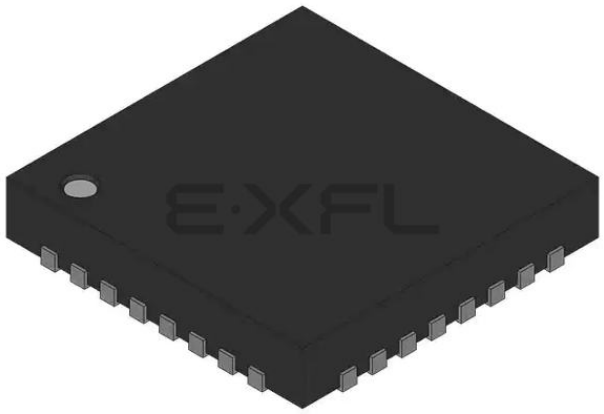


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, 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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atmega48-15mt

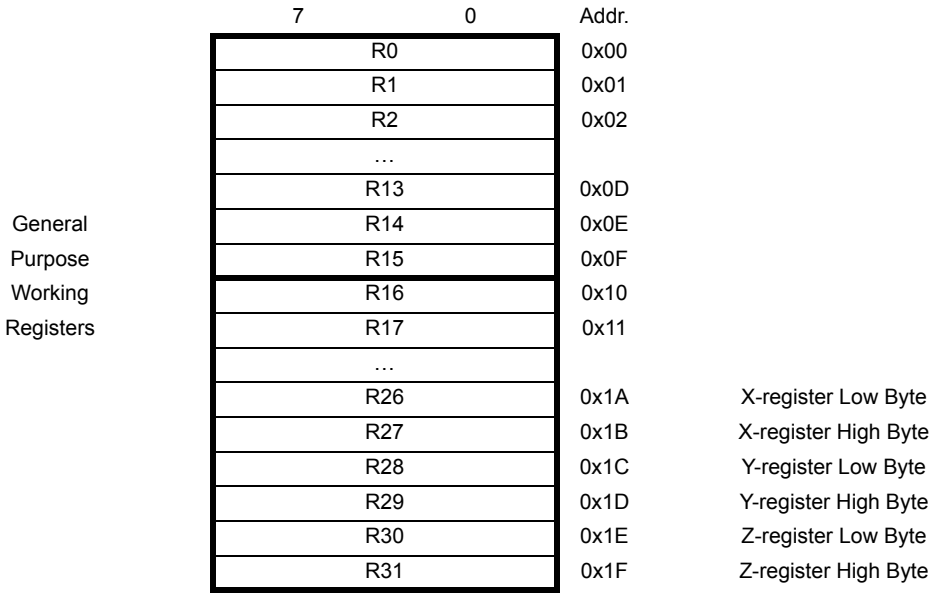
4.5 General Purpose Register File

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



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.

- **Bit 2 – EEMPE: EEPROM Master Write Enable**

The EEMPE bit determines whether setting EEPE to one causes the EEPROM to be written. When EEMPE is set, setting EEPE within four clock cycles will write data to the EEPROM at the selected address If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

- **Bit 1 – EEPE: EEPROM Write Enable**

The EEPROM write enable signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEPE becomes zero.
2. Wait until SELFPRGEN in SPMCSR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMPE bit while writing a zero to EEPE in EECR.
6. Within four clock cycles after setting EEMPE, write a logical one to EEPE.

The EEPROM can not be programmed during a CPU write to the flash memory. The software must check that the flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a boot loader allowing the CPU to program the flash. If the flash is never being updated by the CPU, step 2 can be omitted. See Section 24. “Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168” on page 229 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM master write enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM read enable signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

The calibrated oscillator is used to time the EEPROM accesses. Table 5-2 lists the typical programming time for EEPROM access from the CPU.

Table 5-2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typical Programming Time
EEPROM write (from CPU)	26,368	3.3ms

5.4 I/O Memory

The I/O space definition of the Atmel® ATmega48/88/168 is shown in Section “” on page 285.

All Atmel ATmega48/88/168 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. 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. Refer to the instruction set section for more details. 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 Atmel ATmega48/88/168 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

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

Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR® the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

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

5.4.1 General Purpose I/O Registers

The Atmel ATmega48/88/168 contains three general purpose I/O registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags. General purpose I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

5.4.2 General Purpose I/O Register 2 – GPIOR2

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

5.4.3 General Purpose I/O Register 1 – GPIOR1

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

5.4.4 General Purpose I/O Register 0 – GPIOR0

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATmega168 is:

```
Address      Labels Code                Comments
0x0000              jmp    RESET        ; Reset Handler
0x0002              jmp    EXT_INT0     ; IRQ0 Handler
0x0004              jmp    EXT_INT1     ; IRQ1 Handler
0x0006              jmp    PCINT0       ; PCINT0 Handler
0x0008              jmp    PCINT1       ; PCINT1 Handler
0x000A              jmp    PCINT2       ; PCINT2 Handler
0x000C              jmp    WDT          ; Watchdog Timer Handler
0x000E              jmp    TIM2_COMPA   ; Timer2 Compare A Handler
0x0010              jmp    TIM2_COMPB   ; Timer2 Compare B Handler
0x0012              jmp    TIM2_OVF     ; Timer2 Overflow Handler
0x0014              jmp    TIM1_CAPT    ; Timer1 Capture Handler
0x0016              jmp    TIM1_COMPA   ; Timer1 Compare A Handler
0x0018              jmp    TIM1_COMPB   ; Timer1 Compare B Handler
0x001A              jmp    TIM1_OVF     ; Timer1 Overflow Handler
0x001C              jmp    TIM0_COMPA   ; Timer0 Compare A Handler
0x001E              jmp    TIM0_COMPB   ; Timer0 Compare B Handler
0x0020              jmp    TIM0_OVF     ; Timer0 Overflow Handler
0x0022              jmp    SPI_STC      ; SPI Transfer Complete Handler
0x0024              jmp    USART_RXC    ; USART, RX Complete Handler
0x0026              jmp    USART_UDRE   ; USART, UDR Empty Handler
0x0028              jmp    USART_TXC    ; USART, TX Complete Handler
0x002A              jmp    ADC          ; ADC Conversion Complete Handler
0x002C              jmp    EE_RDY       ; EEPROM Ready Handler
0x002E              jmp    ANA_COMP     ; Analog Comparator Handler
0x0030              jmp    TWI          ; 2-wire Serial Interface Handler
0x0032              jmp    SPM_RDY      ; Store Program Memory Ready Handler
;
0x0033       RESET: ldi                 r16, high(RAMEND); Main program start
0x0034              r16                 ; Set Stack Pointer to top of RAM
0x0035              ldi                 r16, low(RAMEND)
0x0036              out                 SPL,r16
0x0037              sei                 ; Enable interrupts
0x0038              <instr>  xxx
  ...   ...    ...    ...
```

When the BOOTRST fuse is unprogrammed, the boot section size set to 2K bytes and the IVSEL bit in the MCUCR register is set before any interrupts are enabled, the most typical and general program setup for the reset and interrupt vector addresses in ATmega168 is:

```
Address      Labels Code                Comments
0x0000              RESET: ldi          r16,high(RAMEND); Main program start
0x0001              out    SPH,r16      ; Set Stack Pointer to top of RAM
0x0002              ldi    r16,low(RAMEND)
0x0003              out    SPL,r16
0x0004              sei                 ; Enable interrupts
0x0005              <instr>  xxx
;
.org 0xC02
0x1C02              jmp    EXT_INT0     ; IRQ0 Handler
0x1C04              jmp    EXT_INT1     ; IRQ1 Handler
...          ...    ...                 ;
0x1C32              jmp    SPM_RDY      ; Store Program Memory Ready Handler
```

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

When a logic change on any PCINT14..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

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

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

11.6 Pin Change Mask Register 2 – PCMSK2

Bit	7	6	5	4	3	2	1	0	
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT23..16: Pin Change Enable Mask 23..16**

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

11.7 Pin Change Mask Register 1 – PCMSK1

Bit	7	6	5	4	3	2	1	0	
	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	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 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega48/88/168, and will always read as zero.

- **Bit 6..0 – PCINT14..8: Pin Change Enable Mask 14..8**

Each PCINT14..8-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.8 Pin Change Mask Register 0 – PCMSK0

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

Bits 5:4 – COM0**B1:0: 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).

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.

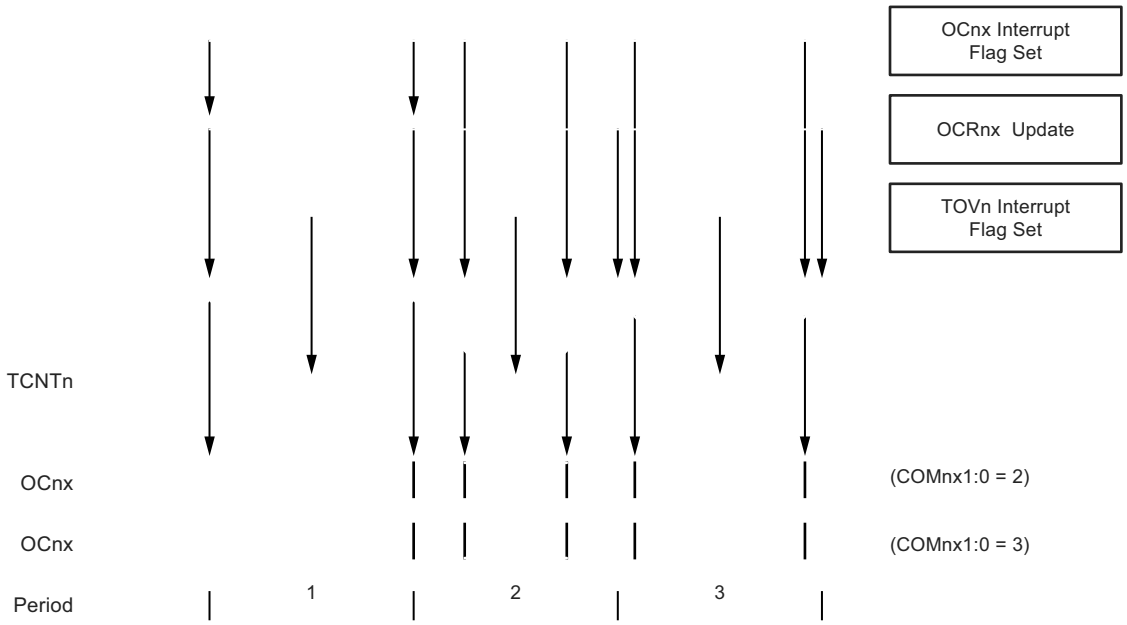
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x1:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk\_I/O}/2$ when OCR2A is set to zero. This feature is similar to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

15.6.4 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 WGM2:0 = 3, and OCR2A when MGM2:0 = 7. In non-inverting compare output mode, the output compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x while upcounting, and set on the compare match while downcounting. 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 15-7. 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 15-7. 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.

Table 15-9. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(no prescaling)
0	1	0	clk_{T2S}/8 (from prescaler)
0	1	1	clk_{T2S}/32 (from prescaler)
1	0	0	clk_{T2S}/64 (from prescaler)
1	0	1	clk_{T2S}/128 (from prescaler)
1	1	0	clk_{T2S}/256 (from prescaler)
1	1	1	clk_{T2S}/1024 (from prescaler)

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

15.8.3 Timer/Counter Register – TCNT2

Bit	7	6	5	4	3	2	1	0	
	TCNT2**[7:0]**								TCNT2
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 Timer/Counter register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. 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.

15.8.4 Output Compare Register A – OCR2A

Bit	7	6	5	4	3	2	1	0	
	OCR2A**[7:0]**								OCR2A
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 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.

15.8.5 Output Compare Register B – OCR2B

Bit	7	6	5	4	3	2	1	0	
	OCR2B**[7:0]**								OCR2B
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 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.

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.

17.5 Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the transmit enable (TXEN) bit in the UCSRnB register. When the transmitter is enabled, the normal port operation of the TxDn pin is overridden by the USART and given 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 XCKn pin will be overridden and used as transmission clock.

17.5.1 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 UDRn 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 register, U2Xn bit or by XCKn depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the data register empty (UDREn) flag. When using frames with less than eight bits, the most significant bits written to the UDRn 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. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

The function simply waits for the transmit buffer to be empty by checking the UDREn 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.

Table 19-6. Status Codes for Slave Receiver Mode (Continued)

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	
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 19-17. Formats and States in the Slave Receiver Mode

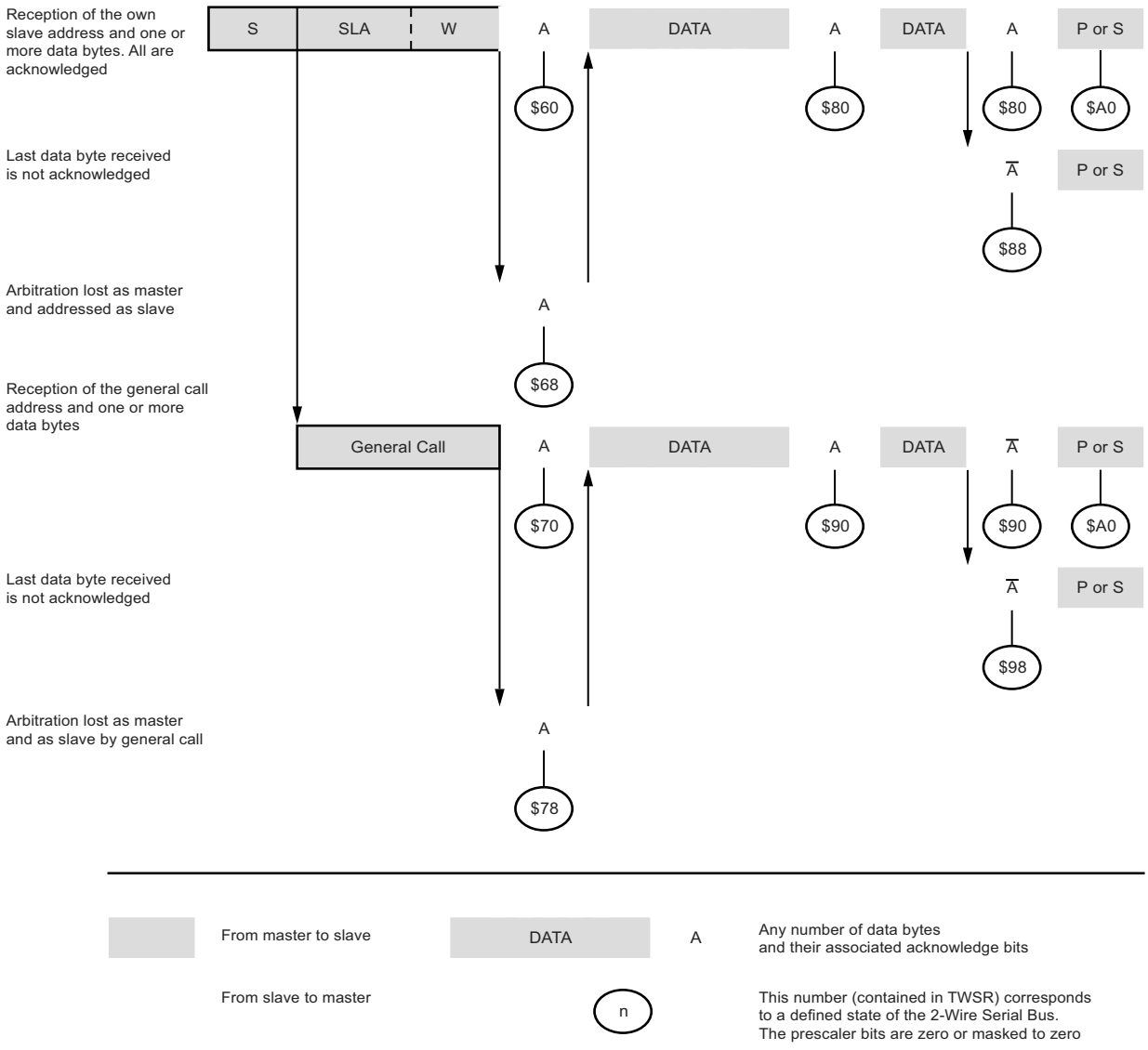
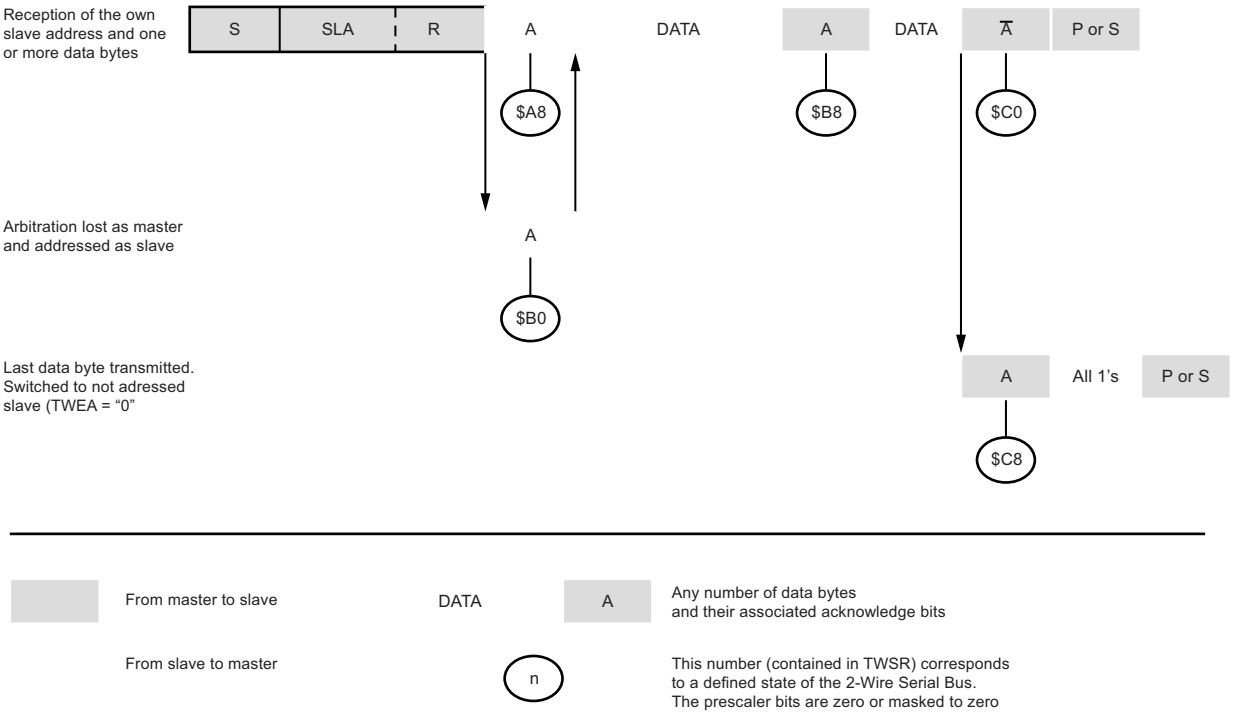


Figure 19-19. Formats and States in the Slave Transmitter Mode



19.8.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 19-8.

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 19-8. Miscellaneous States

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	
0xF8	No relevant state information available; TWINT = “0”	No 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 condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

- **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 system clock frequency and the input clock to the ADC.

Table 21-4. 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

21.6.3 The ADC Data Register – ADCL and ADCH

21.6.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	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	

21.6.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	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 ADC conversion is complete, the result is found in these two registers.

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 is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

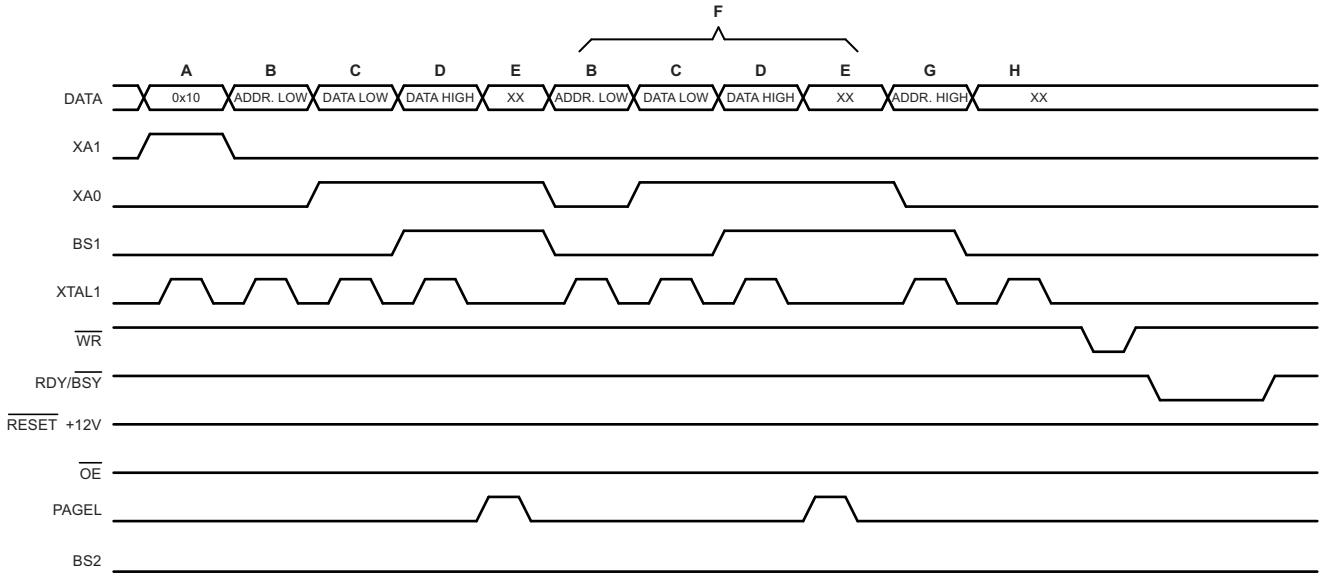
The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in Section 21.6 “ADC Conversion Result” on page 217.

```
        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<<SELFPRGEN)
        call           Do_spm

        ;      re-enable the RWW section
        ldi            spmcrval, (1<<RWWSRE) | (1<<SELFPRGEN)
        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:
        lpm            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
        sbrs           temp1, RWWSB        ; If RWWSB is set, the RWW section is not
ready yet
        ret
        ;      re-enable the RWW section
        ldi            spmcrval, (1<<RWWSRE) | (1<<SELFPRGEN)
        call           Do_spm
        rjmp           Return

Do_spm:
        ;      check for previous SPM complete
Wait_spm:
        in             temp1, SPMCSR
        sbrc           temp1, SELFPRGEN
        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
```

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. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 25-4 for signal waveforms).

28.1.1 Power-Down Supply Current

Figure 28-3. Power-Down Supply Current versus V_{CC} (Watchdog Timer Disabled)

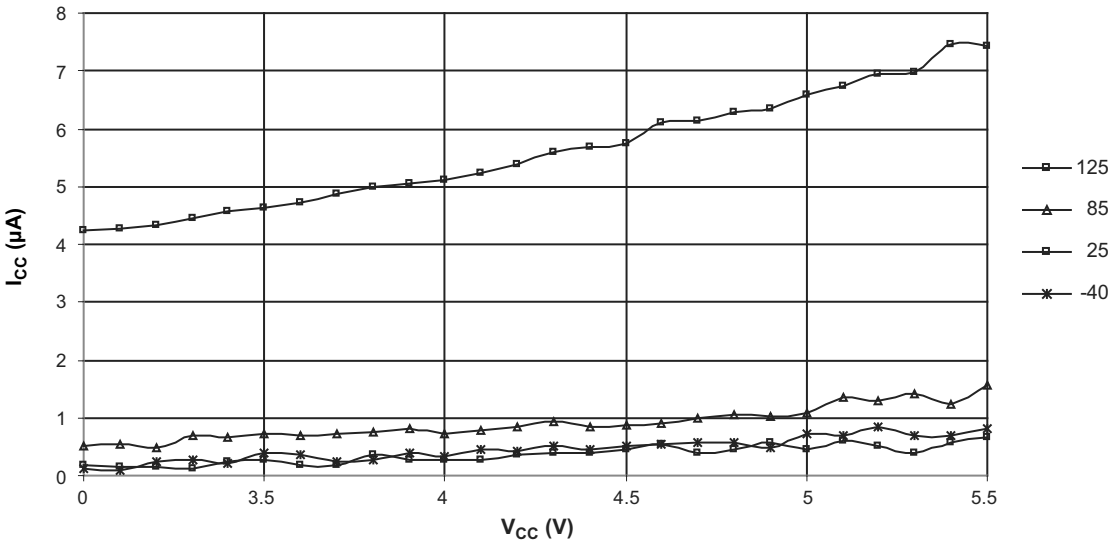
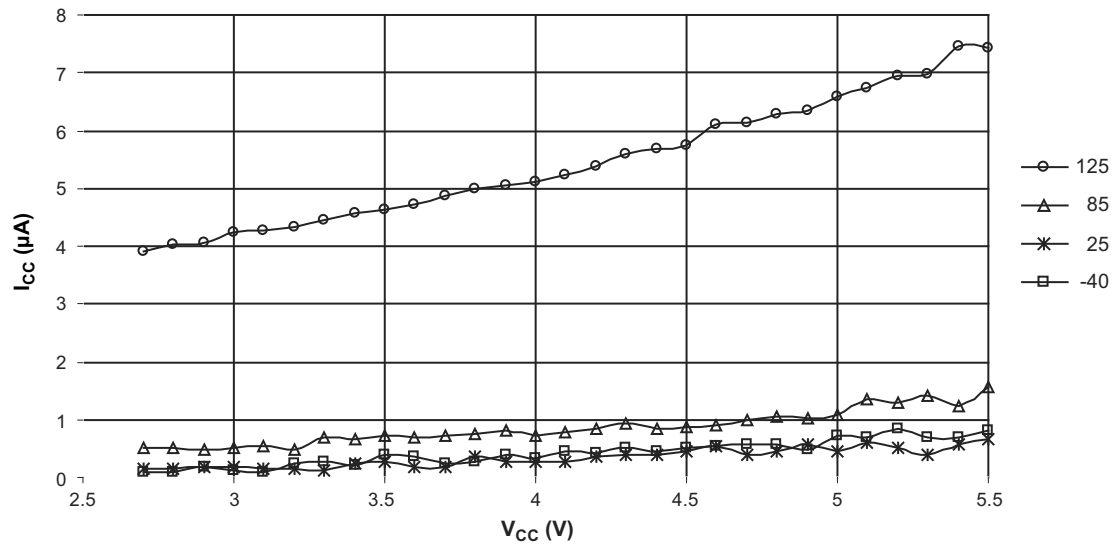


Figure 28-4. Power-Down Supply Current versus V_{CC} (Watchdog Timer Enabled)



28.1.4 Pin Thresholds and Hysteresis

Figure 28-15. I/O Pin Input Threshold versus V_{CC} (VIH, I/O Pin Read as '1

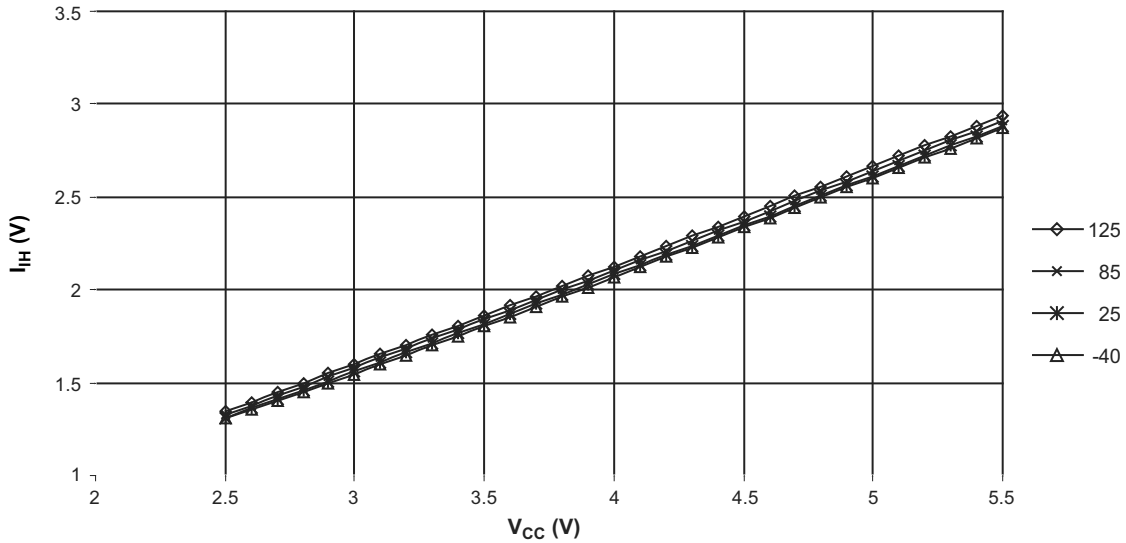


Figure 28-16. I/O Pin Input Threshold versus V_{CC} (VIL, I/O Pin Read as '0')

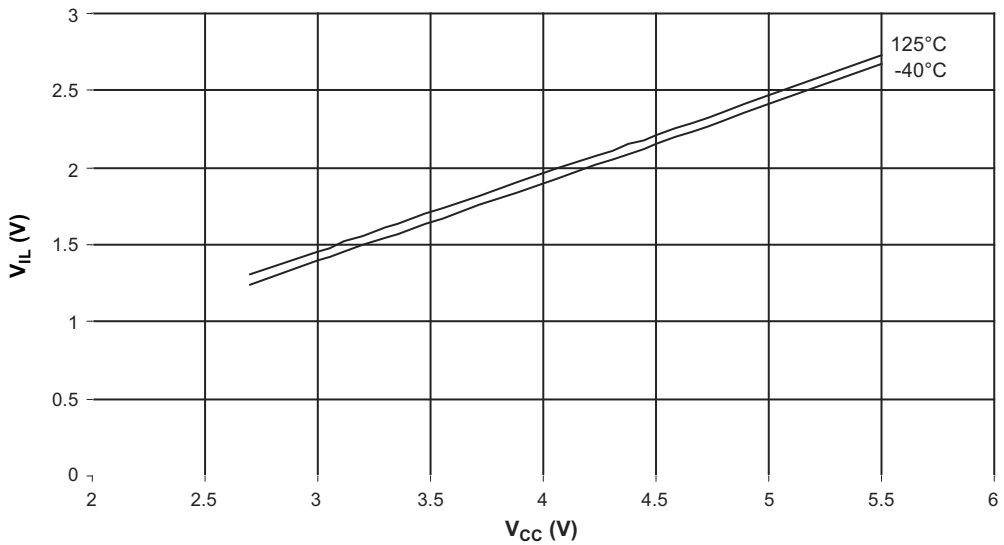


Figure 28-21. Calibrated 8MHz RC Oscillator Frequency versus V_{CC}

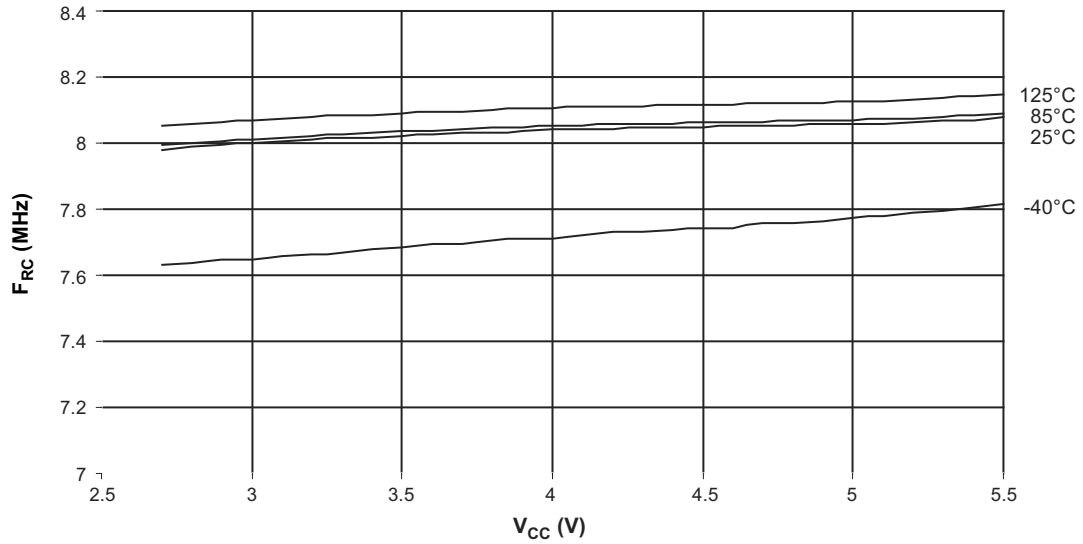
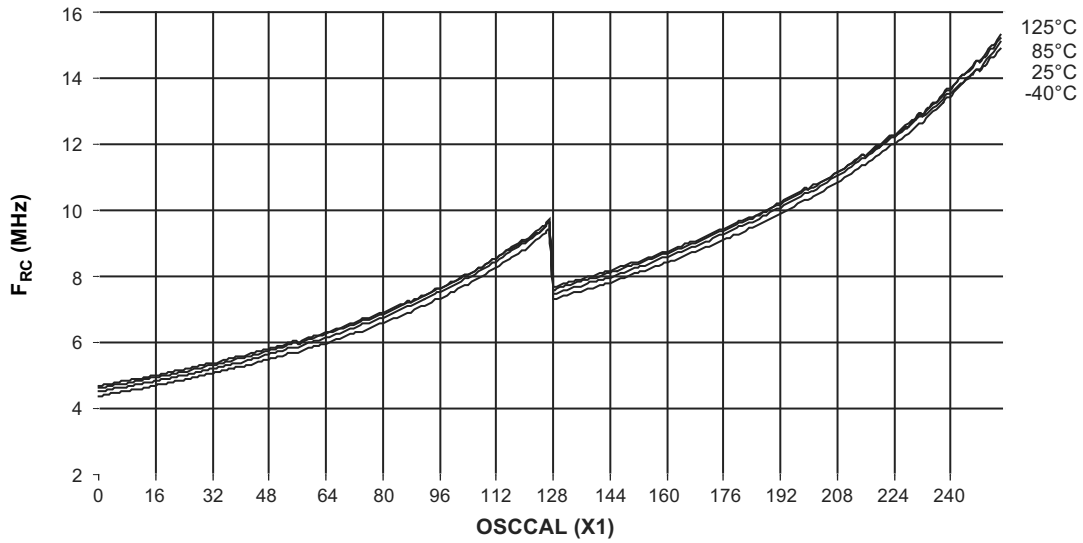


Figure 28-22. Calibrated 8MHz RC Oscillator Frequency versus OSCCAL Value (for ATmega48-15AZ and ATmega168-15AZ)



29. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xB8)	TWBR	2-wire serial interface bit rate register								182
(0xB7)	Reserved	–	–	–	–	–	–	–	–	
(0xB6)	ASSR	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	135
(0xB5)	Reserved	–	–	–	–	–	–	–	–	
(0xB4)	OCR2B	Timer/Counter2 output compare register B								132
(0xB3)	OCR2A	Timer/Counter2 output compare register A								132
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								132
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	131
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	129
(0xAF)	Reserved	–	–	–	–	–	–	–	–	
(0xAE)	Reserved	–	–	–	–	–	–	–	–	
(0xAD)	Reserved	–	–	–	–	–	–	–	–	
(0xAC)	Reserved	–	–	–	–	–	–	–	–	
(0xAB)	Reserved	–	–	–	–	–	–	–	–	
(0xAA)	Reserved	–	–	–	–	–	–	–	–	
(0xA9)	Reserved	–	–	–	–	–	–	–	–	
(0xA8)	Reserved	–	–	–	–	–	–	–	–	
(0xA7)	Reserved	–	–	–	–	–	–	–	–	
(0xA6)	Reserved	–	–	–	–	–	–	–	–	
(0xA5)	Reserved	–	–	–	–	–	–	–	–	
(0xA4)	Reserved	–	–	–	–	–	–	–	–	
(0xA3)	Reserved	–	–	–	–	–	–	–	–	
(0xA2)	Reserved	–	–	–	–	–	–	–	–	
(0xA1)	Reserved	–	–	–	–	–	–	–	–	
(0xA0)	Reserved	–	–	–	–	–	–	–	–	
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	Reserved	–	–	–	–	–	–	–	–	
(0x96)	Reserved	–	–	–	–	–	–	–	–	
(0x95)	Reserved	–	–	–	–	–	–	–	–	

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48/88/168 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. Only valid for Atmel® ATmega88/168

37. Table of Contents