
 clr r18
 out UBRRnH,r18
 out UBRRnL,r18
 ; Setting the XCKn port pin as output, enables master mode.
 sbi XCKn_DDR, XCKn
 ; Set MSPI mode of operation and SPI data mode 0.
 ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
 out UCSRnC,r18
 ; Enable receiver and transmitter.
 ldi r18, (1<<RXENn)|(1<<TXENn)
 out UCSRnB,r18 ; Set baud rate.
 ; IMPORTANT:
 ; The Baud Rate must be set after the transmitter is enabled!
 out UBRRnH, r17
 out UBRRnL, r18
 ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  UBRRn = 0;
  /* Setting the XCKn port pin as output, enables master mode. */
  XCKn_DDR |= (1<<XCKn);
  /* Set MSPI mode of operation and SPI data mode 0. */
  UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
  /* Enable receiver and transmitter. */
  UCSRnB = (1<<RXENn)|(1<<TXENn);
  /* Set baud rate. */
  /* IMPORTANT:     */
  /* The Baud Rate must be set after the transmitter is enabled */
  UBRRn = baud;
 }
```

Note: 1. See "About Code Examples" on page 7

24.5 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to

UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREn) Flag and the Receive Complete (RXCn) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register r16 and the data received will be available in the same register (r16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

Assembly Code Example[1]

```
USART_MSPIM_Transfer:
 ; Wait for empty transmit buffer
 sbis UCSRnA, UDREn
 rjmp USART_MSPIM_Transfer
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
 sbis UCSRnA, RXCn
 rjmp USART_MSPIM_Wait_RXCn
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRnA & (1<<UDREn)) );
  /* Put data into buffer, sends the data */
  UDRn = data;
  /* Wait for data to be received */
  while ( !(UCSRnA & (1<<RXCn)) );
  /* Get and return received data from buffer */
  return UDRn;
}
```

Notes: 1. See "About Code Examples" on page 7

24.5.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and are always read as zero.

24.5.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

24.6 USART MSPIM Register Description

The following section describes the registers used for SPI operation using the USART.

24.6.1 UDRn – USART MSPIM I/O Data Register

The function and bit description of the USART data register (UDR*n*) in MSPI mode is identical to normal USART operation. See "UDR0 – USART0 I/O Data Register" on page 356.

24.6.2 UBRRnL and UBRRnH – USART MSPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode is identical to normal USART operation. See "UBRR0L – USART0 Baud Rate Register Low Byte" on page 360 and "UBRR0H – USART0 Baud Rate Register High Byte" on page 360.

Table 24–3. Comparison of USART in MSPIM mode and SPI pins

USART_MSPIM	SPI	Comment
TxDn	MOSI	Master Out only
RxDn	MISO	Master In only
XCKn	SCK	(Functional identical)
(N/A)	SS	Not supported by USART in MSPIM

24.6.3 UCSR0A – USART0 MSPIM Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C0)	RXC0	TXC0	UDRE0						UCSR0A
Read/Write	R	RW	R						
Initial Value	0	0	0						

- **Bit 7 – RXC0 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

- **Bit 6 – TXC0 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

- **Bit 5 – UDRE0 - USART Data Register Empty**

The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

24.6.4 UCSR0B – USART0 MSPIM Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C1)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0				UCSR0B
Read/Write	RW	RW	RW	RW	RW				
Initial Value	0	0	1	0	0				

- **Bit 7 – RXCIE0 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

- **Bit 6 – TXCIE0 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

- **Bit 5 – UDRIE0 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

- **Bit 4 – RXEN0 - Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxD0 pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXEN0=1 and TXEN0=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXEN0 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXEN0 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

24.6.5 UCSR0C – USART0 MSPIM Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($C2)						UDORD0	UCPHA0	UCPOL0	UCSR0C
Read/Write						RW	RW	RW	
Initial Value						1	1	0	

- **Bit 2 – UDORD0 - Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to section "Frame Formats" for details.

- **Bit 1 – UCPHA0 - Clock Phase**

The UCPHA0 bit setting determines if data is sampled on the leading (first) or tailing (last) edge of XCK0. Refer to the section "SPI Data Modes and Timing" for details.

- **Bit 0 – UCPOL0 - Clock Polarity**

The UCPOL0 bit sets the polarity of the XCK0 clock. The combination of the UCPOL0 and UCPHA0 bit settings determine the timing of the data transfer. Refer to the section "SPI Data Modes and Timing" for details.

24.6.6 UCSR1A – USART1 MSPIM Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C8)	RXC1	TXC1	UDRE1						UCSR1A
Read/Write	R	RW	R						
Initial Value	0	0	0						

- **Bit 7 – RXC1 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC1 bit will become zero. The RXC1 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE1 bit).

- **Bit 6 – TXC1 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR1). The TXC1 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC1 Flag can generate a Transmit Complete interrupt (see description of the TXCIE1 bit).

- **Bit 5 – UDRE1 - USART Data Register Empty**

The UDRE1 Flag indicates if the transmit buffer (UDR1) is ready to receive new data. If UDRE1 is one, the buffer is empty, and therefore ready to be written. The UDRE1 Flag can generate a Data Register Empty interrupt (see description of the UDRIE1 bit). UDRE1 is set after a reset to indicate that the Transmitter is ready.

24.6.7 UCSR1B – USART1 MSPIM Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C9)	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1				UCSR1B
Read/Write	RW	RW	RW	RW	RW				
Initial Value	0	0	1	0	0				

- **Bit 7 – RXCIE1 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC1 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC1 bit in UCSR1A is set.

- **Bit 6 – TXCIE1 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC1 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC1 bit in UCSR1A is set.

- **Bit 5 – UDRIE1 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE1 Flag. A Data Register Empty interrupt will be generated only if the UDRIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE1 bit in UCSR1A is set.

- **Bit 4 – RXEN1 - Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxD1 pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXEN1=1 and TXEN1=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXEN1 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD1 pin when enabled. The disabling of the Transmitter (writing TXEN1 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD1 port.

24.6.8 UCSR1C – USART1 MSPIM Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($CA)						UDORD1	UCPHA1	UCPOL1	UCSR1C
Read/Write						RW	RW	RW	
Initial Value						1	1	0	

- **Bit 2 – UDORD1 - Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to section "Frame Formats" for details.

- **Bit 1 – UCPHA1 - Clock Phase**

The UCPHA1 bit setting determines if data is sampled on the leading (first) or tailing (last) edge of XCK1. Refer to the section "SPI Data Modes and Timing" for details.

- **Bit 0 – UCPOL1 - Clock Polarity**

The UCPOL1 bit sets the polarity of the XCK1 clock. The combination of the UCPOL1 and UCPHA1 bit settings determine the timing of the data transfer. Refer to the section "SPI Data Modes and Timing" for details.

25 2-wire Serial Interface

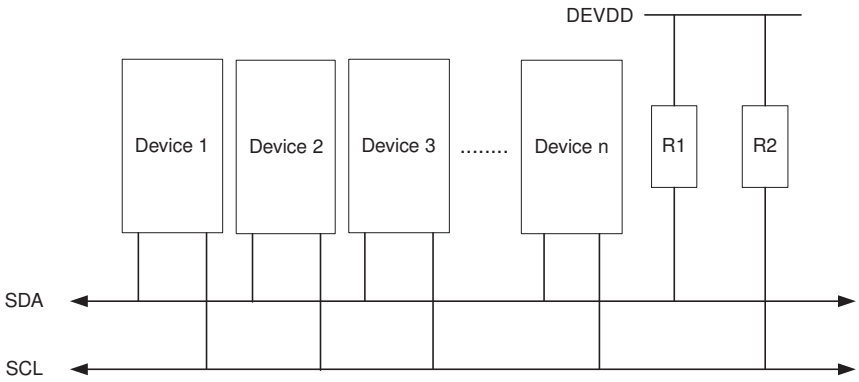
25.1 Features

- **Simple yet powerful and flexible communication interface, only two bus lines needed**
- **Both master and slave operation supported**
- **Device can operate as transmitter or receiver**
- **7-bit address space allows up to 128 different slave addresses**
- **Multi-master arbitration support**
- **Up to 400 kHz data transfer speed**
- **Slew-rate limited output drivers**
- **Noise suppression circuitry rejects spikes on bus lines**
- **Fully programmable slave address with general call support**
- **Address recognition causes wake-up when microcontroller is in sleep mode**

25.2 2-wire Serial Interface Bus Definition

The 2-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 25-1. TWI Bus Interconnection



25.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 25-1. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The Power Reduction TWI bit, PRTWI bit in "PRR0 – Power Reduction Register0" on page 167 must be written to zero to enable the 2-wire Serial Interface.

25.2.2 Electrical Interconnection

As depicted in Figure 25-1 on page 377, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices trim-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

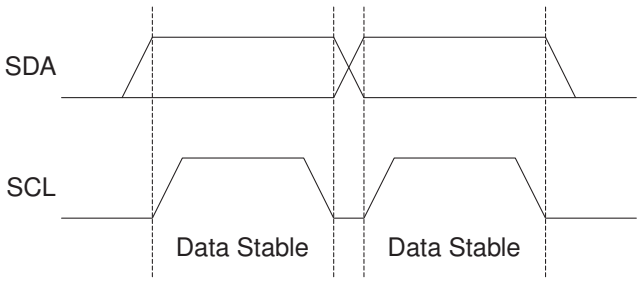
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400 pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in "2-wire Serial Interface Characteristics" on page 503. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

25.3 Data Transfer and Frame Format

25.3.1 Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

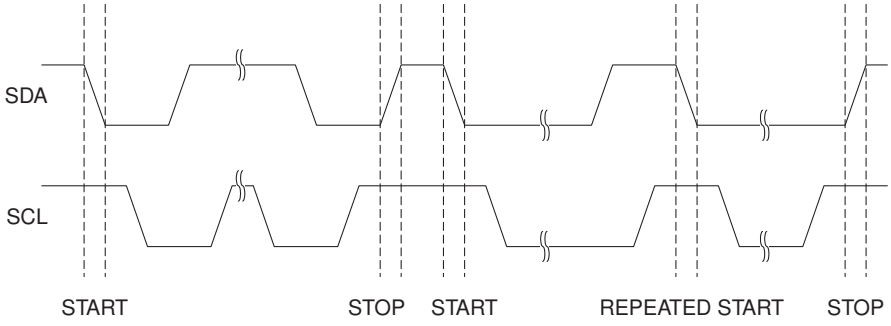
Figure 25-2. Data Validity



25.3.2 START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without relinquishing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.

Figure 25-3. START, REPEATED START and STOP conditions



25.3.3 Address Packet Format

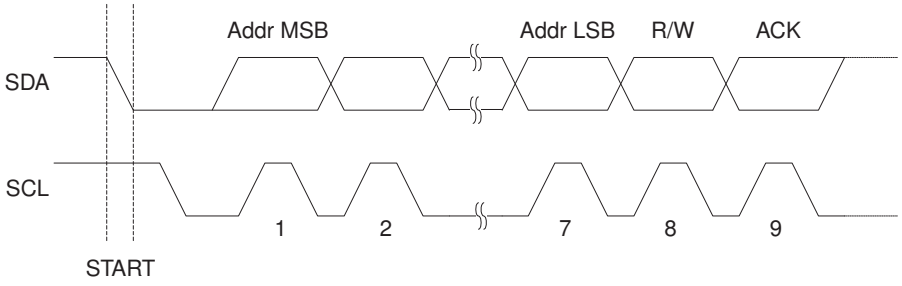
All address packets transmitted on the TWI bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 25-4. Address Packet Format

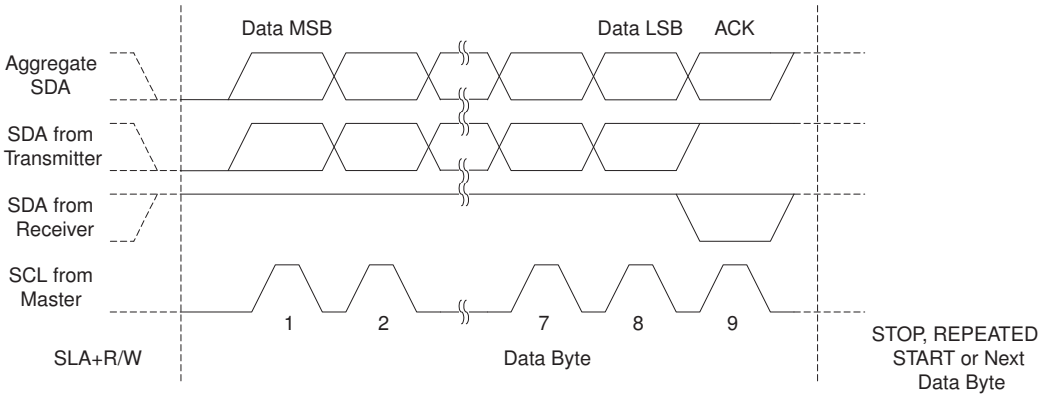


25.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signaled by the Receiver

pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signaled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 25-5. Data Packet Format

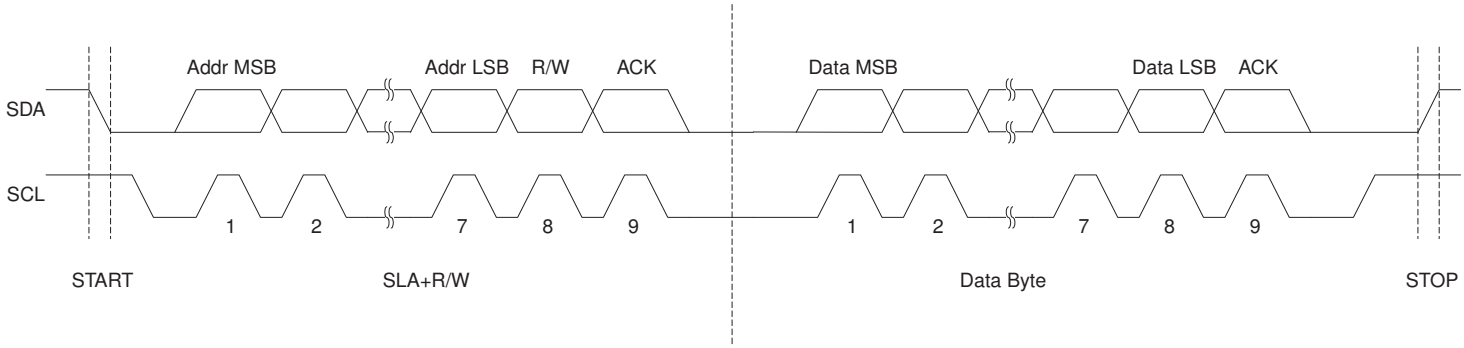


25.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the Wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 25-6 below shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 25-6. Typical Data Transmission



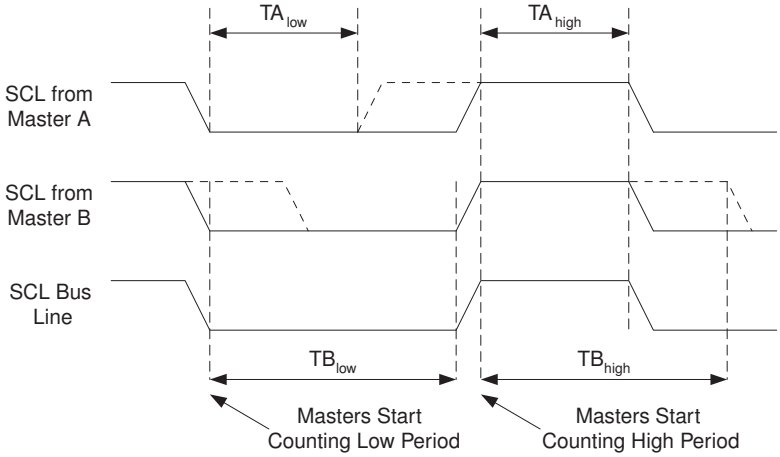
25.4 Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 25-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

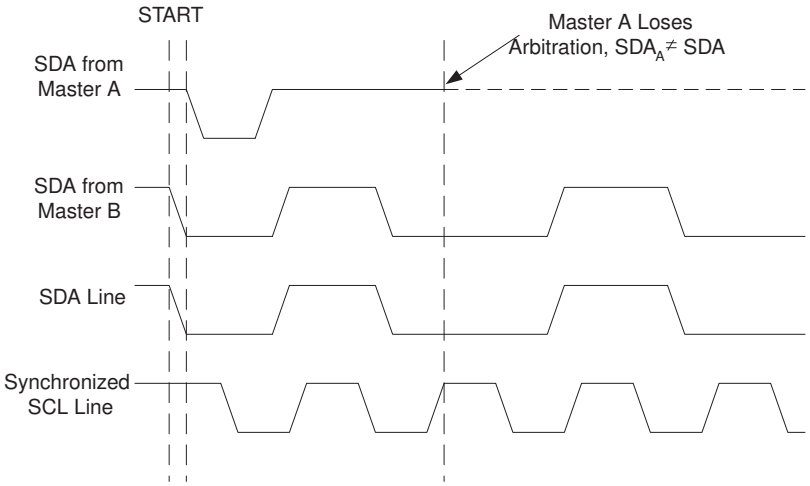
Note that arbitration is not allowed between:

- A REPEATED START condition and a data bit.
- A STOP condition and a data bit.
- A REPEATED START and a STOP condition.

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions

must contain the same number of data packets, otherwise the result of the arbitration is undefined.

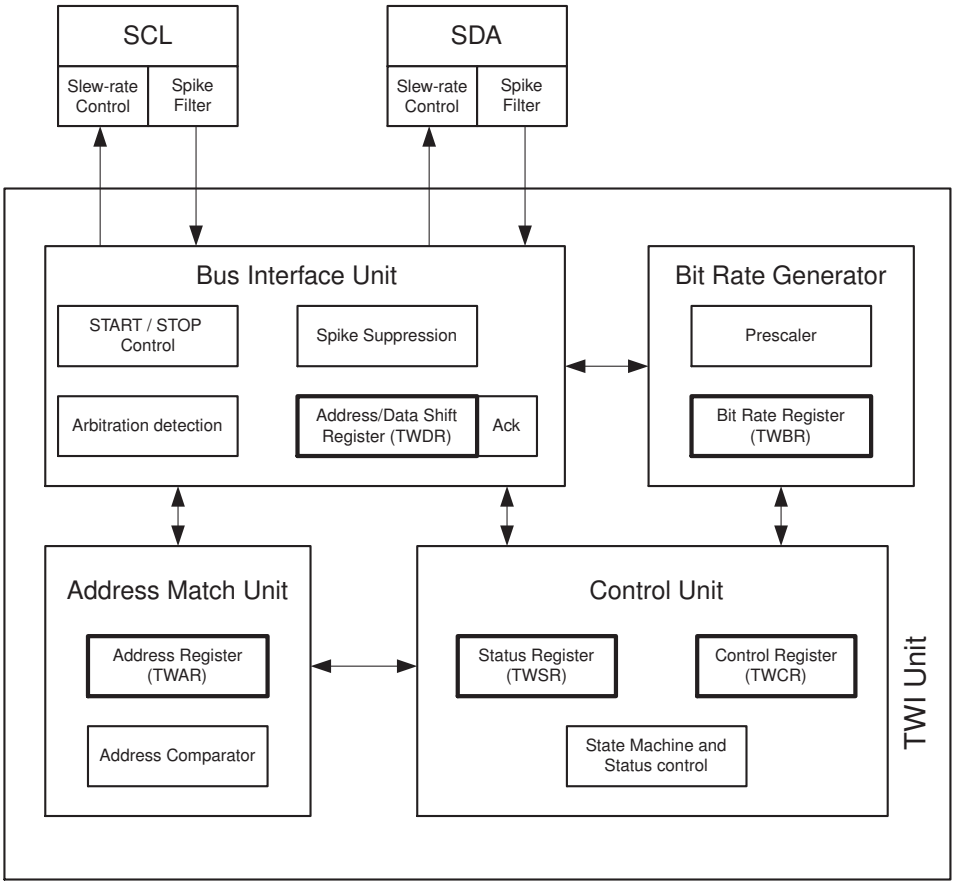
Figure 25-8. Arbitration Between Two Masters



25.5 Overview of the TWI Module

The TWI module is comprised of several sub-modules, as shown in Figure 25-9 below. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 25-9. Overview of the TWI Module



25.5.1 SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

25.5.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$SCL\ frequency = \frac{CPU\ Clock\ frequency}{16 + 2(TWBR) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register.
- TWPS = Value of the prescaler bits in the TWI Status Register.

Note that pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See in "2-wire Serial Interface Characteristics" on page 503 for value of pull-up resistor.

25.5.3 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

25.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to

compare addresses even if the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master. If another interrupt (e.g., INT0) occurs during TWI Power-down address match and wakes up the CPU, the TWI aborts operation and return to it's idle state. If this cause any problems, ensure that TWI Address Match is the only enabled interrupt when entering Power-down.

25.5.5 Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition.
- After the TWI has transmitted SLA+R/W.
- After the TWI has transmitted an address byte.
- After the TWI has lost arbitration.
- After the TWI has been addressed by own slave address or general call.
- After the TWI has received a data byte.
- After a STOP or REPEATED START has been received while still addressed as a Slave.
- When a bus error has occurred due to an illegal START or STOP condition.

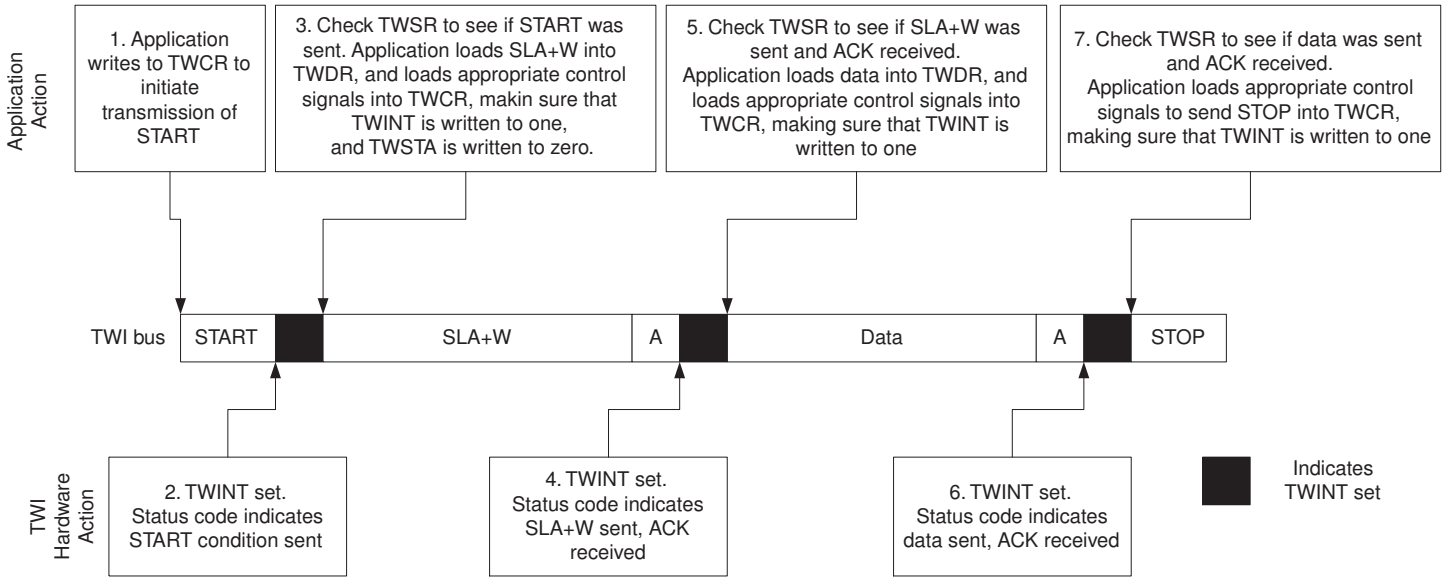
25.6 Using the TWI

The ATmega128RFA1 TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 25-10 on page 385 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 25-10. Interfacing the Application to the TWI in a Typical Transmission



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.
4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT

clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.

6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.

Table 25-2. Code example

	Assembly Code Example	C Example	Comments
1	`ldi r16,(1<<TWINT)|(1<<TWSTA)| (1<<TWEN)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)`	Send START condition

2	`wait1:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait1`
		`while (!(TWCR & (1<<TWINT)));`	Wait for TWINT Flag set. This indicates that the START condition has been transmitted

3	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, START`
	`brne ERROR`
		`if ((TWSR & 0xF8) != START)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR

	`ldi r16, SLA_W`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT)|(1<<TWEN)`
	`out TWCR, r16`
		`TWDR = SLA_W;`
		`TWCR = (1<<TWINT)|(1<<TWEN);`
			Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address

	Assembly Code Example	C Example	Comments
4	wait2: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait2	while (!(TWCR & (1<<TWINT))); 	Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.
5	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_SLA_ACK **brne** ERROR	if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR(); 	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
	ldi r16, DATA **out** TWDR, r16 **ldi** r16, (1<<TWINT)|(1<<TWEN) **out** TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN); 	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data
6	wait3: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait3	while (!(TWCR & (1<<TWINT))); 	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
7	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_DATA_ACK **brne** ERROR	if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR(); 	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
	ldi r16,(1<<TWINT)|(1<<TWEN)| (1<<TWSTO) **out** TWCR, r16	TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO); 	Transmit STOP condition

25.7 Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S:	START condition	**Rs:**	REPEATED START condition
R:	Read bit (high level at SDA)	**W:**	Write bit (low level at SDA)
Data:	8-bit data byte	**P:**	STOP condition
SLA:	Slave Address	**A:**	Acknowledge bit (low level at SDA)
$\overline{A}$:	Not acknowledge bit (high level at SDA)		

In Figure 25-12 on page 389 to Figure 25-18 on page 399 circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

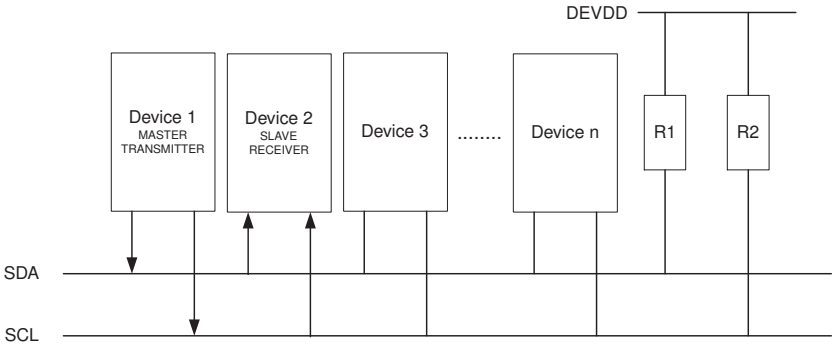
When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and

details of the following serial transfer are given in Table 25-3 on page 390 to Table 25-6 on page 398. Note that the prescaler bits are masked to zero in these tables.

25.7.1 Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 25-11 below). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-11. Data Transfer in Master Transmitter Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

TWEN must be set to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 25-3 on page 390). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in Table 25-3 on page 390.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	X	1	0	X	1	0	X	

After a REPEATED START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Figure 25-12. Formats and States in the Master Transmitter Mode

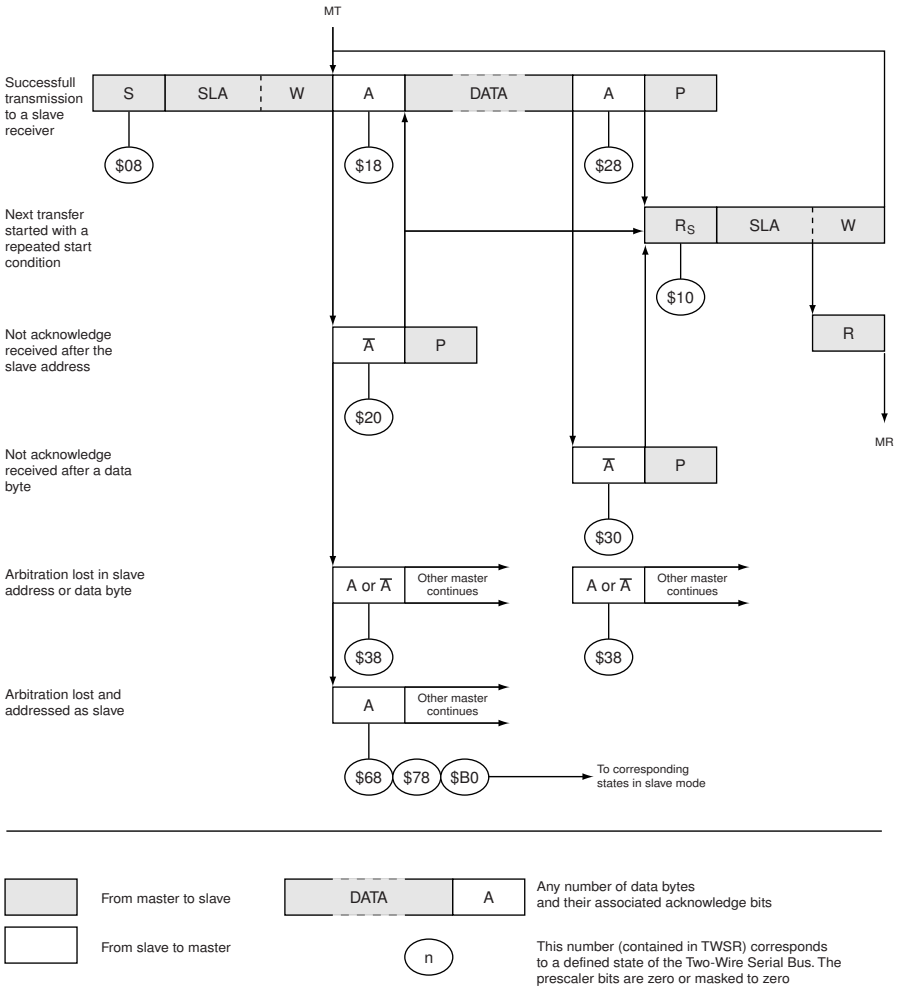


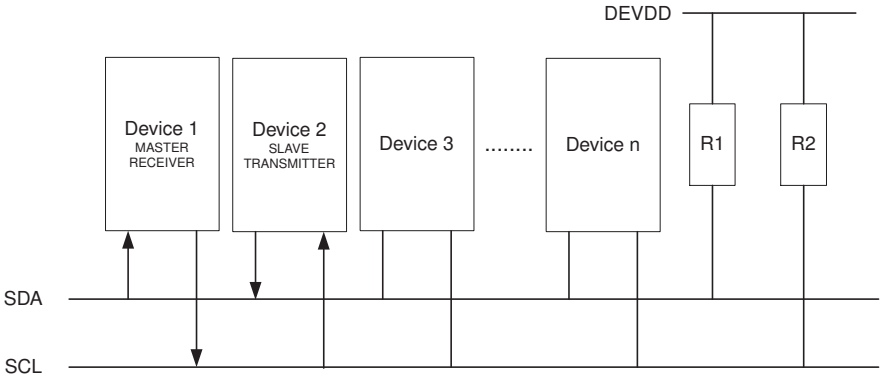
Table 25-3. Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte o	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be rese
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus be-comes free

25.7.2 Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (for Slave see Figure 25-13 on page 391). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-13. Data Transfer in Master Receiver Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

TWEN must be written to one to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 25-4 on page 392). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in Table 25-4 on page 392. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

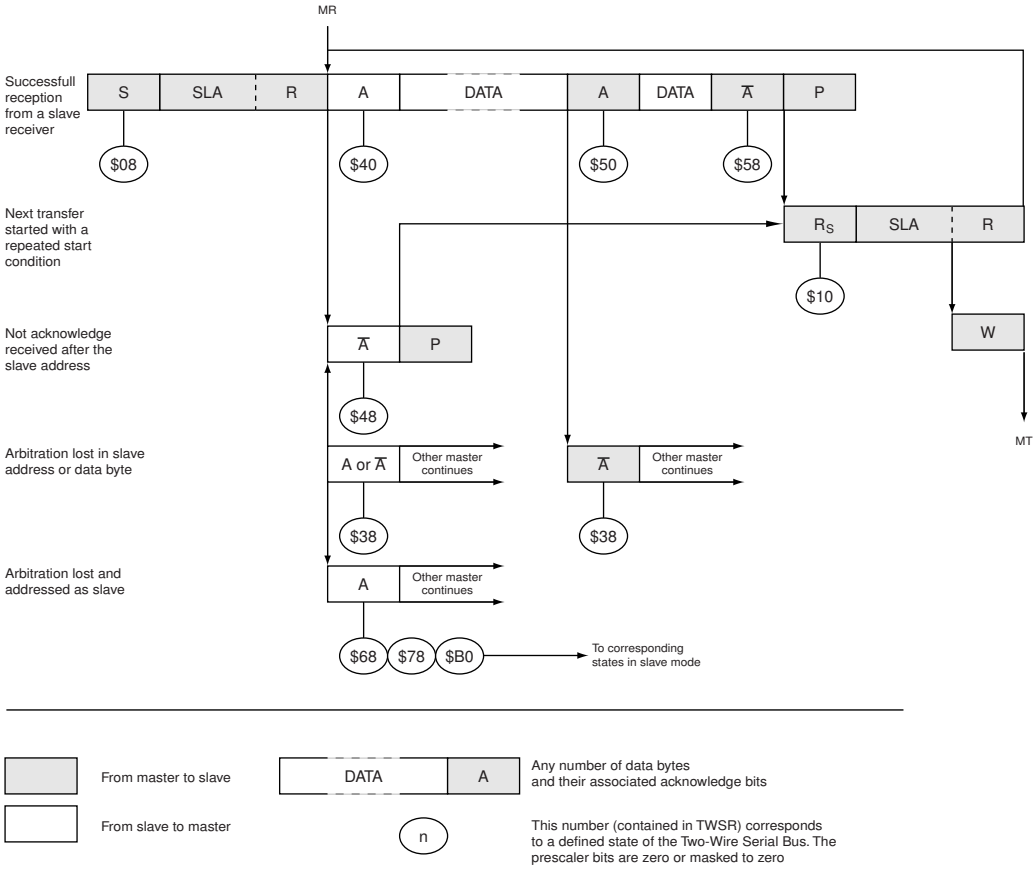
After a repeated START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated

START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 25-4. Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hard-ware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STD	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOTACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

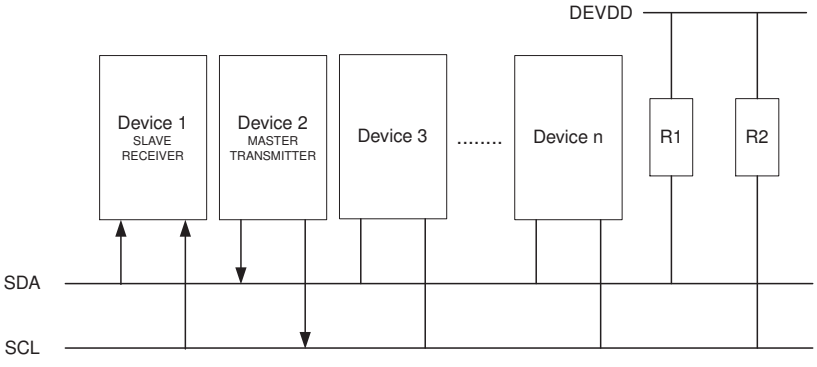
Figure 25-14. Formats and States in the Master Receiver Mode



25.7.3 Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 25-15 below). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-15. Data transfer in Slave Receiver mode



To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 25-5 below. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these Sleep modes.

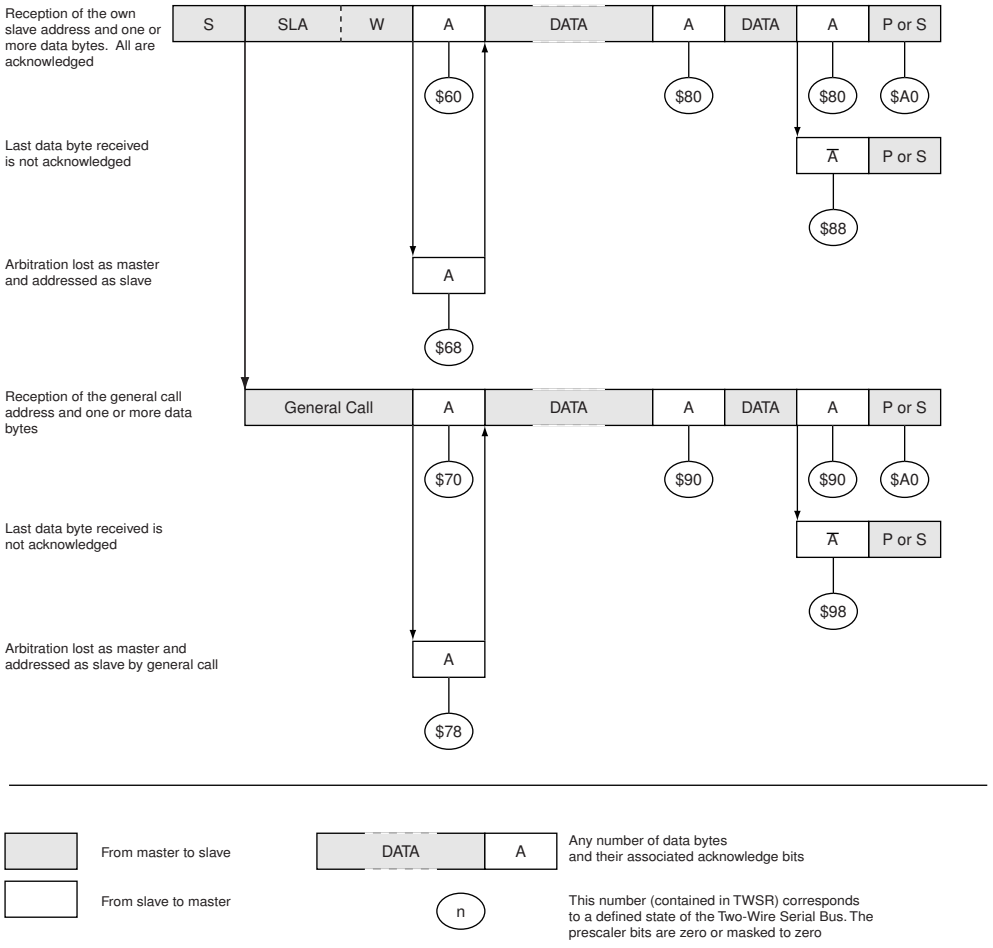
Table 25-5. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned

0x78	Arbitration lost in SLA+R/W as Master;	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
	General call address has been received; ACK has been returned	No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
	has been received; ACK has been returned	Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	has been received; NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0x90	Previously addressed with general call; data	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
	has been re-ceived; ACK has been returned	Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
	has been received; NOT ACK has been returned	Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

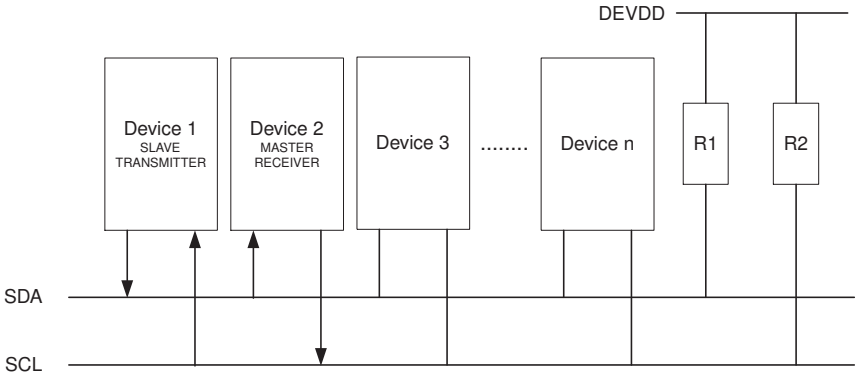
Figure 25-16. Formats and States in the Slave Receiver Mode



25.7.4 Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 25-17 on page 397). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-17. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

The upper seven bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 25-6 on page 398. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part

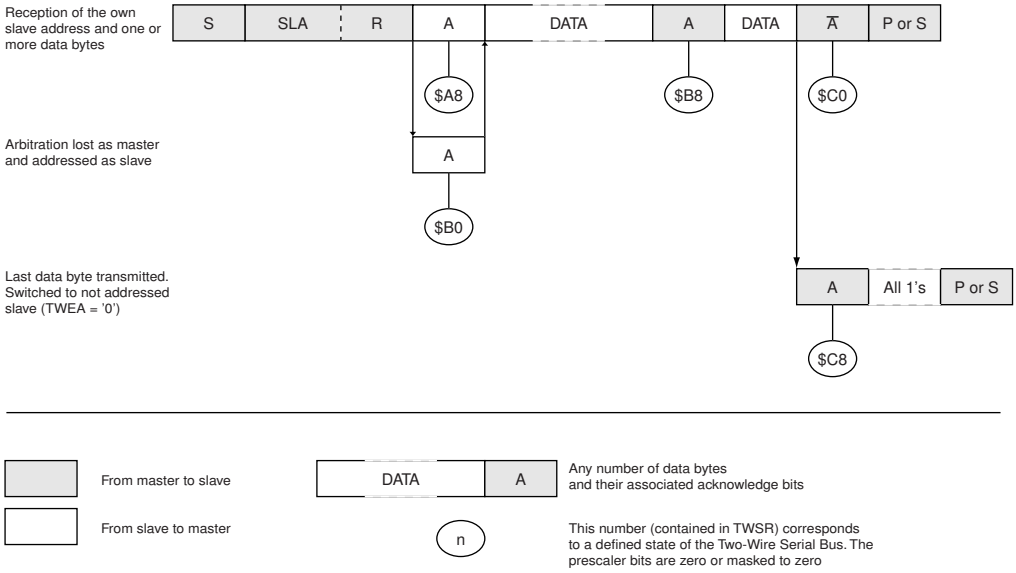
will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

Table 25-6. Status Code for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STD	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xC8	Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 25-18. Formats and States in the Slave Transmitter Mode



25.7.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 25-7 below.

Status 0xF8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status 0x00 indicates that a bus error has occurred during a 2-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 25-7. Miscellaneous States

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hard-ware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xF8	No relevant state information available	TWDR action	No TWCR action				Wait or proceed current transfer
0x00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condi-tion is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

25.7.6 Combining Several TWI Modes

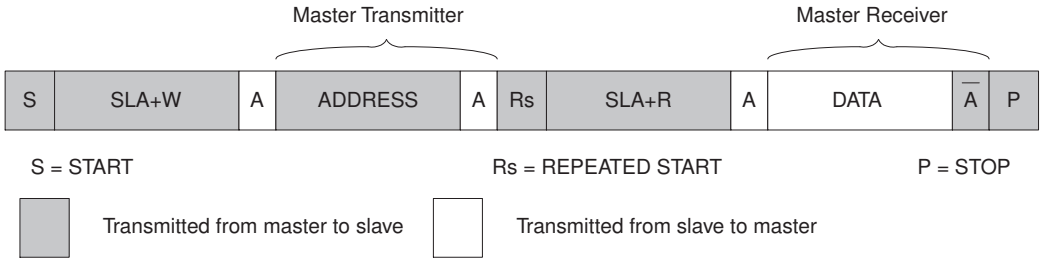
In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.

3. The reading must be performed.
4. The transfer must be finished.

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomic operation. If this principle is violated in a multi-master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

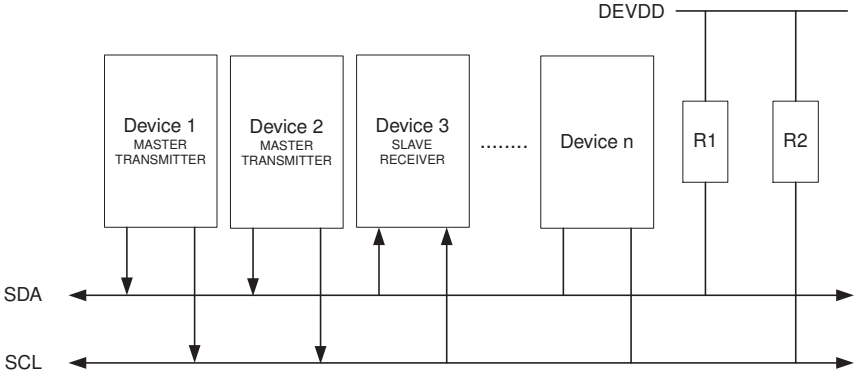
Figure 25-19. Combining Several TWI Modes to Access a Serial EEPROM



25.8 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 25-20. An Arbitration Example



Several different scenarios may arise during arbitration, as described below:

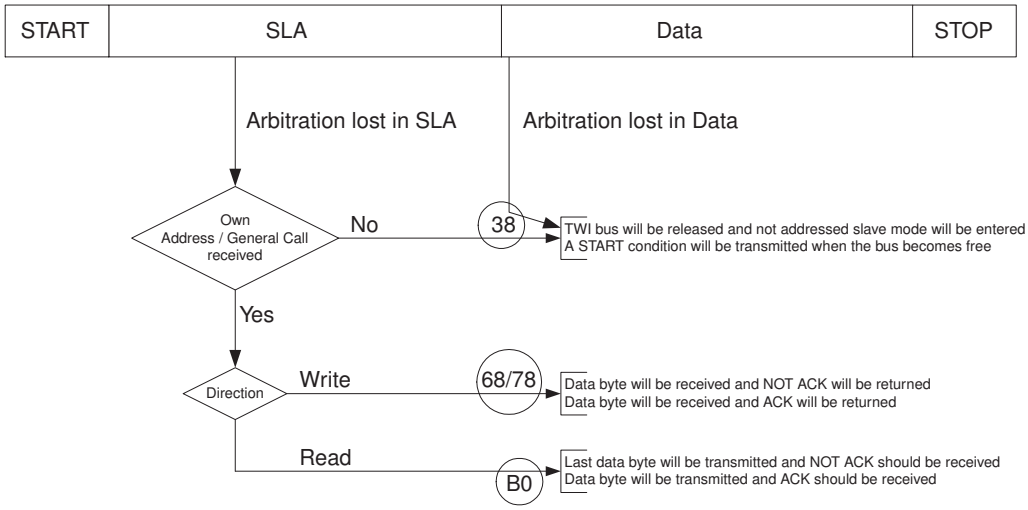
- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention.
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a one on SDA while another Master outputs a zero

will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

- Two or more masters are accessing different slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in Figure 25-21 below. Possible status values are given in circles.

Figure 25-21. Possible Status Codes Caused by Arbitration



25.9 Register Description

25.9.1 TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
NA ($B8)	TWBR7:0								TWBR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency.

- **Bit 7:0 – TWBR7:0 - TWI Bit Rate Register Value**

The TWBR register selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See section "Bit Rate Generator Unit" for calculating bit rates.

25.9.2 TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($BC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	Res	TWIE	TWCR
Read/Write	RW	RW	RW	RW	RW	RW	R	RW	
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are put into the TWDR. It also indicates a write collision if data writing to TWDR is attempted while the register is inaccessible.

- **Bit 7 – TWINT - TWI Interrupt Flag**

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI. So all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR) and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 6 – TWEA - TWI Enable Acknowledge Bit**

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met: 1. The devices own slave address has been received; 2. A general call has been received, while the TWGCE bit in the TWAR is set. 3. A data byte has been received in Master Receiver or Slave Receiver mode. By writing the TWEA bit to zero, the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

- **Bit 5 – TWSTA - TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a Master on the 2-wire Serial Bus. The TWI hardware checks if the bus is available and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO - TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined not-addressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC - TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN - TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O ports connected to the SCL and SDA

pins enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated regardless of any ongoing operation.

- **Bit 1 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 0 – TWIE - TWI Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

25.9.3 TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($B9)	TWS7	TWS6	TWS5	TWS4	TWS3	Res	TWPS1	TWPS0	TWSR
Read/Write	RW	RW	RW	RW	RW	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – TWS4:0 - TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes for both transmitter and receiver mode are described in the following table. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

Table 25-8 TWS Register Bits

Register Bits	Value	Description
TWS4:0	0x00	Bus error due to illegal START or STOP condition.
	0x08	A START condition has been transmitted.
	0x10	A repeated START condition has been transmitted.
	0x18	SLA+W has been transmitted; ACK has been received.
	0x20	SLA+W has been transmitted; NOT ACK has been received.
	0x28	Data byte has been transmitted; ACK has been received.
	0x30	Data byte has been transmitted; NOT ACK has been received.
	0x38	Arbitration lost in SLA+W or data bytes (Transmitter); Arbitration lost in SLA+R or NOT ACK bit (Receiver).
	0x40	SLA+R has been transmitted; ACK has been received.
	0x48	SLA+R has been transmitted; NOT ACK has been received.
	0x50	Data byte has been received; ACK has been

Register Bits	Value	Description
		returned.
	0x58	Data byte has been received; NOT ACK has been returned.
	0x60	Own SLA+W has been received; ACK has been returned.
	0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned.
	0x70	General call address has been received; ACK has been returned.
	0x78	Arbitration lost in SLA+R/W as Master; general call address has been received; ACK has been returned.
	0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned.
	0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned.
	0x90	Previously addressed with general call; data has been received; ACK has been returned.
	0x98	Previously addressed with general call; data has been received; NOT ACK has been returned.
	0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave.
	0xA8	Own SLA+R has been received; ACK has been returned.
	0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned.
	0xB8	Data byte in TWDR has been transmitted; ACK has been received.
	0xC0	Data byte in TWDR has been transmitted; NO ACK has been received.
	0xC8	Last data byte in TWDR has been transmitted (TWEA = 0); ACK has been received.
	0xF8	No relevant state information available; TWINT = 0.

- **Bit 2 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1:0 – TWPS1:0 - TWI Prescaler Bits**

These bits can be read and written and control the bit rate of the prescaler.

Table 25-9 TWPS Register Bits

Register Bits	Value	Description
TWPS1:0	0x00	1

Register Bits	Value	Description
	0x01	4
	0x02	16
	0x03	64

25.9.4 TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($BB)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is automatically controlled by the TWI logic. The CPU cannot access the ACK bit directly.

- **Bit 7:0 – TWD7:0 - TWI Data Register Byte**

25.9.5 TWAR – TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
NA ($BA)	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver. This register is not needed in the Master modes. In multi-master systems TWAR must be set in Masters which can be addressed as Slaves by other Masters. The LSB of TWAR is used to enable the recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bit 7:1 – TWA6:0 - TWI (Slave) Address**

These bits contain the TWI address operated as a Slave device.

- **Bit 0 – TWGCE - TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the 2-wire Serial Bus.

25.9.6 TWAMR – TWI (Slave) Address Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($BD)	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	Res	TWAMR
Read/Write	RW	RW	RW	RW	RW	RW	RW	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 – TWAM6:0 - TWI Address Mask**

The TWAMR can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWI Address Register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR.

- **Bit 0 – Res - Reserved Bit**

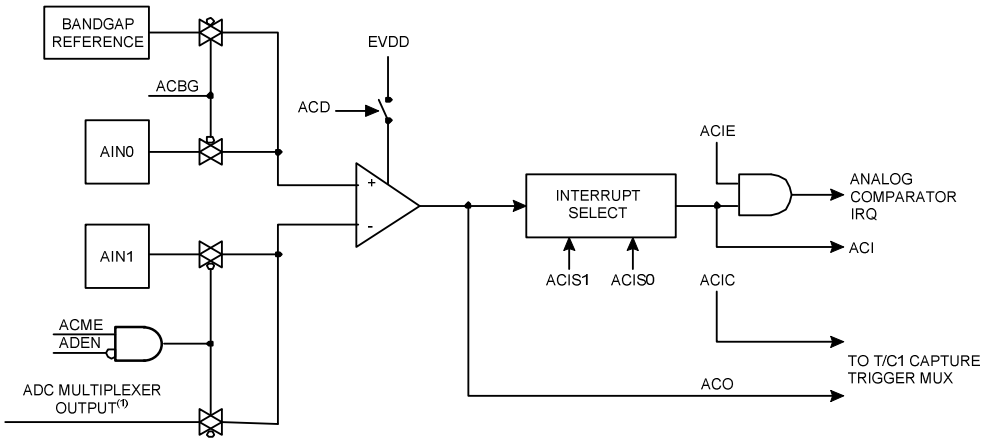
This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

26 AC – Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 26-1 below.

The Power Reduction ADC bit PRADC in PRR0 (see "PRR0 – Power Reduction Register0" on page 167) must be disabled by writing a logical zero to be able to use the ADC input multiplexer.

Figure 26-1. Analog Comparator Block Diagram



Note:
1. See Table 26-1 below.
2. Refer to Figure 1-1 on page 2 and Table 14-9 on page 197 for Analog Comparator pin placement.

26.1 Analog Comparator Multiplexed Input

It is possible to select any of the ADC7:0 pins as the negative input of the Analog Comparator. The ADC multiplexer is used to select this input and consequently the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX5 and MUX2:0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 26-1 below. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 26-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX5	MUX2:0	Analog Comparator Negative Input
0	x	x	xxx	AIN1
1	1	x	xxx	AIN1
1	0	0	000	ADC0
1	0	0	001	ADC1
1	0	0	010	ADC2
1	0	0	011	ADC3
1	0	0	100	ADC4

ACME	ADEN	MUX5	MUX2:0	Analog Comparator Negative Input
1	0	0	101	ADC5
1	0	0	110	ADC6
1	0	0	111	ADC7

26.2 Register Description

26.2.1 ACSR – Analog Comparator Control And Status Register

Bit	7	6	5	4	3	2	1	0	
$30 ($50)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	NA	0	0	0	0	0	

- **Bit 7 – ACD - Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG - Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage connects to the positive input of the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. When the bandgap reference is used as the input of the Analog Comparator, it will take a certain time for the voltage to stabilize. If not stabilized, the first comparison may give a wrong value. See section "Internal Voltage Reference" for details.

- **Bit 5 – ACO - Analog Compare Output**

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1-2 clock cycles.

- **Bit 4 – ACI - Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hard-ware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE - Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC - Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

- **Bit 1:0 – ACIS1:0 - Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in the following table. When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Table 26-2 ACIS Register Bits

Register Bits	Value	Description
ACIS1:0	0x00	Interrupt on Toggle
	0x01	Reserved
	0x02	Interrupt on Falling Edge
	0x03	Interrupt on Rising Edge

26.2.2 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($7B)		ACME							ADCSRB
Read/Write		RW							
Initial Value		0							

- **Bit 6 – ACME - Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer defines the negative input of the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see section "Analog Comparator Multiplexed Input".

26.2.3 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($7F)							AIN1D	AIN0D	DIDR1
Read/Write							RW	RW	
Initial Value							0	0	

- **Bit 1 – AIN1D - AIN1 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

- **Bit 0 – AIN0D - AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

27 ADC – Analog to Digital Converter

27.1 Features

- **10-bit Resolution**
- **Differential Non-Linearity is less than ± 0.5 LSB**
- **2 LSB Integral Non-Linearity**
- **3 - 240 µs Conversion Time**
- **Up to 330 kSPS (Up to 570 kSPS with 8-bit Resolution)**
- **8 Multiplexed Single Ended Input Channels**
- **11 Differential Input Channels**
- **2 Differential Input Channels with an Optional Gain of 10x and 200x**
- **Internal Linear Temperature Sensor**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{AVDD} ADC Input Voltage Range**
- **0 - V_{EVDD} Differential ADC Input Voltage Range**
- **Selectable 1.5V, 1.6V or V_{AVDD} ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceller**

The ATmega128RFA1 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port F. The single-ended voltage inputs refer to 0V (AVSS).

The device also supports multiple differential voltage input combinations. Two of the differential inputs (ADC1 & ADC0 and ADC3 & ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x) or 46 dB (200x) on the differential input voltage before the A/D conversion. The differential input channels are constructed of pairs out of the 8 single-ended inputs. They share a common negative terminal (ADC0, ADC1 or ADC2), while most of the other ADC inputs can be selected as the positive input terminal. If 1x or 10x gain is used, 8 bit resolution can be expected. If 200x gain is used, 6 bit resolution can be expected.

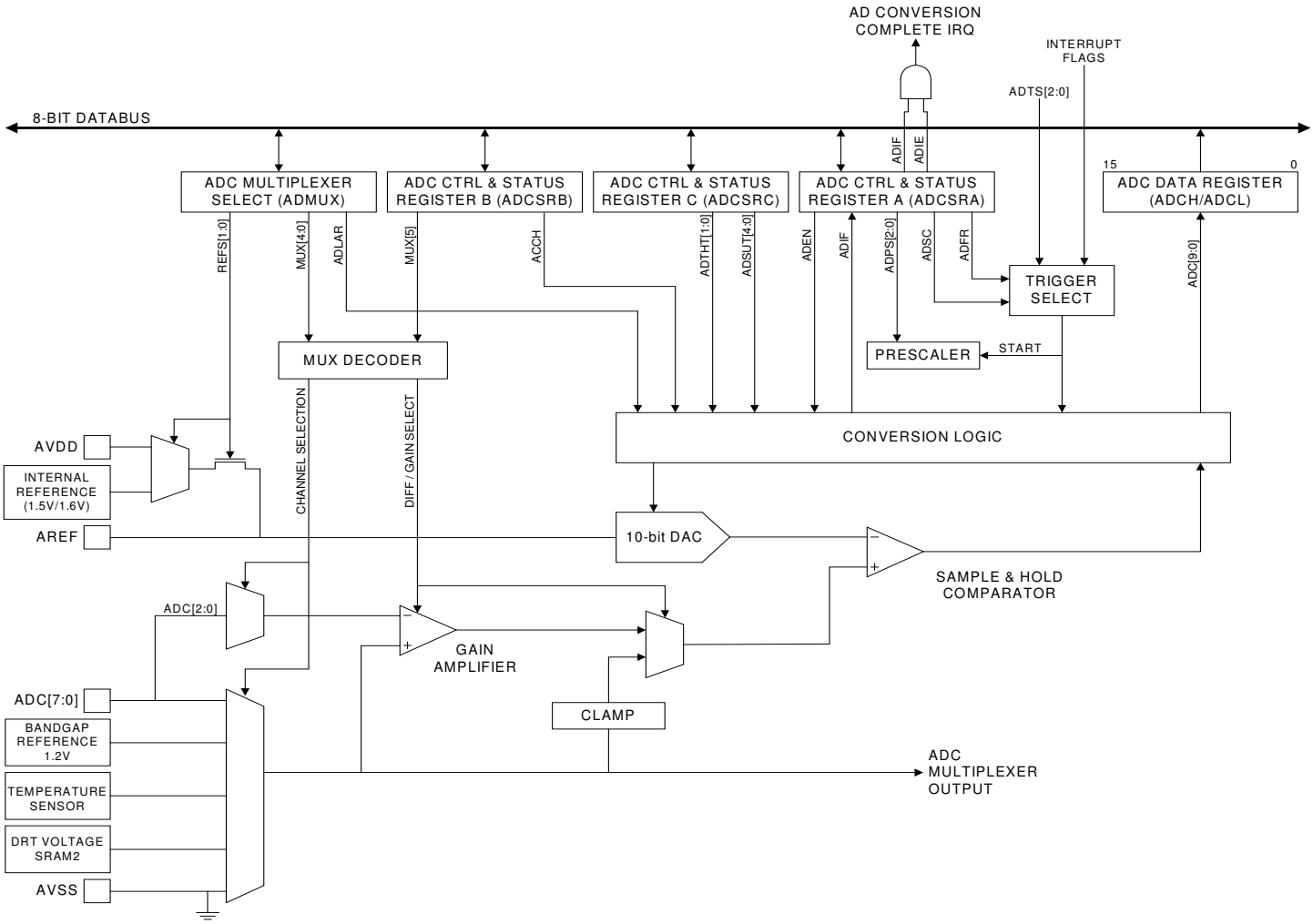
The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 27-1 on page 411.

The analog components of the ADC are supplied from the analog supply voltage AVDD. AVDD is generated from EVDD by an internal voltage generator. The logic part of the ADC is supplied from the digital supply voltage DVDD. DVDD is generated from DEVDD also by an internal voltage generator.

Internal reference voltages of nominally 1.5V, 1.6V or AVDD (1.8V) are provided on-chip. The 1.6V reference is calibrated to ± 1 LSB during manufacturing. The reference voltage can be monitored at the AREF pin. Additional de-coupling capacitance at AREF is not required. A high capacitive loading of AREF will de-stabilize the internal reference voltage generation. An external reference voltage in the range of $0 < V_{AREF,EXT} \leq V_{AVDD}$ may be used but must be supplied with a very low impedance.

The Power Reduction ADC bit, PRADC (see "PRR0 – Power Reduction Register0" on page 167) must be disabled by writing a logical zero to enable the ADC.

Figure 27-1. Analog to Digital Converter Block Schematic



27.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents 0V (conversion result 0x000) and the maximum value in single ended mode represents the reference voltage minus 1 LSB (conversion result 0x3FF). The reference voltage can be measured at the AREF pin. The internal, generated reference voltage can have the values 1.5V, 1.6V or AVDD where the 1.6V has the highest absolute accuracy. The reference voltage is selected by writing to the REFS*n* bits in the ADMUX Register. An external reference voltage can also be selected. Such an external voltage must be supplied with a very low impedance $R_{AREF,EXT}$ (see "ADC Electrical Characteristics" on page 505). The load current $I_{L,AREF}$ (see "ADC Electrical Characteristics" on page 505) seen by the external source is code dependent and changes in the course of the successive approximation process (load current steps). The internal voltage reference (except AVDD) must not be decoupled by an external capacitor. Adding unnecessary external capacitance at the AREF pin will

cause instable operation of the internal reference voltage buffer and will not improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX and ADCSRB. Any of the ADC input pins, as well as AVSS and a fixed bandgap voltage reference can be selected as single ended inputs to the ADC. A choice of ADC input pins can be selected as positive and negative inputs to the differential amplifier. Furthermore the temperature sensor and the DRT voltages of SRAM2 can also be processed with the ADC.

If differential channels are selected, the amplified voltage difference between the selected input channel pair then becomes the input of the ADC. The respective pin voltages for a differential measurement can be in the range from 0V to EVDD. In this way it is possible to handle differential input voltages with a common mode value higher than AVDD e.g. process a 50mV differential signal with a 2.5V common mode voltage. If single ended channels are used, the gain amplifier is bypassed altogether. Any ADC input voltage (single-ended or amplified-differential) exceeding AVDD will be internally clamped to AVDD to avoid damaging the ADC circuitry. Note that the pin input current will not increase if the clamp circuit is active.

The ADC is enabled by setting ADEN bit in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared. It is required to disable the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

27.3 ADC Start-Up

After the ADC is enabled by setting ADEN, it will go through a start-up phase. The analog supply voltage AVDD is turned on. It takes time t_{AVREG} (see "Power Management Electrical Characteristics" on page 503) µs for AVDD to stabilize. A stable AVDD voltage is indicated by the AVDDOK bit in ADCSRB. After this the ADC and, for differential input channels also the gain amplifier, is powered up. The duration of this phase depends on the ADC clock period and the configuration of the Start-Up and Track-And–Hold Time bits, ADSUT4:0 and ADTHT1:0 in ADCSRC. For details about the start-up timing refer to section "Pre-scaling and Conversion Timing" on page 413.

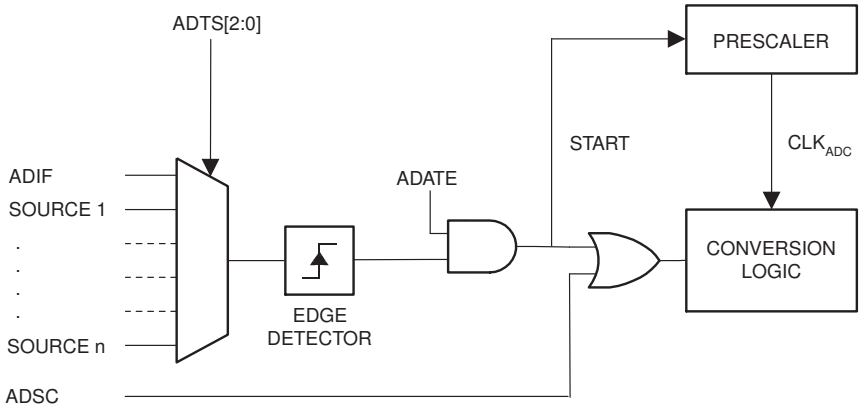
During the ADC start-up phase a conversion start can already be requested by writing a logical one to the ADC Start Conversion bit, ADSC in ADCSRA. In this case a conversion is started directly after the start-up phase. During the start-up phase it is still possible to change the analog input channel until the AVDDOK bit changes to logic one or, if the AVDDOK bit is one, until the ADSC bit is set.

27.4 Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 27-2. ADC Auto Trigger Logic



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.

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.

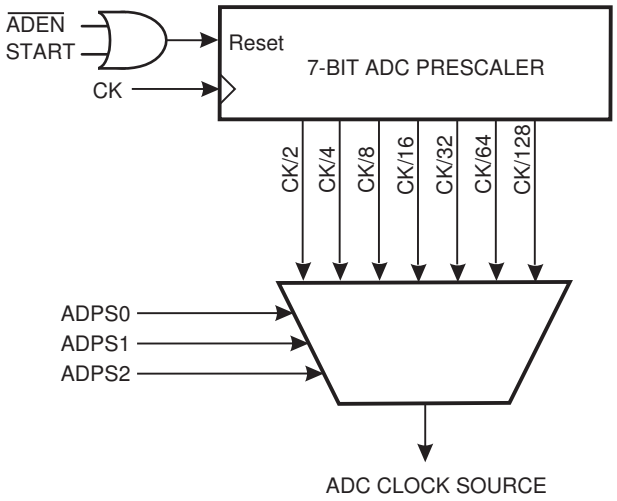
27.5 Pre-scaling and Conversion Timing

27.5.1 Prescaler

By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 4 MHz. If a lower resolution than 10 bits is needed, the input clock

frequency to the ADC can be as high as 8 MHz to get a higher sample rate. For differential input channels the ADC clock speed is restricted to a maximum of 2 MHz.

Figure 27-3. ADC Prescaler



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

27.5.2 Start-Up Timing

The ADC is enabled by setting the ADEN bit in ADCSRA. First the analog voltage regulator is powered up which takes t_{AVREG} (see "Power Management Electrical Characteristics" on page 503). A stable AVDD is indicated by the AVDDOK bit in ADCSRB.

After AVDD has stabilized, the ADC is started. The ADC start-up time has a length of t_{ADSU} and can be adjusted by the Start-Up time bits ADSUT4:0 in ADCSRC. If differential input channels are used, then an additional initialization period t_{AINIT} is required by the gain amplifier. This period is configured by the Track-And-Hold Time bits, ADTHT1:0 in ADCSRC. ADSUT4:0 and ADTHT1:0 are fixed numbers of ADC clock cycles and can be setup for different ADC clock speeds.

The minimum required ADC start-up time is 20 µs. Note that for the maximum ADC speed of 8 MHz the start-up time can not be set higher than 16 µs in ADSUT4:0. Under this condition the user has either to ensure that a conversion is not started earlier than 20 µs after the ADC is enabled or the first conversion result should be discarded.

For a summary of start-up times and sequences see Table 27-1 below, Table 27-2 below, Figure 27-4 on page 415 and Figure 27-5 on page 415.

Table 27-1. Start-Up Time, Single Ended Channels

Parameter	Duration in ADC Clock Cycles
ADC Start-Up Time t_{ADSU}	4(ADSUT+1), minimum 20 µs

Table 27-2. Start-Up Time, Differential Channels

Parameter	Duration in ADC Clock Cycles
ADC Start-Up Time t_{ADSU}	4(ADSUT+1), minimum 20 µs

Parameter	Duration in ADC Clock Cycles
Gain Amplifier Initialization Time t_{AINIT}	2(ADTHT+2)

Figure 27-4. ADC Timing Diagram, Start-Up for Single Ended Channels

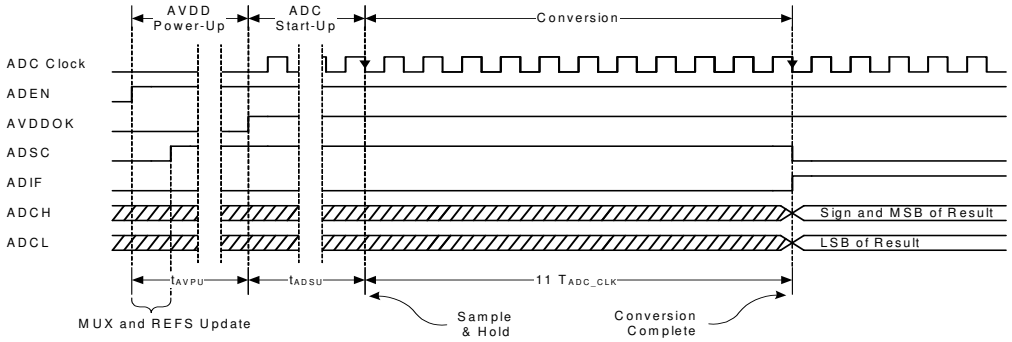
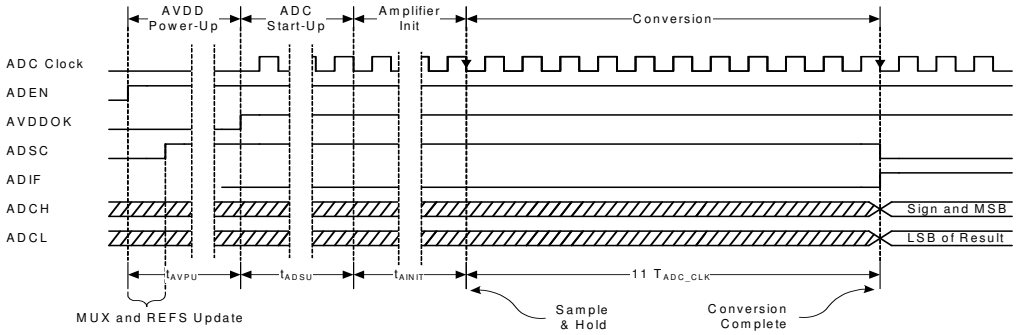


Figure 27-5. ADC Timing Diagram, Start-Up for Differential Channels



27.5.3 Conversion Timing

The delay from requesting a conversion start by setting the ADSC bit in ADCSRA to the moment where the sample-and-hold takes place is fixed. The same fixed delay also applies for auto triggered conversions. In this case three additional CPU clock cycles are used for the trigger event synchronization logic. The delay depends on the prescaler configuration ADPS and if single-ended or differential channels are used. A summary is given in Table 27-3 on page 416. All conversions take 11 ADC clock cycles.

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 at the earliest after the following tracking phase. The tracking phase is required after each conversion. Its duration can be adjusted according to the ADC clock speed by the ADTHT bits in ADCSRC and is different for single-ended and differential channels. For details see Table 27-4 on page 416.

In Free Running mode, a new conversion will be started immediately after the tracking phase of the previous conversion while ADSC remains high. The calculation of the resulting sample rate is given in Table 27-5 on page 416.

For timing diagrams of single and auto triggered and free running conversions see Figure 27-6 on page 416 to Figure 27-8 on page 417.

Table 27-3. Conversion Start Delay

Channel	ADPS	Delay from Conversion Start Request to Sample & Hold t_{SCSMP}
Single-Ended	0, 1	2 CPU clock cycles
	2	4 CPU clock cycles
	3	0 CPU clock cycles
	4…7	0 CPU clock cycles
Differential	0…7	2 ADC clock cycles

Table 27-4. Tracking Time

Channel	Tracking Phase Duration t_{TRCK} in ADC Clock Cycles
Single-Ended	ADTHT+1, minimum 500 ns
Differential	2ADTHT+3

Table 27-5. Sample Rate in Free Running Mode

Channel	Sample Rate in ADC Clock Cycles
Single-Ended	ADTHT+12
Differential	2ADTHT+14

Figure 27-6. ADC Timing Diagram, Single Conversion

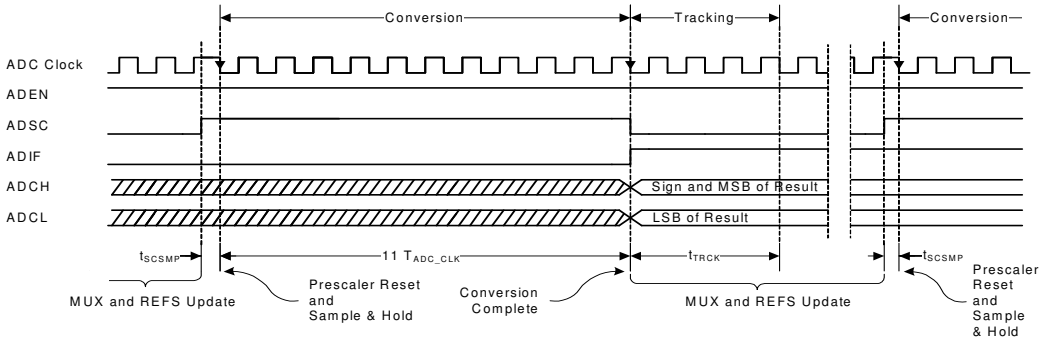


Figure 27-7. ADC Timing Diagram, Auto Triggered Conversion

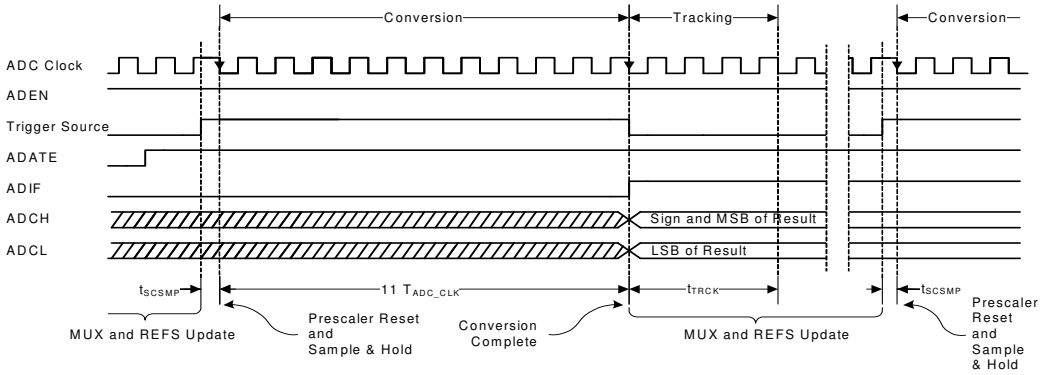
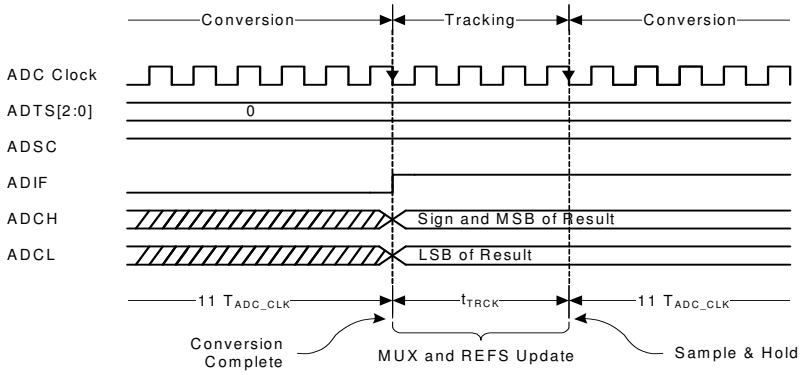


Figure 27-8. ADC Timing Diagram, Free Running Conversion



27.6 Changing Channel or Reference Selection

The MUX*n* and REFS*n* bits in the ADMUX and ADCSRB Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated either during the AVDD power-up phase or until a conversion is started by setting ADSC. After this the channel and reference selection is locked to ensure a sufficient initialization and sampling time for the ADC. Continuous updating of the channel selection resumes after the conversion has completed (ADIF in ADCSRA is set). The reference selection can only be updated if the ADC is disabled and enabled again.

If Auto Triggering is used, the exact time of the triggering event can be undetermined. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN in the ADSCRA Register are written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
2. During a conversion
3. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next A/D conversion.

After the channel or reference voltage selection is updated a settling time is required for the ADC and the gain amplifier or the reference voltage to stabilize. When changing the channel selection while the ADC is enabled the required settling phase is automatically inserted by the ADC interface, see section "ADC Input Channels" on page 418. For consideration on changing the reference voltage selection please refer to section"ADC Voltage Reference" on page 419.

27.6.1 Accessing the ADMUX Register

The channel selection bits MUX4:0 and MUX5 are located in two different register, the ADMUX and the ADCSRB register. To ensure that changes go only into effect after both register have been changed they are internally buffered (see Figure 27-9 on page 419 and Figure 27-10 on page 419). The MUX5 bit has to written first followed by a write access to the MUX4:0 bits which triggers the update of the internal buffer. If only

the MUX4:0 bits need to be modified then a write access to the MUX4:0 bits is sufficient.

27.6.2 ADC Input Channels

The ADC input channels can be changed while the ADC is running under the condition that the previous channel was a single-ended one. Changing between differential channels however requires that the ADC is disabled and enabled again to make the ADC go through the initial start-up phase.

If changing from single-ended to single-ended or from single-ended to differential input channels a settling phase is automatically inserted by the ADC interface logic after the input channel is modified. The settling phase is required by the ADC and the gain amplifier to stabilize. If a conversions start is requested during this settling phase, by setting ADSC or by a trigger event in Auto Triggered mode then the conversion is started only after the settling phase has completed.

In case the MUX*n* bits are altered during an ongoing conversion, the ADC input channel is changed after the conversion has completed. MUX*n* changes occurring during the tracking phase, which follows a conversion, will stop the tracking phase and the ADC settling phase will be entered.

In Free Running mode MUX*n* can also be modified. In this case the ADC input channel is changed after the conversion end or from the subsequent tracking phase. As a consequence the time from one conversion to the next is extended by the duration of the ADC settling phase.

The ADC settling time t_{ASET} depends on the previous and the new channel and on the configuration of the ADSUT4:0 and ADTHT1:0 bits as shown in Table 27-6 below. Additionally a synchronization delay t_{CHDLY} from 2 CPU to 2 ADC Clock cycles is required between changing the ADC input channel selection and the beginning of the settling phase. For details see the timing diagrams Figure 27-9 on page 419 and Figure 27-10 on page 419.

If the analog input signal encounters large variations it can be useful to manually reset the ADC and the gain amplifier before starting a new conversion. To achieve this, the settling phase can be forced without modifying MUX*n* by writing a logic one to the Analog Channel Change bit ACCH in ADCSRB. Using the ACCH bit is only recommended for single-ended input channels. For differential input channels the ADC and the gain amplifier can be reset if the ADC is disabled and enabled again.

Table 27-6. Settling Time after Channel Changes

Channel Transition	Settling Time t_{ASET} in ADC Clock Cycles
Single-Ended or Differential to Single-Ended	ADTHT+2
Single-Ended to Differential	4(ADSUT+1) + 2(ADTHT+2)
Differential to Differential	Requires the ADC to be disabled and enabled again.

Figure 27-9. ADC Timing Diagram, Changing MUX*n* after a Conversion

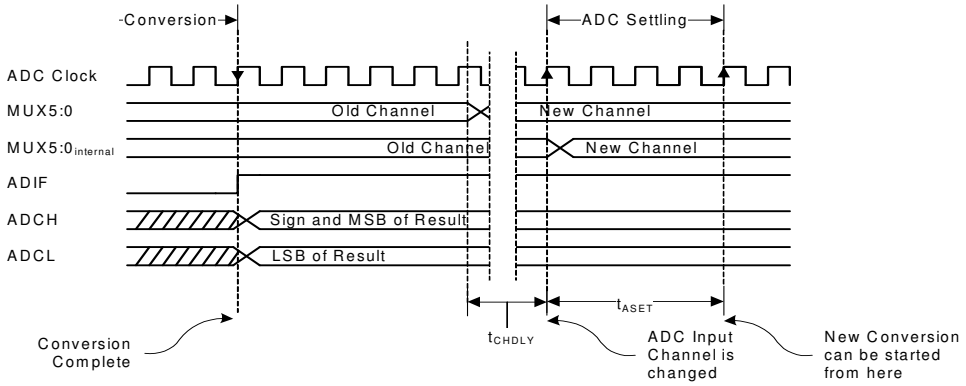
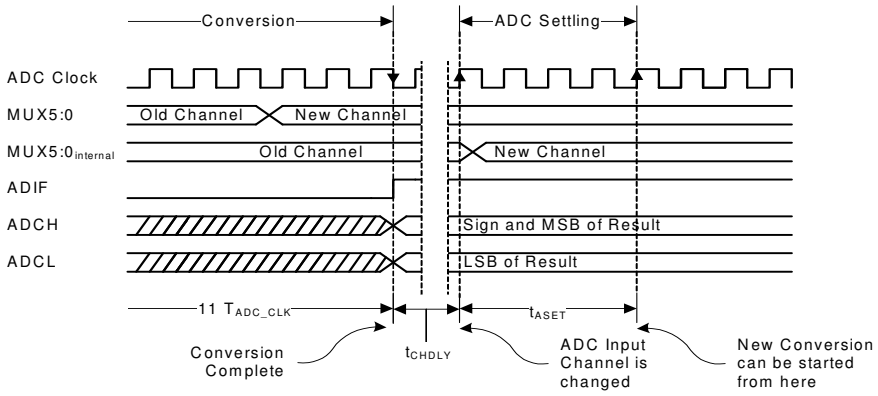


Figure 27-10. ADC Timing Diagram, Changing MUX*n* during a Conversion



27.6.3 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result 0x3FF. V_{REF} can be selected by the REFS*n* bits in the ADMUX register as either AVDD (1.8V), internal 1.5V or 1.6V reference or an external voltage at the AREF pin.

AVDD is connected to the ADC through a passive switch. The internal 1.5V and 1.6V references are generated from a bandgap reference (VBG) through an amplifier. In either case, the external AREF pin is directly connected to the ADC and the reference voltage can be measured at the AREF pin with a high impedance voltmeter. When using the internal 1.5V or 1.6V references no external de-coupling capacitor must be connected to AREF. High capacitive loading will de-stabilize the internal voltage amplifier. The 1.6V reference voltage is calibrated to an absolute accuracy of 1 LSB during the manufacturing process.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. An external reference voltage must be supplied with a very low impedance $\mathbf{R_{AREF,EXT}}$ (see "ADC Electrical Characteristics" on page 505). The load current $\mathbf{I_{L,AREF}}$ (see "ADC Electrical Characteristics" on page 505) seen by the external source is code dependent and changes (current steps) in the course of the successive approximation process. If no external voltage is applied to the AREF pin, the user may switch between AVDD, 1.5V and 1.6V as reference selection.

Changes of the reference selection bits REFS*n* will only take effect until the first conversion start is requested by setting ADSC in ADCSRA. After this the ADC has to be

disabled and enabled again for new reference selections. For internal references a stable voltage is indicated by the REFOK bit in ADCSRB.

27.7 ADC Noise Canceller

The ADC features a noise canceller that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceller can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC Conversion Complete interrupt must be enabled.
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the A/D conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the A/D conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the A/D conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

27.7.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 27-11 on page 421. An analog source applied to ADC*n* is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

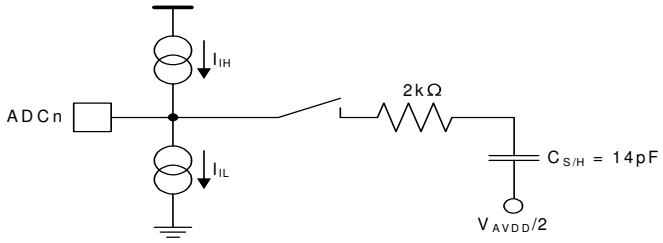
The ADC is optimized for analog signals having output impedance Z_{OUT} of approximately 3 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the correct sampling time will depend on how much time is needed to charge the S/H capacitor, which can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor. The required tracking time (input sampling switch closed) t_{DTRCK} to settle to within 1 LSB can be estimated to

$$t_{DTRCK} = (Z_{OUT} / k\Omega + 2000) \cdot 0.097ns$$

for Z_{OUT} > 3kΩ (worst case: maximum input step). A minimum tracking time of 500ns is guaranteed by the conversion logic. Based on the ADC clock frequency the bits ADTHT[1:0] of register ADCSRC allow the adjustment of the tracking time to the user's requirements.

Tracking time requirements should also be considered for the differential mode. The input signal is sampled by the gain amplifier. The value of the input capacitance $C_{S/H}$ depends on the selected gain (~7pF for 200x gain, <1pF otherwise). The tracking is equal to 50% of the clock period of CK_{ADC2}. Hence in differential mode a slower clock frequency is required for input sources with high impedance.

Figure 27-11. Analog Input Circuitry



Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

27.7.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the ground plane, and keep them well away from high-speed switching digital tracks.
2. Use the ADC noise canceller function to reduce induced noise from the CPU.
3. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

27.7.3 Offset Compensation Schemes

The differential amplifier has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can then be subtracted in software from the measurement results. The offset on any channel can be reduced below one LSB using this kind of software based offset correction.

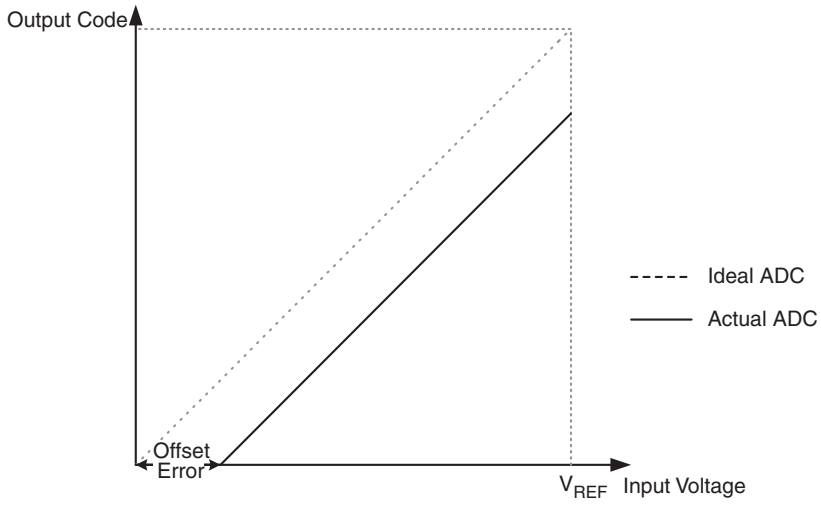
27.7.4 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between 0V and V_{REF} in 2^n steps (LSB's). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

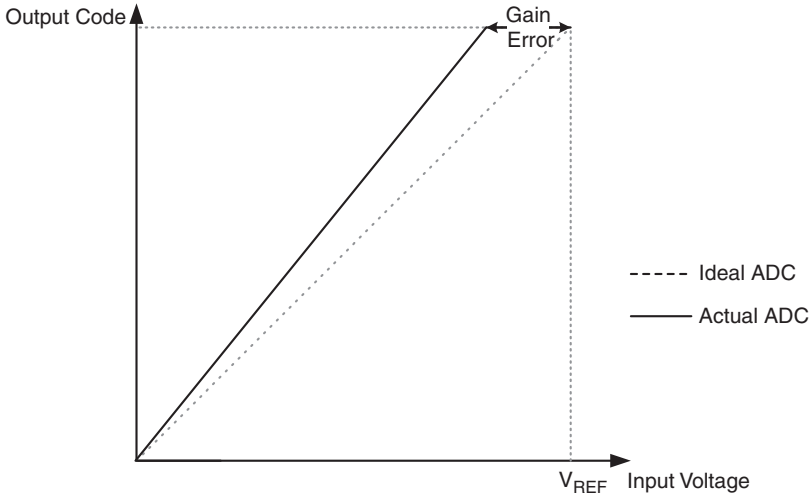
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 27-12. Offset Error



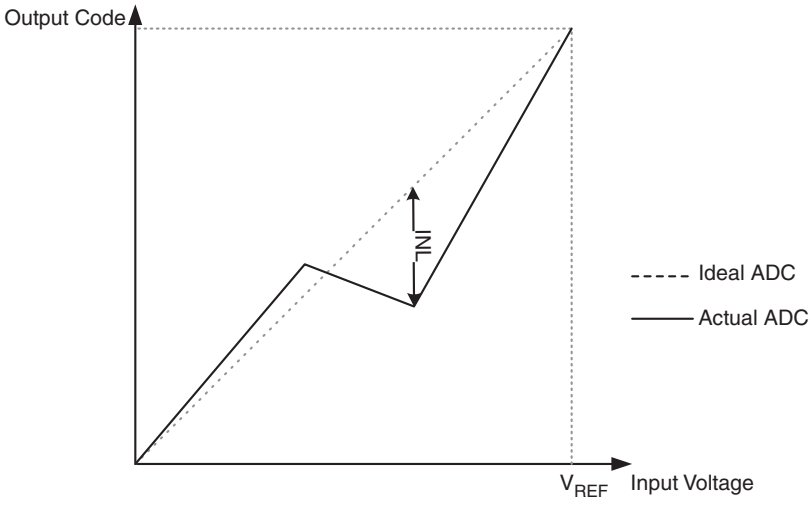
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

Figure 27-13. Gain Error



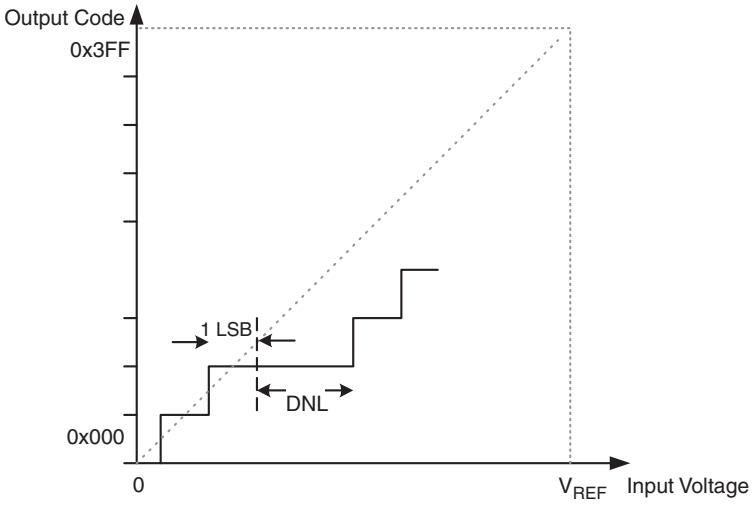
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 27-14. Integral Non-linearity (INL)



- Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 27-15. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. It is always ±0.5 LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5 LSB.

27.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see "Table 27-10" on page 427 and "Table 27-11" on page 428). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the result, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive. Figure 27-16 below shows the decoding of the differential input range.

Table 27-7 on page 425 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of V_{REF}.

Figure 27-16. Differential Measurement Range

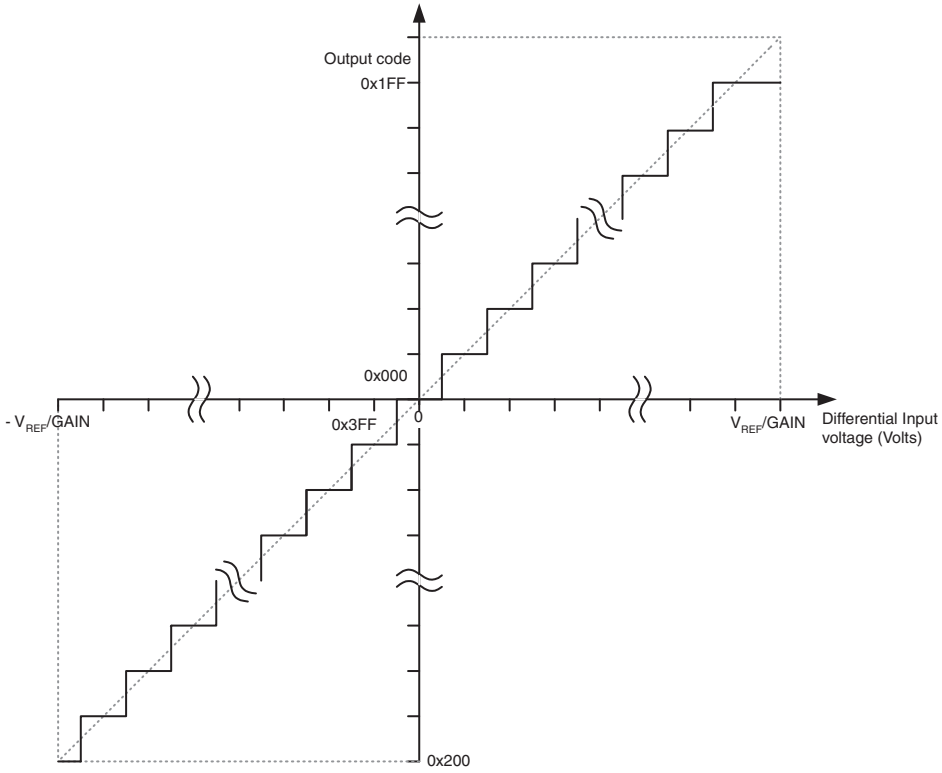


Table 27-7. Correlation Between Input Voltage and Output Codes

V_{ADCn}	Read Code	Corresponding Decimal Value
V_{ADCm} + V_{REF} / GAIN	0x1FF	511
V_{ADCm} + 0.999 V_{REF} / GAIN	0x1FF	511
V_{ADCm} + 0.998 V_{REF} / GAIN	0x1FE	510
…	…	…
V_{ADCm} + 0.001 V_{REF} / GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 0.001 V_{REF} / GAIN	0x3FF	-1
…	…	…
V_{ADCm} - 0.999 V_{REF} / GAIN	0x201	-511
V_{ADCm} - V_{REF} / GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10x gain, 1.6V reference, left adjusted result)

The voltage on ADC3 is 300 mV; the voltage on ADC2 is 425 mV.

ADCR = 512 * 10 * (300 - 425) / 1600 = -400 = 0x270.

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

27.9 Internal Temperature Measurement

The on-chip temperature can be measured using a special setup of the A/D converter inputs. The integrated temperature sensor provides a linear, medium-accurate voltage proportional to the absolute temperature (in Kelvin). This voltage is first amplified with the programmable gain amplifier and then processed with the A/D converter. A low frequency of the conversion clock must be selected due to the nature of the input signal. The absolute accuracy of the temperature measurement is limited by manufacturing tolerances, noise from supply and ground voltages and the exactness of the reference voltage. The following table summarizes the preferred setup of the temperature measurement:

Table 27-8. Recommended ADC Setup for Temperature Measurement

Parameter	Register	Recommended Setup
ADC Channel	ADMUX, ADCSRB	Select the Temperature Sensor, MUX4:0 = 01001; MUX5 = 1;
ADC Clock	ADCSRA	Select a clock frequency of 500kHz or lower;
V_{REF}	ADMUX	Select the internal 1.6V reference voltage;
Start-up time	ADCSRC	Standard requirement of 20µs is sufficient;
Tracking time	ADCSRC	Setting ADTHT = 0 is sufficient;

The A/D conversion result ADC_{TEMP} will always be a positive number. The ideal result can be calculated when using the internal 1.6V reference voltage according to the following equation:

$$ADC_{TEMP} = 241.4 + 0.885 \cdot \theta / °C$$

Similar the Celsius-temperature θ can be extracted from the A/D conversion result with this formula:

$$\theta / °C = 1.13 \cdot ADC_{TEMP} - 272.8$$

Note that the above equations are only valid in the allowed operating temperature range. The translation of the A/D measurement result to a Celsius-temperature value can be easily achieved with a look-up table in software. The accuracy of the temperature reading can be improved by averaging of multiple A/D conversion results. In this way the impact of noise is reduced. The temperature sensor is connected to a differential input channel with a gain of 10. The offset error of the channel can be corrected to the first order by using an appropriate channel (e.g. MUX4:0=01000, MUX5=0, see Table 27-11 on page 428). The in that manner measured error of the differential signal processing is then subtracted from the temperature sensor ADC reading.

Note that changing between the temperature sensor channel and the channel for the offset error correction can lead to a large difference of the analog input voltage. Therefore it is recommended to disable the ADC, select the new channel and then enable the ADC again.

27.10 SRAM DRT Voltage Measurement

The decrease of the supply voltage of SRAM block 2 for the leakage current reduction can also be measured using a special setup of the A/D converter inputs. The details of the SRAM leakage current reduction are described in section "SRAM with Data Retention" on page 163. The supply voltage of a disabled SRAM block can be reduced to save leakage power while maintaining data retention. This feature applies to all four SRAM blocks however only the voltage of SRAM block 2 can be verified using the A/D converter.

The default factory setting for the data retention (DRT) voltage normally guarantees the best leakage performances. Other values are nevertheless possible and can be selected by the application software. The true value of the supply voltage reduction is depending on the manufacturing process and environmental conditions like temperature. The A/D converter allows determining the value of the DRT voltage of SRAM block 2. The same voltage setting results for all practical purposes in the same supply voltage for all other SRAM blocks.

Care must be taken when verifying the DRT voltage of SRAM block 2 with the A/D converter because it will be put into sleep mode and hence it is not available for the application program. Addressing the disabled SRAM will return invalid data (all data read zero). The voltage measurement is split into two parts. One setting allows measuring the voltage drop from DVDD. The other setting allows verifying the voltage shift from DVSS. Both measurements are differential and use the programmable gain amplifier. A low frequency of the conversion clock must be selected due to the high-impedance nature of the input signal. Accurate and stable voltage readings may just be available after a long waiting time of up to 100 ms. This limitation is the consequence of the small leakage currents that discharge the internal de-coupling capacitances before the supply voltage settles to the DRT value. The following table summarizes the preferred setup of the DRT voltage measurement:

Table 27-9. Recommended ADC Setup for DRT Voltage Measurements

Parameter	Register	Recommended Setup
SRAM DRT on	DRTRAM2	Set bits DISPC and ENDRT to 1;
ADC Channel	ADMUX,	Select MUX4:0 = 10100 to measure V_{DRTBBP}; Select MUX4:0 = 11101 to measure V_{DRTBBN};

Parameter	Register	Recommended Setup
	ADCSRB	MUX5 = 1;
ADC Clock	ADCSRA	Select a clock frequency of 500kHz or lower;
V_{REF}	ADMUX	Select the internal 1.6V reference voltage;
Start-up time	ADCSRC	Standard requirement of 20µs is sufficient;
Tracking time	ADCSRC	Setting ADTHT = 0 is sufficient;

The A/D conversion result will always be a positive number for both V_{DRTBBP} and V_{DRTBBN}. The SRAM supply voltage is easily calculated according to the following equation (see chapter "SRAM with Data Retention" on page 163):

$$V_{DD,SRAM,DRT} = V_{DD} - (V_{DRTBBP} + V_{DRTBBN})$$

The conversion result is coded as described in "ADC Conversion Result" on page 423 with a GAIN of 0.5. It is not possible to read both V_{DRTBBP} and V_{DRTBBN} at the same time. However the time required for the A/D conversion is short compared to the time constant of a DRT voltage change.

27.11 Register Description

27.11.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
NA ($7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in the following table. Changes of these bits will only take effect until the first conversion start is requested by setting ADSC. After this the ADC has to be disabled and enabled again for new reference selections. The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 27-10. Reference Voltage Selections for ADC

REFS1	REFS0	Reference Voltage Selection
0	0	AREF, Internal V_{REF} turned off
0	1	AVDD (1.8V)
1	0	Internal 1.5V Voltage Reference (no external capacitor at AREF pin)
1	1	Internal 1.6V Voltage Reference (no external capacitor at AREF pin)

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the A/D conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 432.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs is connected to the ADC. See Table 27-11 on page 428 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in

ADCSRA is set). Note that the MUX5 bit is located in the ADCSRB register. A write access to the MUX4:0 bits triggers the update of the internally buffered MUX5 bit, see "Accessing the ADMUX Register" on page 417 .

27.11.2 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($7B)	AVDDOK	ACME	REFOK	ACCH	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – AVDDOK: AVDD Supply Voltage OK**

The analog functions of the ADC are powered from the AVDD domain. AVDD is supplied from an internal voltage regulator. Setting the ADEN bit in register ADCSRA will power-up the AVDD domain if not already requested by another functional group of the device. The bit allows the user to monitor (poll) the status of the AVDD domain. A status of 1 indicates that AVDD has been powered-up.

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

This bit is used for the Analog Comparator only. See "ADCSRB – ADC Control and Status Register B" on page 409 for details.

- **Bit 5 – REFOK: Reference Voltage OK**

The status of the internal generated reference voltage can be monitored through this bit. Setting the ADEN bit in register ADCSRA will enable the reference voltage for the ADC according to the REFS*n* bits in the ADMUX register. The reference voltage will be available after a start-up delay. A REFOK value of 1 indicates that the internal generated reference voltage is approaching final levels.

- **Bit 4 – ACCH: Analog Channel Change**

The user can force a reset of the analog blocks by setting this bit to 1 without requesting a different channel. The analog blocks of the ADC will be reset to handle possible new voltage ranges. Such a reset phase is especially important for the gain amplifier. It could be temporarily disabled by a large step of its input common voltage leading to erroneous A/D conversion results. ACCH will read as one until the reset phase of the analog blocks can be entered.

- **Bit 3 – MUX5: Analog Channel and Gain Selection Bit**

This bit is used together with MUX4:0 in ADMUX to select the analog input signals connected to the ADC. See the following table for details. If this bit is changed during a conversion, the change will not go in effect until this conversion is complete. Note that the MUX5 bit is internally buffered and a write access to the MUX4:0 bits is required to trigger the update of the MUX5 bit, see "Accessing the ADMUX Register" on page 417 .

Table 27-11. Input Channel Selections

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000110	ADC6			
000111	ADC7			
001000	N/A	ADC0	ADC0	10x
001001		ADC1	ADC0	10x
001010		ADC0	ADC0	200x
001011		ADC1	ADC0	200x
001100		ADC2	ADC2	10x
001101		ADC3	ADC2	10x
001110		ADC2	ADC2	200x
001111		ADC3	ADC2	200x
010000	N/A	ADC0	ADC1	1x
010001		ADC1	ADC1	1x
010010		ADC2	ADC1	1x
010011		ADC3	ADC1	1x
010100		ADC4	ADC1	1x
010101		ADC5	ADC1	1x
010110		ADC6	ADC1	1x
010111		ADC7	ADC1	1x
011000	N/A	ADC0	ADC2	1x
011001		ADC1	ADC2	1x
011010		ADC2	ADC2	1x
011011		ADC3	ADC2	1x
011100		ADC4	ADC2	1x
011101		ADC5	ADC2	1x
011110	1.2V (V_{BG})	N/A		
011111	0V (AVSS)			
100000	Reserved	N/A		
100001	Reserved			
100010	Reserved			
100011	Reserved			
100100	Reserved			
100101	Reserved			
100110	Reserved			
100111	Reserved			
101000	N/A	Reserved		
101001		Temperature Sensor		
101010		Reserved		
101011		Reserved		
101100		Reserved		
101101		Reserved		
101110		Reserved		

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
101111		Reserved		
110000	N/A	Reserved		
110001		Reserved		
110010		Reserved		
110011		Reserved		
110100		SRAM Back-bias Voltage V_{DRTBBP}		
110101		Reserved		
110110		Reserved		
110111		Reserved		
111000	N/A	Reserved		
111001		Reserved		
111010		Reserved		
111011		Reserved		
111100		Reserved		
111101		SRAM Back-bias Voltage V_{DRTBBN}		
111110	Reserved	N/A		
111111	Reserved			

- **Bits 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an A/D conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared, to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS2:0=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 27-12. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

27.11.3 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. The AVDD supply voltage will also be enabled if not already available. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will include a start-up time to initialize the analog blocks of the ADC. The start-up time is defined by the ADSUT bits of register ADCSRC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an A/D conversion is completed and the Data Register are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the CPU frequency and the input clock to the ADC.

Table 27-13. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

27.11.4 ADCSRC – ADC Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($77)	ADTHT1	ADTHT0	Res0	ADSUT4	ADSUT3	ADSUT2	ADSUT1	ADSUT0	ADCSRC
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	1	0	1	0	1	0	0	

This register defines the track-and-hold time for sampling the analog input voltage of the ADC and it defines the start-up time for the analog blocks based on a number of ADC clock cycles. The ADC clock is generated from the system clock with the ADC prescaler. The bits ADPS2:0 of register ADCSRA set the prescaler ratio. Correct start-up and track-and-hold times are important for precise conversion results.

- **Bits 7:6 – ADTHT1:0: ADC Track-and-Hold Time**

These bits define the number of ADC clock cycles for the sampling time of the analog input voltage. For a complete description of this bit, see "Pre-scaling and Conversion Timing" on page 413.

- **Bit 5 – Res0: Reserved**
- **Bits 4:0 – ADSUT4:0: ADC Start-up Time**

These bits define the number of ADC clock cycles for the start-up time of the analog blocks. For a complete description of this bit, see "Pre-scaling and Conversion Timing" on page 413.

27.11.5 ADCL and ADCH – The ADC Data Register

27.11.5.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
NA ($79)	–	–	–	–	–	–	**ADC9**	**ADC8**	**ADCH**
NA ($78)	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADC1**	**ADC0**	**ADCL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

27.11.5.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
NA ($79)	**ADC9**	**ADC8**	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADCH**
NA ($78)	**ADC1**	**ADC0**	–	–	–	–	–	–	**ADCL**
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an A/D conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision (7 bit + sign bit for differential input channels) is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUX*n* bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: A/D Conversion Result**

These bits represent the result from the conversion as detailed in "ADC Conversion Result" on page 423.

27.11.6 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($7E)	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – ADC7D:ADC0D: Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7:0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

27.11.7 DIDR2 – Digital Input Disable Register 2

Bit	7	6	5	4	3	2	1	0	
NA ($7D)	ADC15D	ADC14D	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	DIDR2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

Reserved for future use.

- **Bit 7:0 – ADC15D:ADC8D - Reserved Bits**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero.

27.11.8 BGCR – Reference Voltage Calibration Register

Bit	7	6	5	4	
NA ($67)	Res	BGCAL_FINE3	BGCAL_FINE2	BGCAL_FINE1	BGCR
Read/Write	R	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($67)	BGCAL_FINE0	BGCAL2	BGCAL1	BGCAL0	BGCR
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register contains the calibration values of the reference voltage of the ADC. The values are loaded from the fuse memory after power-up. They can be corrected by the application software e.g. to compensate for temperature changes. The internal 1.6V reference voltage is calibrated and has therefore the highest accuracy compared to the 1.5V or AVDD reference.

- **Bit 7 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 6:3 – BGCAL_FINE3:0 - Fine Calibration Bits**

These bits allow the calibration of the AREF voltage with a resolution of 2mV.

Table 27-14 BGCAL_FINE Register Bits

Register Bits	Value	Description
BGCAL_FINE3:0	0	Center value
	1	Voltage step up
	8	Voltage step down
	7	Setting for highest voltage
	15	Setting for lowest voltage

- **Bit 2:0 – BGCAL2:0 - Coarse Calibration Bits**

These bits allow the calibration of the AREF voltage with a resolution of 10mV.

Table 27-15 BGCAL Register Bits

Register Bits	Value	Description
BGCAL2:0	4	Center value
	3	Voltage step up
	5	Voltage step down
	0	Setting for highest voltage
	7	Setting for lowest voltage

28 JTAG Interface and On-chip Debug System

28.1 Features

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard**
- **Debugger Access to:**
  - **All Internal Peripheral Units**
  - **Internal and External RAM**
  - **The Internal Register File–Program Counter**
  - **EEPROM and Flash Memories**
- **Extensive on-chip debug Support for Break Conditions, Including**
  - **AVR Break Instruction**
  - **Break on Change of Program Memory Flow**
  - **Single Step Break**
  - **Program Memory Breakpoints on Single Address or Address Range**
  - **Data Memory Breakpoints on Single Address or Address Range**
- **Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface**
- **On-chip debugging Supported by AVR Studio®**

28.2 Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

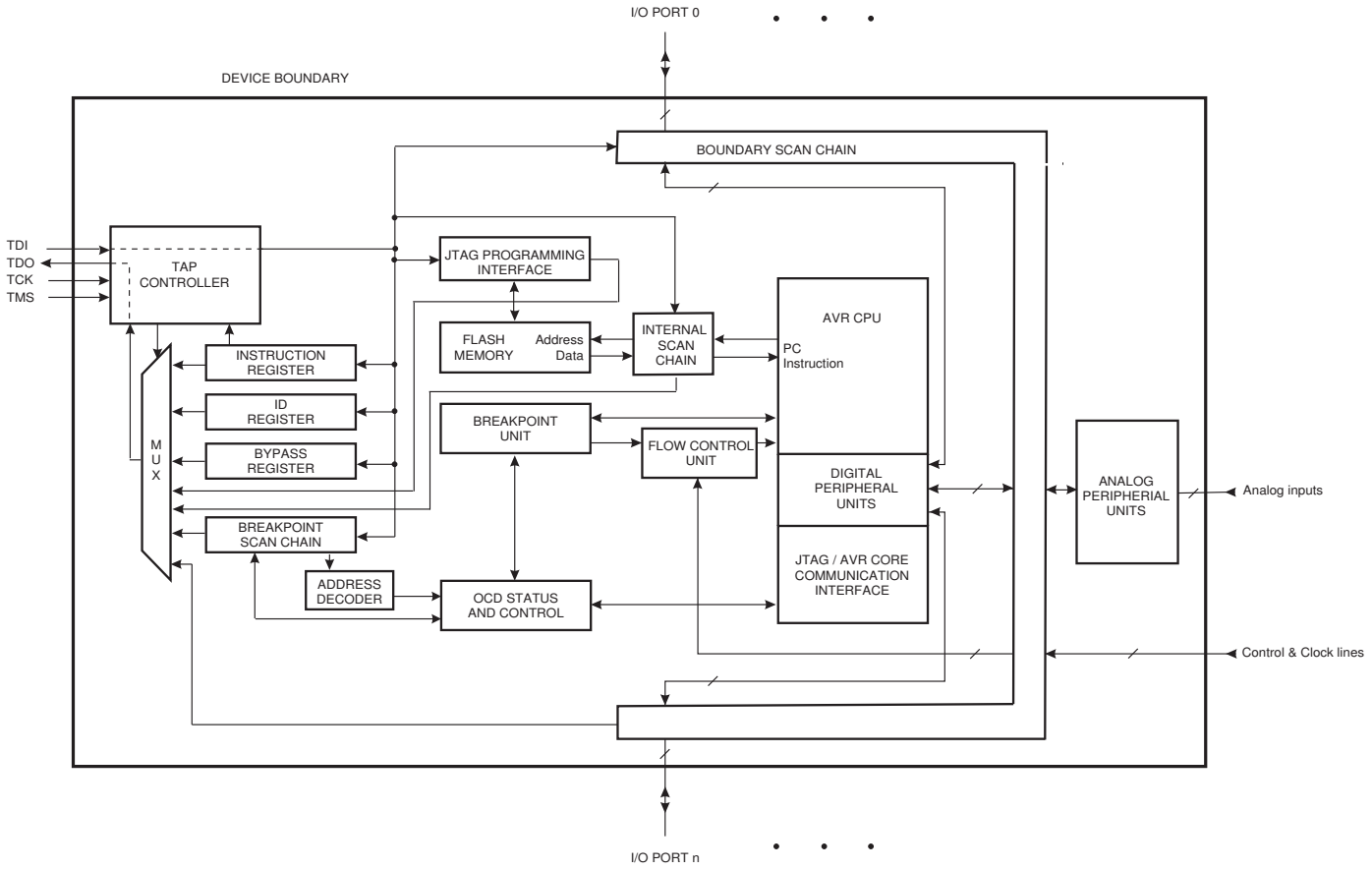
- Testing PCBs by using the JTAG Boundary-scan capability
- Programming the non-volatile memories, Fuses and Lock bits
- On-chip debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-scan Chain can be found in the sections "Programming via the JTAG Interface" on page 481 and "Programming via the JTAG Interface" on page 481, respectively. The on-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 28-1 on page 436 shows a block diagram of the JTAG interface and the on-chip debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (Shift Register) between the TDI – input and TDO – output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the Data Registers used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for serial programming via the JTAG interface. The internal scan-chain and breakpoint scan-chain are used for on-chip debugging only.

Figure 28-1. Block Diagram



28.3 TAP - Test Access Port

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port – TAP. These pins are:

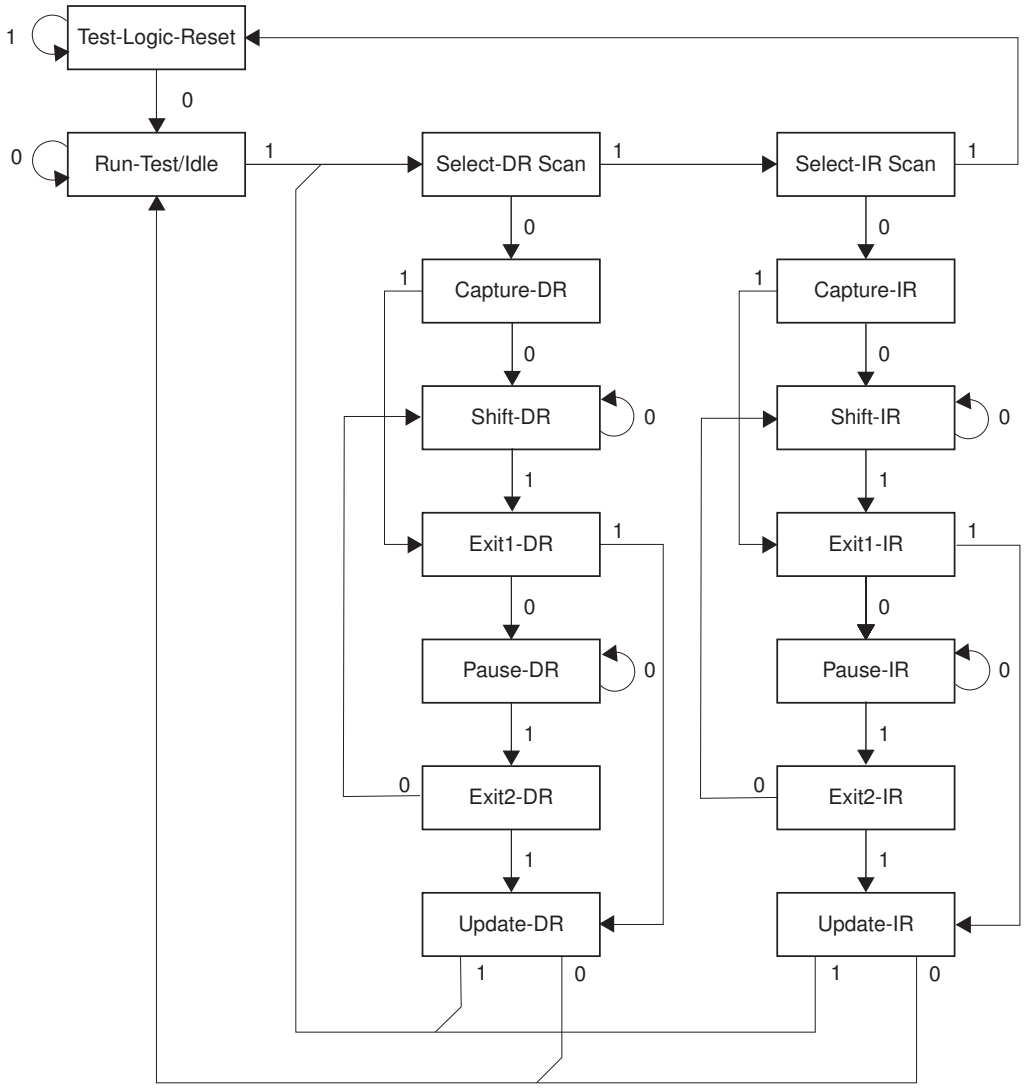
- TMS: Test mode select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK.
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains).
- TDO: Test Data Out. Serial output data from Instruction Register or Data Register.

The IEEE std. 1149.1 also specifies an optional TAP signal; TRST – Test ReSeT – which is not provided.

When the JTAGEN Fuse is un-programmed, these four TAP pins are normal port pins, and the TAP controller is in reset. When programmed the input TAP signals are internally pulled high and the JTAG is enabled for Boundary-scan and programming. The device is shipped with this fuse programmed.

For the on-chip debug system, in addition to the JTAG interface pins, the RESET pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the RESET pin low to reset the whole system, assuming only open collectors on the reset line are used in the application.

Figure 28-2. TAP Controller State Diagram



28.4 TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or on-chip debug system. The state transitions depicted in Figure 28-2 above depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-on Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out on the TDO pin. The JTAG Instruction

selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register – Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note that independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods. For detailed information on the JTAG specification, refer to the literature listed in "Bibliography" on page 440.

28.5 Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 441.

28.6 Using the On-chip Debug System

As shown in Figure 28-1, the hardware support for on-chip debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units.
- Breakpoint unit.
- Communication interface between the CPU and JTAG system.

All read or modify/write operations needed for implementing the debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Breakpoint Unit implements *Break on Change of Program Flow*, *Single Step Break*, two program memory breakpoints and two combined breakpoints. Together, the four breakpoints can be configured as either:

- 4 single program memory breakpoints;
- 3 single program memory breakpoint + 1 single data memory breakpoint;
- 2 single program memory breakpoints + 2 single data memory breakpoints;

- 2 single program memory breakpoints + 1 program memory breakpoint with mask (“range breakpoint”).
- 2 single program memory breakpoints + 1 data memory breakpoint with mask (“range breakpoint”).

A debugger, like the AVR Studio, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the on-chip debug specific JTAG instructions is given in "On-chip Debug Specific JTAG Instructions" below.

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the on-chip debug system to work. As a security feature, the on-chip debug system is disabled when either of the LB1 or LB2 Lock-bits are set. Otherwise, the on-chip debug system would have provided a back-door into a secured device.

The AVR Studio enables the user to fully control execution of programs on an AVR device with on-chip debug capability, AVR In-Circuit Emulator, or the built-in AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly programs assembled with Atmel Corporation’s AVR Assembler and C programs compiled with third party vendors’ compilers. For a full description of the AVR Studio, please refer to the AVR Studio User Guide. Only highlights are presented in this document.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code breakpoints (using the BREAK instruction) and up to two data memory Breakpoints, alternatively combined as a mask (range) breakpoint.

28.7 On-chip Debug Specific JTAG Instructions

The on-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction operation codes are listed for reference.

28.7.1 PRIVATE0; 0x8

Private JTAG instruction for accessing on-chip debug system;

28.7.2 PRIVATE1; 0x9

Private JTAG instruction for accessing on-chip debug system;

28.7.3 PRIVATE2; 0xA

Private JTAG instruction for accessing on-chip debug system;

28.7.4 PRIVATE3; 0xB

Private JTAG instruction for accessing on-chip debug system;

28.8 Using the JTAG Programming Capabilities

Programming of the ATmega128RFA1 via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI, and TDO. These are the only pins that need to be controlled and observed to perform JTAG programming (in addition to power pins). The JTAGEN Fuse must be programmed and the JTD bit in the MCUCR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- Flash programming and verifying.
- EEPROM programming and verifying.
- Fuse programming and verifying.
- Lock bit programming and verifying.

The Lock bit security is exactly as in parallel programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section "Programming via the JTAG Interface" on page 481.

28.9 Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993.
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992.

28.10 On-chip Debug Related Register in I/O Memory

28.10.1 OCDR – On-Chip Debug Register

Bit	7	6	5	4	3	2	1	0	
$31 ($51)	OCDR7:0								OCDR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The OCDR Register provides a communication channel from the running program in the microcontroller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an internal flag; I/O Debug Register Dirty IDRD is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information. In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

- **Bit 7:0 – OCDR7:0 - On-Chip Debug Register Data**

Table 28-16 OCDR Register Bits

Register Bits	Value	Description
OCDR7:0	0	Refer to the debugger documentation for further information on how to use this register.

29 IEEE 1149.1 (JTAG) Boundary-scan

29.1 Features

- **JTAG (IEEE std. 1149.1 compliant) Interface**
- **Boundary-scan Capabilities According to the JTAG Standard**
- **Full Scan of all Port Functions as well as Analog Circuitry having Off-chip Connections**
- **Supports the Optional IDCODE Instruction**
- **Additional Public AVR_RESET Instruction to Reset the ATmega128RFA1**

29.2 System Overview

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long Shift Register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the four TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the Data Register path will show the ID-Code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. Entering reset, the outputs of any port pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESET pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN Fuse must be programmed and the JTD bit in the I/O Register MCUCR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

29.3 Data Registers

The Data Registers relevant for Boundary-scan operations are:

- Bypass Register
- Device Identification Register

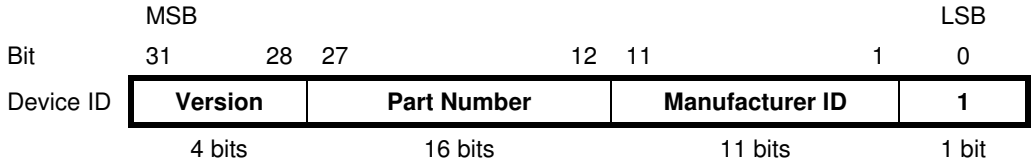
- Reset Register
- Boundary-scan Chain

29.3.1 Bypass Register

The Bypass Register consists of a single Shift Register stage. When the Bypass Register is selected as path between TDI and TDO, the register is reset to 0 when leaving the Capture-DR controller state. The Bypass Register can be used to shorten the scan chain on a system when the other devices are to be tested.

29.3.2 Device Identification Register

Figure 29-1. The Format of the Device Identification Register



29.3.2.1 Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on.

29.3.2.2 Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega128RFA1 is listed in Table 31-6 on page 467.

29.3.2.3 Manufacturer ID

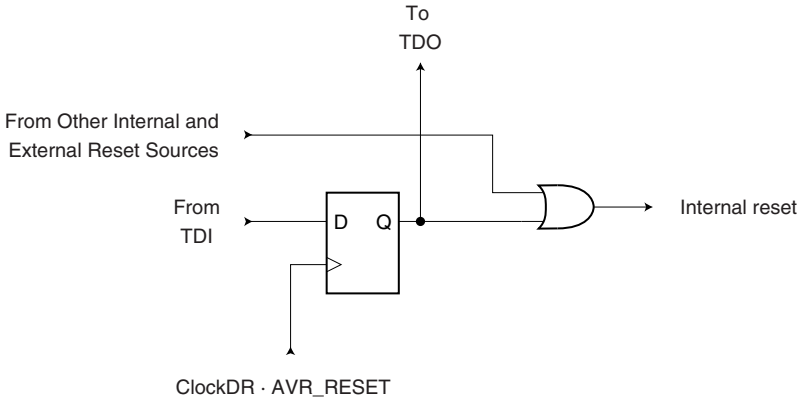
The Manufacturer ID is a 11-bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 31-6 on page 467.

29.3.3 Reset Register

The Reset Register is a test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the fuse settings for the clock options, the part will remain reset for a reset time-out period (see "Clock Sources" on page 148) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 29-2 on page 443.

Figure 29-2. Reset Register



29.3.4 Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections.

See "Boundary-scan Chain" on page 444 for a complete description.

29.4 Boundary-scan Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedance state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

29.4.1 EXTEST; 0x0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-Register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

29.4.2 IDCODE; 0x1

Optional JTAG instruction selecting the 32 bit ID-Register as Data Register. The ID-Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE Register is sampled into the Boundary-scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

29.4.3 SAMPLE_PRELOAD; 0x2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan chain is applied to the output latches. However, the output latches are not connected to the pins.

29.4.4 AVR_RESET; 0xC

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

29.4.5 BYPASS; 0xF

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic “0” into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

29.5 Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connection.

29.5.1 Scanning the Digital Port Pins

Figure 29-3 on page 445 shows the Boundary-scan Cell for a bi-directional port pin. The pull-up function is disabled during Boundary-scan when the JTAG IC contains EXTEST or SAMPLE_PRELOAD. The cell consists of a bi-directional pin cell that combines the three signals Output Control - OCxn, Output Data - ODxn, and Input Data - IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description.

The Boundary-scan logic is not included in the figures in the datasheet. Figure 29-4 on page 446 shows a simple digital port pin as described in the section "I/O-Ports" on page 186. The Boundary-scan details from Figure 29-3 on page 445 replaces the dashed box in Figure 29-4 on page 446.

When no alternate port function is present, the Input Data - ID - corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction - DD Register, and the Pull-up Enable - PUExn – corresponds to logic expression:

$$\overline{PUD} \cdot \overline{DDxn} \cdot PORTxn$$

Digital alternate port functions are connected outside the dotted box Figure 29-4 on page 446 to make the scan chain read the actual pin value. For analog function, there is a direct connection from the external pin to the analog circuit. There is no scan chain on the interface between the digital and the analog circuitry, but some digital control signal to analog circuitry are turned off to avoid driving contention on the pads.

When JTAG IR contains EXTEST or SAMPLE_PRELOAD the clock is not sent out on the port pins even if the CKOUT fuse is programmed. Even though the clock is output when the JTAG IR contains SAMPLE_PRELOAD, the clock is not sampled by the boundary scan.

Figure 29-3. Boundary-scan Cell for Bi-directional Port Pin with Pull-up Function

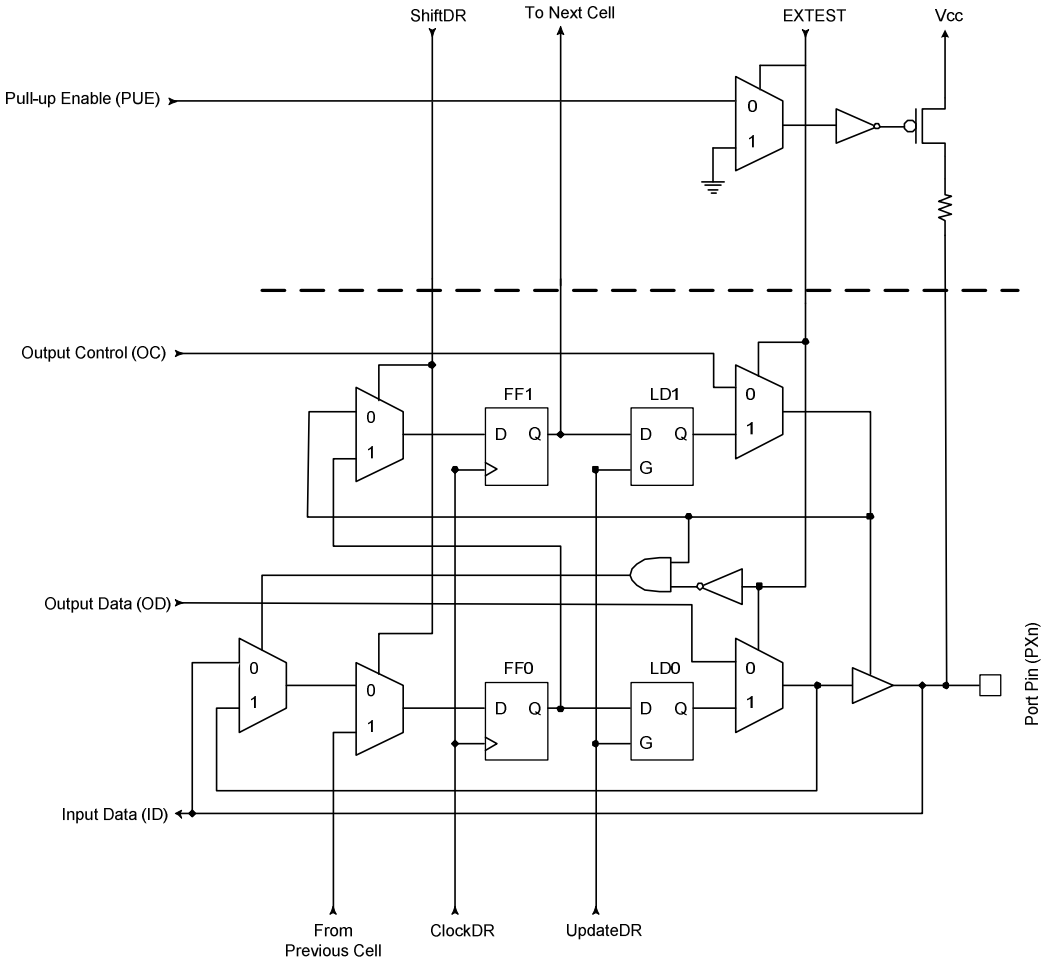
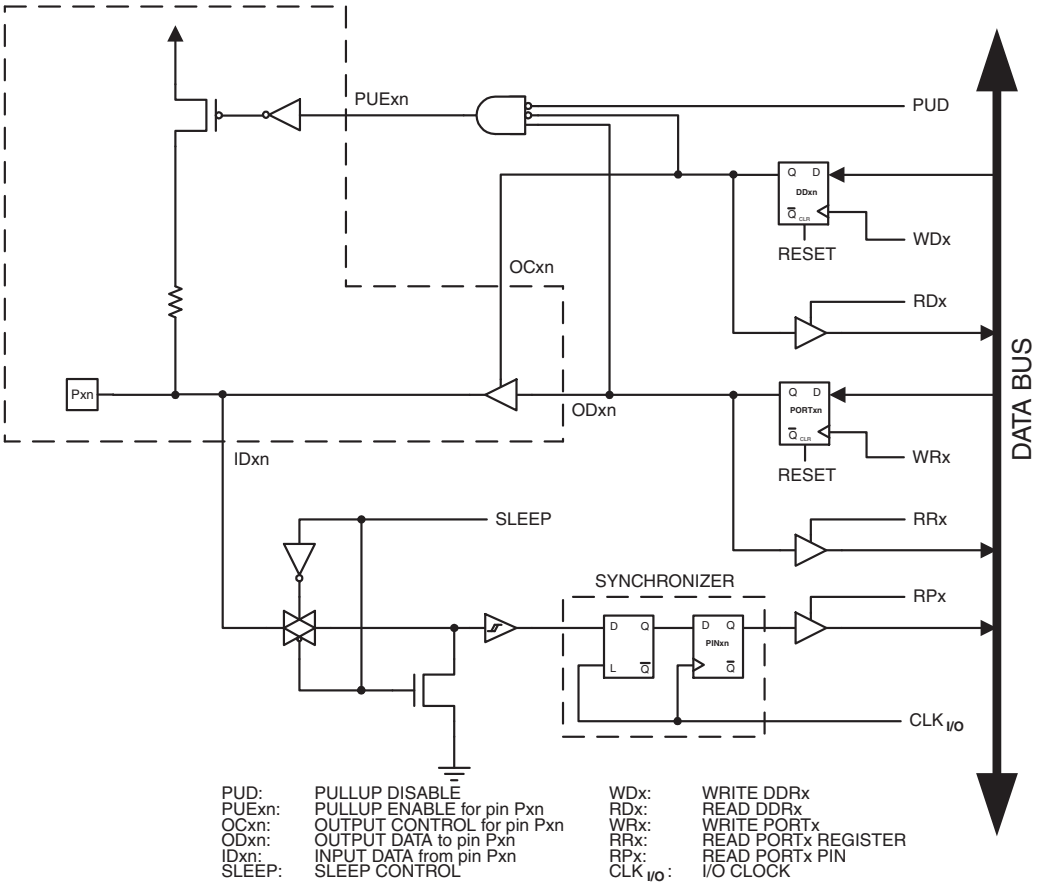


Figure 29-4. General Port Pin Schematic Diagram

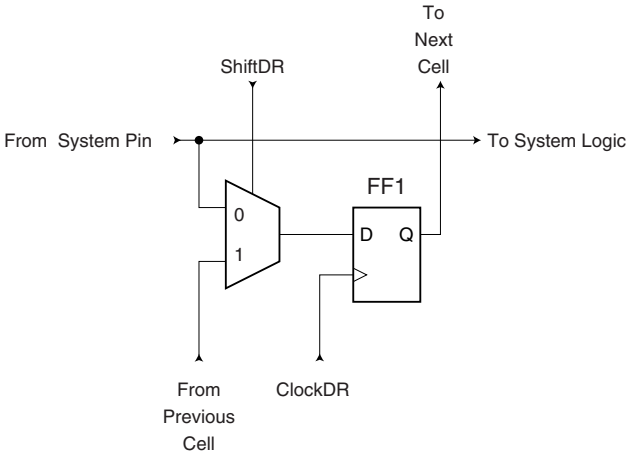
See Boundary-scan Description for Details!



29.5.2 Scanning the RSTN, CLKI and TST Pin

An observe-only cell as shown in Figure 29-5 below is inserted for the active low reset signal RSTN, for the active high programming and test mode enable signal TSTN and for the clock input CLKI.

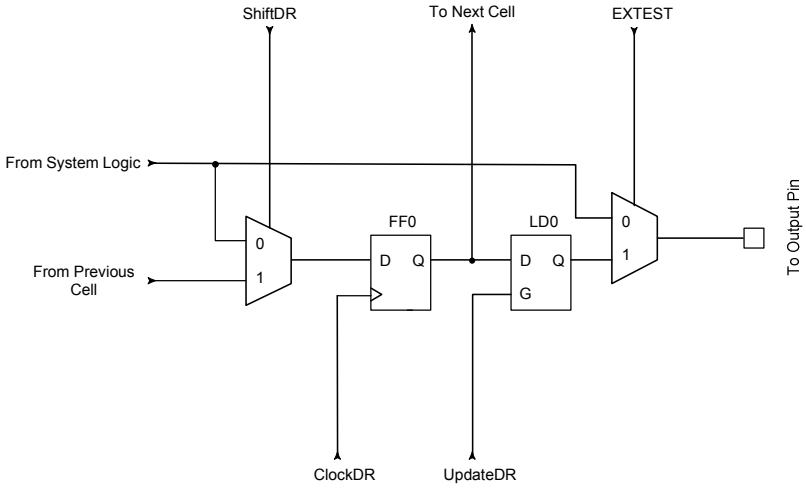
Figure 29-5. Observe-only Cell



29.5.3 Scanning the RSTON Pin

For the low-active reset output pin RSTON a boundary-scan cell as shown in Figure 29-6 below is inserted.

Figure 29-6. Boundary-scan Cell for Output Pins without Pull-up Function



29.6 Boundary-scan Related Register in I/O Memory

29.6.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	JTD								MCUCR
Read/Write	RW								
Initial Value	0								

The MCU Control Register contains control bits for general Microcontroller Unit functions.

- **Bit 7 – JTD - JTAG Interface Disable**

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

29.6.2 MCUSR – MCU Status Register

Bit	7	6	5	4	3	2	1	0	
$34 ($54)				JTRF					MCUSR
Read/Write				RW					
Initial Value				0					

The MCU Status Register provides information on which reset source caused an MCU reset.

- **Bit 4 – JTRF - JTAG Reset Flag**

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

29.7 Boundary-scan Description Language Files

Boundary-scan Description Language (BSDL) files describe Boundary-scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-scan Data Register are included in this description. BSDL files are available for ATmega128RFA1.

29.8 ATmega128RFA1 Boundary-scan Order

Table 29-1 on page 449 shows the Scan order between TDI and TDO when the Boundary-scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order. In Figure 29-3 on page 445, PXn. Data corresponds to FF0, PXn. Control corresponds to FF1, PXn. Bit 4, 5, 6 and 7 of Port F is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled.

Table 29-1. ATmega128RFA1 Boundary-Scan Order

Bit Number	Signal Name	Module
0	PF1.Control	Port F
1	PF1.Data	
2	PF0.Control	
3	PF0.Data	
4	PE7.Control	Port E
5	PE7.Data	
6	PE6.Control	
7	PE6.Data	
8	PE5.Control	
9	PE5.Data	
10	PE4.Control	
11	PE4.Data	
12	PE3.Control	
13	PE3.Data	
14	PE2.Control	
15	PE2.Data	
16	PE1.Control	
17	PE1.Data	
18	PE0.Control	
19	PE0.Data	
20	PB7.Control	Port B
21	PB7.Data	
22	PB6.Control	
23	PB6.Data	
24	PB5.Control	
25	PB5.Data	
26	PB4.Control	
27	PB4.Data	
28	PB3.Control	
29	PB3.Data	
30	PB2.Control	
31	PB2.Data	
32	PB1.Control	
33	PB1.Data	
34	PB0.Control	
35	PB0.Data	
36	CLKI.Data	Clock Input (Input Only)
37	PD7.Control	Port D
38	PD7.Data	
39	PD6.Control	
40	PD6.Data	
41	PD5.Control	
42	PD5.Data	
43	PD4.Control	
44	PD4.Data	
45	PD3.Control	
46	PD3.Data	
47	PD2.Control	
48	PD2.Data	
49	PD1.Control	
50	PD1.Data	
51	PD0.Control	
52	PD0.Data	
53	PG5.Control	Port G
54	PG5.Data	
55	PG4.Control	
56	PG4.Data	
57	PG3.Control	
58	PG3.Data	
59	PG2.Control	
60	PG2.Data	
61	PG1.Control	
62	PG1.Data	
63	PG0.Control	
64	PG0.Data	
65	RSTON.Data	Reset Logic Output (Output Only without Pull-up)
66	RSTT.Data	Reset Logic (Observe Only)
67	TST.Data	Test and Programming Mode Enable (Observe Only)
68	PF3.Control	Port F
69	PF3.Data	
70	PF2.Control	
71	PF2.Data	

30 Boot Loader Support – Read-While-Write Self-Programming

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write that (program) code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself (including erasing) from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

30.1 Features

- **Read-While-Write Self-Programming**
- **Flexible Boot Memory Size**
- **High Security (Separate Boot Lock Bits for a Flexible Protection)**
- **Separate Fuse to Select Reset Vector**
- **Optimized Page(1) Size**
- **Code Efficient Algorithm**
- **Efficient Read-Modify-Write Support**

Note: 1. A page is a section in the Flash consisting of several bytes (see "Table 31-7" on page 467) used during programming. The page organization does not affect normal operation.

30.2 Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections: the Application section and the Boot Loader section (see Figure 30-2 on page 452). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 30-7 on page 461 and Figure 30-2 on page 452. These two sections can have different level of protection since they have different sets of Lock bits.

30.2.1 Application Section

The Application section is the region of the Flash that is used for storing the application code. The protection level for the Application section can be selected by the application Boot Lock bits (Boot Lock bits 0, BLB0), see Table 31-2 on page 464. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

30.2.2 BLS – Boot Loader Section

While the Application section is used for storing the application code, the Boot Loader software must be located in the BLS. The SPM instruction can only initiate programming when executed from the BLS. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1, BLB1), see Table 31-2 on page 464.

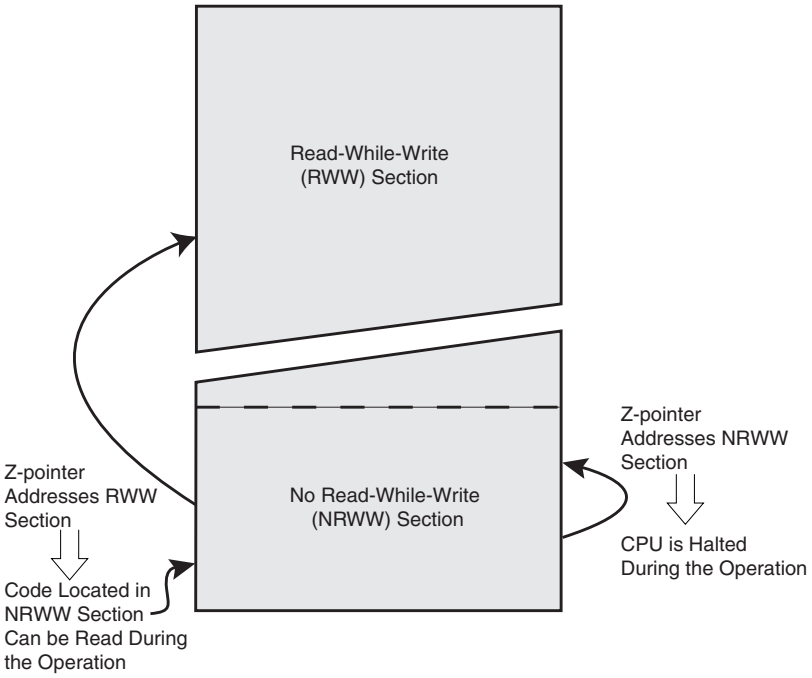
30.3 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on the address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 30-1 on page 452 and Figure 30-1 below. The main differences between the two sections are:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax “Read-While-Write section” refers to the section that is being programmed (erased or written) and not to the section that actually is being read during a Boot Loader software update.

Figure 30-1. Read-While-Write vs. No Read-While-Write



30.3.1 RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by load program memory, call, or jump instructions or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a

programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See "SPMCSR – Store Program Memory Control Register" on page 462 for details on how to clear RWWSB.

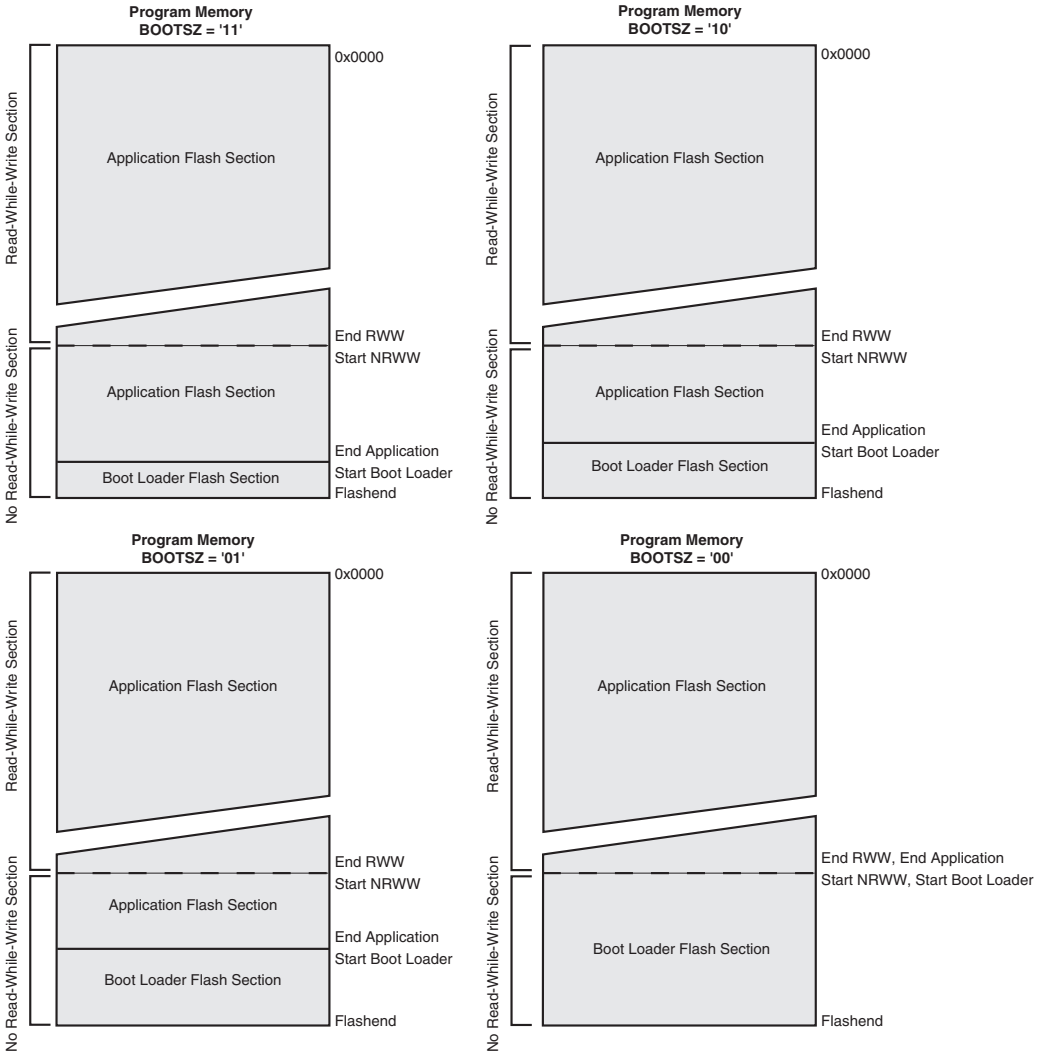
30.3.2 NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire Page Erase or Page Write operation.

Table 30-1. Read-While-Write Features

Which Section does the Z-pointer Address during the Programming?	Which Section can be Read during Programming?	CPU Halted?	Read-While-Write Supported?
RWW Section	NRWW Section	No	Yes
NRWW Section	None	Yes	No

Figure 30-2. Memory Sections



Note: 1. The parameters in the figure above are given in Table 30-7 on page 461.

30.4 Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU.
- To protect only the Boot Loader Flash section from a software update by the MCU.
- To protect only the Application Flash section from a software update by the MCU.
- Allow software update in the entire Flash.

See Table 31-2 on page 464 for further details. The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 1) does not control reading nor writing by (E)LPM/SPM, if it is attempted.

30.4.1 Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

Table 30-2. Boot Reset Fuse(1)

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 30-7 on page 461)

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

30.5 Addressing the Flash During Self-Programming

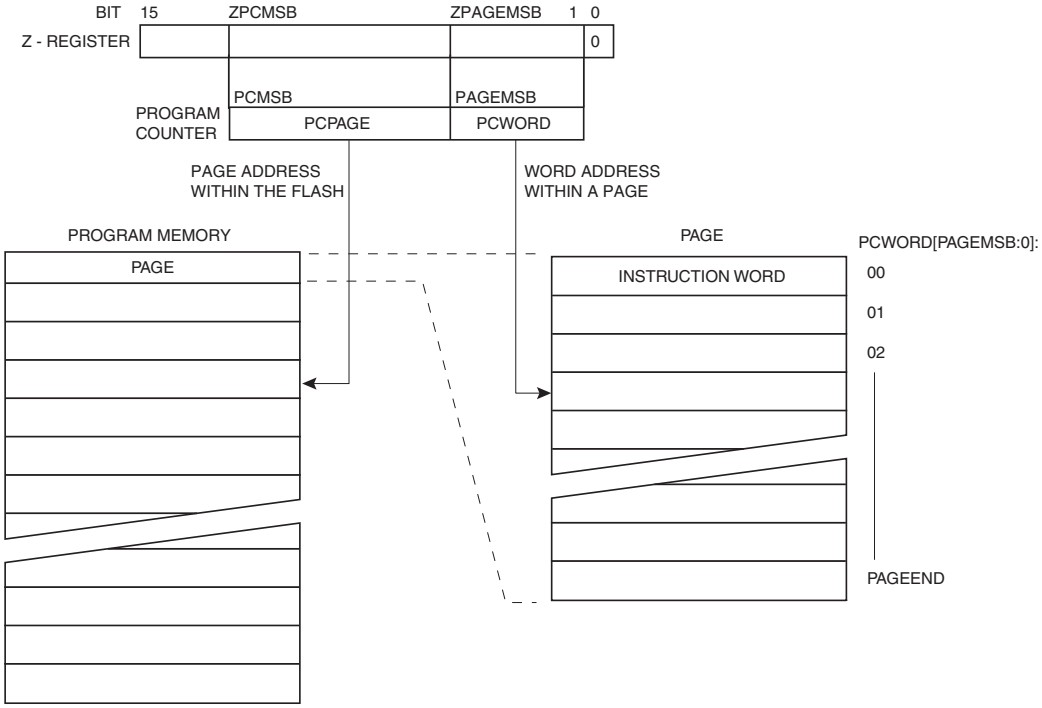
The Z-pointer is used to address the SPM commands. The Z pointer consists of the Z-registers ZL and ZH in the register file, and RAMPZ in the I/O space. The number of bits actually used is implementation dependent. Note that the RAMPZ register is only implemented when the program space is larger than 64K bytes.

Bit	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8
RAMPZ							**RAMPZ1**	**RAMPZ0**
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 31-7" on page 467), 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 is shown in Figure 30-3 below. 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 (E)LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also bit Z0 of the Z-pointer is used.

Figure 30-3. Addressing the Flash during SPM



Note: 1. The different variables used in Figure 30-3 above are listed in Table 30-6 on page 461.

30.6 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;

Alternative 2, fill the buffer after Page Erase

- Perform a Page Erase,

- Fill temporary page buffer,
- Perform a Page Write;

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page. For an assembly code example see "Simple Assembly Code Example for a Boot Loader" on page 458.

30.6.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “X0000011” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the Page Erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

30.6.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will be auto-erased after a Page Write operation or by writing the RWWSRE bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded is still buffered.

30.6.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “X0000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the Page Write.
- Page Write to the NRWW section: The CPU is halted during the operation.

30.6.4 Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCSR is cleared. This means that the interrupt can be used instead of polling the SPMCSR Register in software. When using the SPM interrupt, the Interrupt Vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading. How to move the interrupts is described in "Interrupts" on page 211.

30.6.5 Consideration While Updating BLS

Special care must be taken if the user allows the Boot Loader section to be updated by leaving Boot Lock bit11 un-programmed. An accidental write to the Boot Loader itself can corrupt the entire Boot Loader, and further software updates might be impossible. If it is not necessary to change the Boot Loader software itself, it is recommended to program the Boot Lock bit11 to protect the Boot Loader software from any internal software changes.

30.6.6 Prevent Reading the RWW Section During Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the self programming operation. The RWWSB in the SPMCSR will be set as long as the RWW section is busy. During Self-Programming the Interrupt Vector table should be moved to the BLS as described in "Interrupts" on page 211, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See "Simple Assembly Code Example for a Boot Loader" on page 458 for an example.

30.6.7 Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits and general Lock bits, write the desired data to R0, write “X0001001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	LB2	LB1

See Table 31-2 on page 464 for how the different settings of the Boot Loader bits affect the Flash access.

If bits 5:0 in R0 are cleared (zero), the corresponding Lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SPMEN are set in SPMCSR. The Z-pointer is don’t care during this operation, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the Lock bits). For future compatibility it is also recommended to set bits 7 and 6 in R0 to “1” when writing the Lock bits. When programming the Lock bits the entire Flash can be read during the operation.

30.6.8 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Signature Row, Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

30.6.9 Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the BLBSET and SPMEN bits in SPMCSR. When an (E)LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no (E)LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SPMEN are cleared, (E)LPM will work as described in the Instruction Set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low byte is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the BLBSET and SPMEN bits in SPMCSR. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown on the next page. Refer to (see "Table 31-5" on page 466) for a detailed description and mapping of the Fuse Low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, load 0x0003 in the Z-pointer for reading the Fuse High byte. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Fuse High byte (FHB) will be loaded in the destination register as shown below. Refer to "Table 31-4" on page 465 for detailed description and mapping of the Fuse High byte.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Load 0x0002 in the Z-pointer for reading the Extended Fuse byte. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Extended Fuse byte (EFB) will be loaded in the destination register as shown below. Refer to Table 31-3 on page 465 for detailed description and mapping of the Extended Fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	-	-	-	EFB2	EFB1	EFB0

Fuse and Lock bits that are programmed will be read as zero. Fuse and Lock bits that are un-programmed will be read as one.

30.6.10 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 30-3 below and set the SIGRD and SPMEN bits in SPMCSR. When a LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the Signature Row or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction Set Manual. The Signature Row cannot be read during an EEPROM write/erase operation.

Table 30-3. Signature Row Addressing

Signature Byte	Z-Pointer Address
Device Signature Byte 1	0x0000
Device Signature Byte 2	0x0002
Device Signature Byte 3	0x0004
RC Oscillator Calibration Byte	0x0001

Note: 2. All other addresses are reserved for future use.

30.6.11 Preventing Flash Corruption

During periods of V_{DEVDD}<1.8V, the Flash program can be corrupted because the supply voltage is too low for the CPU and the Flash to operate properly. These issues are the same as for board level systems using Flash, and the same design solutions should be applied.

A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
2. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{DEVDD} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed under the condition that the power supply voltage is sufficient.
3. Keep the AVR core in Power-down sleep mode during periods of low V_{DEVDD}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

30.6.12 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 30-4 below shows the typical programming time for Flash accesses from the CPU.

Table 30-4. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms
Flash write (Page Erase)	7.3 ms	8.9 ms

30.6.13 Simple Assembly Code Example for a Boot Loader

Assembly Code Example[1]

```
;-the routine writes one page of data from RAM to Flash
; the first data location in RAM is pointed to by the Y pointer
; the first data location in Flash is pointed to by the Z-pointer
;-error handling is not included
;-the routine must be placed inside the Boot space
; (at least the Do_spm sub routine). Only code inside NRWW section
; can be read during Self-Programming (Page Erase and Page Write).
;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
; loophi (r25), spmcrval (r20)
; storing and restoring of registers is not included in the routine
; register usage can be optimized at the expense of code size
;-It is assumed that either the interrupt table is moved to the
; Boot loader section or that the interrupts are disabled.
```

Assembly Code Example[1]

```
.equ PAGESIZEB=PAGESIZE*2 ;PAGESIZEB is page in BYTES, not words
.org SMALLBOOTSTART
Write_page:
  ; Page Erase
  ldi spmcrval, (1<<PGERS) | (1<<SPMEN)
  call Do_spm
; re-enable the RWW section
  ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call Do_spm
  ; transfer data from RAM to Flash page buffer
  ldi looplo, low(PAGESIZEB)   ;init loop variable
  ldi loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
Wrloop:
  ld r0, Y+
  ld r1, Y+
  ldi spmcrval, (1<<SPMEN)
  call Do_spm
  adiw ZH:ZL, 2
  sbiw loophi:looplo, 2        ;use subi for PAGESIZEB<=256
  brne Wrloop
  ; execute Page Write
  subi ZL, low(PAGESIZEB)      ;restore pointer
  sbci ZH, high(PAGESIZEB)     ;not required for PAGESIZEB<=256
  ldi spmcrval, (1<<PGWRT) | (1<<SPMEN)
  call Do_spm
  ; re-enable the RWW section
  ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call Do_spm
  ; read back and check, optional
  ldi looplo, low(PAGESIZEB)      ;init loop variable
  ldi loophi, high(PAGESIZEB)     ;not required for PAGESIZEB<=256
  subi YL, low(PAGESIZEB)         ;restore pointer
  sbci YH, high(PAGESIZEB)
Rdloop:
  elpm r0, Z+
  ld r1, Y+
  cpse r0, r1
  jmp Error
  sbiw loophi:looplo, 1           ;use subi for PAGESIZEB<=256
  brne Rdloop
  ; return to RWW section
  ; verify that RWW section is safe to read
Return:
  in temp1, SPMCSR
```

Assembly Code Example[1]

```
    ; If RWWSB is set, the RWW section is not ready yet
    sbrs temp1, RWWSB
    ret
    ; re-enable the RWW section
    ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
    call Do_spm
    rjmp Return
Do_spm:
    ; check for previous SPM complete
Wait_spm:
    in temp1, SPMCSR
    sbrc temp1, SPMEN
    rjmp Wait_spm
    ; input: spmcrval determines SPM action
    ; disable interrupts if enabled, store status
    in temp2, SREG
    cli
    ; check that no EEPROM write access is present
Wait_ee:
    sbic EECR, EEPE
    rjmp Wait_ee
    ; SPM timed sequence
    out SPMCSR, spmcrval
    spm
    ; restore SREG (to enable interrupts if originally enabled)
    out SREG, temp2
    ret
```

Notes: 1. See "About Code Examples" on page 7.

30.6.14 Boot Loader Parameters for 128kByte of Flash Memory

In Table 30-7 on page 461 through Table 30-6 on page 461, the parameters used in the description of the Self-Programming are given.

Table 30-5. Read-While-Write Limit with 128kByte of Flash Memory

Section[1]	Pages	Address
Read-While-Write section (RWW)	480	0x0000 – 0xEFFF
No Read-While-Write section (NRWW)	32	0xF000 – 0xFFFF

Note: 1. For details about these two sections see "NRWW – No Read-While-Write Section" on page 452 .

Table 30-6. Explanation of different variables used in Figure 30-3 on page 454 and the mapping to the Z-pointer for 128kByte of Flash Memory

Variable	Value	Corresponding Z-value[2]	Description[1]
PCMSB	15		Most significant bit in the Program Counter. (The Program Counter is 16 bits PC[15:0])
PAGEMSB	6		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		Z16[3]	Bit in Z-pointer that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z7	Bit in Z-pointer that is mapped to PCMSB. Because Z0 is not used; the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[15:7]	Z16[3]: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)

Notes:
1. Z0: should be zero for all SPM commands, byte select for the (E)LPM instruction.
2. See "Addressing the Flash During Self-Programming" on page 453 for details about the use of Z-pointer during Self-Programming.
3. The Z-register is only 16 bits wide. Bit 16 is located in the RAMPZ register in the I/O map.

Table 30-7. Boot Size Configuration with 128kByte of Flash Memory[1]

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	512 words	4	0x0000 – 0xFDFF	0xFE00 – 0xFFFF	0xFDFF	0xFE00
1	0	1024 words	8	0x0000 – 0xFBFF	0xFC00 – 0xFFFF	0xFBFF	0xFC00
0	1	2048 words	16	0x0000 – 0xF7FF	0xF800 – 0xFFFF	0xF7FF	0xF800
0	0	4096 words	32	0x0000 – 0xEFFF	0xF000 – 0xFFFF	0xEFFF	0xF000

Note:
1. The different BOOTSZ Fuse configurations are shown in Figure 30-2 on page 452.

30.7 Register Description

30.7.1 SPMCSR – Store Program Memory Control Register

Bit	7	6	5	4	3	2	1	0	
$37 ($57)	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	RW	R	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Store Program Memory Control Register contains the control bits needed to control the Boot Loader operations. Note: Only one SPM instruction should be active at any time.

- **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 SPMCR 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. A 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, 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. A LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCR register, will read either the Lock-bits or the Fuse bits (depending on Z0 in the Z pointer) into the destination register.

- **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 - Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLB-SET, 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 remain 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.

30.7.2 NEMCR – Flash Extended-Mode Control-Register

Bit	7	6	5	4	3	2	1	0	
NA ($75)	**Resx7**	**ENEAM**	**AEAM1**	**AEAM0**	**Resx3**	**Resx2**	**Resx1**	**Resx0**	**NEMCR**
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	1	0	1	0	

The Flash Extended-Mode Control-Register handles the extended address-mode of the extra rows.

- **Bit 7 – Resx7 - Reserved**
- **Bit 6 – ENEAM - Enable Extended Address Mode for Extra Rows**

When active high, the extended address mode of the extra rows is enabled. The address is decoded from bits AEAM1:0 of this register.

- **Bit 5:4 – AEAM1:0 - Address for Extended Address Mode of Extra Rows**

These bits are only used when bit ENEAM of this register is set high. Then AEAM1:0 are used to decode the addresses of the extra rows. A value of 0 decodes the default factory row that is also accessible when the extended address mode is deactivated.

Table 30-8 AEAM Register Bits

Register Bits	Value	Description
AEAM1:0	0	Factory Row
	1	User Row 1
	2	User Row 2
	3	User Row 3

- **Bit 3:0 – Resx3:0 - Reserved**

31 Memory Programming

31.1 Program And Data Memory Lock Bits

The ATmega128RFA1 provides six Lock bits which can be left un-programmed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 31-2 below. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 31-1. Lock Bit Byte (1)

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

Note: 1. “1” means un-programmed, “0” means programmed.

Table 31-2. Lock Bit Protection Modes (1)(2)

Memory Lock Bits			Protection Type
LB Mode	**LB2**	**LB1**	
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)
BLB0 Mode	**BL02**	**BL01**	
1	1	1	No restrictions for SPM or (E)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 (E)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	(E)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.

Memory Lock Bits			Protection Type
BLB1 Mode	BL12	BL11	
1	1	1	No restrictions for SPM or (E)LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and (E)LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	(E)LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

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

31.2 Fuse Bits

The ATmega128RFA1 has three Fuse bytes. Table 31-3 below – Table 31-5 on page 466 describe briefly the functionality of all the fuses and how they are mapped into the Fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 31-3. Extended Fuse Byte

Fuse Low Byte	Bit No	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	3	–	1
BODLEVEL2(1)	2	Brown-out Detector trigger level	1 (un-programmed)
BODLEVEL1(1)	1	Brown-out Detector trigger level	1 (un-programmed)
BODLEVEL0(1)	0	Brown-out Detector trigger level	1 (un-programmed)

Notes: 1. See Table 34-23 on page 503 for BODLEVEL Fuse decoding.

Table 31-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
OCDEN(4)	7	Enable On-chip debugging (OCD)	1 (un-programmed, OCD disabled)
JTAGEN	6	Enable JTAG interface	0 (programmed, JTAG enabled)
SPIEN(1)	5	Enable Serial Program and Data Downloading (SPI)	0 (programmed, SPI programming enabled)
WDTON(3)	4	Watchdog Timer always on	1 (un-programmed)

Fuse High Byte	Bit No	Description	Default Value
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (un-programmed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 30-7 on page 461 for details)	0 (programmed)(2)
BOOTSZ0	1	Select Boot Size (see Table 30-7 on page 461for details)	0 (programmed) (2)
BOOTRST	0	Select Reset Vector	1 (un-programmed)

Notes:
1. The SPIEN Fuse is not accessible in serial programming mode.
2. The default value of BOOTSZ1:0 results in maximum Boot Size. See Table 30-7 on page 461 for details.
3. See "WDTCSR – Watchdog Timer Control Register" on page 183 for details.
4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.

Table 31-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8(4)	7	Divide clock by 8	0 (programmed)
CKOUT(3)	6	Clock output	1 (un-programmed)
SUT1	5	Select start-up time	1 (un-programmed)(1)
SUT0	4	Select start-up time	0 (programmed) (1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed) (2)
CKSEL1	1	Select Clock source	1 (un-programmed) (2)
CKSEL0	0	Select Clock source	0 (programmed) (2)

Notes:
1. The default value of SUT1:0 results in maximum start-up time for the default clock source. See "System Control and Reset" on page 176 for details.
2. The default setting of CKSEL3:0 results in internal RC Oscillator @ 8 MHz. See "Table 11-1" on page 148 for details.
3. The CKOUT Fuse allows the system clock to be output on PORTE7. See "Clock Output Buffer" on page 152 for details.
4. See "System Clock Prescaler" on page 152 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

31.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

31.3 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked.

The three bytes reside in a separate address space. For the ATmega128RFA1 the signature bytes are given in Table 31-6 below. Accessing the signature bytes from software is described in section "Reading the Signature Row from Software" on page 457.

Table 31-6. Device and JTAG ID

Part	Signature Byte Number			JTAG	
	0	1	2	Part Number	Manufacturer ID
ATmega128RFA1	0x1E	0xA7	0x01	0xA701	0x1F

31.4 Calibration Byte

The ATmega128RFA1 has a byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

31.5 Page Size

Table 31-7. Number of Words in a Page and Number of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
64k words (128k bytes)	128 words	PC[6:0]	512	PC[15:7]	15

Table 31-8. Number of Bytes in a Page and Number of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
4k bytes	8 bytes	EEA[2:0]	512	EEA[11:3]	11

31.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega128RFA1.

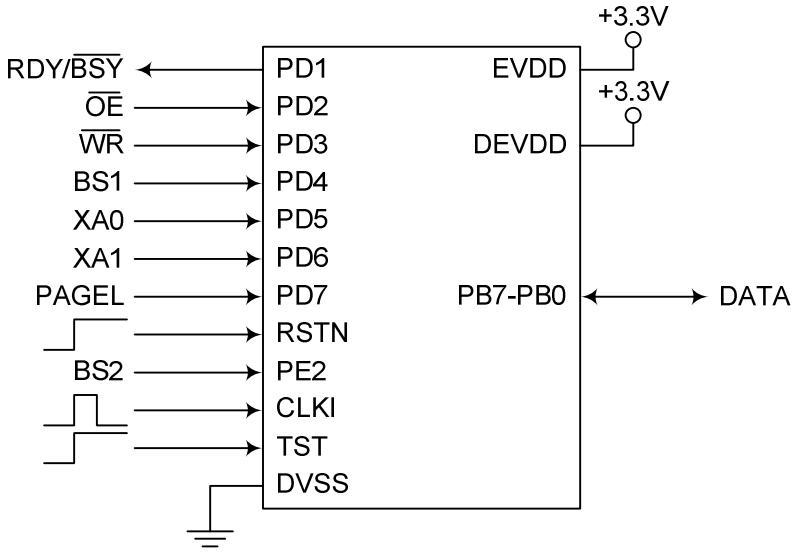
31.6.1 Signal Names

In this section, some pins of the ATmega128RFA1 are referenced by signal names describing their functionality during parallel programming; see Figure 31-1 on page 468 and Table 31-9 on page 468. Pins not described in this table are referenced by their default pin names.

The XA1/XA0 pins determine the action executed when the CLKI pin is given a positive pulse. The bit coding is shown in Table 31-12 on page 469.

When pulsing $\overline{WR}$ or $\overline{OE}$ or, the command loaded determines the action executed. The different commands are shown in Table 31-13 on page 469.

Figure 31-1. Parallel Programming [1]



Note: 1. Unused Pins should be left floating.

Table 31-9. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command.
$\overline{OE}$	PD2	I	Output Enable (Active low).
$\overline{WR}$	PD3	I	Write Pulse (Active low).
BS1	PD4	I	Byte Select 1.
XA0	PD5	I	XTAL Action Bit 0.
XA1	PD6	I	XTAL Action Bit 1.
PAGEL	PD7	I	Program Memory and EEPROM data Page Load.
BS2	PE2	I	Byte Select 2.
DATA	PB7-0	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low).

Table 31-10. BS2 and BS1 Encoding

BS2	BS1	Flash / EEPROM Address	Flash Data Loading / Reading	Fuse Programming	Reading Fuse and Lock Bits
0	0	Low Byte	Low Byte	Low Byte	Fuse Low Byte
0	1	High Byte	High Byte	High Byte	Lock Bits
1	0	Extended High Byte	Reserved	Extended Byte	Extended Fuse Byte
1	1	Reserved	Reserved	Reserved	Fuse High Byte

Table 31-11. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 31-12. XA1 and XA0 Encoding

XA1	XA0	Action when CLKI is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS2 and BS1).
0	1	Load Data (High or Low data byte for Flash determined by BS1).
1	0	Load Command.
1	1	No Action, Idle.

Table 31-13. Command Byte Bit Encoding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

31.7 Parallel Programming

Pulses of CLKI and in the following command sequences are assumed to be at least 250 ns wide unless otherwise noted.

31.7.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 3.3V between DEVDD and DVSS.
2. Set RSTN to 0 and TST to 0.
3. Set the Prog_enable pins listed in Table 31-11 above to “0000” and wait at least 100ns.
4. Set TST to 1. TST can be set high any time before but not after the rising edge of RSTN (t_{TSTRNH}).
5. Set RSTN to 1. Any activity on Prog_enable pins within 100 ns after RSTN is set to 1 will cause the device to fail entering programming mode.
6. Wait at least 50 µs before sending a command.

31.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

31.7.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM [1] memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: 1. The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give CLKI a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/ $\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

31.7.4 Programming the Flash

The Flash is organized in pages; see Table 31-7 on page 467. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give CLKI a positive pulse. This loads the command.

B. Load Address Low byte (Address bits 7:0)

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS2, BS1 to “00”. This selects the address low byte.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give CLKI a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (see Figure 31-3 on page 472 for signal waveforms).

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 31-5 on page 472. Note that if less than eight bits are required to address words in the page (page size < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

G. Load Address High byte (Address bits15:8)

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS2, BS1 to “01”. This selects the address high byte.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the address high byte.

H. Program Page

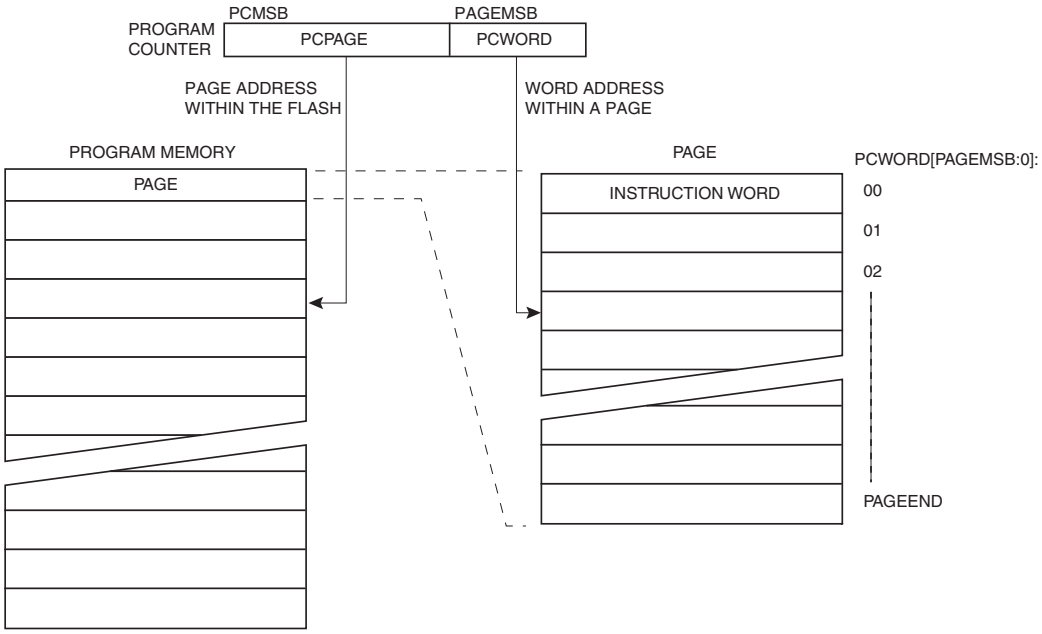
1. Set BS2, BS1 to “00”
2. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high (See Figure 31-3 on page 472 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

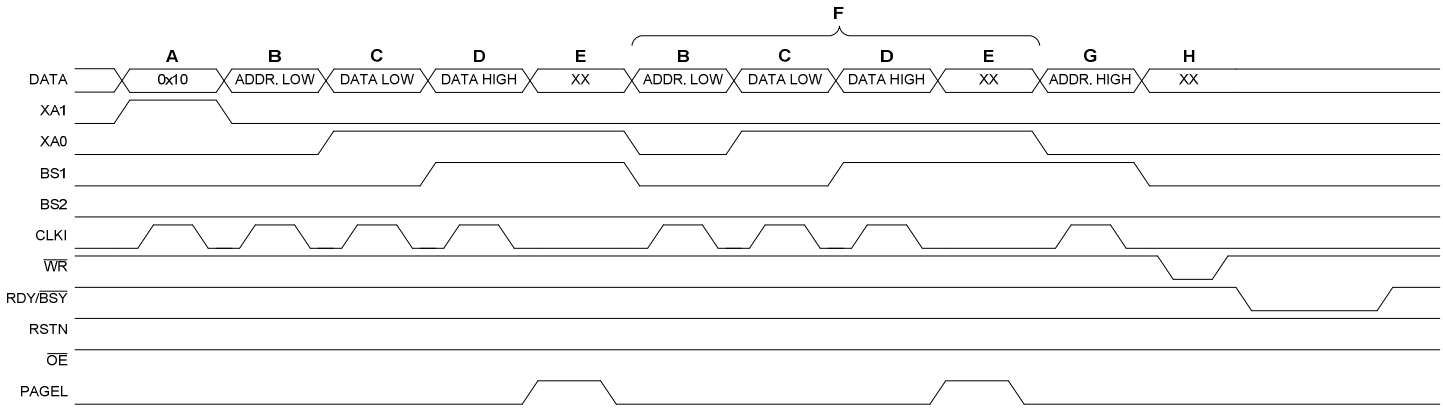
1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give CLKI a positive pulse. This loads the command, and the internal write signals are reset.

Figure 31-5. Addressing the Flash which is Organized in Pages [1]



Note: 1. PCPAGE and PCWORD are listed in Table 31-7 on page 467.

Figure 31-3. Programming the Flash Waveforms [1]



Note: 1. “XX” is don’t care. The letters refer to the programming description above.

31.7.5 Programming the EEPROM

The EEPROM is organized in pages; see Table 31-8 on page 467. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 470 for details on Command, Address and Data loading):

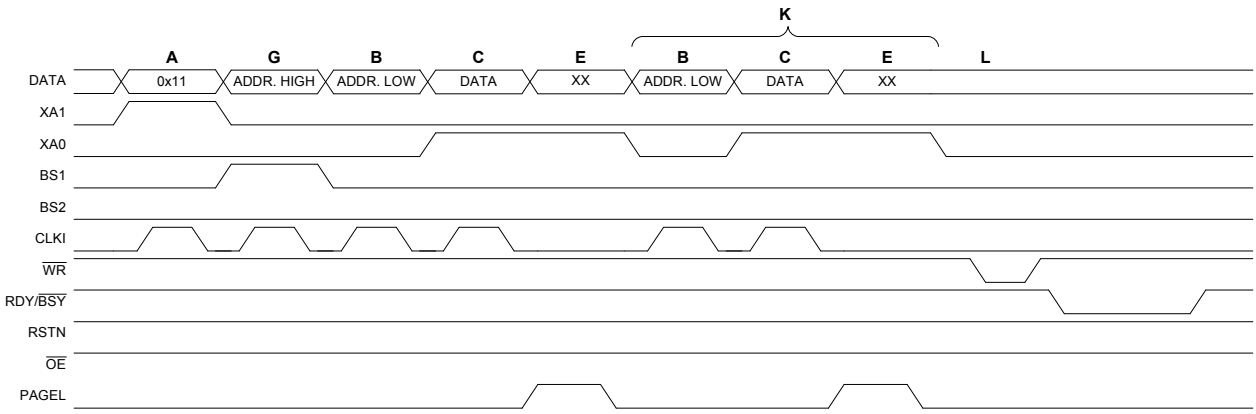
1. A: Load Command “0001 0001”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page

1. Set BS2, BS1 to “00”.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 31-7 below for signal waveforms).

Figure 31-7. Programming the EEPROM Waveforms



31.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 470 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
5. Set BS1 to “1”. The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

31.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 470 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

31.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0100 0000”.

2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

31.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

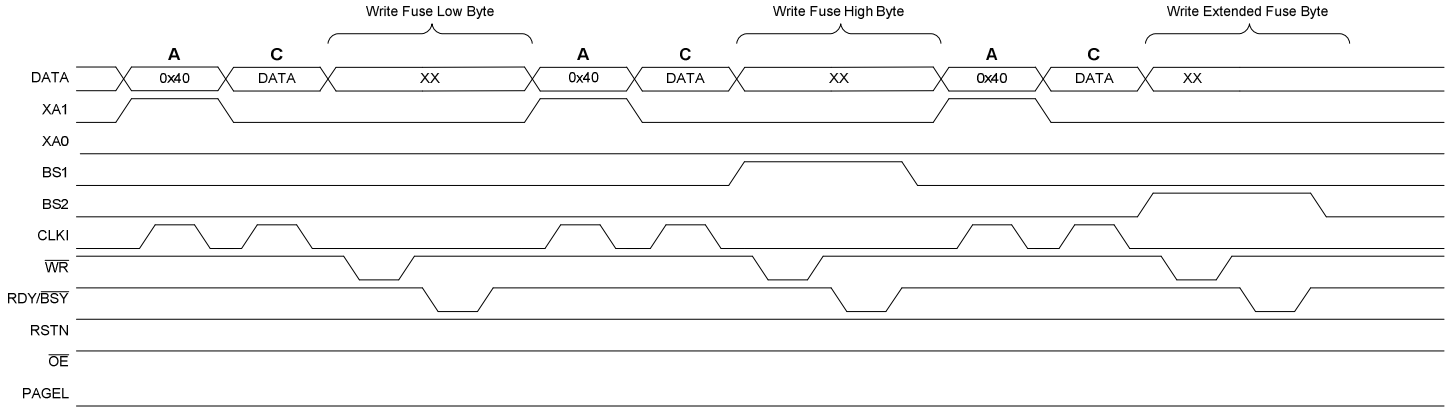
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS2, BS1 to “01”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2, BS1 to “00”. This selects low data byte.

31.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS2, BS1 to “10”. This selects extended data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2, BS1 to “00”. This selects low data byte.

Figure 31-8. Programming the Fuses Waveforms



31.7.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is active (LB1 and LB2 are programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

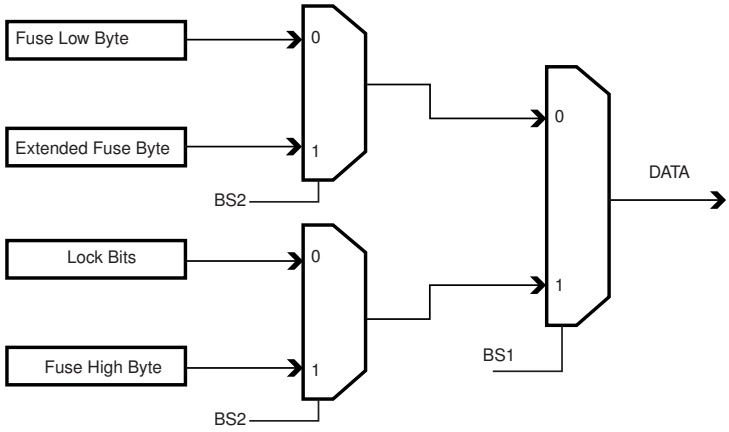
The Lock bits can only be cleared by executing Chip Erase.

31.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, and BS2, BS1 to “00”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, and BS2, BS1 to “11”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set $\overline{OE}$ to “0”, and BS2, BS1 to “10”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, and BS2, BS1 to “01”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 31-9. Mapping between BS1, BS2 and the Fuse and Lock Bits during Read



31.7.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0” and BS to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

31.7.14 Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0” and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

31.7.15 Parallel Programming Characteristics

Figure 31-10. Parallel programming timing including some general timing requirements

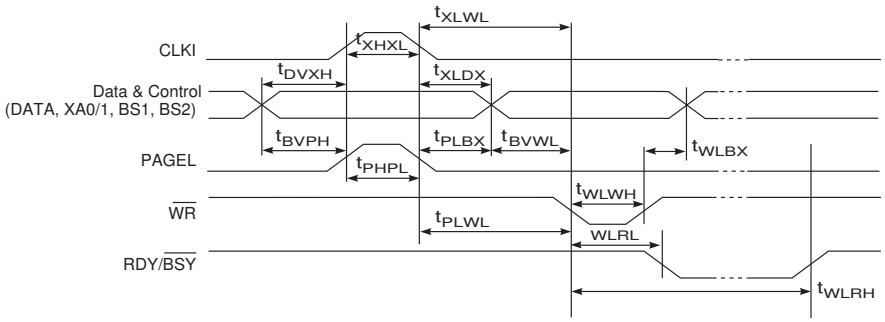
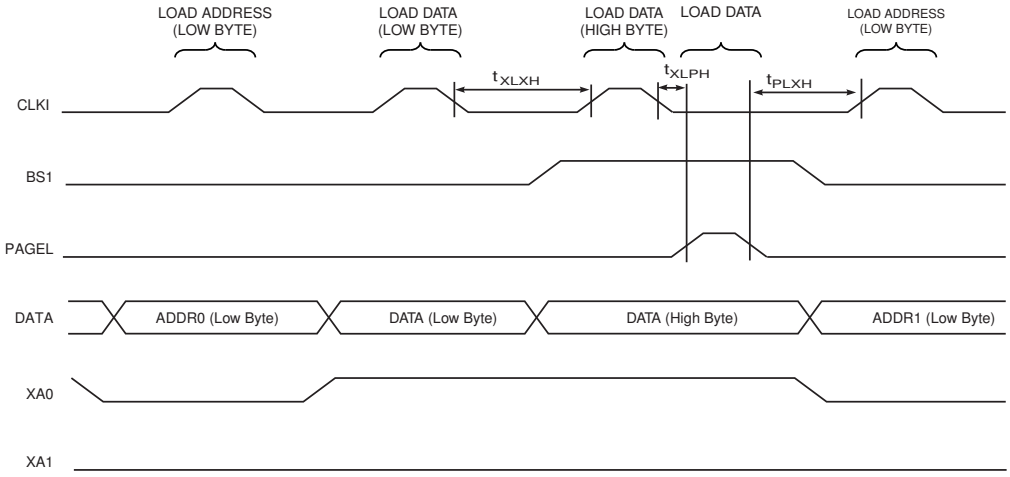
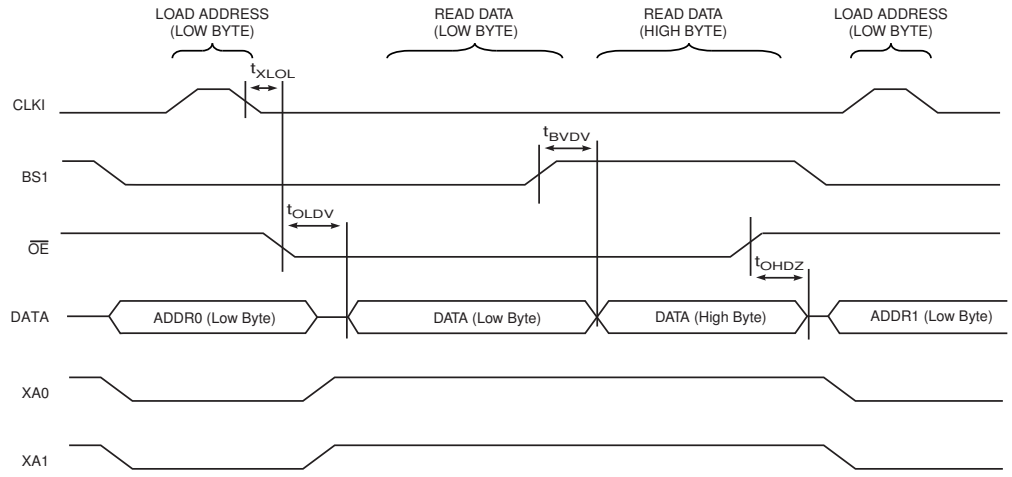


Figure 31-11. Parallel programming loading sequence with timing requirements (1)



Note: 1. The timing requirements shown in Figure 31-10 above (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 31-12. Parallel programming reading sequence (within the same page) with timing requirements (1)



Note: 1. The timing requirements shown in Figure 31-10 above (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 31-14. Parallel Programming Characteristics, V_{DEVDD} = 3.3V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
t_{TSTRNH}	Delay TST High before RSTN High	0			ns
t_{DVXH}	Data and Control Valid before CLKI High	67			ns
t_{XLXH}	CLKI Low to CLKI High	200			ns
t_{XHXL}	CLKI Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after CLKI Low	67			ns
t_{XLWL}	CLKI Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	CLKI Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to CLKI high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS2/1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	12		14.5	ms
t_{XLOL}	CLKI Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

31.8 Serial Downloading

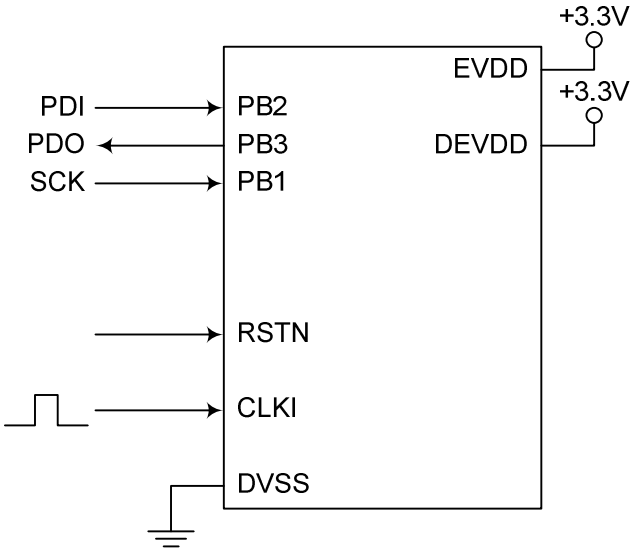
Both the Flash and EEPROM memory arrays can be programmed using a serial programming bus while RSTN is pulled to DVSS. The serial programming interface consists of pins SCK, PDI (input) and PDO (output). After RSTN is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 31-15 below, the pin mapping for serial programming is listed.

31.8.1 Serial Programming Pin Mapping

Table 31-15. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
PDI	PB2	I	Serial Data In
PDO	PB3	O	Serial Data Out
SCK	PB1	I	Serial Clock

Figure 31-13. Serial Programming and Verify [1][2]



Notes:
1. If the device is clocked by the internal Oscillator, it is not required to connect a clock source to the CLKI pin.
2. V_{DEVDD}-0.3V < V_{EVDD} < V_{DEVDD}+0.3V, both V_{EVDD} and V_{DEVDD} must stay in valid supply voltage limits.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz;

High: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz;

31.8.2 Serial Programming Algorithm

When writing serial data to the ATmega128RFA1, data is clocked on the rising edge of SCK.

When reading data from the ATmega128RFA1, data is clocked on the falling edge of SCK. See Figure 31-15 on page 481 for timing details.

To program and verify the ATmega128RFA1 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 31-17 on page 479):

1. Power-up sequence: Apply power between DEVDD and DVSS while RSTN and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, RSTN must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin PDI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or

not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give RSTN a positive pulse and issue a new Programming Enable command.

4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 7 LSB of the address and data together with the Load Program Memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program Memory Page is stored by loading the Write Program Memory Page instruction with the address lines 15:8. If polling (RDY/BSY) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page (see Table 31-16 below). Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte (see Table 31-16 below). In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output PDO.
7. At the end of the programming session, RSTN can be set high to commence normal operation.
8. Power-off sequence (if needed): Set RESET to “1”. Turn DEVDD power off.

Table 31-16. Minimum Wait Delay before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9 ms
$t_{WD\_CHIPERASE}$	14.5 ms

31.8.3 Serial Programming Instruction Set

Table 31-17 below and Figure 31-14 on page 480 describe the Instruction set.

Table 31-17. Serial Programming Instruction Set (4)(5)

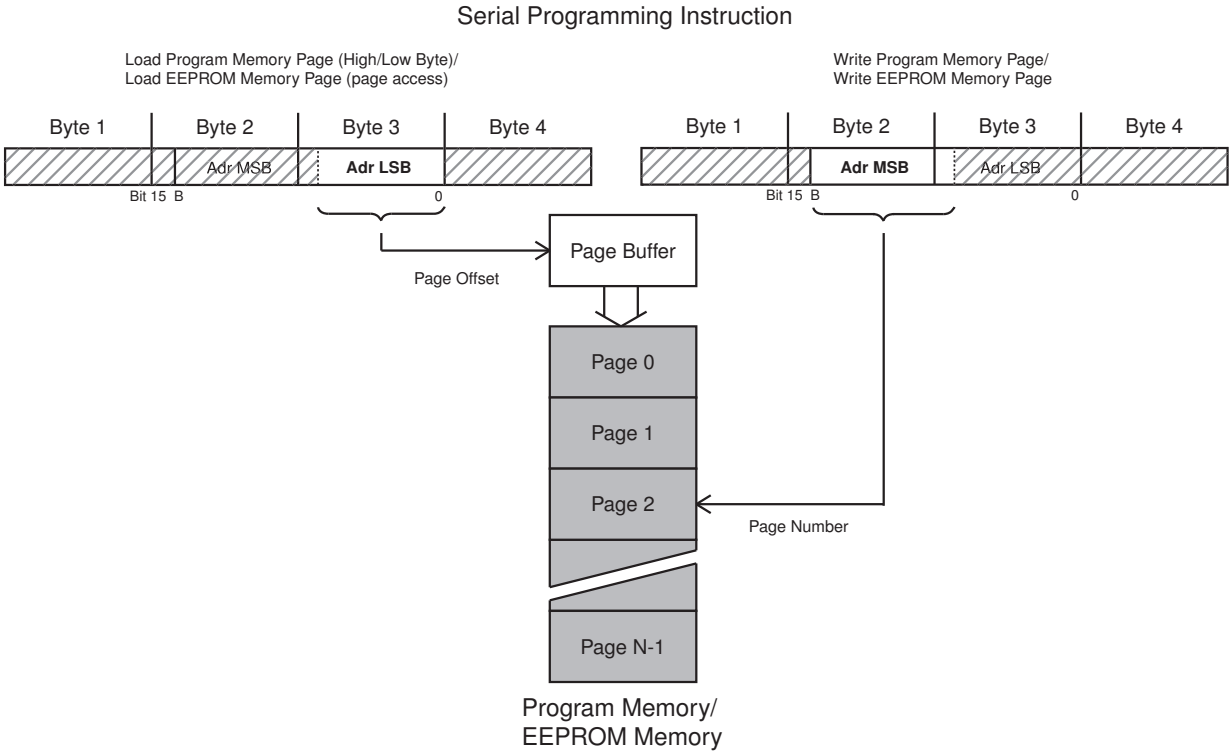
Instruction/Operation	Instruction Format (1)			
	Byte1	Byte2	Byte3	Byte4
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/BSY	$F0	$00	$00	data byte out
Load Instruction				
Load Program Memory Page, High Byte	$48	$00	addr. LSB	high data byte in
Load Program Memory Page, Low Byte	$40	$00	addr. LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	0000 000aa	data byte in
Read Instruction				
Read Program Memory, High byte	$28	addr. MSB	addr. LSB	high data byte out
Read Program Memory, Low byte	$20	addr. MSB	addr. LSB	low data byte out
Read EEPROM Memory	$A0	0000 aaaa	aaaa aaaa	data byte out
Read Lock Bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	0000 000aa	data byte out

	Instruction Format [1]			
Read Fuse Bits	$50	$00	$00	data byte out
Read Fuse High Bits	$58	$08	$00	data byte out
Read Extended Fuse Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions [2][3]				
Write Program Memory Page	$4C	addr. MSB	addr. LSB	$00
Write EEPROM Memory	$C0	0000 aaaa	aaaa aaaa	data byte in
Write EEPROM Memory Page (page access)	$C2	0000 aaaa	aaaa 00	$00
Write Lock Bits	$AC	$E0	$00	data byte in
Write Fuse Bits	$AC	$A0	$00	data byte in
Write Fuse High Bits	$AC	$A8	$00	data byte in
Write Extended Fuse Bits	$AC	$A4	$00	data byte in

Notes:
1. a = address.
2. Bits are programmed ‘0’, un-programmed ‘1’.
3. To ensure future compatibility, unused Fuses and Lock bits should be un-programmed (‘1’).
4. Refer to the corresponding section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
5. See http://www.atmel.com/avr for Application Notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out. Within the same page, the low data byte must be loaded prior to the high data byte. After data is loaded to the page buffer, program the EEPROM page; see Figure 31-14 below.

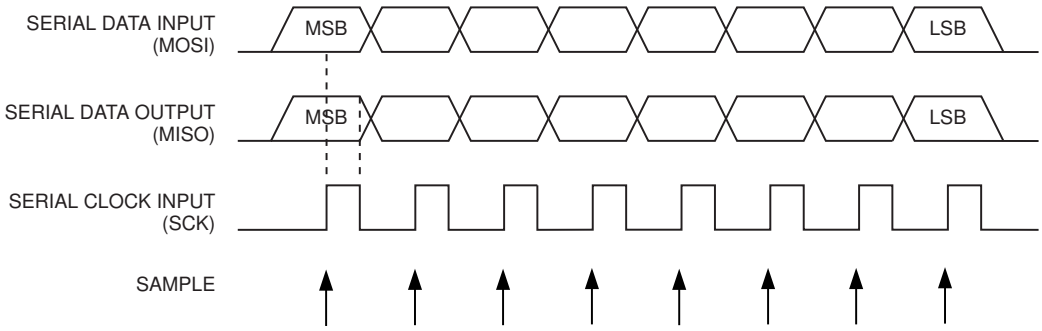
Figure 31-14. Serial Programming Instruction Example



31.8.4 Serial Programming Characteristics

For characteristics of the Serial Programming module see "SPI Timing Characteristics" on page 504.

Figure 31-15. Serial Programming Waveforms



31.9 Programming via the JTAG Interface

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI, and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCR must be cleared. Alternatively, if the JTD bit is set, the external reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using the JTAG pins for Boundary-scan or On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

During programming the clock frequency of the TCK Input must be less than the maximum frequency of the chip. The System Clock Prescaler can not be used to divide the TCK Clock Input into a sufficiently low frequency.

As a definition in this datasheet, the LSB is shifted in and out first of all Shift Registers.

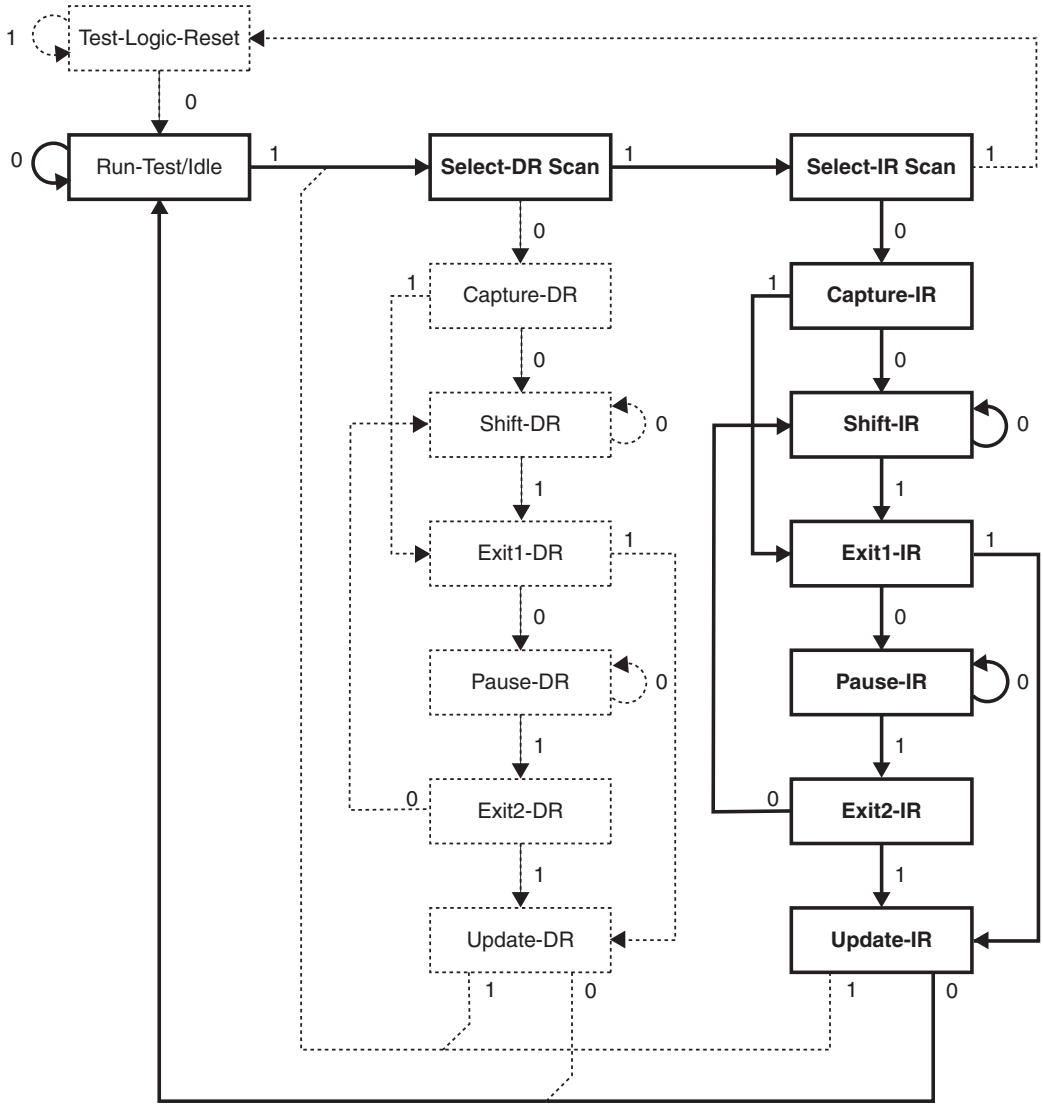
31.9.1 Programming Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. The JTAG instructions useful for programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP-controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences. The state machine sequence for changing the instruction word is shown in Figure 31-16 on page 482.

Figure 31-16. State Machine Sequence for Changing the Instruction Word



31.9.2 AVR_RESET (0xC)

The AVR specific public JTAG instruction is used for setting the AVR device in the Reset mode or taking the device out from the Reset mode. The TAP-controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

31.9.3 PROG_ENABLE (0x4)

The AVR specific public JTAG instruction enables programming via the JTAG port. The 16-bit Programming Enable Register is selected as Data Register. The active states are the following:

- Shift-DR: The programming enable signature is shifted into the Data Register.

- Update-DR: The programming enable signature is compared to the correct value, and Programming mode is entered if the signature is valid.

31.9.4 PROG_COMMANDS (0x5)

The AVR specific public JTAG instruction is used for entering programming commands via the JTAG port. The 15-bit Programming Command Register is selected as Data Register. The active states are the following:

- Capture-DR: The result of the previous command is loaded into the Data Register.
- Shift-DR: The Data Register is shifted by the TCK input, shifting out the result of the previous command and shifting in the new command.
- Update-DR: The programming command is applied to the Flash inputs.
- Run-Test/Idle: One clock cycle is generated, executing the applied command.

31.9.5 PROG_PAGELOAD (0x6)

The AVR specific public JTAG instruction directly loads the Flash data page via the JTAG port. An 8-bit Flash Data Byte Register is selected as the Data Register. This is physically the 8 LSB's of the Programming Command Register. The active states are the following:

- Shift-DR: The Flash Data Byte Register is shifted by the TCK input.
- Update-DR: The content of the Flash Data Byte Register is copied into a temporary register. A write sequence is initiated that within 11 TCK cycles loads the content of the temporary register into the Flash page buffer. The AVR automatically alternates between writing the low and the high byte for each new Update-DR state, starting with the low byte for the first Update-DR encountered after entering the PROG_PAGELOAD command. The Program Counter is pre-incremented before writing the low byte, except for the first written byte. This ensures that the first data is written to the address set up by PROG_COMMANDS, and loading the last location in the page buffer does not make the program counter increment into the next page.

31.9.6 PROG_PAGEREAD (0x7)

The AVR specific public JTAG instruction directly captures the Flash content via the JTAG port. An 8-bit Flash Data Byte Register is selected as the Data Register. This is physically the 8 LSB's of the Programming Command Register. The active states are the following:

- Capture-DR: The content of the selected Flash byte is captured into the Flash Data Byte Register. The AVR automatically alternates between reading the low and the high byte for each new Capture-DR state, starting with the low byte for the first Capture-DR encountered after entering the PROG_PAGEREAD command. The Program Counter is post-incremented after reading each high byte, including the first read byte. This ensures that the first data is captured from the first address set up by PROG_COMMANDS, and reading the last location in the page makes the program counter increment into the next page.
- Shift-DR: The Flash Data Byte Register is shifted by the TCK input.

31.9.7 Data Registers

The Data Registers are selected by the JTAG instruction registers described in section "Programming Specific JTAG Instructions" on page 481. The Data Registers relevant for programming operations are:

- Reset Register

- Programming Enable Register
- Programming Command Register
- Flash Data Byte Register

31.9.8 Reset Register

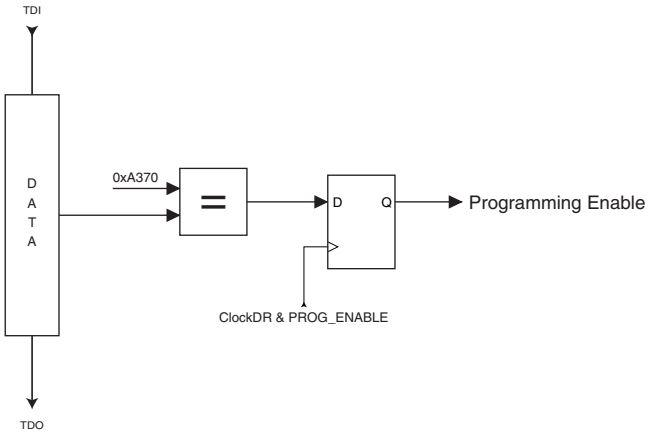
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering Programming mode.

A high value in the Reset Register corresponds to pulling the external reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-out period (refer to "Clock Sources" on page 148) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in "Figure 29-2" on page 443.

31.9.9 Programming Enable Register

The Programming Enable Register is a 16-bit register. The content of this register is compared to the programming enable signature, binary code 1010_0011_0111_0000. When the content of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving Programming mode.

Figure 31-17. Programming Enable Register



31.9.10 Programming Command Register

The Programming Command Register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 31-18 on page 485. The state sequence when shifting in the programming commands is illustrated in Figure 31-19 on page 488.

Figure 31-18. Programming Command Register

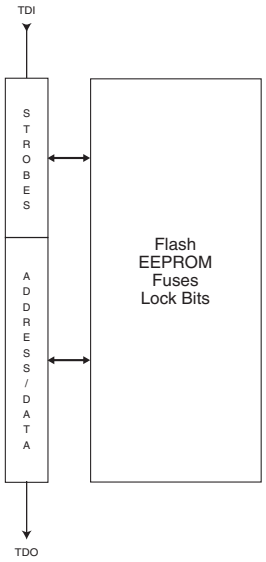


Table 31-18. JTAG Programming Instruction (set **a** = address high bits, **b** = address low bits, **c** = address extended bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care)

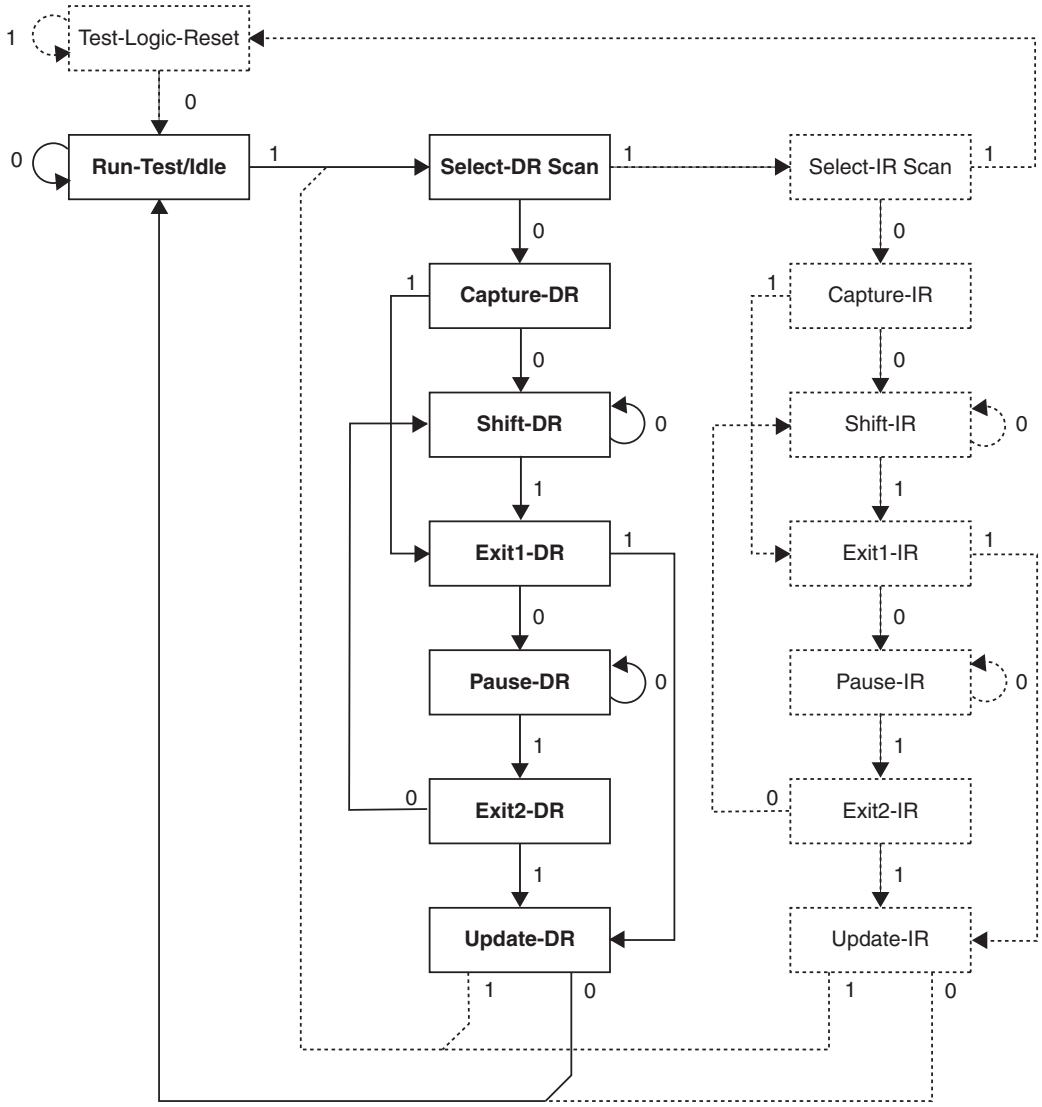
Instruction	TDI Sequence	TDO Sequence	Notes
1a. Chip Erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for Chip Erase Complete	0110011_10000000	xxxxx**o**x_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
2c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
2d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
2e. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2f. Load Data High Byte	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2g. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2i. Poll for Page Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
3c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
3d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
3e. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	 Low byte High byte

Instruction	TDI Sequence	TDO Sequence	Notes
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(10)
4c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write Complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(10)
5c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_**bbbbbbbb** 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(6)
6c. Write Fuse Extended Byte	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6d. Poll for Fuse Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6e. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(7)
6f. Write Fuse High Byte	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Poll for Fuse Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6h. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(8)
6i. Write Fuse Low Byte	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6j. Poll for Fuse Write Complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
7a. Enter Lock Bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte	0010011_11**iiiiii**	xxxxxxx_xxxxxxxx	(4)(9)
7c. Write Lock Bits	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
7d. Poll for Lock Bit Write complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)

Instruction	TDI Sequence	TDO Sequence	Notes
8a. Enter Fuse/Lock Bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Extended Fuse Byte	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(6)(5)
8c. Read Fuse High Byte	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(7)(5)
8d. Read Fuse Low Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(8)(5)
8e. Read Lock Bits	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xx**oooooo**	(9)(5)
8f. Read Fuses and Lock Bits	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	(5) Fuse Ext. byte Fuse High byte Fuse Low byte Lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Notes:
1. This command sequence is not required if the seven MSB's are correctly set by the previous command sequence (which is normally the case).
2. Repeat until o = "1".
3. Set bits to "0" to program the corresponding Fuse, "1" to un-program the Fuse.
4. Set bits to "0" to program the corresponding Lock bit, "1" to leave the Lock bit unchanged.
5. "0" = programmed, "1" = un-programmed.
6. The bit mapping for Fuses Extended byte is listed in Table 31-3 on page 465.
7. The bit mapping for Fuses High byte is listed in Table 31-4 on page 465.
8. The bit mapping for Fuses Low byte is listed in Table 31-5 on page 466
9. The bit mapping for Lock bits byte is listed in Table 31-1 on page 464.
10. Address bits exceeding PCMSB and EEAMSB (Table 31-7 on page 467 and Table 31-8 on page 467) are don't care.
11. All TDI and TDO sequences are represented by binary digits.

Figure 31-19. State Machine Sequence for Changing/Reading the Data Word



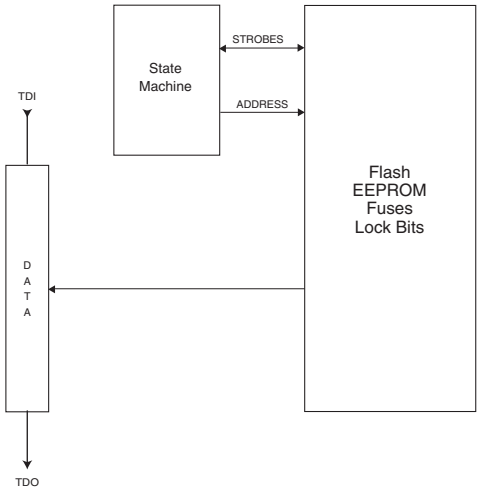
31.9.11 Flash Data Byte Register

The Flash Data Byte Register provides an efficient way to load the entire Flash page buffer before executing Page Write, or to read out/verify the content of the Flash. A state machine sets up the control signals to the Flash and senses the strobe signals from the Flash, thus only the data words need to be shifted in/out.

The Flash Data Byte Register actually consists of the 8-bit scan chain and an 8-bit temporary register. During page load, the Update-DR state copies the content of the scan chain over to the temporary register and initiates a write sequence that within 11 TCK cycles loads the content of the temporary register into the Flash page buffer. The AVR automatically alternates between writing the low and the high byte for each new Update-DR state, starting with the low byte for the first Update-DR encountered after entering the PROG_PAGELOAD command. The Program Counter is pre-incremented before writing the low byte, except for the first written byte. This ensures that the first data is written to the address set up by PROG_COMMANDS, and loading the last location in the page buffer does not make the Program Counter increment into the next page.

During Page Read, the content of the selected Flash byte is captured into the Flash Data Byte Register during the Capture-DR state. The AVR automatically alternates between reading the low and the high byte for each new Capture-DR state, starting with the low byte for the first Capture-DR encountered after entering the PROG_PAGEREAD command. The Program Counter is post-incremented after reading each high byte, including the first read byte. This ensures that the first data is captured from the first address set up by PROG_COMMANDS, and reading the last location in the page makes the program counter increment into the next page.

Figure 31-20. Flash Data Byte Register



The state machine controlling the Flash Data Byte Register is clocked by TCK. During normal operation in which eight bits are shifted for each Flash byte, the clock cycles needed to navigate through the TAP-controller automatically feeds the state machine for the Flash Data Byte Register with sufficient number of clock pulses to complete its operation transparently for the user. However, if too few bits are shifted between each Update-DR state during page load, the TAP-controller should stay in the Run-Test/Idle state for some TCK cycles to ensure that there are at least 11 TCK cycles between each Update-DR state.

31.9.12 Programming Algorithm

All references below of type “1a”, “1b”, and so on, refer to Table 31-18 on page 485.

31.9.13 Entering Programming Mode

1. Enter JTAG instruction AVR_RESET and shift 1 in the Reset Register.
2. Enter instruction PROG_ENABLE and shift 0b1010_0011_0111_0000 in the Programming Enable Register.

31.9.14 Leaving Programming Mode

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by using no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0b0000_0000_0000_0000 in the programming Enable Register.
4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset Register.

31.9.15 Performing Chip Erase

1. Enter JTAG instruction PROG_COMMANDS.
2. Start Chip Erase using programming instruction 1a.
3. Poll for Chip Erase complete using programming instruction 1b, or wait for t_{WLRH_CE} (refer to Table 31-14 on page 477).

31.9.16 Programming the Flash

Before programming the Flash a Chip Erase must be performed, see section "Performing Chip Erase" above.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load High byte of address using programming instruction 2c.
4. Load Low byte of address using programming instruction 2d.
5. Load data using programming instructions 2e, 2f and 2g.
6. Repeat steps 5 and 6 for all instruction words in the page.
7. Write the page using programming instruction 2h.
8. Poll for Flash write complete using programming instruction 2i, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load the page address using programming instructions 2c and 2d. PCWORD (refer to Table 31-7 on page 467) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGELOAD.
5. Load the entire page by shifting in all instruction words in the page byte-by-byte, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. Use Update-DR to copy the contents of the Flash Data Byte Register into the Flash page location and to auto-increment the Program Counter before each new word.
6. Enter JTAG instruction PROG_COMMANDS.
7. Write the page using programming instruction 2h.
8. Poll for Flash write complete using programming instruction 2i, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

31.9.17 Reading the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load address using programming instructions 3c and 3d.
4. Read data using programming instruction 3e.
5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load the page address using programming instructions 3c and 3d. PCWORD (refer to Table 31-7 on page 467) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page (or Flash) by shifting out all instruction words in the page (or Flash), starting with the LSB of the first instruction in the page (Flash) and ending with the MSB of the last instruction in the page (Flash). The Capture-DR state both captures the data from the Flash, and also auto-increments the program counter after each word is read. Note that Capture-DR comes before the shift-DR state. Hence, the first byte which is shifted out contains valid data.
6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

31.9.18 Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed, see "Performing Chip Erase" on page 490.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load High byte of address using programming instruction 4b.
4. Load Low byte of address using programming instruction 4c.
5. Load data using programming instructions 4d and 4e.
6. Repeat steps 4 and 5 for all data bytes in the page.
7. Write the data using programming instruction 4f.
8. Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM.

31.9.19 Reading the EEPROM

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM read using programming instruction 5a.
3. Load address using programming instructions 5b and 5c.
4. Read data using programming instruction 5d.
5. Repeat steps 3 and 4 until all data have been read.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM.

31.9.20 Programming the Fuses

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data high byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse; a “1” will un-program the fuse.
4. Write Fuse High byte using programming instruction 6c.

5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
6. Load data low byte using programming instructions 6e. A “0” will program the fuse, a “1” will un-program the fuse.
7. Write Fuse low byte using programming instruction 6f.
8. Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 31-14 on page 477).

31.9.21 Programming the Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 31-14 on page 477).

31.9.22 Reading the Fuses and Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse/Lock bit read using programming instruction 8a.
3. To read all Fuses and Lock bits, use programming instruction 8e.
   To only read Fuse High byte, use programming instruction 8b.
   To only read Fuse Low byte, use programming instruction 8c.
   To only read Lock bits, use programming instruction 8d.

31.9.23 Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address 0x00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address 0x01 and address 0x02 to read the second and third signature bytes, respectively.

31.9.24 Reading the Calibration Byte

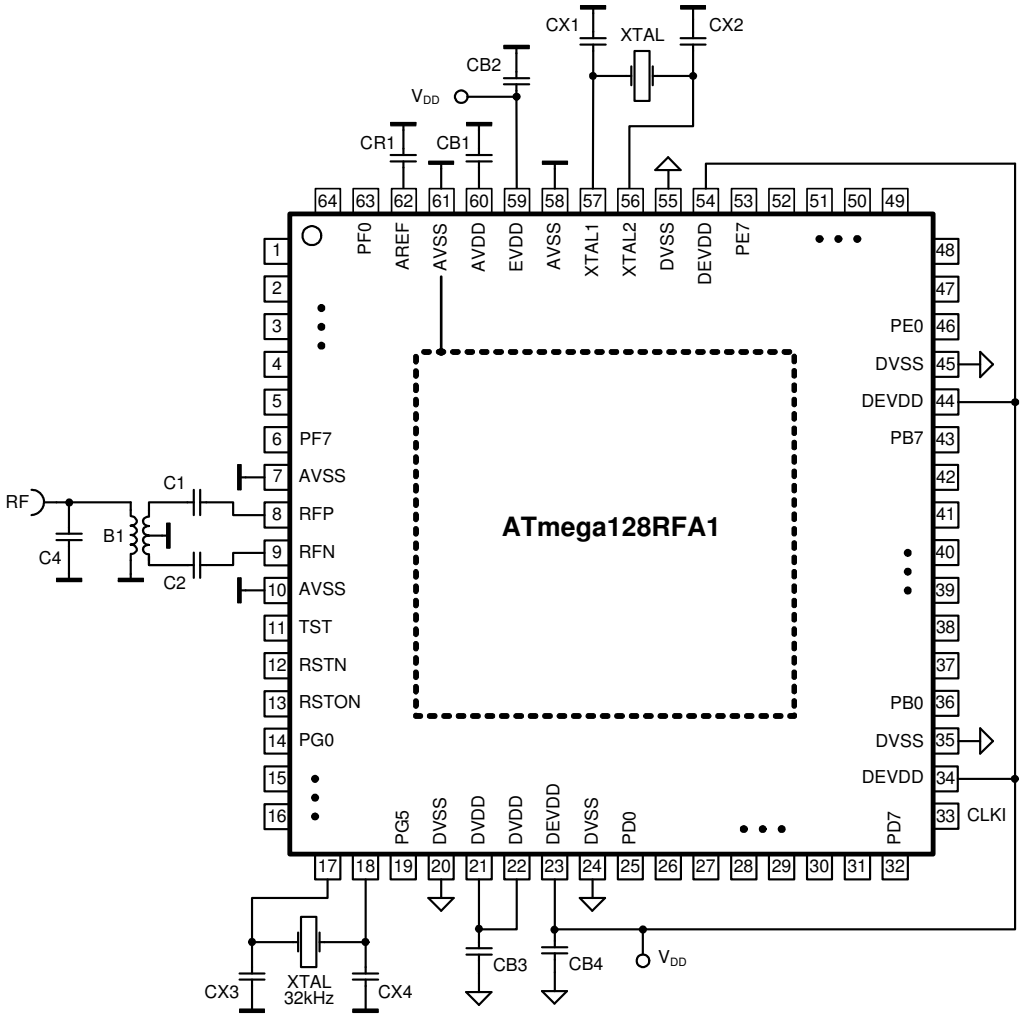
1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address 0x00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

32 Application Circuits

32.1 Basic Application Schematic

A basic application schematic of the ATmega128RFA1 with a single-ended RF connector is shown in Figure 32-1 below. The 50Ω single-ended RF input is transformed to the 100Ω differential RF port impedance using Balun B1. The capacitors C1 and C2 provide AC coupling of the RF input to the RF port, capacitor C4 improves matching.

Figure 32-1. Basic Application schematic (Table 32-1 on page 494)



The power supply bypass capacitors (CB2, CB4) are connected to the external analog supply pin (EVDD, pin 59) and external digital supply pin (DEVDD, pin 23). Pins 34, 44 and 54 supply the digital port pins.

Capacitors CB1 and CB3 are bypass capacitors for the integrated analog and digital voltage regulators to ensure stable operation and to improve noise immunity. Capacitors should be placed as close as possible to the pins and should have a low-resistance and low-inductance connection to ground to achieve the best performance.

The crystal (XTAL), the two load capacitors (CX1, CX2), and the internal circuitry connected to pins XTAL1 and XTAL2 form the 16MHz crystal oscillator for the 2.4GHz transceiver. To achieve the best accuracy and stability of the reference frequency, large parasitic capacitances must be avoided. Crystal lines should be routed as short as possible and not in proximity of digital I/O signals. This is especially required for the High Data Rate Modes.

The 32.768 kHz crystal connected to the internal low power (sub 1µA) crystal oscillator provides a stable time reference for all low power modes including 32 Bit IEEE 802.15.4 Symbol Counter ("MAC Symbol Counter" on page 133) and real time clock application using the asynchronous timer T/C2 ("8-bit Timer/Counter2 with PWM and Asynchronous Operation" on page 309). Total capacitance including CX3, CX4 should not exceed 15pF on both pins. The very low supply current of the oscillator requires careful layout of the PCB and any leakage path must be avoided.

Crosstalk and radiation from switching digital signals to the crystal pins or the RF pins can degrade the system performance. The programming of minimum drive strength settings for the digital output signal is recommended (see "DPDS0 – Port Driver Strength Register 0" on page 174).

Table 32-1. Example Bill of Materials (BoM) for "Basic Application Schematic" on page 493

Designator	Description	Value	Manufacturer	Part Number	Comment
B1	SMD balun SMD balun / filter	2.4 GHz	Wuerth Johanson Technology	748421245 2450FB15L0001	Filter included
CB1 CB3	LDO VREG bypass capacitor	1 µF (100nF minimum)	AVX Murata	0603YD105KAT2A GRM188R61C105KA12D	X5R 10% 16V (0603)
CB2 CB4	Power supply bypass capacitor	1 µF (100nF minimum)			
CX1, CX2	Crystal load capacitor	12 pF	AVX Murata	06035A120JA GRP1886C1H120JA01	COG 5% 50V (0603)
C1, C2	RF coupling capacitor	22 pF	Epcos Epcos AVX	B37930 B37920 06035A220JAT2A	C0G 5% 50V (0402 or 0603)
C4 (optional)	RF matching	0.47 pF	Johnstech		
R1	CLKM low-pass filter resistor	680Ω			Designed for f_{CLKM} = 1 MHz
XTAL	Crystal	CX-4025 16 MHz SX-4025 16 MHz	ACAL Taitjen Siward	XWBBPL-F-1 A207-011	
XTAL 32kHz	Crystal				Rs=100 kOhm

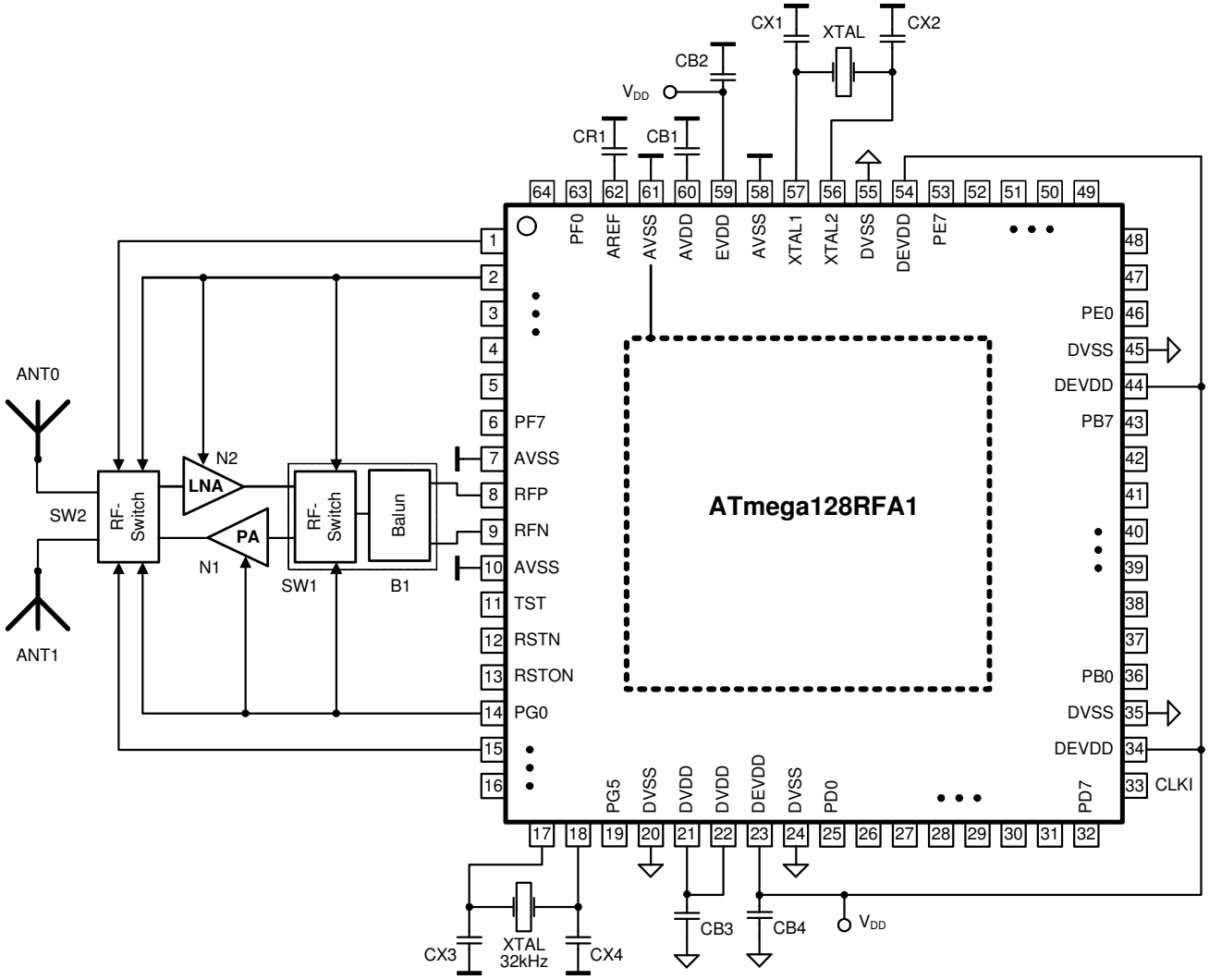
32.2 Extended Feature Set Application Schematic

The ATmega128RFA1 supports additional features like:

- Security Module (AES)
- High Data Rate Mode up to 2MBits/s
- Antenna Diversity using alternate pin function DIG1/2 at Port G and F
- RX/TX Indicator using alternate pin function DIG3/4 at Port G and F

An extended feature set application schematic illustrating the use of the ATmega128RFA1 Extended Feature Set, is shown in Figure 32-2 on page 495.

Figure 32-2. Extended Feature Application schematic



Although this example shows all additional hardware features combined, it is possible to use all features separately or in various combinations.

33 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x1FF)	TRXFBEND	TRXFBEND7	TRXFBEND6	TRXFBEND5	TRXFBEND4	TRXFBEND3	TRXFBEND2	TRXFBEND1	TRXFBEND0	132
...										
(0x180)	TRXFBST	TRXFBST7	TRXFBST6	TRXFBST5	TRXFBST4	TRXFBST3	TRXFBST2	TRXFBST1	TRXFBST0	132
(0x17F)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17E)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17D)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17C)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17B)	TST_RX_LENGTH	RX_LENGTH7	RX_LENGTH6	RX_LENGTH5	RX_LENGTH4	RX_LENGTH3	RX_LENGTH2	RX_LENGTH1	RX_LENGTH0	131
(0x17A)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x179)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x178)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x177)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x176)	TST_CTRL_DIGI	Res7	Res6	Res5	Res4	TST_CTRL_DIG3	TST_CTRL_DIG2	TST_CTRL_DIG1	TST_CTRL_DIG0	131
(0x175)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...										
(0x173)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x172)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x171)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x170)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x16F)	CSMA_BE	MAX_BE3	MAX_BE2	MAX_BE1	MAX_BE0	MIN_BE3	MIN_BE2	MIN_BE1	MIN_BE0	130
(0x16E)	CSMA_SEED_1	AACK_FVN_MODE1	AACK_FVN_MODE0	AACK_SET_PD	AACK_DIS_ACK	AACK_I_AM_COORD	CSMA_SEED_12	CSMA_SEED_11	CSMA_SEED_10	129
(0x16D)	CSMA_SEED_0	CSMA_SEED_07	CSMA_SEED_06	CSMA_SEED_05	CSMA_SEED_04	CSMA_SEED_03	CSMA_SEED_02	CSMA_SEED_01	CSMA_SEED_00	128
(0x16C)	XAH_CTRL_0	MAX_FRAME_RETRES3	MAX_FRAME_RETRES2	MAX_FRAME_RETRES1	MAX_FRAME_RETRES0	MAX_CSMA_RETRES2	MAX_CSMA_RETRES1	MAX_CSMA_RETRES0	SLOTTED_OPERATION	127
(0x16B)	IEEE_ADDR_7	IEEE_ADDR_77	IEEE_ADDR_76	IEEE_ADDR_75	IEEE_ADDR_74	IEEE_ADDR_73	IEEE_ADDR_72	IEEE_ADDR_71	IEEE_ADDR_70	126
(0x16A)	IEEE_ADDR_6	IEEE_ADDR_67	IEEE_ADDR_66	IEEE_ADDR_65	IEEE_ADDR_64	IEEE_ADDR_63	IEEE_ADDR_62	IEEE_ADDR_61	IEEE_ADDR_60	126
(0x169)	IEEE_ADDR_5	IEEE_ADDR_57	IEEE_ADDR_56	IEEE_ADDR_55	IEEE_ADDR_54	IEEE_ADDR_53	IEEE_ADDR_52	IEEE_ADDR_51	IEEE_ADDR_50	126
(0x168)	IEEE_ADDR_4	IEEE_ADDR_47	IEEE_ADDR_46	IEEE_ADDR_45	IEEE_ADDR_44	IEEE_ADDR_43	IEEE_ADDR_42	IEEE_ADDR_41	IEEE_ADDR_40	126
(0x167)	IEEE_ADDR_3	IEEE_ADDR_37	IEEE_ADDR_36	IEEE_ADDR_35	IEEE_ADDR_34	IEEE_ADDR_33	IEEE_ADDR_32	IEEE_ADDR_31	IEEE_ADDR_30	125
(0x166)	IEEE_ADDR_2	IEEE_ADDR_27	IEEE_ADDR_26	IEEE_ADDR_25	IEEE_ADDR_24	IEEE_ADDR_23	IEEE_ADDR_22	IEEE_ADDR_21	IEEE_ADDR_20	125
(0x165)	IEEE_ADDR_1	IEEE_ADDR_17	IEEE_ADDR_16	IEEE_ADDR_15	IEEE_ADDR_14	IEEE_ADDR_13	IEEE_ADDR_12	IEEE_ADDR_11	IEEE_ADDR_10	125
(0x164)	IEEE_ADDR_0	IEEE_ADDR_07	IEEE_ADDR_06	IEEE_ADDR_05	IEEE_ADDR_04	IEEE_ADDR_03	IEEE_ADDR_02	IEEE_ADDR_01	IEEE_ADDR_00	124
(0x163)	PAN_ID_1	PAN_ID_17	PAN_ID_16	PAN_ID_15	PAN_ID_14	PAN_ID_13	PAN_ID_12	PAN_ID_11	PAN_ID_10	124
(0x162)	PAN_ID_0	PAN_ID_07	PAN_ID_06	PAN_ID_05	PAN_ID_04	PAN_ID_03	PAN_ID_02	PAN_ID_01	PAN_ID_00	124
(0x161)	SHORT_ADDR_1	SHORT_ADDR_17	SHORT_ADDR_16	SHORT_ADDR_15	SHORT_ADDR_14	SHORT_ADDR_13	SHORT_ADDR_12	SHORT_ADDR_11	SHORT_ADDR_10	124
(0x160)	SHORT_ADDR_0	SHORT_ADDR_07	SHORT_ADDR_06	SHORT_ADDR_05	SHORT_ADDR_04	SHORT_ADDR_03	SHORT_ADDR_02	SHORT_ADDR_01	SHORT_ADDR_00	123
(0x15F)	MAN_ID_1	MAN_ID_17	MAN_ID_16	MAN_ID_15	MAN_ID_14	MAN_ID_13	MAN_ID_12	MAN_ID_11	MAN_ID_10	123
(0x15E)	MAN_ID_0	MAN_ID_07	MAN_ID_06	MAN_ID_05	MAN_ID_04	MAN_ID_03	MAN_ID_02	MAN_ID_01	MAN_ID_00	123
(0x15D)	VERSION_NUM	VERSION_NUM7	VERSION_NUM6	VERSION_NUM5	VERSION_NUM4	VERSION_NUM3	VERSION_NUM2	VERSION_NUM1	VERSION_NUM0	122
(0x15C)	PART_NUM	PART_NUM7	PART_NUM6	PART_NUM5	PART_NUM4	PART_NUM3	PART_NUM2	PART_NUM1	PART_NUM0	122
(0x15B)	PLL_DCU	PLL_DCU_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	121
(0x15A)	PLL_CF	PLL_CF_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	121
(0x159)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x158)	FTN_CTRL	FTN_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	120
(0x157)	XAH_CTRL_1	Res1	Res0	AACK_FLTR_RES_FT	AACK_UPLD_RES_FT	Res	AACK_ACK_TIME	AACK_PROM_MODE	Res	119
...	Reserved									
(0x155)	RX_SYN	RX_PDT_DIS	Res2	Res1	Res0	RX_PDT_LEVEL3	RX_PDT_LEVEL2	RX_PDT_LEVEL1	RX_PDT_LEVEL0	118
...	Reserved									
(0x152)	XOSC_CTRL	XTAL_MODE3	XTAL_MODE2	XTAL_MODE1	XTAL_MODE0	XTAL_TRIM3	XTAL_TRIM2	XTAL_TRIM1	XTAL_TRIM0	117
(0x151)	BATMON	BAT_LOW	BAT_LOW_EN	BATMON_OK	BATMON_HR	BATMON_VTH3	BATMON_VTH2	BATMON_VTH1	BATMON_VTH0	116
(0x150)	VREG_CTRL	AVREG_EXT	AVDD_OK	Res5	Res4	Res3	DVDD_OK	Res1	Res0	115
(0x14F)	IRQ_STATUS	AWAKE	TX_END	AMI	CCA_ED_DONE	RX_END	RX_START	PLL_UNLOCK	PLL_LOCK	114
(0x14E)	IRQ_MASK	AWAKE_EN	TX_END_EN	AMI_EN	CCA_ED_DONE_EN	RX_END_EN	RX_START_EN	PLL_UNLOCK_EN	PLL_LOCK_EN	114
(0x14D)	ANT_DIV	ANT_SEL	Res2	Res1	Res0	ANT_DIV_EN	ANT_EXT_SW_EN	ANT_CTRL1	ANT_CTRL0	112
(0x14C)	TRX_CTRL_2	RX_SAFE_MODE	Res4	Res3	Res2	Res1	Res0	OQPSK_DATA_RATE1	OQPSK_DATA_RATE0	112
(0x14B)	SFD_VALUE	SFD_VALUE7	SFD_VALUE6	SFD_VALUE5	SFD_VALUE4	SFD_VALUE3	SFD_VALUE2	SFD_VALUE1	SFD_VALUE0	111
(0x14A)	RX_CTRL	Res7	Res6	Res5	Res4	PDT_THRES3	PDT_THRES2	PDT_THRES1	PDT_THRES0	111
(0x149)	CCA_THRES	CCA_CS_THRES3	CCA_CS_THRES2	CCA_CS_THRES1	CCA_CS_THRES0	CCA_ED_THRES3	CCA_ED_THRES2	CCA_ED_THRES1	CCA_ED_THRES0	110
(0x148)	PHY_CC_CCA	CCA_REQUEST	CCA_MODE1	CCA_MODE0	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0	109
(0x147)	PHY_ED_LEVEL	ED_LEVEL7	ED_LEVEL6	ED_LEVEL5	ED_LEVEL4	ED_LEVEL3	ED_LEVEL2	ED_LEVEL1	ED_LEVEL0	108
(0x146)	PHY_RSSI	RX_CRC_VALID	RND_VALUE1	RND_VALUE0	RSSI4	RSSI3	RSSI2	RSSI1	RSSI0	107
(0x145)	PHY_TX_PWR	PA_BUF_LT1	PA_BUF_LT0	PA_LT1	PA_LT0	TX_PWR3	TX_PWR2	TX_PWR1	TX_PWR0	106

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x144)	TRX_CTRL_1	PA_EXT_EN	IRQ_2_EXT_EN	TX_AUTO_CRC_ON	Res4	Res3	Res2	Res1	Res0	105
(0x143)	TRX_CTRL_0	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	105
(0x142)	TRX_STATE	TRAC_STATUS2	TRAC_STATUS1	TRAC_STATUS0	TRX_CMD4	TRX_CMD3	TRX_CMD2	TRX_CMD1	TRX_CMD0	104
(0x141)	TRX_STATUS	CCA_DONE	CCA_STATUS	TST_STATUS	TRX_STATUS4	TRX_STATUS3	TRX_STATUS2	TRX_STATUS1	TRX_STATUS0	102
...	Reserved									
(0x13F)	AES_KEY	AES_KEY7	AES_KEY6	AES_KEY5	AES_KEY4	AES_KEY3	AES_KEY2	AES_KEY1	AES_KEY0	102
(0x13E)	AES_STATE	AES_STATE7	AES_STATE6	AES_STATE5	AES_STATE4	AES_STATE3	AES_STATE2	AES_STATE1	AES_STATE0	102
(0x13D)	AES_STATUS	AES_ER	Res5	Res4	Res3	Res2	Res1	Res0	AES_DONE	101
(0x13C)	AES_CTRL	AES_REQUEST	Res	AES_MODE	Res	AES_DIR	AES_IM	Res1	Res0	100
(0x13B)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...	Reserved									
(0x139)	TRXPR	Res3	Res2	Res1	Res0	Res3	Res2	SLPTR	TRXRST	169
...	Reserved									
(0x137)	DPDS1	Res5	Res4	Res3	Res2	Res1	Res0	PGDRV1	PGDRV0	175
(0x136)	DPDS0	PFDRV1	PFDRV0	PEDRV1	PEDRV0	PDDRV1	PDDRV0	PBDRV1	PBDRV0	174
(0x135)	DRTRAM0	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	170
(0x134)	DRTRAM1	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	170
(0x133)	DRTRAM2	Res7	Res	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	171
(0x132)	DRTRAM3	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	171
(0x131)	LLDRH	Res2	Res1	Res0	LLDRH4	LLDRH3	LLDRH2	LLDRH1	LLDRH0	173
(0x130)	LLDRL	Res3	Res2	Res1	Res0	LLDRL3	LLDRL2	LLDRL1	LLDRL0	173
(0x12F)	LLCR	Res1	Res0	LLDONE	LLCOMP	LLCAL	LLTCO	LLSHORT	LLENCAL	172
...	Reserved									
(0x12D)	OCR5CH	OCR5CH7	OCR5CH6	OCR5CH5	OCR5CH4	OCR5CH3	OCR5CH2	OCR5CH1	OCR5CH0	300
(0x12C)	OCR5CL	OCR5CL7	OCR5CL6	OCR5CL5	OCR5CL4	OCR5CL3	OCR5CL2	OCR5CL1	OCR5CL0	301
(0x12B)	OCR5BH	OCR5BH7	OCR5BH6	OCR5BH5	OCR5BH4	OCR5BH3	OCR5BH2	OCR5BH1	OCR5BH0	299
(0x12A)	OCR5BL	OCR5BL7	OCR5BL6	OCR5BL5	OCR5BL4	OCR5BL3	OCR5BL2	OCR5BL1	OCR5BL0	300
(0x129)	OCR5AH	OCR5AH7	OCR5AH6	OCR5AH5	OCR5AH4	OCR5AH3	OCR5AH2	OCR5AH1	OCR5AH0	299
(0x128)	OCR5AL	OCR5AL7	OCR5AL6	OCR5AL5	OCR5AL4	OCR5AL3	OCR5AL2	OCR5AL1	OCR5AL0	299
(0x127)	ICR5H	ICR5H7	ICR5H6	ICR5H5	ICR5H4	ICR5H3	ICR5H2	ICR5H1	ICR5H0	301
(0x126)	ICR5L	ICR5L7	ICR5L6	ICR5L5	ICR5L4	ICR5L3	ICR5L2	ICR5L1	ICR5L0	301
(0x125)	TCNT5H	TCNT5H7	TCNT5H6	TCNT5H5	TCNT5H4	TCNT5H3	TCNT5H2	TCNT5H1	TCNT5H0	298
(0x124)	TCNT5L	TCNT5L7	TCNT5L6	TCNT5L5	TCNT5L4	TCNT5L3	TCNT5L2	TCNT5L1	TCNT5L0	298
...	Reserved									
(0x122)	TCCR5C	FOC5A	FOC5B	FOC5C	Res4	Res3	Res2	Res1	Res0	297
(0x121)	TCCR5B	ICNC5	ICES5	Res	WGM53	WGM52	CS52	CS51	CS50	296
(0x120)	TCCR5A	COM5A1	COM5A0	COM5B1	COM5B0	COM5C1	COM5C0	WGM51	WGM50	294
...	Reserved									
(0xF8)	SCOCR1HH	SCOCR1HH7	SCOCR1HH6	SCOCR1HH5	SCOCR1HH4	SCOCR1HH3	SCOCR1HH2	SCOCR1HH1	SCOCR1HH0	140
(0xF7)	SCOCR1HL	SCOCR1HL7	SCOCR1HL6	SCOCR1HL5	SCOCR1HL4	SCOCR1HL3	SCOCR1HL2	SCOCR1HL1	SCOCR1HL0	141
(0xF6)	SCOCR1LH	SCOCR1LH7	SCOCR1LH6	SCOCR1LH5	SCOCR1LH4	SCOCR1LH3	SCOCR1LH2	SCOCR1LH1	SCOCR1LH0	141
(0xF5)	SCOCR1LL	SCOCR1LL7	SCOCR1LL6	SCOCR1LL5	SCOCR1LL4	SCOCR1LL3	SCOCR1LL2	SCOCR1LL1	SCOCR1LL0	141
(0xF4)	SCOCR2HH	SCOCR2HH7	SCOCR2HH6	SCOCR2HH5	SCOCR2HH4	SCOCR2HH3	SCOCR2HH2	SCOCR2HH1	SCOCR2HH0	141
(0xF3)	SCOCR2HL	SCOCR2HL7	SCOCR2HL6	SCOCR2HL5	SCOCR2HL4	SCOCR2HL3	SCOCR2HL2	SCOCR2HL1	SCOCR2HL0	142
(0xF2)	SCOCR2LH	SCOCR2LH7	SCOCR2LH6	SCOCR2LH5	SCOCR2LH4	SCOCR2LH3	SCOCR2LH2	SCOCR2LH1	SCOCR2LH0	142
(0xF1)	SCOCR2LL	SCOCR2LL7	SCOCR2LL6	SCOCR2LL5	SCOCR2LL4	SCOCR2LL3	SCOCR2LL2	SCOCR2LL1	SCOCR2LL0	142
(0xF0)	SCOCR3HH	SCOCR3HH7	SCOCR3HH6	SCOCR3HH5	SCOCR3HH4	SCOCR3HH3	SCOCR3HH2	SCOCR3HH1	SCOCR3HH0	142
(0xEF)	SCOCR3HL	SCOCR3HL7	SCOCR3HL6	SCOCR3HL5	SCOCR3HL4	SCOCR3HL3	SCOCR3HL2	SCOCR3HL1	SCOCR3HL0	143
(0xEE)	SCOCR3LH	SCOCR3LH7	SCOCR3LH6	SCOCR3LH5	SCOCR3LH4	SCOCR3LH3	SCOCR3LH2	SCOCR3LH1	SCOCR3LH0	143
(0xED)	SCOCR3LL	SCOCR3LL7	SCOCR3LL6	SCOCR3LL5	SCOCR3LL4	SCOCR3LL3	SCOCR3LL2	SCOCR3LL1	SCOCR3LL0	143
(0xEC)	SCTSRHH	SCTSRHH7	SCTSRHH6	SCTSRHH5	SCTSRHH4	SCTSRHH3	SCTSRHH2	SCTSRHH1	SCTSRHH0	138
(0xEB)	SCTSRHL	SCTSRHL7	SCTSRHL6	SCTSRHL5	SCTSRHL4	SCTSRHL3	SCTSRHL2	SCTSRHL1	SCTSRHL0	139
(0xEA)	SCTSRLH	SCTSRLH7	SCTSRLH6	SCTSRLH5	SCTSRLH4	SCTSRLH3	SCTSRLH2	SCTSRLH1	SCTSRLH0	139
(0xE9)	SCTSRLL	SCTSRLL7	SCTSRLL6	SCTSRLL5	SCTSRLL4	SCTSRLL3	SCTSRLL2	SCTSRLL1	SCTSRLL0	139
(0xE8)	SCBTSRHH	SCBTSRHH7	SCBTSRHH6	SCBTSRHH5	SCBTSRHH4	SCBTSRHH3	SCBTSRHH2	SCBTSRHH1	SCBTSRHH0	139
(0xE7)	SCBTSRHL	SCBTSRHL7	SCBTSRHL6	SCBTSRHL5	SCBTSRHL4	SCBTSRHL3	SCBTSRHL2	SCBTSRHL1	SCBTSRHL0	140
(0xE6)	SCBTSRLH	SCBTSRLH7	SCBTSRLH6	SCBTSRLH5	SCBTSRLH4	SCBTSRLH3	SCBTSRLH2	SCBTSRLH1	SCBTSRLH0	140
(0xE5)	SCBTSRLL	SCBTSRLL7	SCBTSRLL6	SCBTSRLL5	SCBTSRLL4	SCBTSRLL3	SCBTSRLL2	SCBTSRLL1	SCBTSRLL0	140
(0xE4)	SCCNTHH	SCCNTHH7	SCCNTHH6	SCCNTHH5	SCCNTHH4	SCCNTHH3	SCCNTHH2	SCCNTHH1	SCCNTHH0	137
(0xE3)	SCCNTHL	SCCNTHL7	SCCNTHL6	SCCNTHL5	SCCNTHL4	SCCNTHL3	SCCNTHL2	SCCNTHL1	SCCNTHL0	138
(0xE2)	SCCNTLH	SCCNTLH7	SCCNTLH6	SCCNTLH5	SCCNTLH4	SCCNTLH3	SCCNTLH2	SCCNTLH1	SCCNTLH0	138
(0xE1)	SCCNTLL	SCCNTLL7	SCCNTLL6	SCCNTLL5	SCCNTLL4	SCCNTLL3	SCCNTLL2	SCCNTLL1	SCCNTLL0	138
(0xE0)	SCIRQS	Res2	Res1	Res0	IRQSBO	IRQSOF	IRQSCP3	IRQSCP2	IRQSCP1	145
(0xDF)	SCIRQM	Res2	Res1	Res0	IRQMBO	IRQMOF	IRQMCP3	IRQMCP2	IRQMCP1	146

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xDE)	SCSR	Res6	Res5	Res4	Res3	Res2	Res1	Res0	SCBSY	145
(0xDD)	SCCR1	Res6	Res5	Res4	Res4	Res3	Res2	Res1	SCENBO	144
(0xDC)	SCCR0	SCRES	SCMBTS	SCEN	SCCKSEL	SCTSE	SCCMP3	SCCMP2	SCCMP1	143
...	Reserved									
(0xD1)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0xD0)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...	Reserved									
(0xCE)	UDR1	UDR17	UDR16	UDR15	UDR14	UDR13	UDR12	UDR11	UDR10	360
(0xCD)	UBRR1H	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	364
(0xCC)	UBRR1L	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	365
...	Reserved									
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UDORD1	UCPHA1	UCPOL1	376
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	375
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	375
...	Reserved									
(0xC6)	UDR0	UDR07	UDR06	UDR05	UDR04	UDR03	UDR02	UDR01	UDR00	356
(0xC5)	UBRR0H	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	360
(0xC4)	UBRR0L	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	360
...	Reserved									
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UDORD0	UCPHA0	UCPOL0	374
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	374
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	373
...	Reserved									
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	Res	406
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	Res	TWIE	402
(0xBB)	TWDR	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	405
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	405
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	Res	TWPS1	TWPS0	403
(0xB8)	TWBR	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	401
...	Reserved									
(0xB6)	ASSR	EXCLKAMR	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	328
...	Reserved									
(0xB4)	OCR2B	OCR2B7	OCR2B6	OCR2B5	OCR2B4	OCR2B3	OCR2B2	OCR2B1	OCR2B0	327
(0xB3)	OCR2A	OCR2A7	OCR2A6	OCR2A5	OCR2A4	OCR2A3	OCR2A2	OCR2A1	OCR2A0	327
(0xB2)	TCNT2	TCNT27	TCNT26	TCNT25	TCNT24	TCNT23	TCNT22	TCNT21	TCNT20	327
(0xB1)	TCCR2B	FOC2A	FOC2B	Res1	Res0	WGM22	CS22	CS21	CS20	326
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	Res1	Res0	WGM21	WGM20	325
...	Reserved									
(0xAD)	OCR4CH	OCR4CH7	OCR4CH6	OCR4CH5	OCR4CH4	OCR4CH3	OCR4CH2	OCR4CH1	OCR4CH0	291
(0xAC)	OCR4CL	OCR4CL7	OCR4CL6	OCR4CL5	OCR4CL4	OCR4CL3	OCR4CL2	OCR4CL1	OCR4CL0	292
(0xAB)	OCR4BH	OCR4BH7	OCR4BH6	OCR4BH5	OCR4BH4	OCR4BH3	OCR4BH2	OCR4BH1	OCR4BH0	291
(0xAA)	OCR4BL	OCR4BL7	OCR4BL6	OCR4BL5	OCR4BL4	OCR4BL3	OCR4BL2	OCR4BL1	OCR4BL0	291
(0xA9)	OCR4AH	OCR4AH7	OCR4AH6	OCR4AH5	OCR4AH4	OCR4AH3	OCR4AH2	OCR4AH1	OCR4AH0	290
(0xA8)	OCR4AL	OCR4AL7	OCR4AL6	OCR4AL5	OCR4AL4	OCR4AL3	OCR4AL2	OCR4AL1	OCR4AL0	290
(0xA7)	ICR4H	ICR4H7	ICR4H6	ICR4H5	ICR4H4	ICR4H3	ICR4H2	ICR4H1	ICR4H0	292
(0xA6)	ICR4L	ICR4L7	ICR4L6	ICR4L5	ICR4L4	ICR4L3	ICR4L2	ICR4L1	ICR4L0	292
(0xA5)	TCNT4H	TCNT4H7	TCNT4H6	TCNT4H5	TCNT4H4	TCNT4H3	TCNT4H2	TCNT4H1	TCNT4H0	289
(0xA4)	TCNT4L	TCNT4L7	TCNT4L6	TCNT4L5	TCNT4L4	TCNT4L3	TCNT4L2	TCNT4L1	TCNT4L0	289
...	Reserved									
(0xA2)	TCCR4C	FOC4A	FOC4B	FOC4C	Res4	Res3	Res2	Res1	Res0	288
(0xA1)	TCCR4B	ICNC4	ICES4	Res	WGM43	WGM42	CS42	CS41	CS40	287
(0xA0)	TCCR4A	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	285
...	Reserved									
(0x9D)	OCR3CH	OCR3CH7	OCR3CH6	OCR3CH5	OCR3CH4	OCR3CH3	OCR3CH2	OCR3CH1	OCR3CH0	282
(0x9C)	OCR3CL	OCR3CL7	OCR3CL6	OCR3CL5	OCR3CL4	OCR3CL3	OCR3CL2	OCR3CL1	OCR3CL0	283
(0x9B)	OCR3BH	OCR3BH7	OCR3BH6	OCR3BH5	OCR3BH4	OCR3BH3	OCR3BH2	OCR3BH1	OCR3BH0	282
(0x9A)	OCR3BL	OCR3BL7	OCR3BL6	OCR3BL5	OCR3BL4	OCR3BL3	OCR3BL2	OCR3BL1	OCR3BL0	282
(0x99)	OCR3AH	OCR3AH7	OCR3AH6	OCR3AH5	OCR3AH4	OCR3AH3	OCR3AH2	OCR3AH1	OCR3AH0	281
(0x98)	OCR3AL	OCR3AL7	OCR3AL6	OCR3AL5	OCR3AL4	OCR3AL3	OCR3AL2	OCR3AL1	OCR3AL0	281
(0x97)	ICR3H	ICR3H7	ICR3H6	ICR3H5	ICR3H4	ICR3H3	ICR3H2	ICR3H1	ICR3H0	283
(0x96)	ICR3L	ICR3L7	ICR3L6	ICR3L5	ICR3L4	ICR3L3	ICR3L2	ICR3L1	ICR3L0	283
(0x95)	TCNT3H	TCNT3H7	TCNT3H6	TCNT3H5	TCNT3H4	TCNT3H3	TCNT3H2	TCNT3H1	TCNT3H0	280
(0x94)	TCNT3L	TCNT3L7	TCNT3L6	TCNT3L5	TCNT3L4	TCNT3L3	TCNT3L2	TCNT3L1	TCNT3L0	280
...	Reserved									

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x92)	TCCR3C	FOC3A	FOC3B	FOC3C	Res4	Res3	Res2	Res1	Res0	279
(0x91)	TCCR3B	ICNC3	ICES3	Res	WGM33	WGM32	CS32	CS31	CS30	278
(0x90)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	276
...	Reserved									
(0x8D)	OCR1CH	OCR1CH7	OCR1CH6	OCR1CH5	OCR1CH4	OCR1CH3	OCR1CH2	OCR1CH1	OCR1CH0	273
(0x8C)	OCR1CL	OCR1CL7	OCR1CL6	OCR1CL5	OCR1CL4	OCR1CL3	OCR1CL2	OCR1CL1	OCR1CL0	273
(0x8B)	OCR1BH	OCR1BH7	OCR1BH6	OCR1BH5	OCR1BH4	OCR1BH3	OCR1BH2	OCR1BH1	OCR1BH0	272
(0x8A)	OCR1BL	OCR1BL7	OCR1BL6	OCR1BL5	OCR1BL4	OCR1BL3	OCR1BL2	OCR1BL1	OCR1BL0	272
(0x89)	OCR1AH	OCR1AH7	OCR1AH6	OCR1AH5	OCR1AH4	OCR1AH3	OCR1AH2	OCR1AH1	OCR1AH0	271
(0x88)	OCR1AL	OCR1AL7	OCR1AL6	OCR1AL5	OCR1AL4	OCR1AL3	OCR1AL2	OCR1AL1	OCR1AL0	271
(0x87)	ICR1H	ICR1H7	ICR1H6	ICR1H5	ICR1H4	ICR1H3	ICR1H2	ICR1H1	ICR1H0	273
(0x86)	ICR1L	ICR1L7	ICR1L6	ICR1L5	ICR1L4	ICR1L3	ICR1L2	ICR1L1	ICR1L0	274
(0x85)	TCNT1H	TCNT1H7	TCNT1H6	TCNT1H5	TCNT1H4	TCNT1H3	TCNT1H2	TCNT1H1	TCNT1H0	270
(0x84)	TCNT1L	TCNT1L7	TCNT1L6	TCNT1L5	TCNT1L4	TCNT1L3	TCNT1L2	TCNT1L1	TCNT1L0	271
...	Reserved									
(0x82)	TCCR1C	FOC1A	FOC1B	FOC1C	Res4	Res3	Res2	Res1	Res0	270
(0x81)	TCCR1B	ICNC1	ICES1	Res	WGM13	WGM12	CS12	CS11	CS10	268
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	266
(0x7F)	DIDR1							AIN1D	AIN0D	409
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	433
(0x7D)	DIDR2	ADC15D	ADC14D	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	433
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	427
(0x7B)	ADCSRB	AVDDOK	ACME	REFOK	ACCH	MUX5	ADTS2	ADTS1	ADTS0	428
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	430
(0x79)	ADCH	ADCH7	ADCH6	ADCH5	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	432
(0x78)	ADCL	ADCL7	ADCL6	ADCL5	ADCL4	ADCL3	ADCL2	ADCL1	ADCL0	432
(0x77)	ADCSRC	ADTHT1	ADTHT0	Res0	ADSUT4	ADSUT3	ADSUT2	ADSUT1	ADSUT0	431
...	Reserved									
(0x75)	NEMCR	Res7	ENEAM	AEAM1	AEAM0	Res3	Res2	Res1	Res0	463
(0x74)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x73)	TIMSK5	Res1	Res0	ICIE5	Res	OCIE5C	OCIE5B	OCIE5A	TOIE5	302
(0x72)	TIMSK4	Res1	Res0	ICIE4	Res	OCIE4C	OCIE4B	OCIE4A	TOIE4	293
(0x71)	TIMSK3	Res1	Res0	ICIE3	Res	OCIE3C	OCIE3B	OCIE3A	TOIE3	284
(0x70)	TIMSK2	Res4	Res3	Res2	Res1	Res0	OCIE2B	OCIE2A	TOIE2	323
(0x6F)	TIMSK1	Res1	Res0	ICIE1	Res	OCIE1C	OCIE1B	OCIE1A	TOIE1	274
(0x6E)	TIMSK0	Res4	Res3	Res2	Res1	Res0	OCIE0B	OCIE0A	TOIE0	242
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	224
(0x6C)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	224
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	225
(0x6A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	220
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	219
(0x68)	PCICR	Res4	Res3	Res2	Res1	Res0	PCIE2	PCIE1	PCIE0	223
(0x67)	BGCR	Res	BGCAL_FINE3	BGCAL_FINE2	BGCAL_FINE1	BGCAL_FINE0	BGCAL2	BGCAL1	BGCAL0	433
(0x66)	OSCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	153
(0x65)	PRR1	Res	PRTRX24	PRTIM5	PRTIM4	PRTIM3			PRUSART1	168
(0x64)	PRR0	PRTWI	PRTIM2	PRTIM0	PRPGA	PRTIM1	PRSPI	PRUSART0	PRADC	167
(0x63)	PRR2	Res3	Res2	Res1	Res0	PRRAM3	PRRAM2	PRRAM1	PRRAM0	168
...	Reserved									
(0x61)	CLKPR	CLKPCE	Res2	Res1	Res0	CLKPS3	CLKPS2	CLKPS1	CLKPS0	154
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	183
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	14
...	Reserved									
0x3B (0x5B)	RAMPZ	Res5	Res4	Res3	Res2	Res1	Res0	RAMPZ1	RAMPZ0	14
...	Reserved									
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	462
...	Reserved									
0x35 (0x55)	MCUCR	JTD	Res1	Res0	PUD	Res1	Res0	IVSEL	IVCE	204
0x34 (0x54)	MCUSR	Res2	Res1	Res0	JTRF	WDRF	BORF	EXTRF	PORF	183
0x33 (0x53)	SMCR	Res3	Res2	Res1	Res0	SM2	SM1	SM0	SE	166
...	Reserved									
0x31 (0x51)	OCDR	OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	440
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	408

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
...	Reserved									
0x2E (0x4E)	SPDR	SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0	338
0x2D (0x4D)	SPSR	SPIF	WCOL	Res4	Res3	Res2	Res1	Res0	SPI2X	337
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	336
0x2B (0x4B)	GPIOR2	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	26
0x2A (0x4A)	GPIOR1	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	26
...	Reserved									
0x28 (0x48)	OCR0B	OCR0B_7	OCR0B_6	OCR0B_5	OCR0B_4	OCR0B_3	OCR0B_2	OCR0B_1	OCR0B_0	242
0x27 (0x47)	OCR0A	OCR0A_7	OCR0A_6	OCR0A_5	OCR0A_4	OCR0A_3	OCR0A_2	OCR0A_1	OCR0A_0	241
0x26 (0x46)	TCNT0	TCNT0_7	TCNT0_6	TCNT0_5	TCNT0_4	TCNT0_3	TCNT0_2	TCNT0_1	TCNT0_0	241
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	Res1	Res0	WGM02	CS02	CS01	CS00	240
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	Res1	Res0	WGM01	WGM00	238
0x23 (0x43)	GTCCR	TSM	Res4	Res3	Res2	Res1	Res0	PSRASY	PSRSYNC	329
0x22 (0x42)	EEARH	Res3	Res2	Res1	Res0	EEAR11	EEAR10	EEAR9	EEAR8	23
0x21 (0x41)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	23
0x20 (0x40)	EEDR	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	23
0x1F (0x3F)	EECR	Res1	Res0	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	24
0x1E (0x3E)	GPIOR0	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	26
0x1D (0x3D)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	222
0x1C (0x3C)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	222
0x1B (0x3B)	PCIFR	Res4	Res3	Res2	Res1	Res0	PCIF2	PCIF1	PCIF0	223
0x1A (0x3A)	TIFR5	Res1	Res0	ICF5	Res	OCF5C	OCF5B	OCF5A	TOV5	302
0x19 (0x39)	TIFR4	Res1	Res0	ICF4	Res	OCF4C	OCF4B	OCF4A	TOV4	293
0x18 (0x38)	TIFR3	Res1	Res0	ICF3	Res	OCF3C	OCF3B	OCF3A	TOV3	284
0x17 (0x37)	TIFR2	Res4	Res3	Res2	Res1	Res0	OCF2B	OCF2A	TOV2	324
0x16 (0x36)	TIFR1	Res1	Res0	ICF1	Res	OCF1C	OCF1B	OCF1A	TOV1	275
0x15 (0x35)	TIFR0	Res4	Res3	Res2	Res1	Res0	OCF0B	OCF0A	TOV0	243
0x14 (0x34)	PORTG	Res1	Res0	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	209
0x13 (0x33)	DDRG	Res1	Res0	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	209
0x12 (0x32)	PING	Res1	Res0	PING5	PING4	PING3	PING2	PING1	PING0	210
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	208
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	208
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	209
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	207
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	207
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	208
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	206
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	207
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	207
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	27
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	28
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	28
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	205
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	206
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	206
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	27
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	27
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	27

Notes:
1. Reserved registers, bits and I/O memory addresses (marked as Res*) may not be modified.
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 the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. 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 device is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Op-code for the IN and OUT instructions. For the Extended I/O space from 0x60 – 0x1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

34 Electrical Characteristics

34.1 Absolute Maximum Ratings

Note that 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 are not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
T_{STOR}	Storage temperature		-50		150	°C
T_{LEAD}	Lead temperature	T = 10s, (soldering profile compliant with IPC/JEDEC J-STD-020B)			260	°C
V_{ESD}	ESD robustness	Compl. to [3], Compl. to [4]	4 750			kV V
P_{RF}	Input RF level				+14	dBm
V_{DIG}	Voltage on all pins (except pins 4, 5, 13, 14, 29)		-0.3		V_{DD}+0.3	V
V_{ANA}	Voltage on pins 4, 5, 13, 14, 29		-0.3		2.0	V

34.1.1 Recommended Operating Range

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
T_{OP}	Operating temperature range		-40		+85	°C
V_{DD}	Supply voltage	Voltage on pins 15,28[2]	1.8	3.0	3.6	V
$V_{DD1.8}$	Supply voltage (on pins 13, 14, 29)	External voltage supply[1]	1.7	1.8	1.9	V

Notes:
1. Register VREG_CTRL needs to be programmed to disable internal voltage regulators and supply blocks by an external 1.8V supply, refer to section "Voltage Regulators (AVREG, DVREG)" on page 163.
2. Even if an implementation uses the external 1.8V voltage supply $V_{DD1.8}$ it is required to connect V_{DD}.

34.1.2 Digital Pin Characteristics

Test Conditions: T_{OP} = 25°C (unless otherwise stated)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IH}	High level input voltage[1]		V_{DD} – 0.4			V
V_{IL}	Low level input voltage[1]				0.4	V
V_{OH}	High level output voltage[1]	For all output driver strengths defined in DPDS0, DPDS1	V_{DD} – 0.4			V
V_{OL}	Low level output voltage[1]	For all output driver strengths defined in DPDS0, DPDS1			0.4	V

Note:
5. The capacitive load should not be larger than 50 pF for all I/Os when using the default driver strength settings, refer to section "DPDS0 – Port Driver Strength Register 0" on page 174 and "DPDS1 – Port Driver Strength Register 1" on page 175. Generally, large load capacitances increase the overall current consumption.

34.2 Clock Characteristics

34.2.1 Calibrated Internal RC Oscillator Accuracy

Table 34-2. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{DEVDD}	Temperature	Calibration Accuracy
Factory Calibration	16 MHz	3.0V	25°C	± TBD %
User Calibration	TBD	1.8V – 3.6V	-TBD°C - TBD°C	± TBD %

34.2.2 External Clock Drive

Figure 34-1 External Clock Drive Waveforms

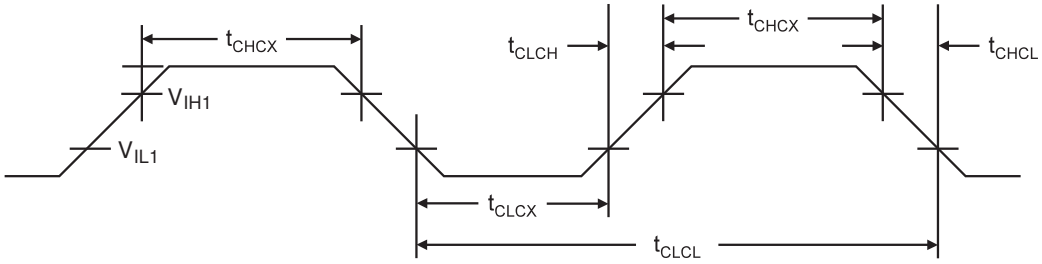


Table 34-3. External Clock Drive

Symbol	Parameter	Min.	Max.	Units
$1/t_{CLCL}$	Oscillator Frequency		16	MHz
t_{CLCL}	Clock Period	62.5		ns
t_{CHCX}	High Time	25		ns
t_{CLCX}	Low Time	25		ns
t_{CLCH}	Rise Time		0.1	µs
t_{CHCL}	Fall Time		0.1	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		1	%

34.3 System and Reset Characteristics

Table 34-4. Reset, Brown-out and Internal Voltage Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			TBD		V
	Power-on Reset Threshold Voltage (falling)[1]			0.3		V
V_{PSR}	Power-on slope rate			TBD		V/ms
V_{RST}	RSTN Pin Threshold Voltage		$0.2V_{DEVDD}$		$0.9V_{DEVDD}$	V
t_{RST}	Minimum pulse width on RSTN Pin			TBD		ns
V_{HYS}	Brown-out Detector Hysteresis			50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset			100		ns
V_{BG}	Bandgap reference voltage	V_{DEVDD} = 3.0V, T_A = 25°C		1.2		V

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

Table 34-23. BODLEVEL Fuse Coding

BODLEVEL2:0 Fuses	Min V_{BOD}	Typ V_{BOD}	Max V_{BOD}	Units
111	BOD Disabled			
110		1.8		V
101		1.9		V
100		2.0		V
011		2.1		V
010		2.2		V
001		2.3		V
000		2.4		V

Note: V_{BOT} may be below nominal minimum operating voltage. The device is operated down to V_{DEVDD} = V_{BOT} during the production test. This guarantees that a Brown-Out Reset will occur before V_{DEVDD} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 110 for 16 MHz operation of the ATmega128RFA1.

34.4 Power Management Electrical Characteristics

34.4.1 Power Switches

Table 34-6. Timing Characteristics of the Power Switches

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{POR}	Power-on reset time	Applies if the device is powered up		TBD		µs
t_{BG}	Bandgap startup time			7		µs
t_{DRT_ON}	DRT switch switch-on time			2		µs
t_{PWRSW_ON}	Power switch switch-on time			2		µs

34.4.2 Voltage Regulators

Table 34-7. Timing Characteristics of the Voltage regulators

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{AVREG}	Power up time AVREG	external capacity on pin AVDD TBD		TBD		µs
t_{DVREG}	Power up time DVREG	Startup after wakeup, Startup after deep sleep, external capacity on pin DVDD TBD TRX24 and all SRAM modules enabled)		30		µs
t_{BG}	Power up time bandgap			TBD		µs

34.5 2-wire Serial Interface Characteristics

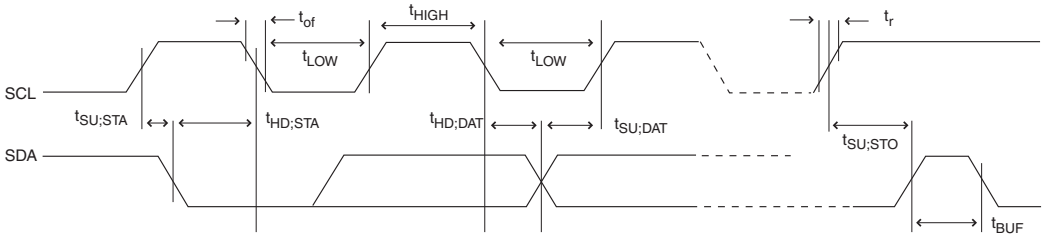
The timing characteristics refer to Table 34-8.

Table 34-8 2-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min.	Max.	Units
V_{IL}	Input Low-voltage		-0.5	$0.3V_{DEVDD}$	V

Symbol	Parameter	Condition	Min.	Max.	Units
V_{IH}	Input High-voltage		$0.7V_{DEVDD}$	V_{DEVDD} +0.5	V
V_{hys}	Hysteresis of Schmitt Trigger Inputs				V
V_{OL}	Output Low-voltage	3mA sink current	0	0.4	V
t_r	Rise Time for both SDA and SCL			300	ns
t_{of}	Output Fall Time from V_{IHmin} to V_{ILmax}			250	ns
t_{SP}	Spikes suppressed by the input filter			50	ns

Figure 34-2



34.6 SPI Timing Characteristics

See Figure 34-3and Figure 34-4 for details.

Table 34-9. SPI Timing Parameters

Description	Mode	Min	Typ	Max	Units
SCK period	Master		See "SPCR – SPI Control Register" on page 336.		ns
SCK high/low	Master		50% duty cycle		
Rise/fall time	Master		TBD		
Setup	Master		TBD		
Hold	Master		TBD		
Out to SCK	Master		0.5 t_{SCK}		
SCK to out	Master		TBD		
SCK to out high	Master		TBD		
$\overline{SS}$ low to out	Slave		TBD		
SCK period	Slave	4 t_{CK}			
SCK high/low(1)	Slave	2 t_{CK}			
Rise/fall time	Slave			TBD	
Setup	Slave	TBD			
Hold	Slave	t_{CK}			
SCK to out	Slave		TBD		
SCK to $\overline{SS}$ high	Slave	TBD			
$\overline{SS}$ high to tri-state	Slave		TBD		
$\overline{SS}$ low to SCK	Slave	TBD			

Note: 1. In SPI Programming mode the minimum SCK high/low period is 2 t_{CLCL} for f_{CK} < 12 MHz and 3 t_{CLCL} for f_{CK} > 12 MHz.

Figure 34-3. SPI timing Requirements (Master Mode)

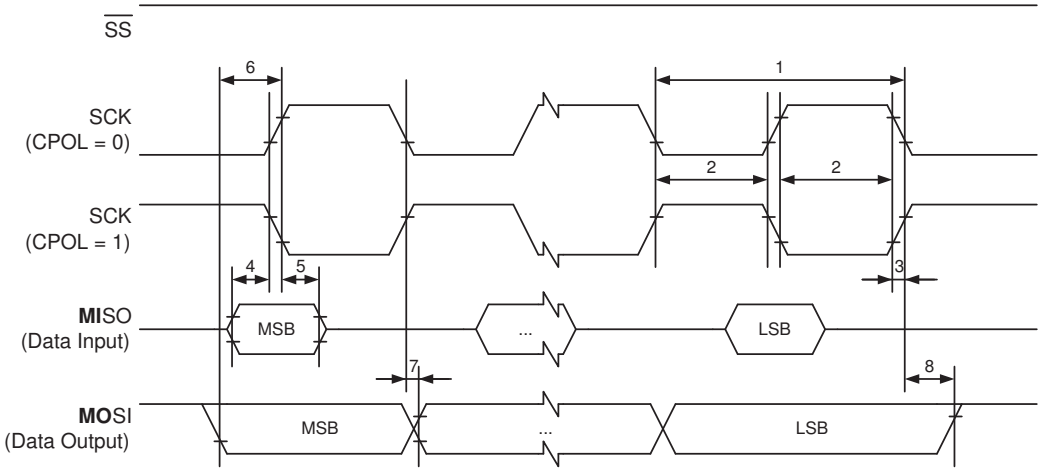
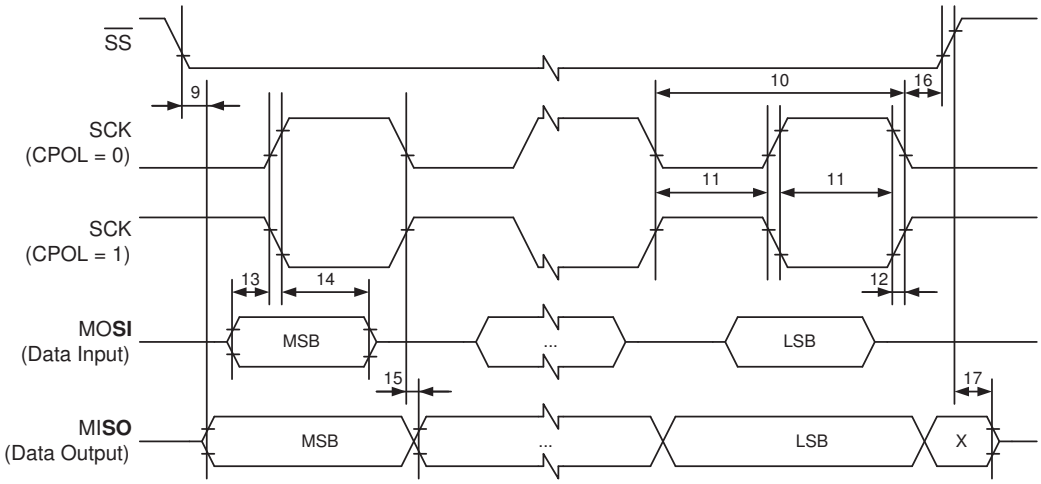


Figure 34-4. SPI timing Requirements (Slave Mode)



34.7 ADC Characteristics

Table 34-10. ADC Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{INT1}	Internal Voltage Reference			1.5		V
V_{INT2}	Internal Voltage Reference			1.6		V
V_{INT3}	Internal Voltage Reference			AVDD		V
$R_{AREF,EXT}$	External Voltage Impedance					Ω
$I_{L,AREF}$	Load Current					A

Table 34-11. ADC Characteristics, Single Ended Channels

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution	Single Ended Conversion CLKADC <= 4 MHz		10		Bits
		Single Ended Conversion CLKADC > 8 MHz		8		Bits
	Absolute accuracy (Including	Single Ended Conversion		TBD		LSB

Symbol	Parameter	Condition	Min[1]	Typ[1]	Max[1]	Units
	INL, DNL, quantization error, gain and offset error)	VREF = 1.6V CLKADC = 200kHz				
		Single Ended Conversion VREF = 1.6V CLKADC = 1MHz		TBD		LSB
		Single Ended Conversion VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		TBD		LSB
	Integral Non-Linearity (INL)	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		0.8		LSB
	Differential Non-Linearity (DNL)	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz	-0.5			LSB
	Gain Error	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		1		LSB
	Offset Error	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		1.5		LSB
	Conversion Time	Free Running Conversion	3		240	µs
	Clock Frequency	Single Ended Conversion			8000	kHz
V_{REF}	Reference Voltage		1.5		AVDD	V
V_{IN}	Input Voltage		0		AVDD	V
	Input Bandwith		20			kHz
R_{REF}	Reference Input Resistance			TBD		kΩ
R_{AIN}	Analog Input Resistance			2		kΩ

Note: 1. Values are guidelines only
2. All values are valid for EVDD = 3.0V

Table 34-12. PGA and ADC Characteristics, Differential Channels

Symbol	Parameter	Condition	Min[1]	Typ[1]	Max[1]	Units
	Resolution	Gain = 1x		10		Bits
		Gain = 10x		10		Bits
		Gain = 200x		10		Bits
	Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Gain = 1x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
	Integral Non-Linearity (INL)	Gain = 1x VREF = 1.6V CLKADC = 2MHz			3	LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz			5	LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz			10	LSB
	Differential Non-Linearity (DNL)	Gain = 1x VREF = 1.6V CLKADC = 2MHz	-0.5			LSB

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
		Gain = 10x VREF = 1.6V CLKADC = 2MHz	-0.75			LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz	TBD			LSB
	Gain Error	Gain = 1x		1		LSB
		Gain = 10x		1.5		
		Gain = 200x		10		
	Offset Error	Gain = 1x VREF = 1.6V CLKADC = 2MHz		0.7		LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz		0.75		LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz		13		LSB
	Conversion Time	Free Running Conversion	100			µs
	Clock Frequency	Single Ended Conversion			2000	kHz
V_{REF}	Reference Voltage		1.5		AVDD	V
V_{IN}	Input Voltage		0		AVDD	V
V_{DIFF}	Input Differential Voltage		-AVDD		AVDD	V
	ADC Conversion Output		-512		511	LSB
	Input Bandwith		20			kHz
R_{REF}	Reference Input Resistance			TBD		kΩ
R_{AIN}	Analog Input Resistance			2		kΩ

Note: 1. Values are guidelines only
2. All values are valid for EVDD = 3.0V

34.8 Transceiver Electrical Characteristics

34.8.1 Digital Interface Timing Characteristics

Test Conditions: T_{OP} = 25°C, V_{DD} = 3.0V, C_L = 50 pF (unless otherwise stated)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{12}	AES core cycle time			24		µs
t_{IRQ}	Interrupt event latency	Relative to the event to be indicated		9		µs

34.8.2 General RF Specifications

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_{RF}	Frequency range	As specified in [1],[2]	2405		2480	MHz
f_{CH}	Channel spacing	As specified in [1],[2]		5		MHz
f_{HDR}	Header bit rate (SHR, PHR)	As specified in [1],[2]		250		kb/s

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_{PSDU}	PSDU bit rate	As specified in [1],[2] OQPSK_DATA_RATE = 1 OQPSK_DATA_RATE = 2 OQPSK_DATA_RATE = 3		250 500 1000 2000		kb/s kb/s kb/s kb/s
f_{CHIP}	Chip rate	As specified in [1],[2]		2000		kchip/s
f_{CLK}	Crystal oscillator frequency	Reference oscillator		16		MHz
t_{XTAL}	Reference oscillator settling time	Leaving SLEEP state to clock available at pin 17 (CLKM)		215	1000	µs
	Symbol rate deviation Reference frequency accuracy for correct functionality	PSDU bit rate 250 kb/s 500 kb/s 1000 kb/s 2000 kb/s	-60(1) -40 -40 -30		+60 +40 +40 +30	ppm ppm ppm ppm
B_{20dB}	20 dB bandwidth			2.8		MHz

Note: 6. A reference frequency accuracy of ±40 ppm is required by [1], [2].

34.8.3 Transmitter Characteristics

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
P_{TX}	TX Output power	Maximum configurable TX output power value Register bit TX_PWR = 0	0	+3.5	+6	dBm
P_{RANGE}	Output power range	16 steps, configurable in register 0x05 (PHY_TX_PWR)		20		dB
P_{ACC}	Output power tolerance				±3	dB
	TX Return loss	100Ω differential impedance, P_{TX} = +3.5 dBm		10		dB
	EVM			8		%rms
P_{HARM}	Harmonics 2nd harmonic 3rd harmonic			 -38 -45		 dBm dBm
P_{SPUR}	Spurious Emissions 30 – ≤ 1000 MHz >1 – 12.75 GHz 1.8 – 1.9 GHz 5.15 – 5.3 GHz	Complies with EN 300 328/440, FCC-CFR-47 part 15, ARIB STD-66, RSS-210		 -36 -30 -47 -47		 dBm dBm dBm dBm

34.8.4 Receiver Characteristics

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, PSDU bit rate = 250 kb/s, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
P_{SENS}	Receiver sensitivity 250 kb/s 500 kb/s 1000 kb/s 2000 kb/s	AWGN channel, PER ≤ 1%, PSDU length 20 octets High Data Rate Modes: PSDU length 20 octets		 -100 -96 -94 -86		 dBm dBm dBm dBm
	Antenna Diversity	250 kb/s, PSDU 20 octets		-99		dBm
RL	Return loss	100Ω differential impedance		10		dB
NF	Noise figure			6		dB
P_{RXMAX}	Maximum RX input level	PER ≤ 1%, PSDU length of 20 octets		10		dBm
P_{ACRN}	Adjacent channel rejection: -5 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		34		dB
P_{ACRP}	Adjacent channel rejection: +5 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		38		dB
P_{AACRN}	Alternate channel rejection: -10 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		54		dB
P_{AACRP}	Alternate channel rejection: +10 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		54		dB
P_{SPUR}	Spurious emissions: LO leakage 30 – ≤ 1000 MHz >1 – 12.75 GHz			 -71	 -57 -47	 dBm dBm dBm
$f_{RXTXOFFS}$	TX/RX carrier frequency offset	Sensitivity loss < 2 dB	-300[1]		+300	kHz
IIP3	3rd – order intercept point	At maximum gain Offset freq. interf. 1 = 5 MHz Offset freq. interf. 2 = 10 MHz		-14		dBm
IIP2	2nd – order intercept point	At maximum gain Offset freq. interf. 1 = 60 MHz Offset freq. interf. 2 = 62 MHz		17		dBm
	RSSI tolerance	Tolerance within gain step			±5	dB
	RSSI dynamic range			81		dB
	RSSI resolution			3		dB
	RSSI sensitivity	Defined as RSSI_BASE_VAL		-90		dBm
	Minimum RSSI value	P_{RF} ≤ RSSI_BASE_VAL		0		
	Maximum RSSI value	P_{RF} > RSSI_BASE_VAL + 81 dB		28		

Note: 1. Offset equals ±120 ppm

34.8.5 Current Consumption Specifications

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{BUSY_TX}	Supply current transmit state	P_{TX} = 3 dBm P_{TX} = 1 dBm P_{TX} = -3 dBm P_{TX} = -17 dBm (current consumption is reduced at V_{DD} = 1.8V for each output power level)		14.5 10 9 8		mA mA mA mA
I_{RX_ON}	Supply current RX_ON state	RX_ON state		12.5		mA
$I_{RX\_ON\_P}$	Supply current RX_ON state	RX_ON state, with register setting RX_PDT_LEVEL > 0[1]		12.0		mA
I_{PLL_ON}	Supply current PLL_ON state	PLL_ON state		5.7		mA
I_{TRX_OFF}	Supply current TRX_OFF state	TRX_OFF state		0.4		mA
I_{SLEEP}	Supply current SLEEP state	SLEEP state		0.02		µA

Note: 1. Refer to section "Figure 32-1" on page 493

34.8.6 Crystal Parameter Requirements

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_0	Crystal frequency			16		MHz
C_L	Load capacitance		8		14	pF
C_0	Static capacitance				7	pF
R_1	Series resistance				100	Ω

35 Typical Characteristics

35.1 Internal Oscillator Speed

t.b.d.

36 Ordering Information

ATmega128RFA1

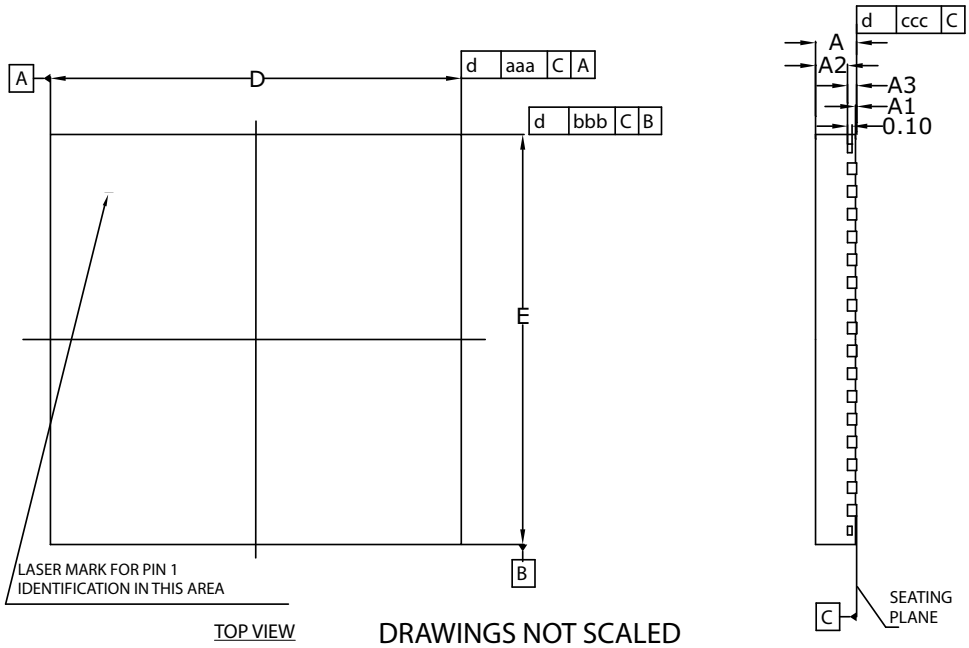
Speed (MHz)	Power Supply	Ordering Code	Package	Packing	Operation Range
16	1.8 – 3.6V	ATmega128RFA1-ZU	PI	Tray	Industrial (-40ºC to 85ºC)
16	1.8 – 3.6V	ATmega128RFA1-ZUR	PI	Tape & Reel	Industrial (-40ºC to 85ºC)

Notes: 1. Pb-free packaging, complies to European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

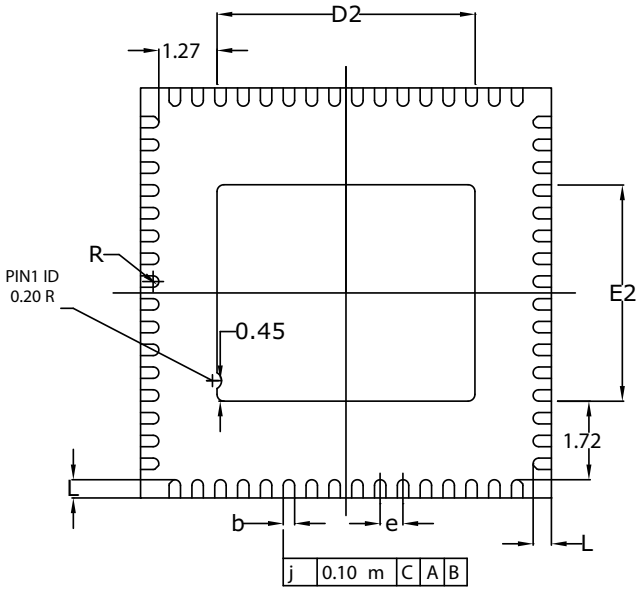
Package Type	
PI	64-lead, 9 x 9 x 0.9 mm Body, Quad Flat No-lead Package (QFN)

37 Packaging Information

PI



DRAWINGS NOT SCALED



SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	---	---	0.90	---	---	0.035
A1	---	---	0.05	---	---	0.001
A2	---	0.65	0.70	---	0.026	0.028
A3	0.20 REF.			0.008 REF.		
b	0.18	0.25	0.30	0.007	0.010	0.012
D	9.00 bsc			0.354 bsc		
D2	5.55	5.65	5.75	0.219	0.222	0.226
E	9.00 bsc			0.354 bsc		
E2	4.65	4.75	4.85	0.183	0.187	0.191
L	0.35	0.40	0.45	0.014	0.016	0.018
e	0.50 bsc			0.020 bsc		
R	0.09	---	---	0.004	---	---
TOLERANCES OF FORM AND POSITION						
aaa	0.10			0.004		
bbb	0.10			0.004		
ccc	0.05			0.002		

ALL DIMENSIONS ARE IN MILLIMETERS.

PACKAGE WARPAGE MAX 0.08 mm.

02/12/2008

Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	TITLE PI - 64 leads - 9.0 x 9.0 mm - pitch 0.5mm Quad Flat No Lead Package QFN	DRAWING No. PI	REV. A

38 Errata

38.1 ATmega128RFA1 revision D (1.2)

- **Power-Chain turns off when power supply drops below 1.6V**
- **JTAG interface reads wrong data**
- **CSMA back-off calculation has reduced degree of randomness**
- **Update of internal temporary registers for CSMA_SEED register may fail**
- **Interrupt TRX24_CCA_ED_DONE may occur twice**

38.2 ATmega128RFA1 revision C (1.1)

- **Power-Chain turns off when power supply drops below 1.6V**
- **JTAG interface reads wrong data**
- **CSMA back-off calculation has reduced degree of randomness**
- **Update of internal temporary registers for CSMA_SEED register may fail**
- **Interrupt TRX24_CCA_ED_DONE may occur twice**
- **DVREG_EXT bit is not write-protected**
- **ENDRT bits have wrong reset value**

38.3 ATmega128RFA1 revision AB (1.0)

Not sampled.

38.4 Compiler package WinAVR-20090313

In the compiler package WinAVR-20090313 the SRAM start address has a wrong value of 0x100. In this case the variables are randomly allocated across the extended I/O space 0x100 to 0x1FF. It causes an unpredictable behavior by random overwrite of registers (see also "JTAG interface reads wrong data" on page 514 and "DVREG_EXT bit is not write-protected" on page 514).

Problem Fix/Workaround

Use the linker option -Wl,--section-start=.data=0x800200

38.5 Detailed errata description

38.5.1 Power-Chain turns off when power supply drops below 1.6V

If the voltage of the pins DEVDD drops below 1.6V, the internal power chain turns off. Some hardware settings (e.g. clock source) can alter their state unintentionally. Raising the supply voltage above 1.8V again does not bring the circuit back to normal operation. This condition can happen either by lowering the power supply voltage below 1.6V or turn-off the supply source while other external devices are feeding DEVDD by the internal ESD diodes of the IO stages (e.g. hardware debugger attached to the JTAG interface) (2606).

If the power supply drops below 1.6V while being in Deep Sleep mode, the internal power chain is not affected.

Problem Fix/Workaround

Turn on the Brown-Out Detector at any voltage level. The supply current in Deep Sleep does not increase.

38.5.2 JTAG interface reads wrong data

If the Power Reduction Register bits associated with the SRAM's (PRRAM3…0 in PRR2) and the 2.4GHz Transceiver (PRTRX24 in PRR1) are set, the JTAG interface reads wrong data. (2613).

Problem Fix/Workaround

Do not use PRRAM3…0 in PRR2 and PRTRX24 in PRR1. Force pin RSTN=0 and the JTAG interface can erase the program memory.

38.5.3 CSMA back-off calculation has reduced degree of randomness

The CSMA back-off calculation in the transceiver extended operating modes has a reduced degree of randomness (e.g. transceiver is in the state TX_ARET_ON) (2665).

Problem Fix/Workaround

Initialize CSMA_SEED registers with a random value.

38.5.4 Update of internal temporary registers for CSMA_SEED register may fail

The update of the internal temporary registers of the CSMA_SEED registers may fail. Read/write operation to the CSMA_SEED registers itself works as expected (2646).

Problem Fix/Workaround

A sleep cycle of the transceiver updates the internal temporary registers.

38.5.5 Interrupt TRX24_CCA_ED_DONE may occur twice

When requesting a manually initiated CCA measurement in BUSY_RX state and during an internal ED measurement, a TRX24_CCA_ED_DONE interrupt could be issued immediately after the request. In this case the register bit CCA_DONE is equal to 0 and an additional TRX24_CCA_ED_DONE interrupt is issued after finishing the CCA measurement and register bit CCA_DONE is set to 1 (2000).

Problem Fix/Workaround

Prevent a frame reception during manually initiated CCA measurement

- make sure that TRX_STATUS is not in RX_BUSY (i.e. start from state PLL_ON)
- set bit RX_PDT_DIS=1
- switch TRX_STATE to RX_ON
- perform CCA measurement
- set bit RX_PDT_DIS=0

38.5.6 DVREG_EXT bit is not write-protected

The external mode of the DVDD voltage regulator is not write-protected. If it is enabled (DVREG_EXT=1 in the register VREG_CTRL) with no external power supply for DVDD, the device leaves normal operation and can't be recovered by the Watchdog (2658).

Problem Fix/Workaround

Do not write the bit DVREG_EXT in the register VREG_CTRL.

38.5.7 ENDRT bits have wrong reset value

The ENDRT bits in the registers DRTRAM3…0 have the wrong reset value. The data retention of the associated SRAM in DEEP_SLEEP is disabled (2495).

Problem Fix/Workaround

Set ENDRT=1 in DRTRAM3…0 at the beginning of the firmware program.

39 Revision history

Please note that the referring page numbers in this section are referring to this document. The referring revision in this section are referring to the document revision

Rev. 8266A-MCU Wireless-12/09

1. Initial release

Table of Contents

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature