



Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

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

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	4KB (2K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atmega48-15at1

4.5 General Purpose Register File

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

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

Figure 4-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4-2. AVR CPU General Purpose Working Registers

	7	0	Addr.	
General Purpose Working Registers	R0		0x00	
	R1		0x01	
	R2		0x02	
	...			
	R13		0x0D	
	R14		0x0E	
	R15		0x0F	
	R16		0x10	
	R17		0x11	
	...			
	R26		0x1A	X-register Low Byte
	R27		0x1B	X-register High Byte
	R28		0x1C	Y-register Low Byte
	R29		0x1D	Y-register High Byte
	R30		0x1E	Z-register Low Byte
	R31		0x1F	Z-register High Byte

Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

6.1.4 Asynchronous Timer Clock – clk_{ASY}

The asynchronous timer clock allows the asynchronous Timer/Counter to be clocked directly from an external clock or an external 32kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

6.1.5 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

6.2 Clock Sources

The device has the following clock source options, selectable by flash fuse bits as shown below. The clock from the selected source is input to the AVR[®] clock generator, and routed to the appropriate modules.

Table 6-1. Device Clocking Options Select⁽¹⁾

Device Clocking Option	CKSEL3..0
Low power crystal oscillator	1111 - 1000
Full swing crystal oscillator	0111 - 0110
Low frequency crystal oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External clock	0000
Reserved	0001

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

6.2.1 Default Clock Source

The device is shipped with internal RC oscillator at 8.0MHz and with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock. The startup time is set to maximum and time-out period enabled. (CKSEL = “0010”, SUT = “10”, CKDIV8 = “0”). The default setting ensures that all users can make their desired clock source setting using any available programming interface.

6.2.2 Clock Startup Sequence

Any clock source needs a sufficient V_{CC} to start oscillating and a minimum number of oscillating cycles before it can be considered stable.

To ensure sufficient V_{CC} , the device issues an internal reset with a time-out delay (t_{TOUR}) after the device reset is released by all other reset sources. Section 8. “System Control and Reset” on page 38 describes the start conditions for the internal reset. The delay (t_{TOUR}) is timed from the watchdog oscillator and the number of cycles in the delay is set by the SUTx and CKSELx fuse bits. The selectable delays are shown in Table 6-2. The frequency of the watchdog oscillator is voltage dependent as shown in Section “” on page 285.

Table 6-2. Number of Watchdog Oscillator Cycles

Typ Time-out ($V_{CC} = 5.0V$)	Typ Time-out ($V_{CC} = 3.0V$)	Number of Cycles
0ms	0ms	0
4.1ms	4.3ms	4K (4,096)
65ms	69ms	8K (8,192)

7.2 Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter idle mode, stopping the CPU but allowing the SPI, USART, analog comparator, ADC, 2-wire serial interface, Timer/Counters, watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH} , while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the timer overflow and USART transmit complete interrupts. If wake-up from the analog comparator interrupt is not required, the analog comparator can be powered down by setting the ACD bit in the analog comparator control and status register – ACSR. This will reduce power consumption in idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

7.3 ADC Noise Reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC noise reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the 2-wire serial interface address watch, Timer/Counter2, and the watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU} , and clk_{FLASH} , while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC conversion complete interrupt, only an external reset, a watchdog system reset, a watchdog interrupt, a brown-out reset, a 2-wire serial interface address match, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or INT1 or a pin change interrupt can wake up the MCU from ADC noise reduction mode.

7.4 Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter power-down mode. In this mode, the external oscillator is stopped, while the external interrupts, the 2-wire serial interface address watch, and the watchdog continue operating (if enabled). Only an external reset, a watchdog system reset, a watchdog interrupt, a brown-out reset, a 2-wire serial interface address match, an external level interrupt on INT0 or INT1, or a pin change interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 11. “External Interrupts” on page 73 for details.

When waking up from power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL fuses that define the reset time-out period, as described in Section 6.2 “Clock Sources” on page 24.

7.5 Power-save Mode

When the SM2..0 bits are written to 011, the SLEEP instruction makes the MCU enter power-save mode. This mode is identical to power-down, with one exception:

If Timer/Counter2 is enabled, it will keep running during sleep. The device can wake up from either timer overflow or output compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK2, and the global interrupt enable bit in SREG is set.

If Timer/Counter2 is not running, power-down mode is recommended instead of power-save mode.

The Timer/Counter2 can be clocked both synchronously and asynchronously in power-save mode. If Timer/Counter2 is not using the asynchronous clock, the Timer/Counter oscillator is stopped during sleep. If Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Note that even if the synchronous clock is running in power-save, this clock is only available for Timer/Counter2.

10.3.3 Alternate Functions of Port C

The port C pins with alternate functions are shown in Table 10-6.

Table 10-6. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC6	$\overline{\text{RESET}}$ (reset pin) PCINT14 (pin change interrupt 14)
PC5	ADC5 (ADC input channel 5) SCL (2-wire serial bus clock line) PCINT13 (pin change interrupt 13)
PC4	ADC4 (ADC input channel 4) SDA (2-wire serial bus data input/output line) PCINT12 (pin change interrupt 12)
PC3	ADC3 (ADC input channel 3) PCINT11 (pin change interrupt 11)
PC2	ADC2 (ADC input channel 2) PCINT10 (pin change interrupt 10)
PC1	ADC1 (ADC input channel 1) PCINT9 (pin change interrupt 9)
PC0	ADC0 (ADC input channel 0) PCINT8 (pin change interrupt 8)

The alternate pin configuration is as follows:

- **$\overline{\text{RESET}}$ /PCINT14 – Port C, Bit 6**

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

If PC6 is used as a reset pin, DDC6, PORTC6 and PINC6 will all read 0.

PCINT14: pin change interrupt source 14. The PC6 pin can serve as an external interrupt source.

- **SCL/ADC5/PCINT13 – Port C, Bit 5**

SCL, 2-wire serial interface clock: When the TWEN bit in TWCR is set (one) to enable the 2-wire serial interface, pin PC5 is disconnected from the port and becomes the serial clock I/O pin for the 2-wire serial interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

PC5 can also be used as ADC input channel 5. Note that ADC input channel 5 uses digital power.

PCINT13: pin change interrupt source 13. The PC5 pin can serve as an external interrupt source.

- **SDA/ADC4/PCINT12 – Port C, Bit 4**

SDA, 2-wire serial interface data: When the TWEN bit in TWCR is set (one) to enable the 2-wire serial interface, pin PC4 is disconnected from the port and becomes the serial data I/O pin for the 2-wire serial interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

PC4 can also be used as ADC input Channel 4. Note that ADC input channel 4 uses digital power.

PCINT12: pin change interrupt source 12. The PC4 pin can serve as an external interrupt source.

- **ADC3/PCINT11 – Port C, Bit 3**

PC3 can also be used as ADC input channel 3. Note that ADC input channel 3 uses analog power.

PCINT11: pin change interrupt source 11. The PC3 pin can serve as an external interrupt source.

Bits 5:4 – COM0B1:0: Compare Match Output B Mode

These bits control the output compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B 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 OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting.

Table 12-5 on page 88 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 12-5. Compare Output Mode, non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

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

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

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

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

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

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

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

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

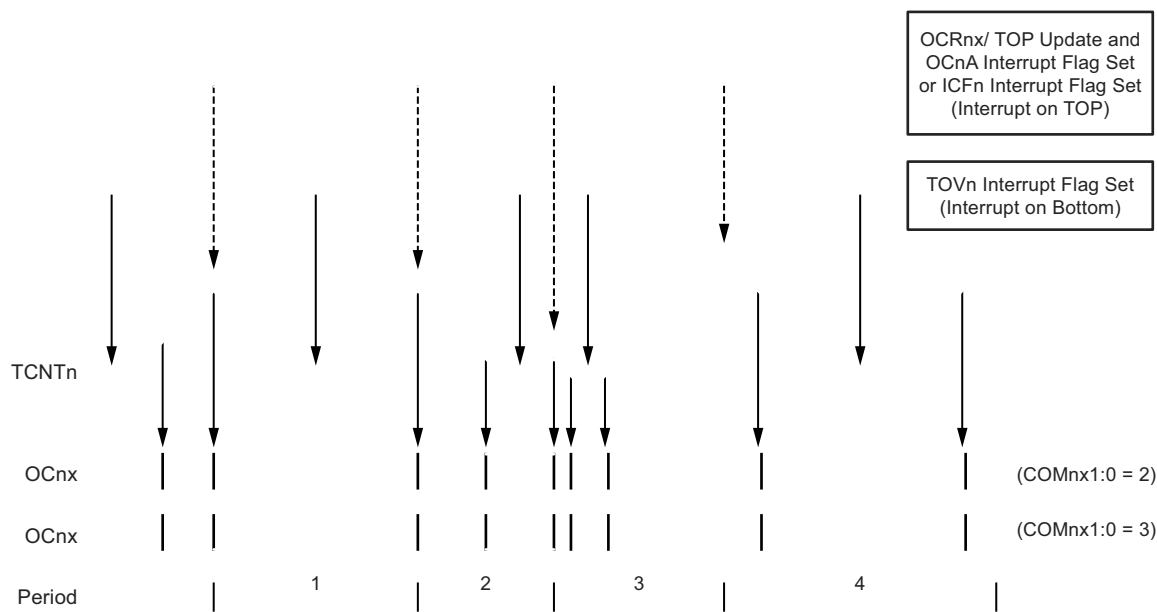
• Bits 3, 2 – Res: Reserved Bits

These bits are reserved bits in the Atmel® ATmega48/88/168 and will always read as zero.

• Bits 1:0 – WGM01:0: Waveform Generation Mode

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 12-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 12.6 “Modes of Operation” on page 81).

Figure 14-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag is set accordingly at the same timer clock cycle as the OCR1x registers are updated with the double buffer value (at TOP). The interrupt flags can 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 TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x registers are written. As the third period shown in Figure 14-8 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x register. Since the OCR1x 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 differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on 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 generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table on page 114). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

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

The extreme values for the OCR1x register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x 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.

Table 14-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{I/O} /1 (no prescaling)
0	1	0	clk _{I/O} /8 (from prescaler)
0	1	1	clk _{I/O} /64 (from prescaler)
1	0	0	clk _{I/O} /256 (from prescaler)
1	0	1	clk _{I/O} /1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

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.

14.10.3 Timer/Counter1 Control Register C – TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**
- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B 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/FOC1B bits are always read as zero.

14.10.4 Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

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 14.2 “Accessing 16-bit Registers” on page 96.

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.

14.10.5 Output Compare Register 1 A – OCR1AH and OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

14.10.6 Output Compare Register 1 B – OCR1BH and OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
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	

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 OC1x 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 14.2 “Accessing 16-bit Registers” on page 96.

14.10.7 Input Capture Register 1 – ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
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	

The input capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the analog comparator output for Timer/Counter1). The input capture 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 14.2 “Accessing 16-bit Registers” on page 96.

14.10.8 Timer/Counter1 Interrupt Mask Register – TIMSK1

Bit	7	6	5	4	3	2	1	0	
	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega48/88/168, and will always read as zero.

- **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 (see Section 9. “Interrupts” on page 48) is executed when the ICF1 flag, located in TIFR1, is set.

- **Bit 4, 3 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega48/88/168, and will always read as zero.

- **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 (see Section 9. “Interrupts” on page 48) 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 (see Section 9. “Interrupts” on page 48) 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 (see Section 8.9 “Watchdog Timer” on page 44) is executed when the TOV1 flag, located in TIFR1, is set.

14.10.9 Timer/Counter1 Interrupt Flag Register – TIFR1

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega48/88/168, and will always read as zero.

- **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, 3 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATmega48/88/168, and will always read as zero.

- **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. In normal and CTC modes, the TOV1 flag is set when the timer overflows. Refer to Table 14-5 on page 114 for the TOV1 flag behavior when using another WGM13:0 bit setting.

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.

The function simply waits for data to be present in the receive buffer by checking the RXCn flag, before reading the buffer and returning the value.

17.6.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8n bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FEn, DORn and UPEn status flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8n, FEn, DORn and UPEn 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⁾
<pre> 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 </pre>
C Code Example ⁽¹⁾
<pre> 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); } </pre>

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

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.

19.2.2 Electrical Interconnection

As depicted in Figure 19-1 on page 175, 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 tri-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.

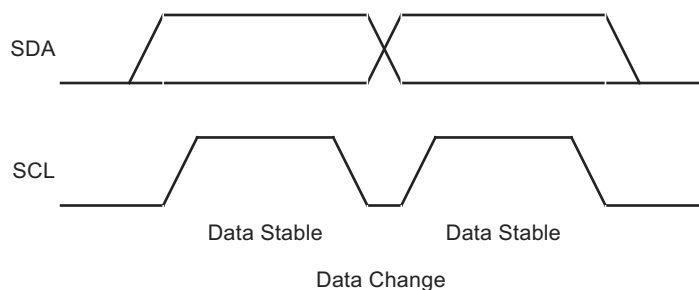
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in Section 27. “2-wire Serial Interface Characteristics” on page 266. Two different sets of specifications are presented there, one relevant for bus speeds below 100kHz, and one valid for bus speeds up to 400kHz.

19.3 Data Transfer and Frame Format

19.3.1 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.

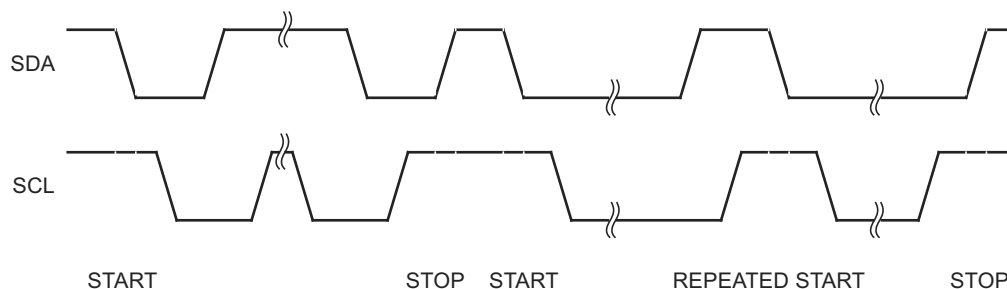
Figure 19-2. Data Validity



19.3.2 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 signalled by changing the level of the SDA line when the SCL line is high.

Figure 19-3. START, REPEATED START and STOP Conditions



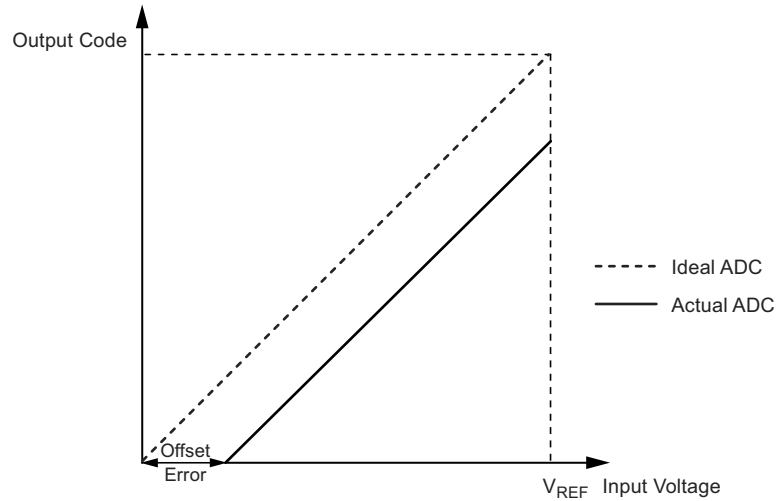
21.5.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). 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 21-10. 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 21-11. Gain Error

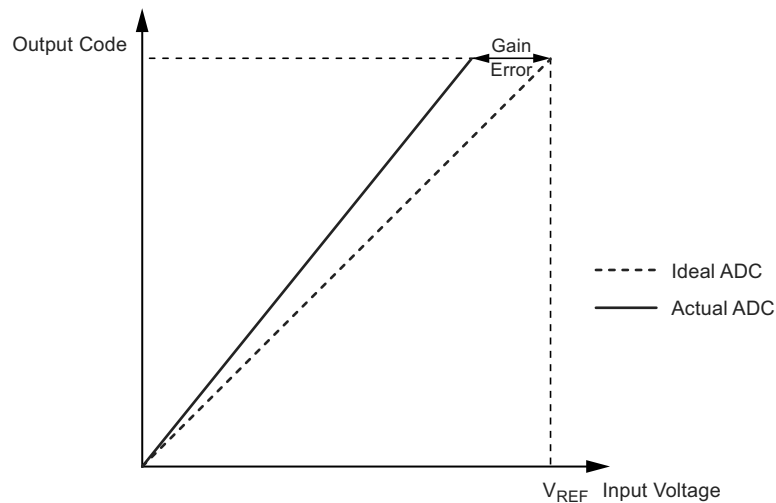


Figure 24-1. Read-While-Write versus No Read-While-Write

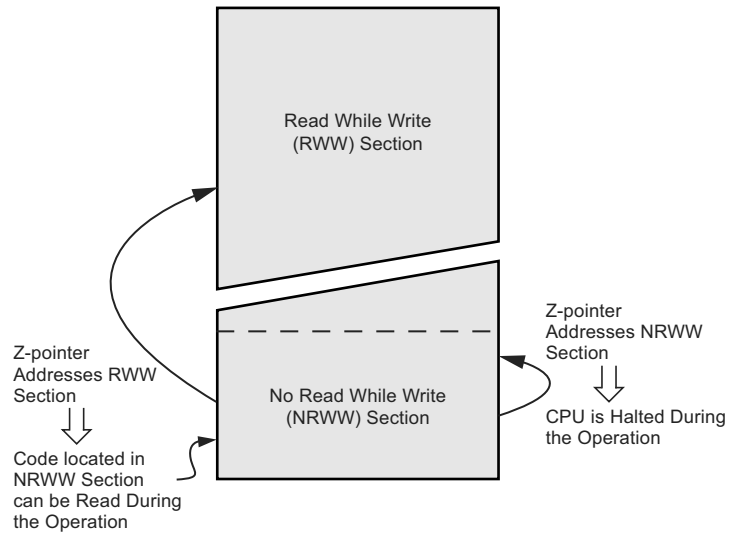
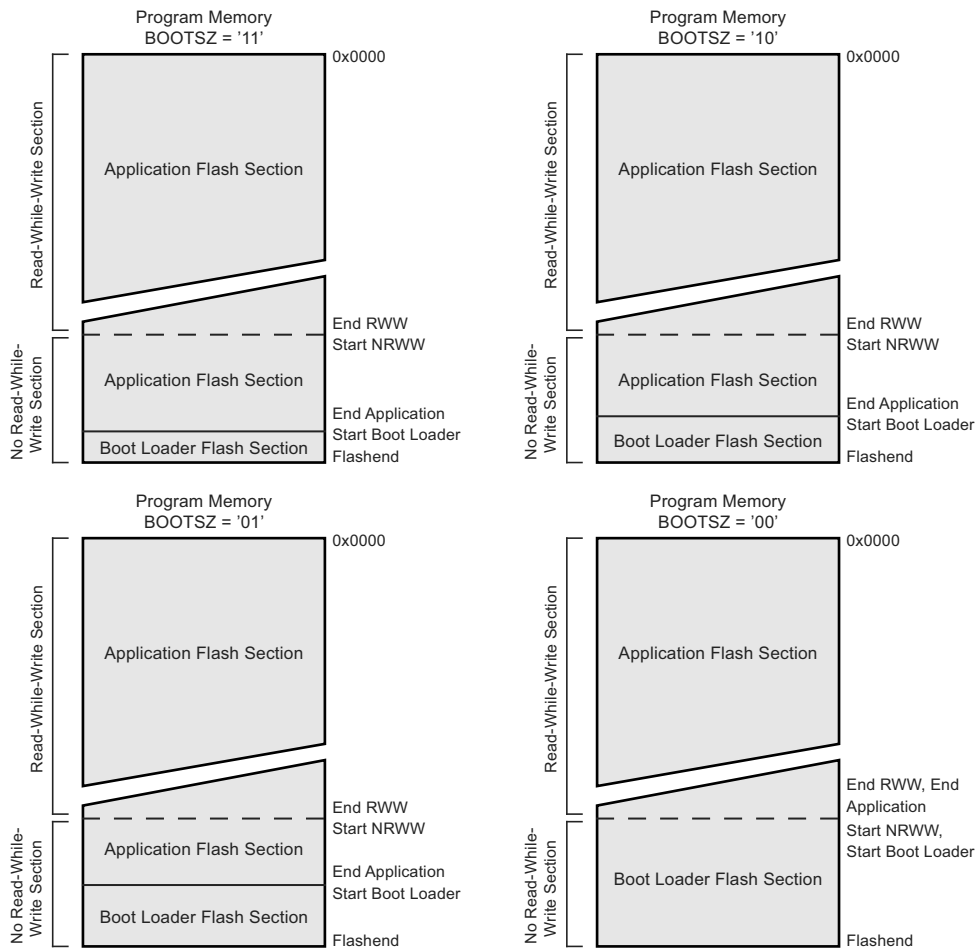


Figure 24-2. Memory Sections



Note: 1. The parameters in the figure above are given in Table 24-6 on page 240.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SELFPRGEN, 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 SELFPRGEN, 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 – SELFPRGEN: Self Programming Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT or PGERS, the following SPM instruction will have a special meaning, see description above. If only SELFPRGEN 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 SELFPRGEN 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 SELFPRGEN bit remains high until the operation is completed.

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

24.6 Addressing the Flash During Self-Programming

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

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

Since the flash is organized in pages (see Table 25-12 on page 247), 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 24-3 on page 235. Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the boot loader software addresses the same page in both the page erase and page write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

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

24.7.2 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 “0000001” 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 auto-erase 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 will be lost.

24.7.3 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.

24.7.4 Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SELFPRGEN 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 Section 8.9 “Watchdog Timer” on page 44.

24.7.5 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 unprogrammed. 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.

24.7.6 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 Section 8.9 “Watchdog Timer” on page 44, 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 Section 24.7.12 “Simple Assembly Code Example for a Boot Loader” on page 238 for an example.

24.7.7 Setting the Boot Loader Lock Bits by SPM

To set the boot loader lock bits, write the desired data to R0, write “X0001001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The only accessible lock bits are the boot lock bits that may prevent the application and boot loader section from any software update by the MCU.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

See Table 24-2 on page 232 and Table 24-3 on page 232 for how the different settings of the boot loader bits affect the flash access.

If bits 5..2 in R0 are cleared (zero), the corresponding boot lock bit will be programmed if an SPM instruction is executed within four cycles after BLBSET and SELFPRGEN 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 IO_{ck} bits). For future compatibility it is also recommended to set bits 7, 6, 1, and 0 in R0 to “1” when writing the lock bits. When programming the lock bits the entire flash can be read during the operation.

25.7.2 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.

25.7.3 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 XTAL1 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.

25.7.4 Programming the Flash

The flash is organized in pages, see Table 25-12 on page 247. 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 XTAL1 a positive pulse. This loads the command.

B. Load address low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = address low byte (0x00 - 0xFF).
4. Give XTAL1 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 XTAL1 a positive pulse. This loads the data byte.

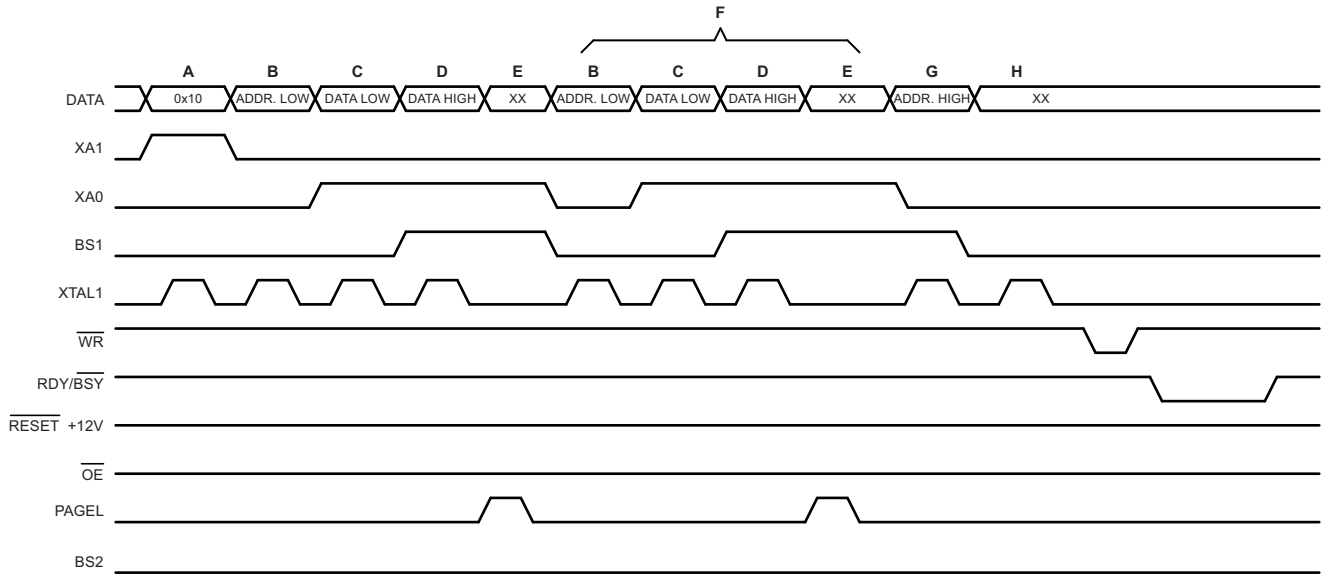
D. Load data high byte

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 XTAL1 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 25-3 on page 250 for signal waveforms)

Figure 25-3. Programming the Flash Waveforms⁽¹⁾



Note: 1. "XX" is don't care. The letters refer to the programming description above.

25.7.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 25-13 on page 247. 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 Section 25.7.4 "Programming the Flash" on page 248 for details on command, address and data loading):

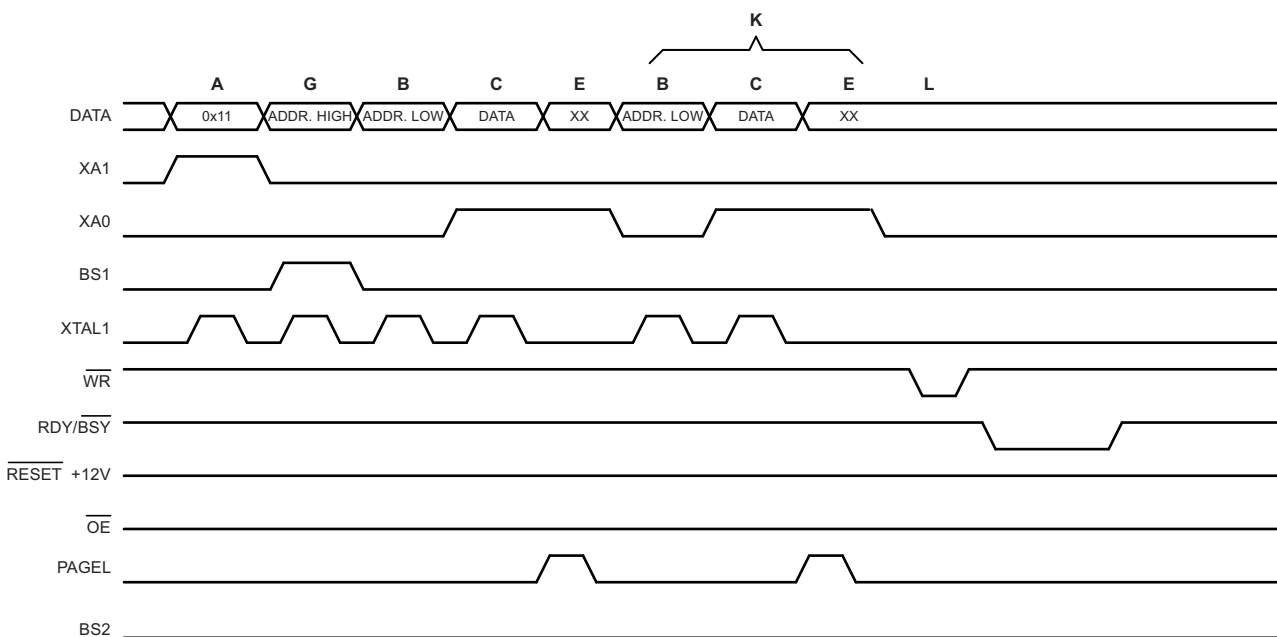
1. A: Load command "0001 0001".
2. G: Load address high byte (0x00 - 0xFF).
3. B: Load address low byte (0x00 - 0xFF).
4. C: Load data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page

1. Set BS1 to "0".
2. Give \overline{WR} a negative pulse. This starts programming of the EEPROM page. $\overline{RDY/BSY}$ goes low.
3. Wait until $\overline{RDY/BSY}$ goes high before programming the next page (See Figure 25-4 for signal waveforms).

Figure 25-4. Programming the EEPROM Waveforms



25.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 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”.

25.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 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”.

25.7.8 Programming the Fuse Low Bits

The algorithm for programming the fuse low bits is as follows (refer to Section 25.7.4 “Programming the Flash” on page 248 for details on command and data loading):

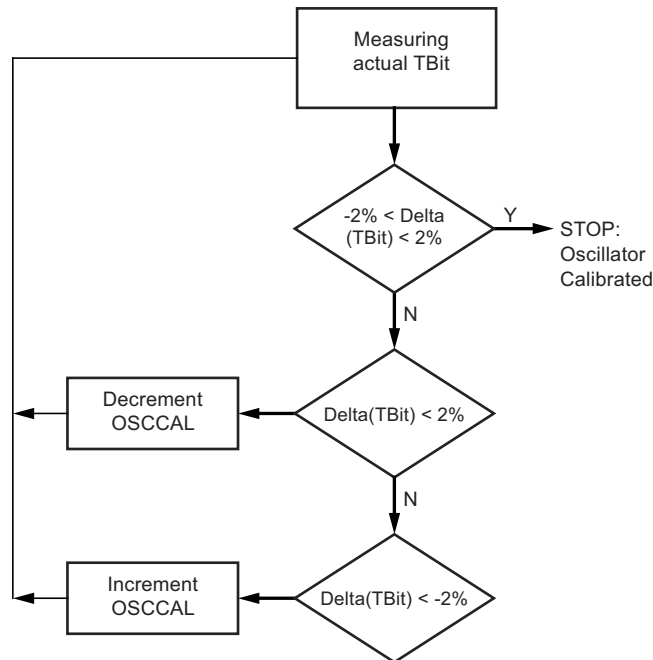
1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Give \overline{WR} a negative pulse and wait for $\overline{RDY/BSY}$ to go high.

26.6 LIN Re-synchronization Algorithm

26.7 Synchronization Algorithm

The possibility to change the value of OSCCAL during the oscillator operation allows for in-situ calibration of the slave node to entering Master frames. The principle of operation is to measure the TBit during the SYNCH byte and to change the calibration value of OSCCAL to recover from local frequency drifts due to local voltage or temperature deviation. The algorithm used for the synchronization of the internal RC oscillator is depicted in Figure 26-3 on page 264.

Figure 26-3. Dichotomic Algorithm Used for LIN Slave Clock Re-synchronization



26.8 Precaution Against OSCCAL Discontinuity

The Figure 28-23 on page 281 illustrates the on-purpose discontinuity of RC frequency versus OSCCAL value. For one correct re-synchronization, the frequency change must be kept on the same side of the discontinuity (no change of OSCCAL[7]). Since there will be no device having frequency changed by more than 10% (see Figure 28-21 on page 280), thus no reason to change the frequency value by more than 10%. Therefore, when calibration tries to cross the border because of subsequent increase (or decrease) in OSCCAL values, then the routine must be stopped.

Example: For parts operating in the lower part of the curve, if $\text{new_OSCCAL} > 127$ then $\text{new_OSCCAL} = 127$. Similar for parts operating on the high side of the discontinuity.

Figure 28-9. Output High Voltage versus Output High Current ($V_{CC} = 3V$)

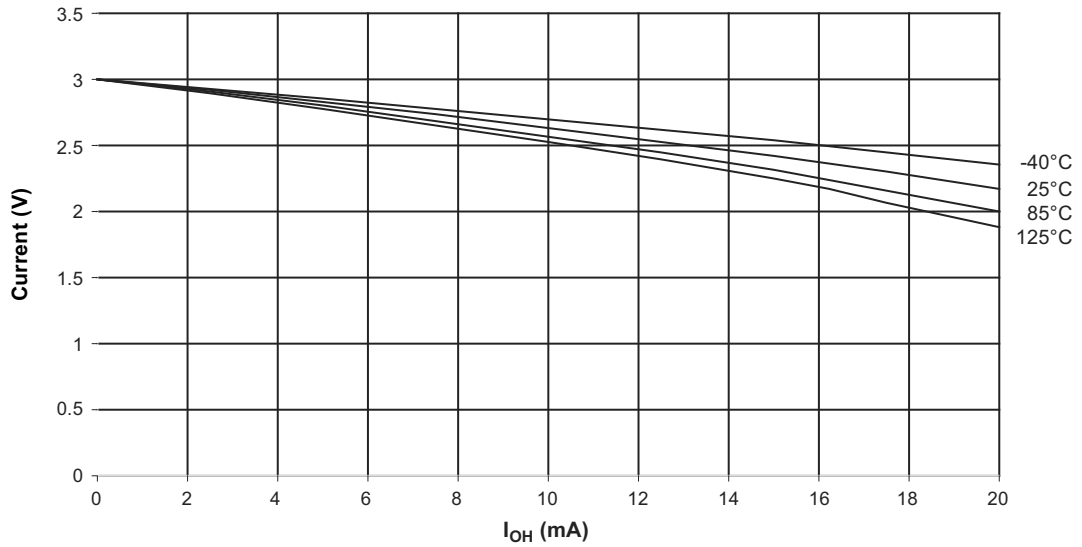


Figure 28-10. Reset Pull-up Resistor Current versus Reset Pin Voltage ($V_{CC} = 5V$)

