Features

- High Performance, Low Power AVR ® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 1 MIPS throughput per MHz
  - On-chip 2-cycle Multiplier
- Data and Non-Volatile Program Memory
  - 8K Bytes Flash of In-System Programmable Program Memory
    - Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
  - 512 Bytes of In-System Programmable EEPROM
- Endurance: 100,000 Write/Erase Cycles
  - 512 Bytes Internal SRAM
  - Programming Lock for Flash Program and EEPROM Data Security
- On Chip Debug Interface (debugWIRE)
- Peripheral Features
  - Two 12-bit High Speed PSC (Power Stage Controllers) with 4-bit Resolution Enhancement
    - Non Overlapping Inverted PWM Output Pins With Flexible Dead-Time
    - Variable PWM duty Cycle and Frequency
    - Synchronous Update of all PWM Registers
    - Auto Stop Function for Event Driven PFC Implementation
    - Less than 25 Hz Step Width at 150 kHz Output Frequency
    - PSC2 with four Output Pins and Output Matrix
  - One 8-bit General purpose Timer/Counter with Separate Prescaler and Capture Mode
  - One 16-bit General purpose Timer/Counter with Separate Prescaler, Compare Mode and Capture Mode
  - Master/Slave SPI Serial Interface
  - 10-bit ADC
    - 8 Single Ended Channels and 1 Fully Differential ADC Channel Pair
    - Programmable Gain (5x, 10x, 20x, 40x on Differential Channel)
    - Internal Reference Voltage
  - Two Analog Comparator with Resistor-Array to Adjust Comparison Voltage
  - 4 External Interrupts
  - Programmable Watchdog Timer with Separate On-Chip Oscillator
- Special Microcontroller Features
  - Low Power Idle, Noise Reduction, and Power Down Modes
  - Power On Reset and Programmable Brown Out Detection
  - Flag Array in Bit-programmable I/O Space (4 bytes)
  - In-System Programmable via SPI Port
  - Internal Calibrated RC Oscillator (8 MHz)
  - On-chip PLL for fast PWM (32 MHz, 64 MHz) and CPU (16 MHz)

8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

AT90PWM1

4378C–AVR–09/08

- **Operating Voltage: 2.7V - 5.5V**
- **Extended Operating Temperature:**
  - **-40°C to +105°**

1. History

Product	Revision
AT90PWM1	First revision of parts

2. Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

3. Pin Configurations

Figure 3-1. SOIC 24-pin Package

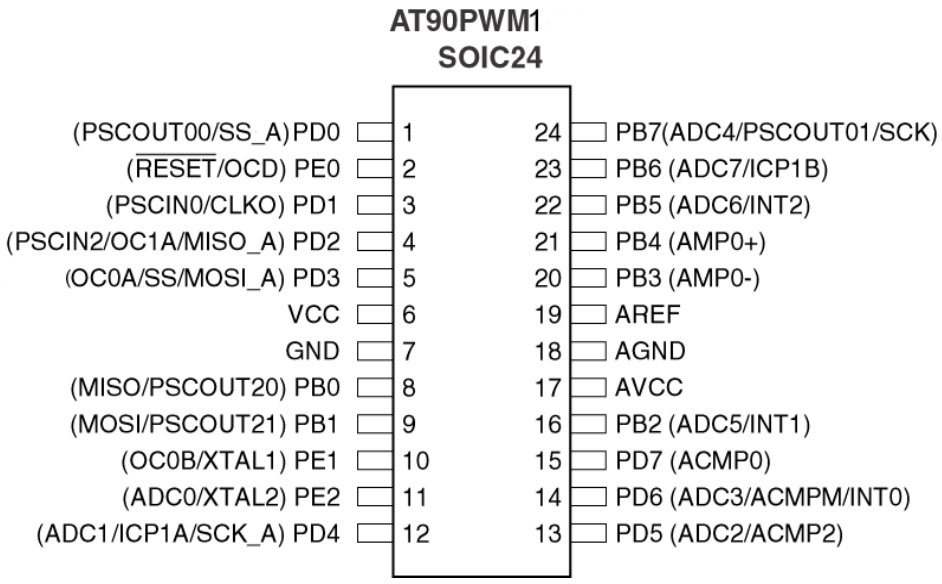
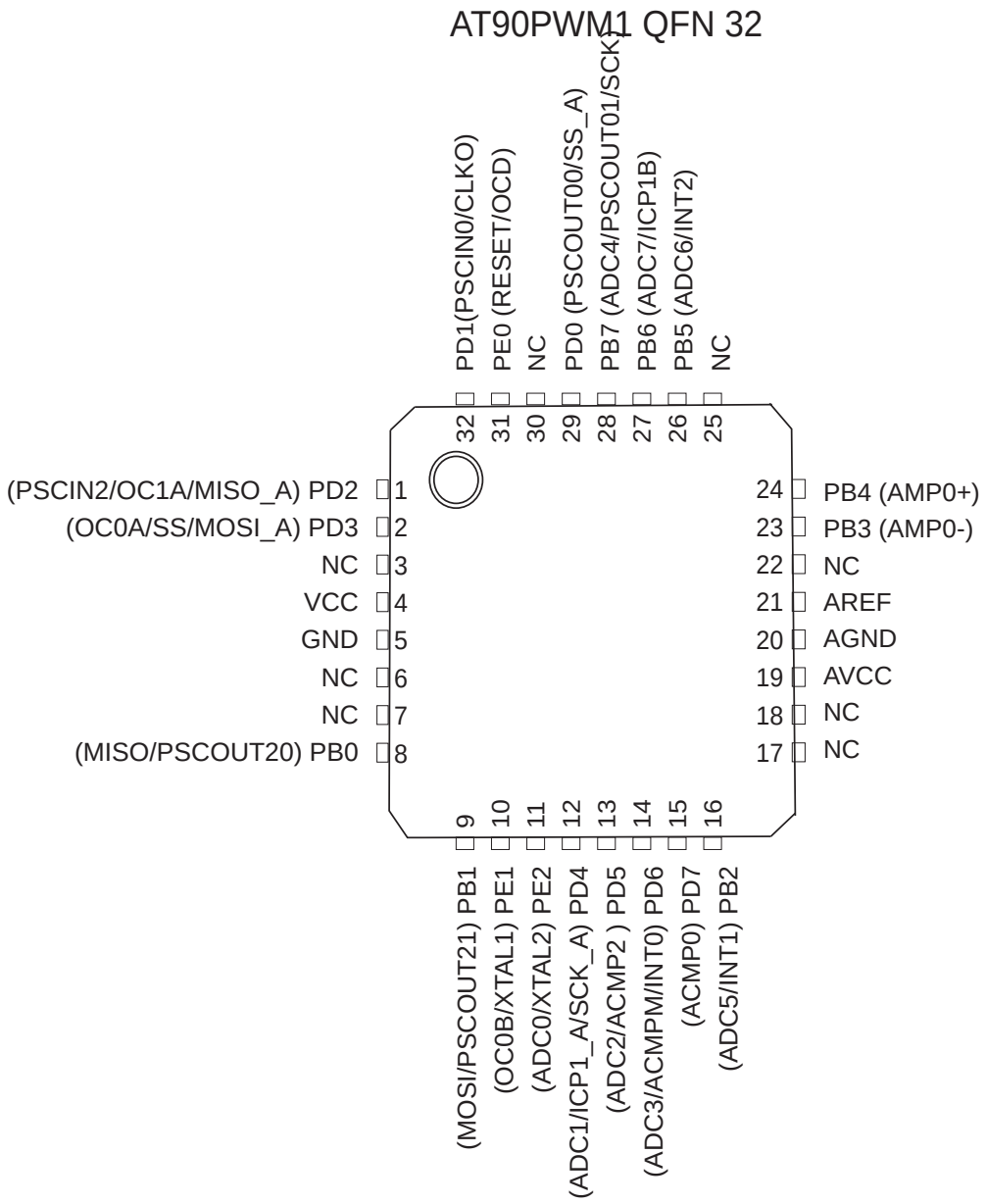


Figure 3-2. QFN 32 -pin Package



3.1 Pin Descriptions

:

Table 3-1. Pin out description

QFN32	S024 Pin Number	Mnemonic	Type	Name, Function & Alternate Function
5	7	GND	Power	**Ground:** 0V reference
20	18	AGND	Power	**Analog Ground:** 0V reference for analog part
4	6	VCC	power	**Power Supply:**
19	17	AVCC	Power	**Analog Power Supply:** This is the power supply voltage for analog part For a normal use this pin must be connected.
21	19	AREF	Power	**Analog Reference :** reference for analog converter . This is the reference voltage of the A/D converter. As output, can be used by external analog
8	8	PBO	I/O	MISO (SPI Master In Slave Out) PSCOUT20 output
9	9	PB1	I/O	MOSI (SPI Master Out Slave In) PSCOUT21 output
16	16	PB2	I/O	ADC5 (Analog Input Channel5) INT1
23	20	PB3	I/O	AMP0- (Analog Differential Amplifier 0 Input Channel)
24	21	PB4	I/O	AMP0+ (Analog Differential Amplifier 0 Input Channel)
26	22	PB5	I/O	ADC6 (Analog Input Channel 6) INT 2
27	23	PB6	I/O	ADC7 (Analog Input Channel 7) ICP1B (Timer 1 input capture alternate input) PSCOUT11 output
28	24	PB7	I/O	PSCOUT01 output ADC4 (Analog Input Channel 4) SCK (SPI Clock)
29	1	PD0	I/O	PSCOUT00 output XCK (UART Transfer Clock) SS_A (Alternate SPI Slave Select)
32	3	PD1	I/O	PSCIN0 (PSC 0 Digital Input) CLKO (System Clock Output)
1	4	PD2	I/O	PSCIN2 (PSC 2 Digital Input) OC1A (Timer 1 Output Compare A) MISO_A (Programming & alternate SPI Master In Slave Out)
2	5	PD3	I/O	TXD (Dali/UART Tx data) OC0A (Timer 0 Output Compare A) SS (SPI Slave Select) MOSI_A (Programming & alternate Master Out SPI Slave In)

Table 3-1. Pin out description (Continued)

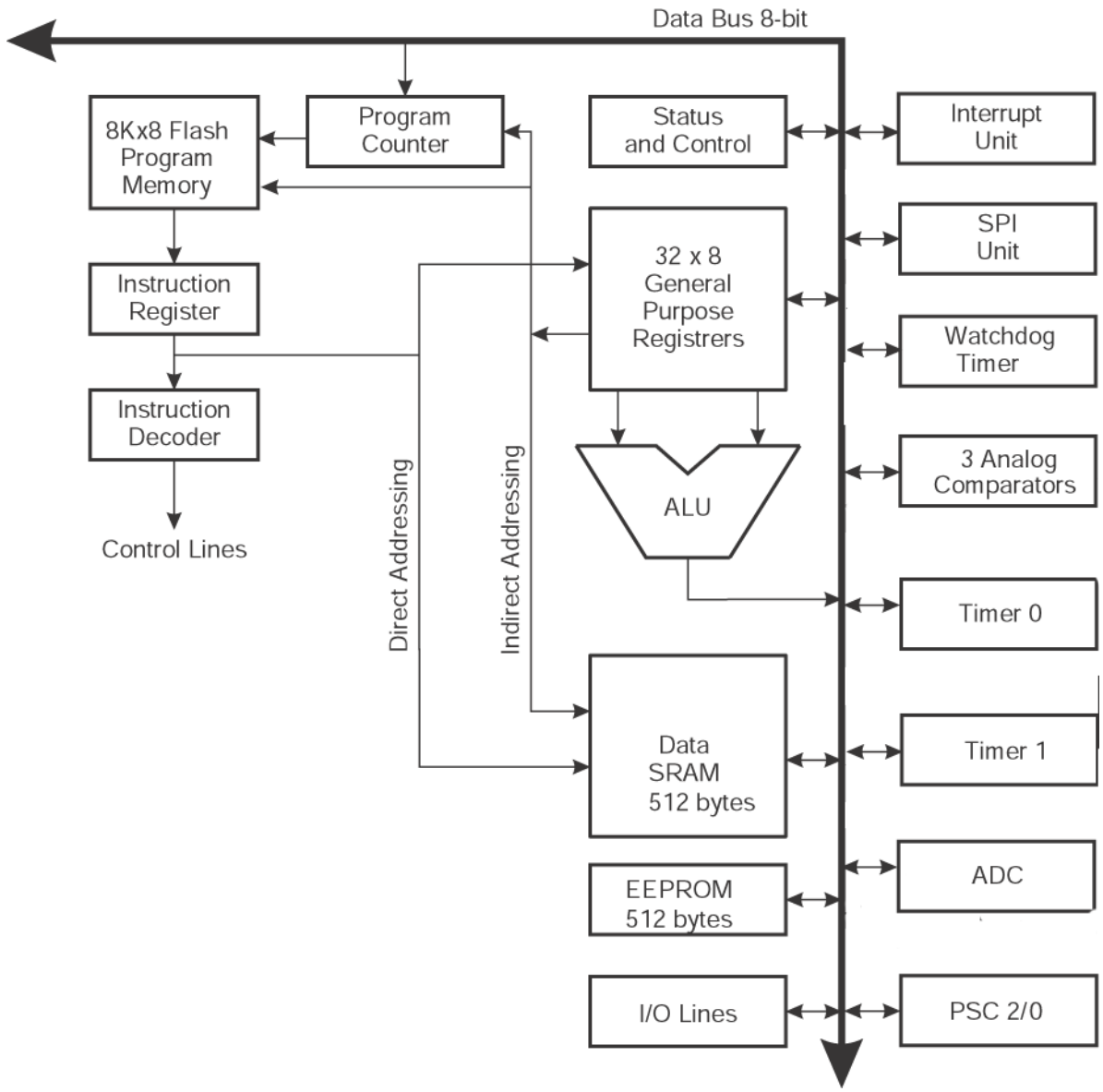
QFN32	S024 Pin Number	Mnemonic	Type	Name, Function & Alternate Function
12	12	PD4	I/O	ADC1 (Analog Input Channel 1) RXD (Dali/UART Rx data) ICP1A (Timer 1 input capture) SCK_A (Programming & alternate SPI Clock)
13	13	PD5	I/O	ADC2 (Analog Input Channel 2) ACMP2 (Analog Comparator 2 Positive Input)
14	14	PD6	I/O	ADC3 (Analog Input Channel 3) ACMPM reference for analog comparators INT0
15	15	PD7	I/O	ACMP0 (Analog Comparator 0 Positive Input)
31	2	PE0	I/O or I	RESET (Reset Input) OCD (On Chip Debug I/O)
10	10	PE1	I/O	XTAL1: XTAL Input OC0B (Timer 0 Output Compare B)
11	11	PE2	I/O	XTAL2: XTAL OuTput ADC0 (Analog Input Channel 0)

4. Overview

The AT90PWM1 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90PWM1 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

4.1 Block Diagram

Figure 4-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The AT90PWM1 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, 2 Power Stage Controllers, two flexible Timer/Counters with compare modes and PWM, an 8-channel 10-bit ADC with two differential

input stage with programmable gain, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, an On-chip Debug system and four software selectable power saving modes.

The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI ports and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. The ADC Noise Reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel AT90PWM1 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90PWM1 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

4.2 Pin Descriptions

4.2.1 VCC

Digital supply voltage.

4.2.2 GND

Ground.

4.2.3 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the AT90PWM1 as listed on page 65.

4.2.4 Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the AT90PWM1 as listed on page 68.

4.2.5 Port E (PE2..0) $\overline{\text{RESET}}$/ XTAL1/ XTAL2

Port E is an 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

If the RSTDISBL Fuse is programmed, PE0 is used as an I/O pin. Note that the electrical characteristics of PE0 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PE0 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 9-1 on page 43. Shorter pulses are not guaranteed to generate a Reset.

Depending on the clock selection fuse settings, PE1 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PE2 can be used as output from the inverting Oscillator amplifier.

The various special features of Port E are elaborated in “Alternate Functions of Port E” on page 71 and “Clock Systems and their Distribution” on page 27.

4.2.6 AVCC

AVCC is the supply voltage pin for the A/D Converter on Port F. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

4.2.7 AREF

This is the analog reference pin for the A/D Converter.

4.3 About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

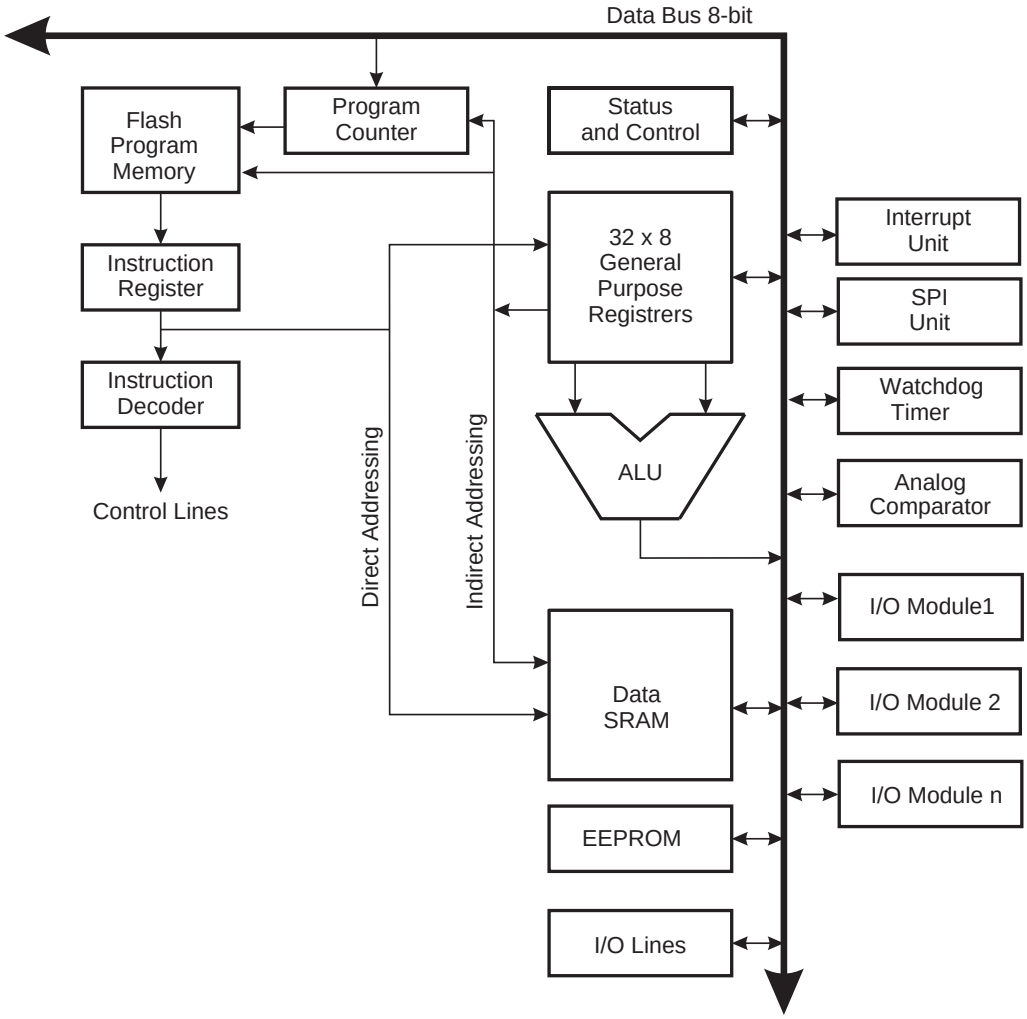
5. AVR CPU Core

5.1 Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

5.2 Architectural Overview

Figure 5-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM (Store Program Memory) instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher is the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the AT90PWM1 has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

5.3 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

5.4 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set to enabled the interrupts. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

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

- **Bit 5 – H: Half Carry Flag**

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

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

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

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

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

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

- **Bit 1 – Z: Zero Flag**

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

- **Bit 0 – C: Carry Flag**

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

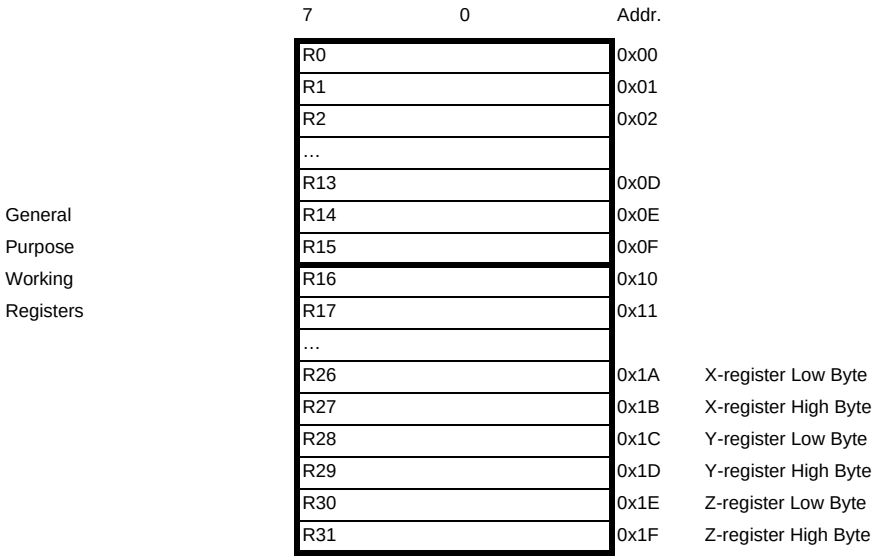
5.5 General Purpose Register File

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 16-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 5-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 5-2. AVR CPU General Purpose Working Registers



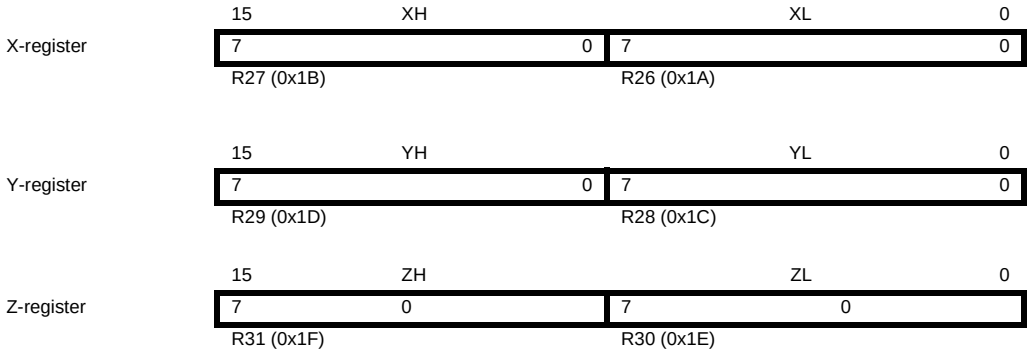
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 5-2, 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.

5.5.1 The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 5-3.

Figure 5-3. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

5.6 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x100. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

5.7 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 5-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 5-4. The Parallel Instruction Fetches and Instruction Executions

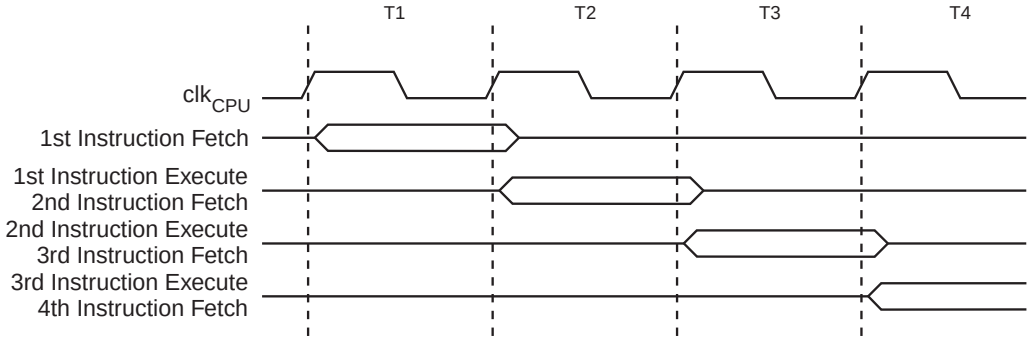
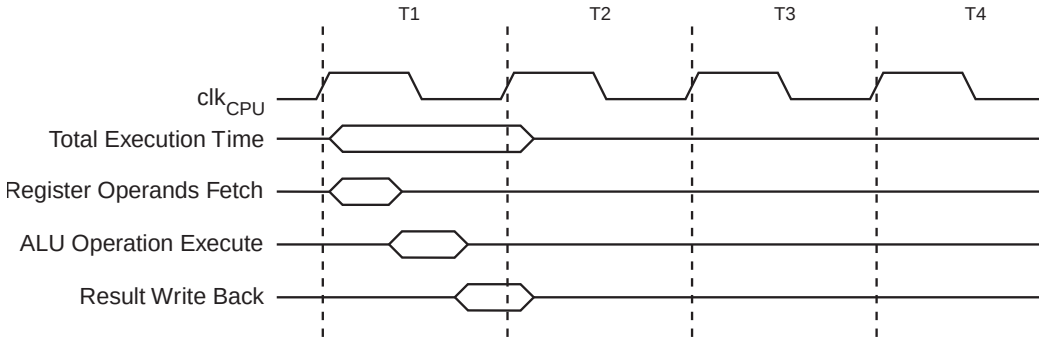


Figure 5-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 5-5. Single Cycle ALU Operation



5.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section “Memory Programming” on page 219 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 53. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is PSC2 CAPT – the PSC2 Capture Event. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to “Interrupts” on page 53 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by

programming the BOOTRST Fuse, see “Boot Loader Support – Read-While-Write Self-Programming” on page 205.

5.8.1 Interrupt Behavior

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence..

Assembly Code Example

```
in  r16, SREG     ; store SREG value
cli               ; disable interrupts during timed sequence
sbi EECR, EEMWE   ; start EEPROM write
sbi EECR, EEWE
out SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;      /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG;      /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

`sei  ; set Global Interrupt Enable`
`sleep; enter sleep, waiting for interrupt`
`; note: will enter sleep before any pending`
`; interrupt(s)`

C Code Example

`_SEI(); /* set Global Interrupt Enable */`
`_SLEEP(); /* enter sleep, waiting for interrupt */`
`/* note: will enter sleep before any pending interrupt(s) */`

5.8.2 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

6. Memories

This section describes the different memories in the AT90PWM1. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the AT90PWM1 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

6.1 In-System Reprogrammable Flash Program Memory

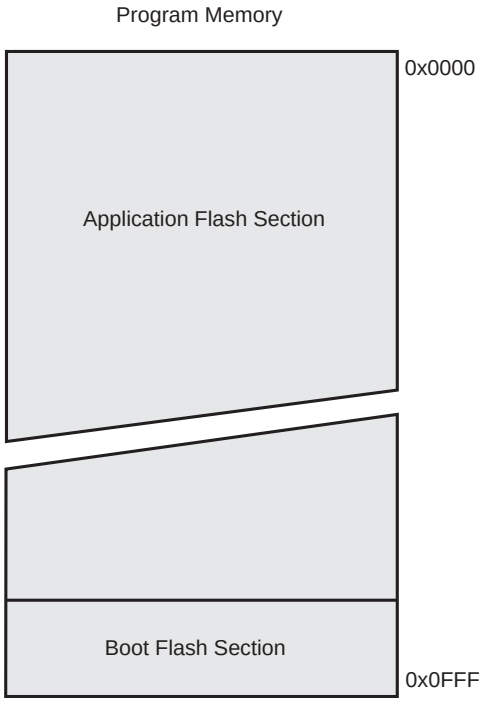
The AT90PWM1 contains 8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The AT90PWM1 Program Counter (PC) is 12 bits wide, thus addressing the 4K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in “Boot Loader Support – Read-While-Write Self-Programming” on page 205. “Memory Programming” on page 219 contains a detailed description on Flash programming in SPI or Parallel programming mode.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory.

Timing diagrams for instruction fetch and execution are presented in “Instruction Execution Timing” on page 13.

Figure 1. Program Memory Map



6.2 SRAM Data Memory

Figure 2 shows how the AT90PWM1 SRAM Memory is organized.

The AT90PWM1 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the 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.

The lower 768 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

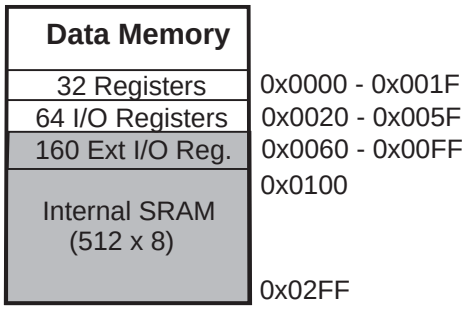
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 512 bytes of internal data SRAM in the AT90PWM1 are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 12.

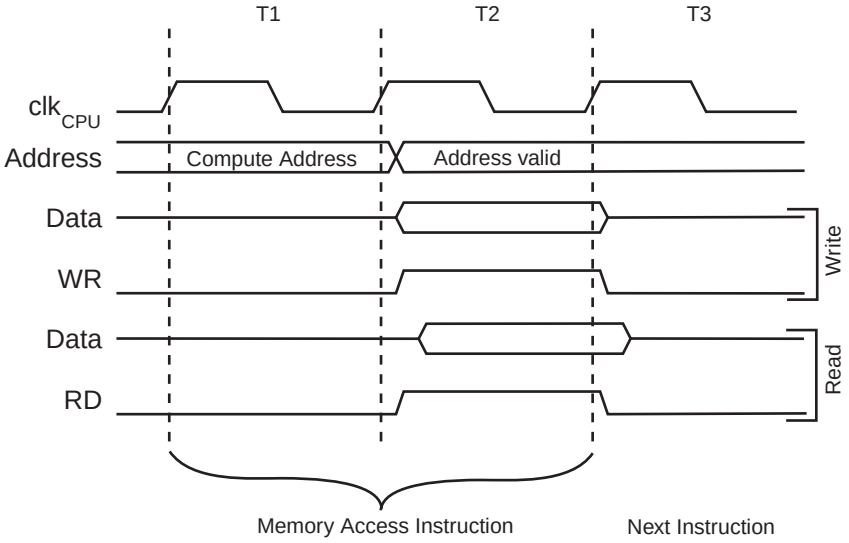
Figure 2. Data Memory Map



6.2.1 SRAM Data Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 3.

Figure 3. On-chip Data SRAM Access Cycles



6.3 EEPROM Data Memory

The AT90PWM1 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI and Parallel data downloading to the EEPROM, see “Serial Downloading” on page 234, and “Parallel Programming Parameters, Pin Mapping, and Commands” on page 223 respectively.

6.3.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. For details on how to avoid problems in these situations seeSee “Preventing EEPROM Corruption” on page 24.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

6.3.2 The EEPROM Address Registers – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	**EEAR8**	**EEARH**

	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	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	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

6.3.3 The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	EEDR7	**EEDR6**	**EEDR5**	**EEDR4**	**EEDR3**	**EEDR2**	**EEDR1**	**EEDR0**	**EEDR**
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	

- **Bits 7..0 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

6.3.4 The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	**–**	**EEPM1**	**EEPM0**	**EERIE**	**EEMWE**	**EEWE**	**EERE**	**EECR**
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bits 7..6 – Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **Bits 5..4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits**

The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEWE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 1. While EEWE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 1. EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared. The interrupt will not be generated during EEPROM write or SPM.

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

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

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

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, 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 EEWE becomes zero.
2. Wait until SPMEN (Store Program Memory Enable) in SPMCSR (Store Program Memory Control and Status Register) 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 EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

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 “Boot Loader Support – Read-While-Write Self-Programming” on page 205 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 EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE 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 EEWE 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 2 lists the typical programming time for EEPROM access from the CPU.

Table 2. EEPROM Programming Time.

Symbol	Number of Calibrated RC Oscillator Cycles	Typ Programming Time
EEPROM write (from CPU)	26368	3.3 ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

TABLE 1.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write (unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

TABLE 2.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

6.3.5 Preventing EEPROM Corruption

During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

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 detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

6.4 I/O Memory

The I/O space definition of the AT90PWM1 is shown in "Register Summary" on page 277.

All AT90PWM1 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 AT90PWM1 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's, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

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

6.5 General Purpose I/O Registers

The AT90PWM1 contains four General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags.

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

6.5.1 General Purpose I/O Register 0 – GPIOR0

Bit	7	6	5	4	3	2	1	0	
	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	GPIOR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6.5.2 General Purpose I/O Register 1 – GPIOR1

Bit	7	6	5	4	3	2	1	0	
	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	GPIOR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6.5.3 General Purpose I/O Register 2 – GPIOR2

Bit	7	6	5	4	3	2	1	0	
	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	GPIOR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

6.5.4 General Purpose I/O Register 3– GPIOR3

Bit	7	6	5	4	3	2	1	0

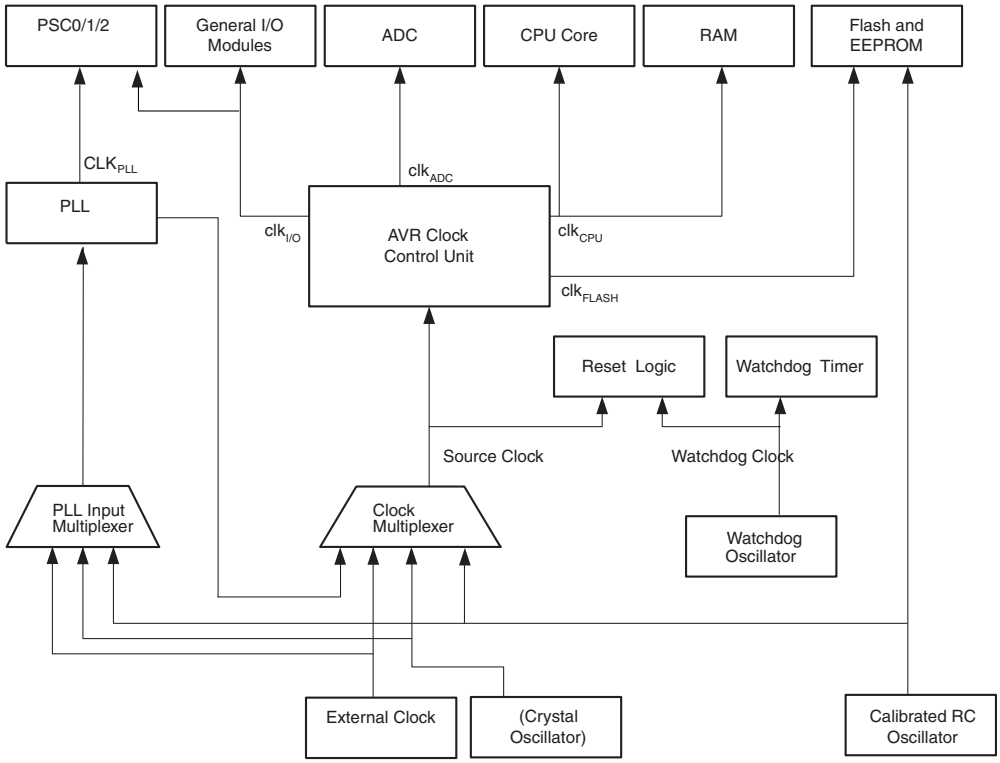
	GPIOR37	GPIOR36	GPIOR35	GPIOR34	GPIOR33	GPIOR32	GPIOR31	GPIOR30	GPIOR3
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7. System Clock

7.1 Clock Systems and their Distribution

Figure 7-1 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to unused modules can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 37. The clock systems are detailed below.

Figure 7-1. Clock Distribution AT90PWM2/3



7.1.1 CPU Clock – clk_{CPU}

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

7.1.2 I/O Clock – $clk_{I/O}$

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

7.1.3 Flash Clock – clk_{FLASH}

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

7.1.4 PLL Clock – clk_{PLL}

The PLL clock allows the PSC modules to be clocked directly from a 64/32 MHz clock. A 16 MHz clock is also derived for the CPU.

7.1.5 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.

7.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as illustrated Table 7-1. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 7-1. Device Clocking Options Select AT90PWM1

Device Clocking Option	System Clock	PLL Input	CKSEL3..0 (1)
External Crystal/Ceramic Resonator	Ext Osc (2)	RC Osc (3)	1111 - 1000
PLL output divided by 4 : 16 MHz / PLL driven by External Crystal/Ceramic Resonator	Ext Osc	Ext Osc	0100
PLL output divided by 4 : 16 MHz / PLL driven by External Crystal/Ceramic Resonator	PLL / 4	Ext Osc	0101
Reserved	N/A	N/A	0111- 0110
PLL output divided by 4 : 16 MHz	PLL / 4	RC Osc	0011
Calibrated Internal RC Oscillator	RC Osc	RC Osc	0010
PLL output divided by 4 : 16 MHz / PLL driven by External clock	PLL / 4	Ext Clk (4)	0001
External Clock	Ext Clk	RC Osc	0000

1.For all fuses “1” means unprogrammed while “0” means programmed
2.Ext Osc : External Osc
3.RC Osc : Internal RC Oscillator
4.Ext Clk : External Clock Input

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before starting normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 7-2. The

frequency of the Watchdog Oscillator is voltage dependent as shown in “Watchdog Oscillator Frequency vs. VCC” on page 270.

Table 7-2. Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 5.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

7.3 Default Clock Source

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

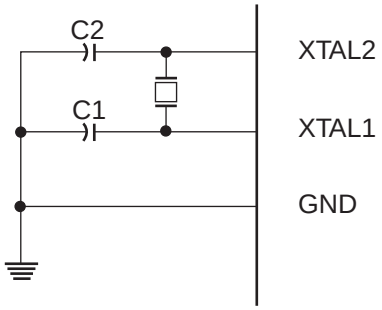
7.4 Low Power Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 7-2. Either a quartz crystal or a ceramic resonator may be used.

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

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 7-3. For ceramic resonators, the capacitor values given by the manufacturer should be used. For more information on how to choose capacitors and other details on Oscillator operation, refer to the Multi-purpose Oscillator Application Note.

Figure 7-2. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 7-3.

Table 7-3. Crystal Oscillator Operating Modes

CKSEL3..1	Frequency Range[1] (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100[2]	0.4 - 0.9	–

Table 7-3. Crystal Oscillator Operating Modes

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

Notes: 1. The frequency ranges are preliminary values. Actual values are TBD.
2. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 7-4.

Table 7-4. Start-up Times for the Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	14CK + 4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	14CK + 65 ms	Ceramic resonator, slowly rising power
0	10	1K CK[2]	14CK	Ceramic resonator, BOD enabled
0	11	1K CK[2]	14CK + 4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK[2]	14CK + 65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	14CK	Crystal Oscillator, BOD enabled
1	10	16K CK	14CK + 4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	14CK + 65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

7.5 Calibrated Internal RC Oscillator

By default, the Internal RC OScillator provides an approximate 8.0 MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. The device is shipped with the CKDIV8 Fuse programmed. See “System Clock Prescaler” on page 35 for more details.

This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 7-5. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL Register and thereby automatically cal-

ibrates the RC Oscillator. The accuracy of this calibration is shown as Factory calibration in Table 23-1 on page 241.

By changing the OSCCAL register from SW, see "Oscillator Calibration Register – OSCCAL" on page 31, it is possible to get a higher calibration accuracy than by using the factory calibration. The accuracy of this calibration is shown as User calibration in Table 23-1 on page 241.

When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 223.

Table 7-5. Internal Calibrated RC Oscillator Operating Modes(1)(2)

Frequency Range (MHz)	CKSEL3..0
7.3 - 8.1	0010

Notes: 1. The device is shipped with this option selected.
2. If 8 MHz frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 7-6 on page 31.

Table 7-6. Start-up times for the internal calibrated RC Oscillator clock selection

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

Note: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4.1 ms to ensure programming mode can be entered.
2. The device is shipped with this option selected.

Table 7-7. Oscillator Calibration Register – OSCCAL

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

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

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

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

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

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

7.6 PLL

To generate high frequency and accurate PWM waveforms, the ‘PSC’s need high frequency clock input. This clock is generated by a PLL. To keep all PWM accuracy, the frequency factor of PLL must be configurable by software. With a system clock of 8 MHz, the PLL output is 32Mhz or 64Mhz.

7.6.1 Internal PLL for PSC

The internal PLL in AT90PWM1 generates a clock frequency that is 64x multiplied from nominally 1 MHz input. The source of the 1 MHz PLL input clock is the output of the internal RC Oscillator which is divided down to 1 MHz. See the Figure 7-3 on page 33.

The PLL is locked on the RC Oscillator and adjusting the RC Oscillator via OSCCAL Register will adjust the fast peripheral clock at the same time. However, even if the possibly divided RC Oscillator is taken to a higher frequency than 1 MHz, the fast peripheral clock frequency saturates at 70 MHz (worst case) and remains oscillating at the maximum frequency. It should be noted that the PLL in this case is not locked any more with the RC Oscillator clock.

Therefore it is recommended not to take the OSCCAL adjustments to a higher frequency than 1 MHz in order to keep the PLL in the correct operating range. The internal PLL is enabled only when the PLLE bit in the register PLLCSR is set. The bit PLOCK from the register PLLCSR is set when PLL is locked.

Both internal 1 MHz RC Oscillator and PLL are switched off in Power-down and Standby sleep modes

.

Table 7-8. Start-up Times when the PLL is selected as system clock

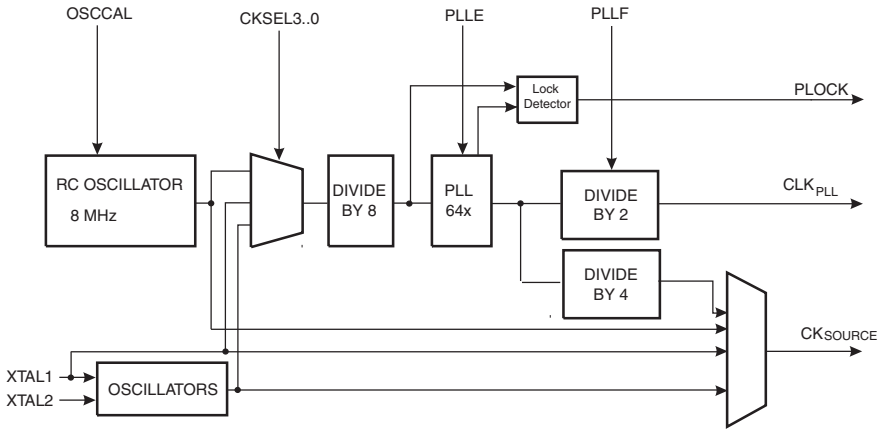
CKSEL 3..0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)
0011 RC Osc	00	1K CK	14CK
	01	1K CK	14CK + 4 ms
	10	1K CK	14CK + 64 ms
	11	16K CK	14CK
0101 Ext Osc	00	1K CK	14CK
	01	1K CK	14CK + 4 ms
	10	16K CK	14CK + 4 ms
	11	16K CK	14CK + 64 ms

Table 7-8. Start-up Times when the PLL is selected as system clock

CKSEL 3..0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)
0001 Ext Clk	00	6 CK (1)	14CK
	01	6 CK (2)	14CK + 4 ms
	10	6 CK (3)	14CK + 64 ms
	11	Reserved	

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

Figure 7-3. PCK Clocking System AT90PWM2/3



7.6.2 PLL Control and Status Register – PLLCSR

Bit	7	6	5	4	3	2	1	0	
$29 ($29)	–	–	–	–	–	PLLF	PLLE	PLOCK	PLLCSR
Read/Write	R	R	R	R	R	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0/1	0	

- **Bit 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 and always read as zero.

- **Bit 2 – PLLF: PLL Factor**

The PLLF bit is used to select the division factor of the PLL.

If PLLF is set, the PLL output is 64Mhz.

If PLLF is clear, the PLL output is 32Mhz.

- **Bit 1 – PLLE: PLL Enable**

When the PLLE is set, the PLL is started and if not yet started the internal RC Oscillator is started as PLL reference clock. If PLL is selected as a system clock source the value for this bit is always 1.

- **Bit 0 – PLOCK: PLL Lock Detector**

When the PLOCK bit is set, the PLL is locked to the reference clock, and it is safe to enable CLK_{PLL} for PSC. After the PLL is enabled, it takes about 100 ms for the PLL to lock.

7.7 128 kHz Internal Oscillator

The 128 kHz internal Oscillator is a low power Oscillator providing a clock of 128 kHz. The frequency is nominal at 3V and 25°C. This clock is used by the Watchdog Oscillator.

7.8 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 7-4. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000”.

Figure 7-4. External Clock Drive Configuration

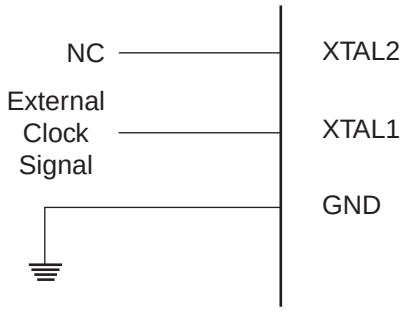


Table 7-9. External Clock Frequency

CKSEL3..0	Frequency Range
0000	0 - 16 MHz

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 7-10.

Table 7-10. Start-up Times for the External Clock Selection

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

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to “System Clock Prescaler” on page 35 for details.

7.9 Clock Output Buffer

When the CKOUT Fuse is programmed, the system Clock will be output on CLKO. This mode is suitable when chip clock is used to drive other circuits on the system. The clock will be output also during reset and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including internal RC Oscillator, can be selected when CLKO serves as clock output. If the System Clock Prescaler is used, it is the divided system clock that is output (CKOUT Fuse programmed).

7.10 System Clock Prescaler

The AT90PWM1 system clock can be divided by setting the Clock Prescale Register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 7-11.

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting. The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler - even if it were readable, and the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2 * T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

7.10.1 Clock Prescaler Register – CLKPR

Bit	7	6	5	4	3	2	1	0	
	CLKPCE	–	–	–	**CLKPS3**	**CLKPS2**	**CLKPS1**	**CLKPS0**	**CLKPR**
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaniosly written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 3..0 – CLKPS3..0: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 7-11.

The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selcted clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 Fuse programmed.

Table 7-11. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

8. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the five sleep modes, the SE bit in SMCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the SMCR Register select which sleep mode (Idle, ADC Noise Reduction, Power-down, Power-save, or Standby) will be activated by the SLEEP instruction. See Table 3 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 7-1 on page 27 presents the different clock systems in the AT90PWM1, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

8.0.1 Sleep Mode Control Register – SMCR

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 3..1 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the five available sleep modes as shown in Table 3.

Table 3. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Reserved

Note: 1. Standby mode is only recommended for use with external crystals or resonators.

- **Bit 1 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

8.1 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 SPI, USART, Analog Comparator, ADC, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halt 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.

8.2 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, Timer/Counter (if their clock source is external - T0 or T1) 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 Reset, a Brown-out Reset, a Timer/Counter interrupt, an SPM/EEPROM ready interrupt, an External Level Interrupt on INT3:0 can wake up the MCU from ADC Noise Reduction mode.

8.3 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 and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a PSC Interrupt, an External Level Interrupt on INT3:0 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 “External Interrupts” on page 74 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 “Clock Sources” on page 28.

8.4 Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down

with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Table 4. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators	Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PLL}	Main Clock Source Enabled	INT3..0	PSC	SPM/EEPROM Ready	ADC	WDT	OtherI/O
Idle			X	X	X	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X(2)	X	X	X	X	
Power-down							X(2)	X			X	
Standby(1)						X	X(2)				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. Only level interrupt.

8.5 Power Reduction Register

The Power Reduction Register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

A full predictible behaviour of a peripheral is not guaranteed during and after a cycle of stopping and starting of its clock. So its recommended to stop a peripheral before stopping its clock with PRR register.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

8.5.1 Power Reduction Register - PRR

Bit	7	6	5	4	3	2	0	
	PRPSC2	**PRPSC1**	**PRPSC0**	**PRTIM1**	**PRTIM0**	**PRSPI**	**PRADC**	**PRR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	

- **Bit 7 - PRPSC2: Power Reduction PSC2**

Writing a logic one to this bit reduces the consumption of the PSC2 by stopping the clock to this module. When waking up the PSC2 again, the PSC2 should be re initialized to ensure proper operation.

- **Bit 6 - PRPSC1: Power Reduction PSC1**

Writing a logic one to this bit reduces the consumption of the PSC1 by stopping the clock to this module. When waking up the PSC1 again, the PSC1 should be re initialized to ensure proper operation.

- **Bit 5 - PRPSC0: Power Reduction PSC0**

Writing a logic one to this bit reduces the consumption of the PSC0 by stopping the clock to this module. When waking up the PSC0 again, the PSC0 should be re initialized to ensure proper operation.

- **Bit 4 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit reduces the consumption of the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the setting of this bit.

- **Bit 3 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit reduces the consumption of the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the setting of this bit.

- **Bit 2 - PRSPI: Power Reduction Serial Peripheral Interface**

Writing a logic one to this bit reduces the consumption of the Serial Peripheral Interface by stopping the clock to this module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

- **Bit 0 - PRADC: Power Reduction ADC**

Writing a logic one to this bit reduces the consumption of the ADC by stopping the clock to this module. The ADC must be disabled before using this function. The analog comparator cannot use the ADC input MUX when the clock of ADC is stopped.

8.6 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

8.6.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "CROSS REFERENCE REMOVED" for details on ADC operation.

8.6.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 176 for details on how to configure the Analog Comparator.

8.6.3 Brown-out Detector

If the Brown-out Detector is not needed by the application, this module should be turned off. If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute sig-

nificantly to the total current consumption. Refer to “Brown-out Detection” on page 45 for details on how to configure the Brown-out Detector.

8.6.4 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to “Internal Voltage Reference” on page 47 for details on the start-up time.

8.6.5 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to “Watchdog Timer” on page 48 for details on how to configure the Watchdog Timer.

8.6.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section “I/O-Ports” on page 58 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the Digital Input Disable Registers (DIDR1 and DIDR0). Refer to “Digital Input Disable Register 1– DIDR1” and “Digital Input Disable Register 0 – DIDR0” on page 180 and page 199 for details.

8.6.7 On-chip Debug System

If the On-chip debug system is enabled by OCDEN Fuse and the chip enter sleep mode, the main clock source is enabled, and hence, always consumes power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

9. System Control and Reset

9.0.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 9-1 shows the reset logic. Table 9-1 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in “Clock Sources” on page 28.

9.0.2 Reset Sources

The AT90PWM1 has four sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.

Figure 9-1. Reset Logic

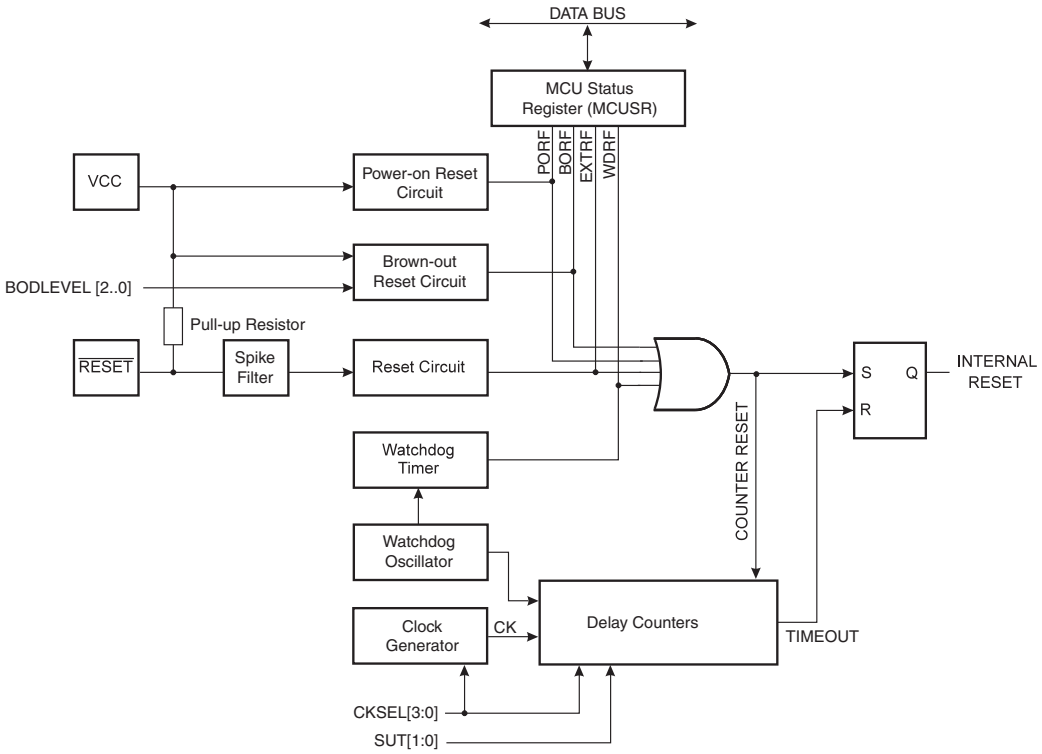


Table 9-1. Reset Characteristics[1]

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			1.4	2.3	V
	Power-on Reset Threshold Voltage (falling)[2]			1.3	2.3	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.2Vcc		0.85Vcc	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin			400		ns

Notes: 1. Values are guidelines only..
2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

9.0.3 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 9-1. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 9-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

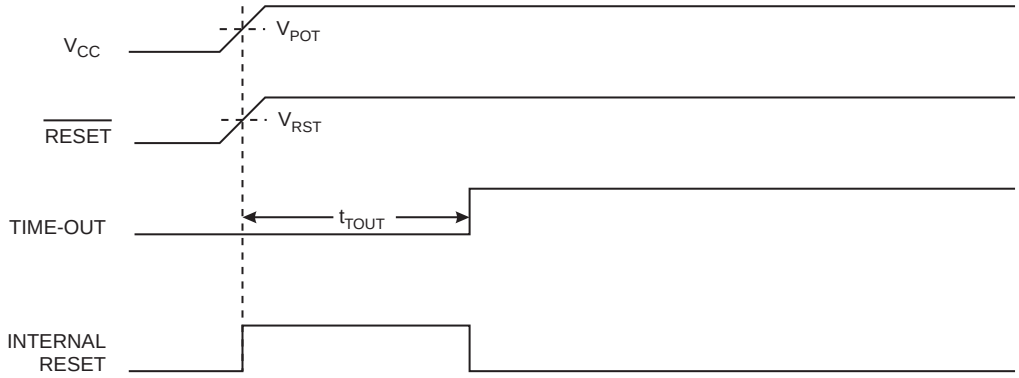
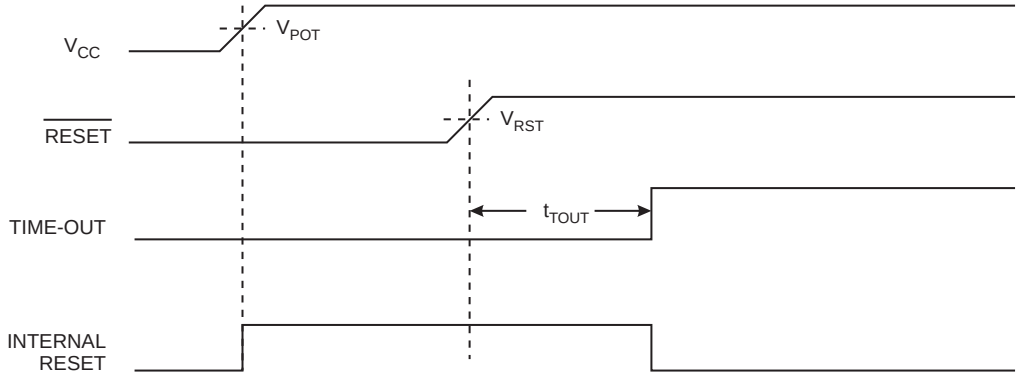


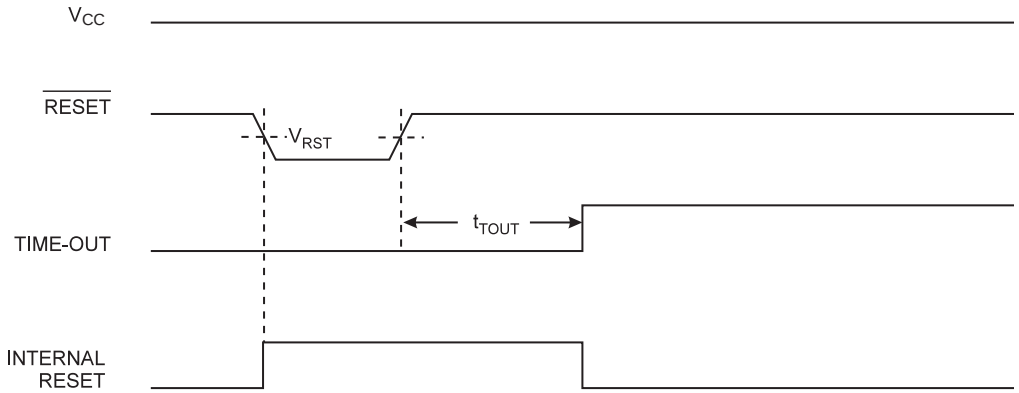
Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



9.0.4 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 9-1) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period – t_{TOUT} – has expired.

Figure 9-4. External Reset During Operation



9.0.5 Brown-out Detection

AT90PWM1 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

Table 9-2. BODLEVEL Fuse Coding[1][2]

BODLEVEL 2..0 Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110		4.5		V
101		2.7		V
100		4.3		V
011		4.4		V
010		4.2		V
001		2.8		V
000		2.6		V

Notes: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-Out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 010 for Low Operating Voltage and BODLEVEL = 101 for High Operating Voltage .
2. Values are guidelines only.

Table 9-3. Brown-out Characteristics[1]

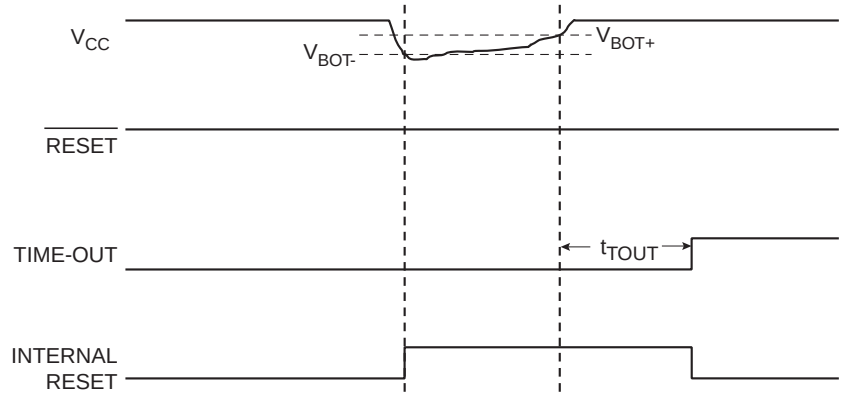
Symbol	Parameter	Min.	Typ.	Max.	Units
V_{HYST}	Brown-out Detector Hysteresis		70		mV
t_{BOD}	Min Pulse Width on Brown-out Reset		2		µs

Notes: 1. Values are guidelines only.

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-5), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 9-3.

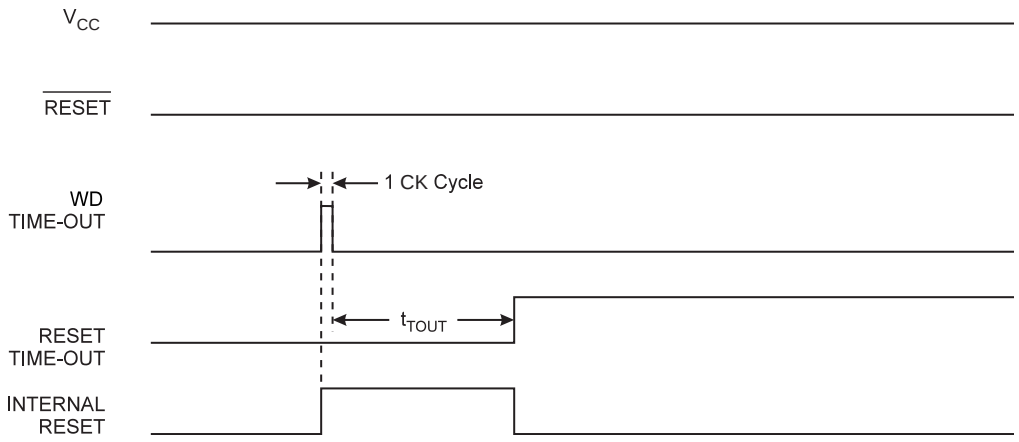
Figure 9-5. Brown-out Reset During Operation



9.0.6 Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 48 for details on operation of the Watchdog Timer.

Figure 9-6. Watchdog Reset During Operation



9.0.7 MCU Status Register – MCUSR

The MCU Status Register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the reset flags.

9.1 Internal Voltage Reference

AT90PWM1 features an internal bandgap reference. This reference is used for Brown-out Detection. It can also be used as a voltage reference for the DAC and/or the ADC, and can be used as analog input for the analog comparators. In order to use the internal Vref, it is necessary to configure it thanks to the REFS1 and REFS0 bits in the ADMUX register and to set an analog feature which requires it.

9.1.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 9-4. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.
4. When the DAC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC or the DAC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC or DAC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

9.1.2 Voltage Reference Characteristics

Table 9-4. Internal Voltage Reference Characteristics[1]

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{BG}	Bandgap reference voltage			1.1		V
t_{BG}	Bandgap reference start-up time			40		µs
I_{BG}	Bandgap reference current consumption			15		µA

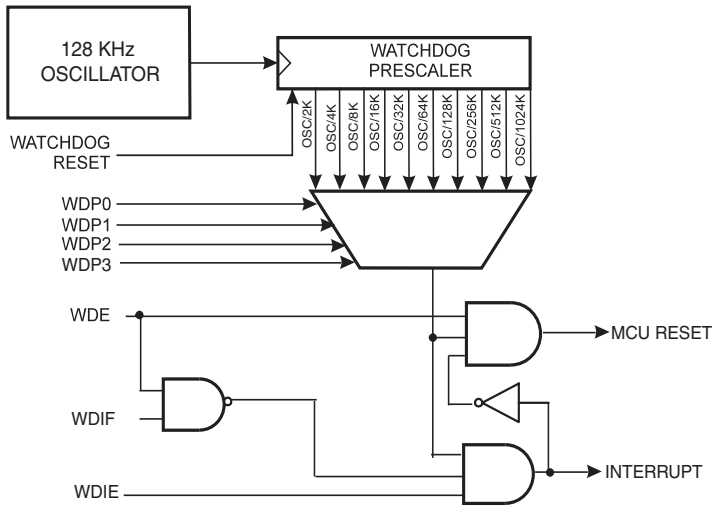
Note: 1. Values are guidelines only.

9.2 Watchdog Timer

AT90PWM1 has an Enhanced Watchdog Timer (WDT). The main features are:

- **Clocked from separate On-chip Oscillator**
- **3 Operating modes**
  - **Interrupt**
  - **System Reset**
  - **Interrupt and System Reset**
- **Selectable Time-out period from 16ms to 8s**
- **Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode**

Figure 9-7. Watchdog Timer



The Watchdog Timer (WDT) is a timer counting cycles of a separate on-chip 128 kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

In Interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In System Reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, Interrupt and System Reset mode, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The “Watchdog Timer Always On” (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and Watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example[1]

```
WDT_off:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Clear WDRF in MCUSR
  in    r16, MCUSR
  andi  r16, (0xff & (0<<WDRF))
  out   MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional time-out
  lds r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  sts WDTCSR, r16
  ; Turn off WDT
  ldi   r16, (0<<WDE)
  sts WDTCSR, r16
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Clear WDRF in MCUSR */
  MCUSR &= ~(1<<WDRF);
  /* Write logical one to WDCE and WDE */
  /* Keep old prescaler setting to prevent unintentional time-out */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

Note: 1. The example code assumes that the part specific header file is included.

Note: If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the Watchdog Timer will stay enabled. If the code is not set up to handle the Watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the Watchdog System Reset Flag (WDRF) and the WDE control bit in the initialisation routine, even if the Watchdog is not in use.

The following code example shows one assembly and one C function for changing the time-out value of the Watchdog Timer.

Assembly Code Example[1]

```
WDT_Prescaler_Change:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Start timed sequence
  lds r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  sts WDTCSR, r16
  ; --  Got four cycles to set the new values from here -
  ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  sts WDTCSR, r16
  ; --  Finished setting new values, used 2 cycles -
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed  equence */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note: 1. The example code assumes that the part specific header file is included.

Note: The Watchdog Timer should be reset before any change of the WDP bits, since a change in the WDP bits can result in a time-out when switching to a shorter time-out period;

9.2.1 Watchdog Timer Control Register - WDTCSR

Bit	7	6	5	4	3	2	1	0	
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- **Bit 7 - WDIF: Watchdog Interrupt Flag**

This bit is set when a time-out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

- **Bit 6 - WDIE: Watchdog Interrupt Enable**

When this bit is written to one and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if time-out in the Watchdog Timer occurs.

If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode). This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 9-5. Watchdog Timer Configuration

WDTON(1)	WDE	WDIE	Mode	Action on Time-out
0	0	0	Stopped	None
0	0	1	Interrupt Mode	Interrupt
0	1	0	System Reset Mode	Reset
0	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
1	x	x	System Reset Mode	Reset

Note: 1. For the WDTON Fuse “1” means unprogrammed while “0” means programmed.

- **Bit 4 - WDCE: Watchdog Change Enable**

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set.

Once written to one, hardware will clear WDCE after four clock cycles.

- **Bit 3 - WDE: Watchdog System Reset Enable**

WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

- **Bit 5, 2..0 - WDP3..0: Watchdog Timer Prescaler 3, 2, 1 and 0**

The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is running. The different prescaling values and their corresponding time-out periods are shown in Table 9-6 on page 52.

Table 9-6. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

10. Interrupts

This section describes the specifics of the interrupt handling as performed in AT90PWM1. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 14.

10.1 Interrupt Vectors in AT90PWM1

Table 5. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and Emulation AVR Reset
2	0x0001	PSC2 CAPT	PSC2 Capture Event
3	0x0002	PSC2 EC	PSC2 End Cycle
4	0x0003	PSC1 CAPT	PSC1 Capture Event
5	0x0004	PSC1 EC	PSC1 End Cycle
6	0x0005	PSC0 CAPT	PSC0 Capture Event
7	0x0006	PSC0 EC	PSC0 End Cycle
8	0x0007	ANACOMP 0	Analog Comparator 0
9	0x0008	ANACOMP 1	Analog Comparator 1
10	0x0009	ANACOMP 2	Analog Comparator 2
11	0x000A	INT0	External Interrupt Request 0
12	0x000B	TIMER1 CAPT	Timer/Counter1 Capture Event
13	0x000C	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	0x000D	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	0x000E		
16	0x000F	TIMER1 OVF	Timer/Counter1 Overflow
17	0x0010	TIMER0 COMPA	Timer/Counter0 Compare Match A
18	0x0011	TIMER0 OVF	Timer/Counter0 Overflow
19	0x0012	ADC	ADC Conversion Complete
20	0x0013	INT1	External Interrupt Request 1
21	0x0014	SPI, STC	SPI Serial Transfer Complete
22	0x0015		
23	0x0016		
24	0x0017		
25	0x0018	INT2	External Interrupt Request 2
26	0x0019	WDT	Watchdog Time-Out Interrupt
27	0x001A	EE READY	EEPROM Ready
28	0x001B	TIMER0 COMPB	Timer/Counter0 Compare Match B

Table 5. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
29	0x001C	INT3	External Interrupt Request 3
30	0x001D		
31	0x001E		
32	0x001F	SPM READY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 205.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

Table 6 shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 6. Reset and Interrupt Vectors Placement in AT90PWM1(1)

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

Note: 1. The Boot Reset Address is shown in Table 70 on page 218. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in AT90PWM1 is:

```
Address  Labels Code                 Comments
0x000           rjmp   RESET         ; Reset Handler
0x001           rjmp   PSC2_CAPT      ; PSC2 Capture event Handler
0x002           rjmp   PSC2_EC        ; PSC2 End Cycle Handler
0x003           rjmp   PSC1_CAPT      ; PSC1 Capture event Handler
0x004           rjmp   PSC1_EC        ; PSC1 End Cycle Handler
0x005           rjmp   PSC0_CAPT      ; PSC0 Capture event Handler
0x006           rjmp   PSC0_EC        ; PSC0 End Cycle Handler
0x007           rjmp   ANA_COMP_0     ; Analog Comparator 0 Handler
0x008           rjmp   ANA_COMP_1     ; Analog Comparator 1 Handler
0x009           rjmp   ANA_COMP_2     ; Analog Comparator 2 Handler
0x00A           rjmp   EXT_INT0       ; IRQ0 Handler
0x00B           rjmp   TIM1_CAPT      ; Timer1 Capture Handler
0x00C           rjmp   TIM1_COMPA     ; Timer1 Compare A Handler
0x00D           rjmp   TIM1_COMPB     ; Timer1 Compare B Handler

0x00F           rjmp   TIM1_OVF       ; Timer1 Overflow Handler
```

```
0x010           rjmp   TIM0_COMPA      ; Timer0 Compare A Handler
0x011           rjmp   TIM0_OVF        ; Timer0 Overflow Handler
0x012           rjmp   ADC             ; ADC Conversion Complete Handler
0x013           rjmp   EXT_INT1        ; IRQ1 Handler
0x014           rjmp   SPI_STC         ; SPI Transfer Complete Handler
0x018           rjmp   EXT_INT2        ; IRQ2 Handler
0x019           rjmp   WDT             ; Watchdog Timer Handler
0x01A           rjmp   EE_RDY          ; EEPROM Ready Handler
0x01B           rjmp   TIM0_COMPB      ; Timer0 Compare B Handler
0x01C           rjmp   EXT_INT3        ; IRQ3 Handler
0x01F           rjmp   SPM_RDY         ; Store Program Memory Ready Handler
;
0x020RESET:     ldi    r16, high(RAMEND); Main program start
0x021           out    SPH,r16         ; Set Stack Pointer to top of RAM
0x022           ldi    r16, low(RAMEND)
0x023         out    SPL,r16
0x024           sei                    ; Enable interrupts
0x025           <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 AT90PWM1 is:

```
Address  Labels Code                  Comments
0x000    RESET: ldi    r16,high(RAMEND); Main program start
0x001           out    SPH,r16         ; Set Stack Pointer to top of RAM
0x002           ldi    r16,low(RAMEND)
0x003           out    SPL,r16
0x004           sei                    ; Enable interrupts
0x005           <instr>  xxx
;
.org 0xC01
0xC01           rjmp   PSC2_CAPT       ; PSC2 Capture event Handler
0xC02           rjmp   PSC2_EC         ; PSC2 End Cycle Handler
...             ...    ...             ;
0xC1F           rjmp   SPM_RDY         ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 2K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses in AT90PWM1 is:

```
Address  Labels Code             Comments
.org 0x001
0x001           rjmp   PSC2_CAPT       ; PSC2 Capture event Handler
0x002           rjmp   PSC2_EC         ; PSC2 End Cycle Handler
...             ...    ...             ;
0x01F           rjmp   SPM_RDY         ; Store Program Memory Ready Handler
;
.org 0xC00
0xC00    RESET: ldi    r16,high(RAMEND); Main program start
```

```
0xC01          out    SPH,r16         ; Set Stack Pointer to top of RAM
0xC02          ldi    r16,low(RAMEND)
0xC03          out    SPL,r16
0xC04          sei                    ; Enable interrupts
0xC05          <instr>  xxx
```

When the BOOTRST Fuse is programmed, 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 AT90PWM1 is:

```
Address  Labels Code                  Comments
;
.org 0xC00
0xC00          rjmp   RESET           ; Reset handler
0xC01          rjmp   PSC2_CAPT       ; PSC2 Capture event Handler
0xC02          rjmp   PSC2_EC         ; PSC2 End Cycle Handler
...            ...    ...             ;
0xC1F          rjmp   SPM_RDY         ; Store Program Memory Ready Handler
;
0xC20    RESET: ldi    r16,high(RAMEND); Main program start
0xC21          out    SPH,r16         ; Set Stack Pointer to top of RAM
0xC22          ldi    r16,low(RAMEND)
0xC23          out    SPL,r16
0xC24          sei                    ; Enable interrupts
0xC25          <instr>  xxx
```

10.1.1 Moving Interrupts Between Application and Boot Space

The MCU Control Register controls the placement of the Interrupt Vector table.

10.1.2 MCU Control Register – MCUCR

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

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

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to the section “Boot Loader Support – Read-While-Write Self-Programming” on page 205 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to

IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 205 for details on Boot Lock bits.

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

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

Assembly Code Example

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

C Code Example

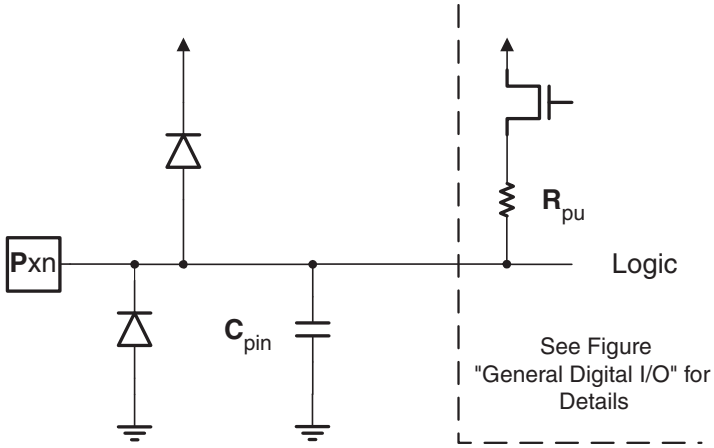
```
void Move_interrupts(void)
{
  /* Enable change of Interrupt Vectors */
  MCUCR = (1<<IVCE);
  /* Move interrupts to Boot Flash section */
  MCUCR = (1<<IVSEL);
}
```

11. I/O-Ports

11.1 Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 11-1. Refer to “Electrical Characteristics(1)” on page 238 for a complete list of parameters.

Figure 11-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in “Register Description for I/O-Ports”.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

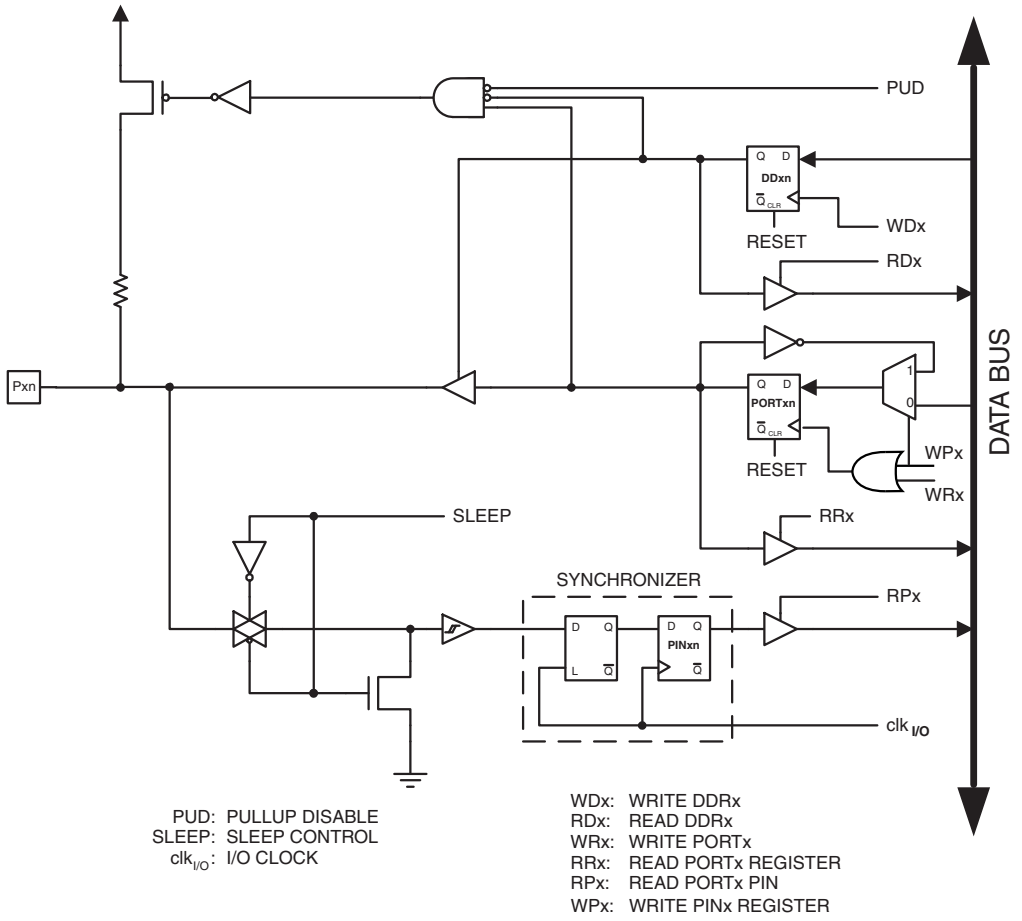
Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O”. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in “Alternate Port Functions” on page 63. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

11.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 11-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 11-2. General Digital I/O[1]



Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

11.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O-Ports" on page 72, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin

The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

11.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

11.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 7 summarizes the control signals for the pin value.

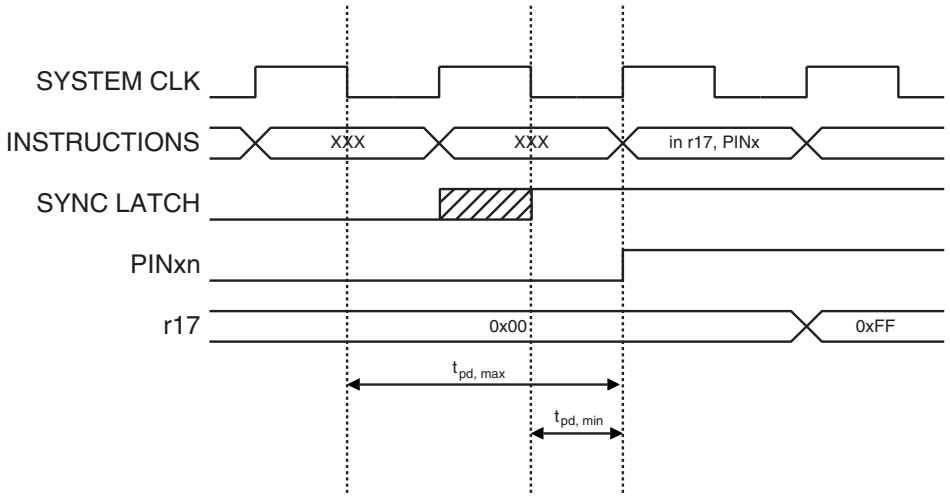
Table 7. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Default configuration after Reset. Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

11.2.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 11-2, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 11-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

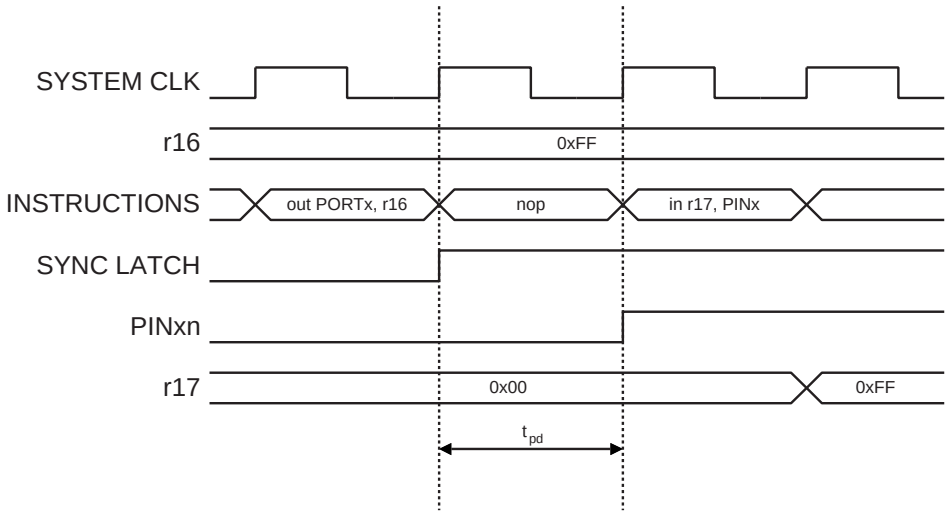
Figure 11-3. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 11-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay t_{pd} through the synchronizer is 1 system clock period.

Figure 11-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin

values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

TABLE 2.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16, (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi  r17, (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out  PORTB, r16
out  DDRB, r17
; Insert nop for synchronization
nop
; Read port pins
in   r16, PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

11.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 11-2, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode, Power-save mode, and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

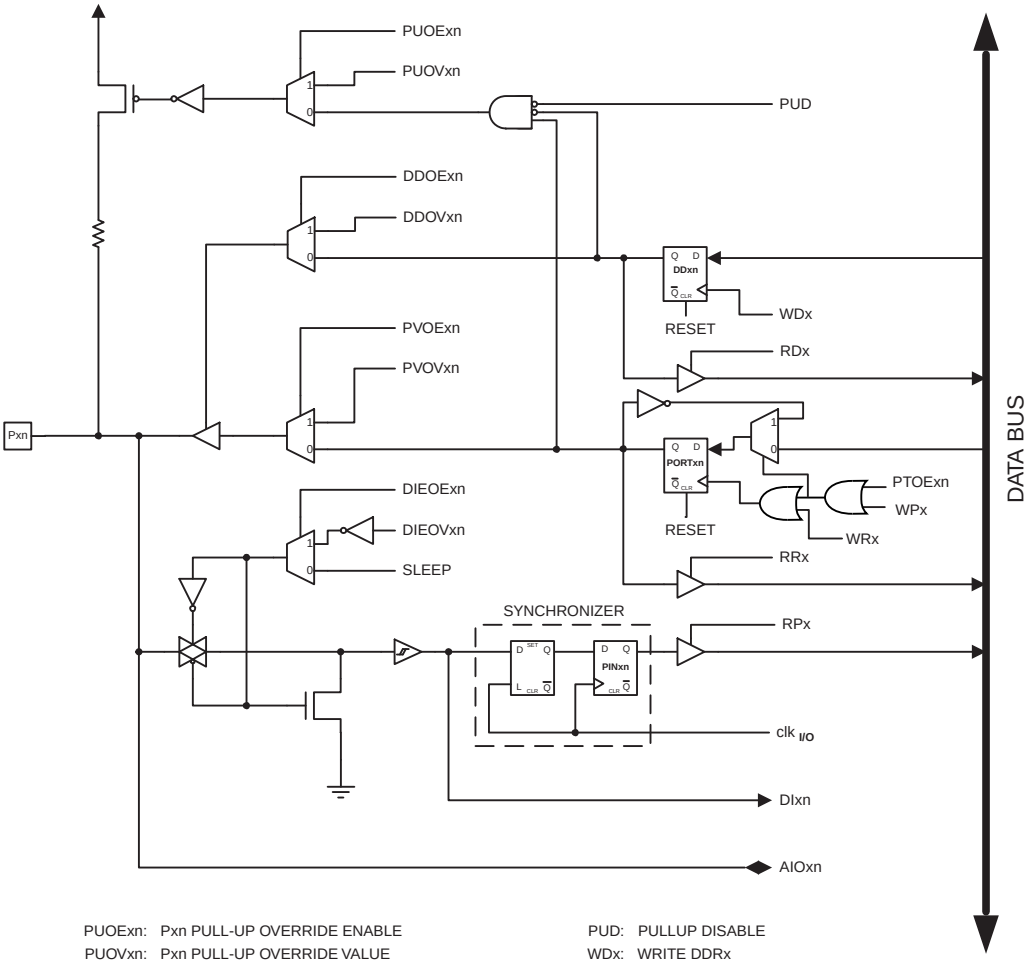
SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in “Alternate Port Functions” on page 63.

If a logic high level (“one”) is present on an Asynchronous External Interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is not enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.

11.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 11-5 shows how the port pin control signals from the simplified Figure 11-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 11-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	$clk_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 8 summarizes the function of the overriding signals. The pin and port indexes from Figure 11-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 8. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

11.3.1 MCU Control Register – MCUCR

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

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). Se

11.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 9.

Table 9. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	PSCOUT01 output ADC4 (Analog Input Channel 4) SCK (SPI Bus Serial Clock)
PB6	ADC7 (Analog Input Channel 7) ICP1B (Timer 1 input capture alternate input) PSCOUT11 output (see note 4)
PB5	ADC6 (Analog Input Channel 6) INT2
PB4	AMP0+ (Analog Differential Amplifier 0 Input Channel)
PB3	AMP0- (Analog Differential Amplifier 0 Input Channel)
PB2	ADC5 (Analog Input Channel5) INT1
PB1	MOSI (SPI Master Out Slave In) PSCOUT21 output
PB0	MISO (SPI Master In Slave Out) PSCOUT20 output

The alternate pin configuration is as follows:

- **PSCOUT01/ADC4/SCK – Bit 7**

PSCOUT01: Output 1 of PSC 0.

ADC4, Analog to Digital Converter, input channel 4.

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.

- **ADC7/ICP1B/PSCOUT11 – Bit 6**

ADC7, Analog to Digital Converter, input channel 7.

ICP1B, Input Capture Pin: The PB6 pin can act as an Input Capture Pin for Timer/Counter1.

PSCOUT11: Output 1 of PSC 1.

- **ADC6/$\overline{\text{INT2}}$ – Bit 5**

ADC6, Analog to Digital Converter, input channel 6.

INT2, External Interrupt source 2. This pin can serve as an External Interrupt source to the MCU.

- **APM0+ – Bit 4**

AMP0+, Analog Differential Amplifier 0 Positive Input Channel.

- **AMP0- – Bit 3**

AMP0-, Analog Differential Amplifier 0 Negative Input Channel.

- **ADC5/$\overline{\text{INT1}}$ – Bit 2**

ADC5, Analog to Digital Converter, input channel 5.

INT1, External Interrupt source 1. This pin can serve as an external interrupt source to the MCU.

- **MOSI/PSCOUT21 – Bit 1**

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB1 When the SPI is enabled as a master, the data direction of this pin is controlled by DDB1. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB1 and PUD bits.

PSCOUT21: Output 1 of PSC 2.

- **MISO/PSC20 – Bit 0**

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB0. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 and PUD bits.

PSCOUT20: Output 0 of PSC 2.

Table 10 and Table 11 relates the alternate functions of Port B to the overriding signals shown in Figure 11-5 on page 63.

Table 10. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/ADC4/ PSCOUT01/SCK	PB6/ADC7/ PSCOUT11/ ICP1B	PB5/ADC6/ INT2	PB4/AMP0+
PUOE	SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$	0	0	0
PUOV	PB7 • $\overline{\text{PUD}}$ • $\overline{\text{SPIPS}}$	0	0	0
DDOE	SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$ + PSCen01	PSCen11	0	0
DDOV	PSCen01	1	0	0
PVOE	SPE • MSTR • $\overline{\text{SPIPS}}$	PSCen11	0	0
PVOV	PSCout01 • SPIPS + PSCout01 • PSCen01 • $\overline{\text{SPIPS}}$ + PSCout01 • PSCen01 • $\overline{\text{SPIPS}}$	PSCOUT11	0	0
DIEOE	ADC4D	ADC7D	ADC6D + In2en	AMP0ND
DIEOV	0	0	In2en	0
DI	$\overline{\text{SCKin}}$ • $\overline{\text{SPIPS}}$ • $\overline{\text{ireset}}$	ICP1B	INT2	
AIO	ADC4	ADC7	ADC6	AMP0+

Table 11. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/AMP0-	PB2/ADC5/INT1	PB1/MOSI/ PSCOUT21	PB0/MISO/ PSCOUT20
PUOE	0	0	–	–
PUOV	0	0	–	–
DDOE	0	0	–	–
DDOV	0	0	–	–
PVOE	0	0	–	–
PVOV	0	0	–	–
DIEOE	AMP0ND	ADC5D + In1en	0	0
DIEOV	0	In1en	0	0
DI		INT1	$\overline{\text{MOSI_IN}}$ • $\overline{\text{SPIPS}}$ • $\overline{\text{ireset}}$	$\overline{\text{MISO_IN}}$ • $\overline{\text{SPIPS}}$ • $\overline{\text{ireset}}$
AIO	AMP0-	ADC5	–	–

11.3.3 Alternate Functions of Port D

The Port D pins with alternate functions are shown in Table 12.

Table 12. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	ACMP0 (Analog Comparator 0 Positive Input)
PD6	ADC3 (Analog Input Channel 3) ACMPM reference for analog comparators INT0
PD5	ADC2 (Analog Input Channel 2) ACMP2 (Analog Comparator 2 Positive Input)
PD4	ADC1 (Analog Input Channel 1) ICP1 (Timer 1 input capture) SCK_A (Programming & alternate SPI Clock)
PD3	OC0A (Timer 0 Output Compare A) SS (SPI Slave Select) MOSI_A (Programming & alternate SPI Master Out Slave In)
PD2	PSCIN2 (PSC 2 Digital Input) OC1A (Timer 1 Output Compare A) MISO_A (Programming & alternate Master In SPI Slave Out)
PD1	PSCIN0 (PSC 0 Digital Input) CLKO (System Clock Output)
PD0	PSCOUT00 output SS_A (Alternate SPI Slave Select)

The alternate pin configuration is as follows:

- **ACMP0 – Bit 7**

ACMP0, Analog Comparator 0 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **ADC3/ACMPM/$\overline{\text{INT0}}$ – Bit 6**

ADC3, Analog to Digital Converter, input channel 3.

ACMPM, Analog Comparators Negative Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

INT0, External Interrupt source 0. This pin can serve as an external interrupt source to the MCU.

- **ADC2/ACMP2 – Bit 5**

ADC2, Analog to Digital Converter, input channel 2.

ACMP2, Analog Comparator 1 Positive Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.

- **ADC1/ICP1/SCK_A – Bit 4**

ADC1, Analog to Digital Converter, input channel 1.

ICP1 – Input Capture Pin1: This pin can act as an input capture pin for Timer/Counter1.

SCK_A: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD4. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD4 bit.

- **OC0A/SS/MOSI_A, Bit 3**

OC0A, Output Compare Match A output: This pin can serve as an external output for the Timer/Counter0 Output Compare A. The pin has to be configured as an output (DDD3 set “one”) to serve this function. The OC0A pin is also the output pin for the PWM mode

$\overline{SS}$: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD3. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD3 bit.

MOSI_A: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD3 When the SPI is enabled as a master, the data direction of this pin is controlled by DDD3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD3 bit.

- **PSCIN2/OC1A/MISO_A, Bit 2**

PCSIN2, PSC 2 Digital Input.

OC1A, Output Compare Match A output: This pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDD2 set “one”) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

MISO_A: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDD2. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDD2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD2 bit.

- **PSCIN0/CLKO – Bit 1**

PCSIN0, PSC 0 Digital Input.

CLKO, Divided System Clock: The divided system clock can be output on this pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTD1 and DDD1 settings. It will also be output during reset.

- **PSCOUT00/SS_A – Bit 0**

PSCOUT00: Output 0 of PSC 0.

$\overline{SS}$_A: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDD0. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDD0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTD0 bit.

Table 13 and Table 14 relates the alternate functions of Port D to the overriding signals shown in Figure 11-5 on page 63.

Table 13. Overriding Signals for Alternate Functions PD7..PD4

Signal Name	PD7/ ACMP0	PD6/ADC3/ ACMPM/INT0	PD5/ADC2/ ACMP2	PD4/ADC1/ ICP1A/SCK_A
PUOE	0	0	0	SPE • $\overline{\text{MSTR}}$ • SPIPS
PUOV	0	0	0	PD4 • PUD
DDOE	0	0	0	SPE • $\overline{\text{MSTR}}$ • SPIPS
DDOV	0	0	0	0
PVOE	0	0	0	SPE • MSTR • SPIPS
PVOV	0	0	0	–
DIEOE	ACMP0D	ADC3D + In0en	ADC2D	ADC1D
DIEOV	0	In0en	0	0
DI	–	INT0		ICP1A
AIO	ACOMP0	ADC3 ACMPM	ADC2 ACOMP2	ADC1

Table 14. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/OC0A/ SS/MOSI_A	PD2/PSCIN2/ OC1A/MISO_A	PD1/PSCIN0/ CLKO	PD0/PSCOUT00/ SS_A
PUOE	SPE • $\overline{\text{MSTR}}$ • SPIPS	–	0	SPE • $\overline{\text{MSTR}}$ • SPIPS
PUOV	SPE • $\overline{\text{MSTR}}$ • $\overline{\text{SPIPS}}$ • PD3 • $\overline{\text{PUD}}$	–	0	PD0 • $\overline{\text{PUD}}$
DDOE	SPE • $\overline{\text{MSTR}}$ • SPIPS	–	0	PSCen00 + SPE • $\overline{\text{MSTR}}$ • SPIPS
DDOV		0	0	PSCen00
PVOE	OC0en + SPE • MSTR • SPIPS	–	0	PSCen00 + UMSEL
PVOV	TXD + $\overline{\text{TXEN}}$ • ($\overline{\text{OC0en}}$ • OC0 + $\overline{\text{OC0en}}$ • SPIPS • MOSI)	–	0	–
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SS MOSI_Ain			SS_A
AIO				

11.3.4 Alternate Functions of Port E

The Port E pins with alternate functions are shown in Table 15.

Table 15. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE2	XTAL2: XTAL Output ADC0 (Analog Input Channel 0)
PE1	XTAL1: XTAL Input OC0B (Timer 0 Output Compare B)
PE0	RESET# (Reset Input) OCD (On Chip Debug I/O)

The alternate pin configuration is as follows:

- **XTAL2/ADC0 – Bit 2**

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

ADC0, Analog to Digital Converter, input channel 0.

- **XTAL1/OC0B – Bit 1**

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

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

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

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

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

Table 16 relates the alternate functions of Port E to the overriding signals shown in Figure 11-5 on page 63.

Table 16. Overriding Signals for Alternate Functions in PE2..PE0

Signal Name	PE2/ADC0/ XTAL2	PE1/OC0B	PE0/$\overline{\text{RESET}}$/ OCD
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	OC0Ben	0
PVOV	0	OC0B	0
DIEOE	ADC0D	0	0
DIEOV	0	0	0
DI			
AIO	Osc Output ADC0	Osc / Clock input	

11.4 Register Description for I/O-Ports

11.4.1 Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

11.4.2 Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

11.4.3 Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

11.4.4 Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

11.4.5 Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

11.4.6 Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

11.4.7 Port E Data Register – PORTE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PORTE2	PORTE1	PORTE0	PORTE
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.8 Port E Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	DDE2	DDE1	DDE0	DDRE
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

11.4.9 Port E Input Pins Address – PINE

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PINE2	PINE1	PINE0	PINE
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	N/A	N/A	N/A	

12. External Interrupts

The External Interrupts are triggered by the INT3:0 pins. Observe that, if enabled, the interrupts will trigger even if the INT3:0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The External Interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Registers – EICRA (INT3:0). When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT3:0 requires the presence of an I/O clock, described in “Clock Systems and their Distribution” on page 27. The I/O clock is halted in all sleep modes except Idle mode.

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. This makes the MCU less sensitive to noise. The changed level is sampled twice by the Watchdog Oscillator clock. The period of the Watchdog Oscillator is 1 µs (nominal) at 5.0V and 25°C. The frequency of the Watchdog Oscillator is voltage dependent as shown in the “Electrical Characteristics(1)” on page 238. The MCU will wake up if the input has the required level during this sampling or if it is held until the end of the start-up time. The start-up time is defined by the SUT fuses as described in “System Clock” on page 27. If the level is sampled twice by the Watchdog Oscillator clock but disappears before the end of the start-up time, the MCU will still wake up, but no interrupt will be generated. The required level must be held long enough for the MCU to complete the wake up to trigger the level interrupt.

12.0.1 External Interrupt Control Register A – EICRA

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
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	

- **Bits 7..0 – ISC31, ISC30 – ISC01, ISC00: External Interrupt 3 - 0 Sense Control Bits**

The External Interrupts 3 - 0 are activated by the external pins INT3:0 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in Table 17. Edges on INT3..INT0 are registered asynchronously.The value on the INT3:0 pins are sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. Observe that CPU clock frequency can be lower than the XTAL frequency if the XTAL divider is enabled. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low.

Table 17. Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request
1	0	The falling edge between two samples of INTn generates an interrupt request.
1	1	The rising edge between two samples of INTn generates an interrupt request.

Note: 1. n = 3, 2, 1 or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

12.0.2 External Interrupt Mask Register – EIMSK

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	INT3	INT2	INT1	IINT0	EIMSK
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	

- **Bits 3..0 – INT3 – INT0: External Interrupt Request 3 - 0 Enable**

When an INT3 – INT0 bit is written to one and the I-bit in the Status Register (SREG) is set (one), the corresponding external pin interrupt is enabled. The Interrupt Sense Control bits in the External Interrupt Control Register – EICRA – defines whether the external interrupt is activated on rising or falling edge or level sensed. Activity on any of these pins will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

12.0.3 External Interrupt Flag Register – EIFR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	INTF3	INTF2	INTF1	IINTF0	EIFR
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	

- **Bits 3..0 – INTF3 - INTF0: External Interrupt Flags 3 - 0**

When an edge or logic change on the INT3:0 pin triggers an interrupt request, INTF3:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT3:0 in EIMSK, are set (one), the MCU will jump to the 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. These flags are always cleared when INT3:0 are configured as level interrupt.

13. Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

13.0.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 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 $f_{CLK_I/O}/1024$.

13.0.2 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 Timer/Counter1 and Timer/Counter0. 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 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to 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).

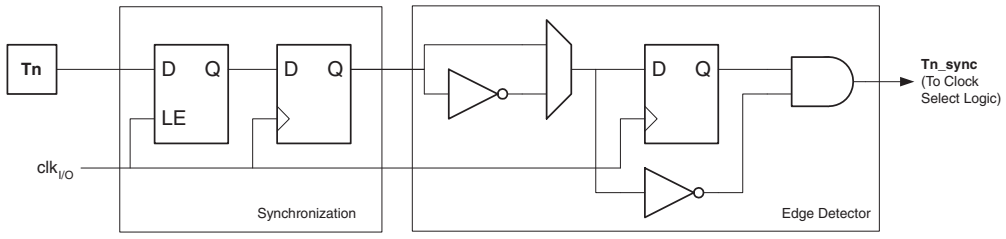
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

13.0.3 External Clock Source

An external clock source applied to the Tn/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The Tn/T0 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 13-1 shows a functional equivalent block diagram of the Tn/T0 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_{T1}/clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 13-1. Tn/T0 Pin Sampling



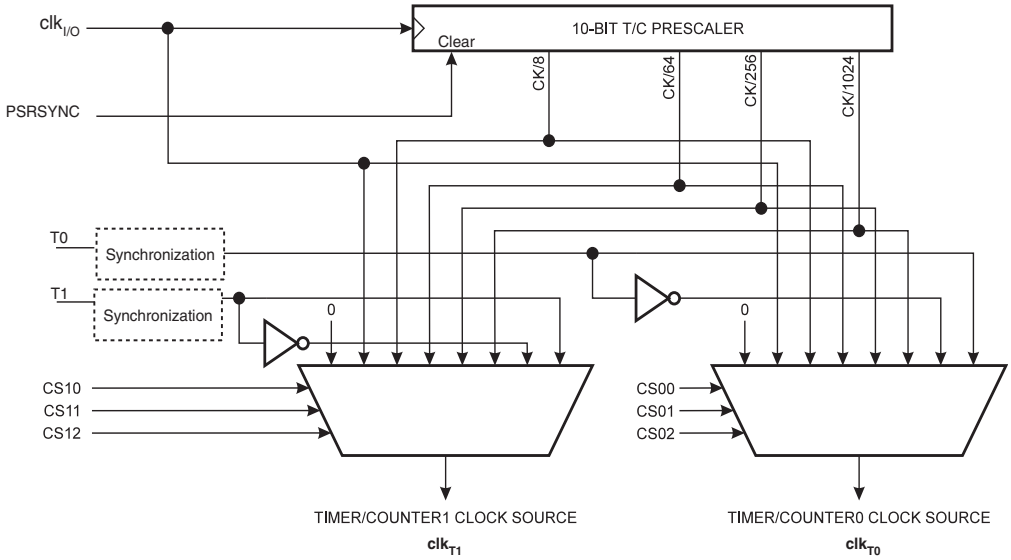
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the Tn/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when Tn/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied 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 an external clock it can detect 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 that maximum frequency of an external clock source is less than $f_{clk_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 13-2. Prescaler for Timer/Counter0 and Timer/Counter1[1]



Note: 1. The synchronization logic on the input pins (Tn/T0) is shown in Figure 13-1.

13.0.4 General Timer/Counter Control Register – GTCCR

Bit	7	6	5	4	3	2	1	0	
	TSM	ICPSEL1	–	–	–	–	–	PSRSYNC	GTCCR
Read/Write	R/W	R/W	R	R	R	R	R	R/W	
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 PSRSYNC bit 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 configuration. When the TSM bit is written to zero, the PSRSYNC bit is cleared by hardware, and the Timer/Counters start counting simultaneously.

- **Bit6 – ICPSEL1: Timer 1 Input Capture selection**

Timer 1 capture function has two possible inputs ICP1A (PD4) and ICP1B (PB6). The selection is made thanks to ICPSEL1 bit as described in Table .

Table 18. ICPSEL1

ICPSEL1	Description
0	Select ICP1A as trigger for timer 1 input capture
1	Select ICP1B as trigger for timer 1 input capture

- **Bit 0 – PSRSYNC: Prescaler Reset**

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

14. 8-bit Timer/Counter0 with PWM

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

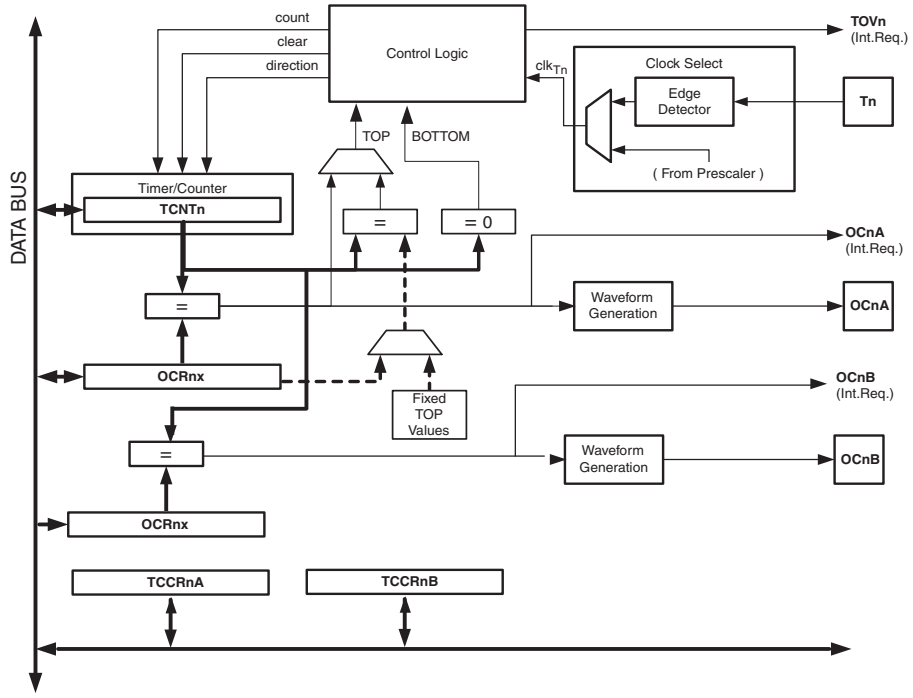
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **Three Independent Interrupt Sources (TOV0, OCF0A, and OCF0B)**

14.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 14-1. For the actual placement of I/O pins, refer to “Pin Descriptions” on page 7. 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 “8-bit Timer/Counter Register Description” on page 89.

The PRTIM0 bit in “Power Reduction Register” on page 39 must be written to zero to enable Timer/Counter0 module.

Figure 14-1. 8-bit Timer/Counter Block Diagram



14.1.1 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 19 are also used extensively throughout the document.

Table 19. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 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 OCR0A Register. The assignment is dependent on the mode of operation.

14.1.2 Registers

The Timer/Counter (TCNT0) and Output Compare Registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge 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_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) 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 (OC0A and OC0B). See “Using the Output Compare Unit” on page 106. for details. The compare match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

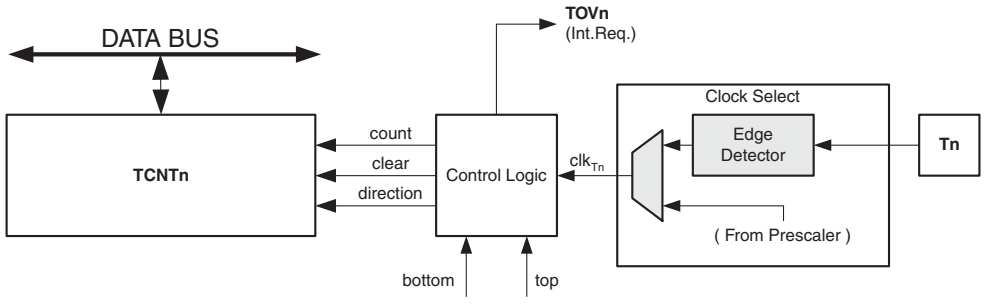
14.2 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 76.

14.3 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 14-2 shows a block diagram of the counter and its surroundings.

Figure 14-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} 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 WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 84.

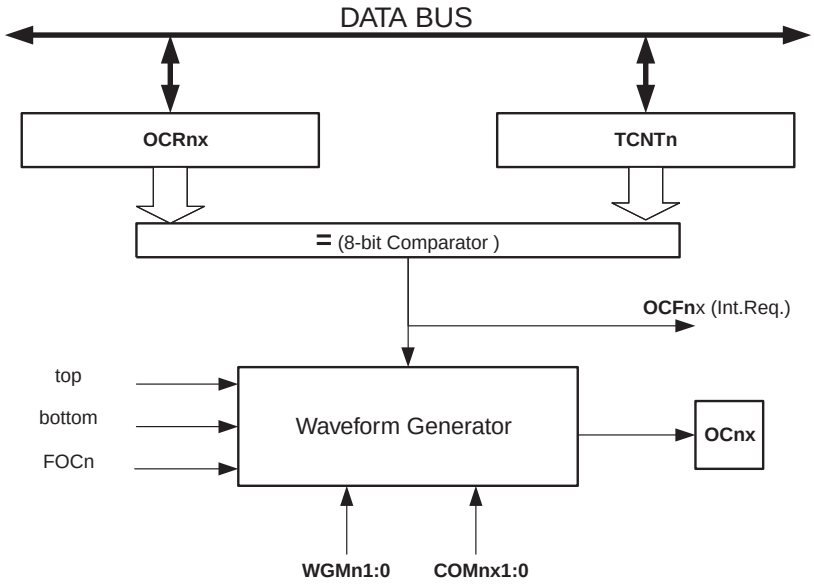
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

14.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) 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 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 WGM02:0 bits and Compare Output mode (COM0x1:0) 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 (“Modes of Operation” on page 84).

Figure 14-3 shows a block diagram of the Output Compare unit.

Figure 14-3. Output Compare Unit, Block Diagram



The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

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

14.4.1 Force Output Compare

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

14.4.2 Compare Match Blocking by TCNT0 Write

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

14.4.3 Using the Output Compare Unit

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

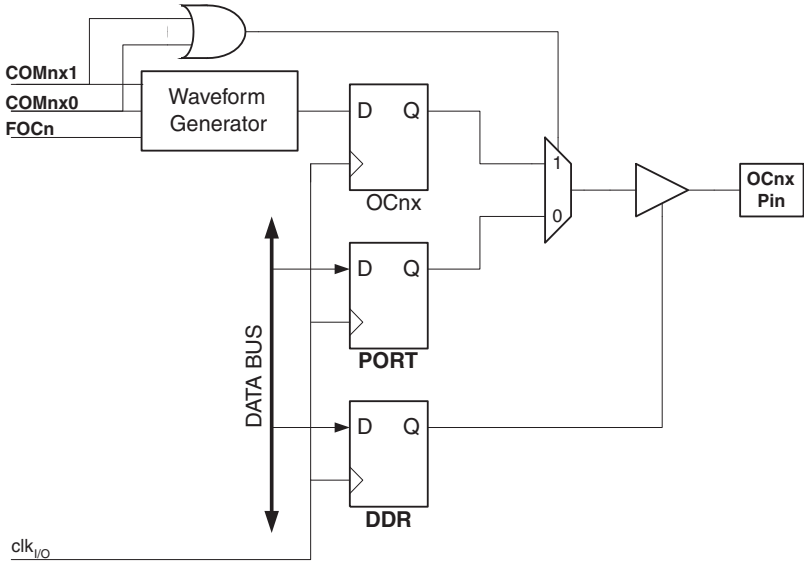
The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation modes.

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

14.5 Compare Match Output Unit

The Compare Output mode (COM0x1:0) bits have two functions. The Waveform Generator uses the COM0x1:0 bits for defining the Output Compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 14-4 shows a simplified schematic of the logic affected by the COM0x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x Register, not the OC0x pin. If a system reset occur, the OC0x Register is reset to “0”.

Figure 14-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0x) from the Waveform Generator if either of the COM0x1:0 bits are set. However, the OC0x 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 OC0x pin (DDR_OC0x) must be set as output before the OC0x 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 OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See “8-bit Timer/Counter Register Description” on page 89.

14.5.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0x1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next compare match. For compare output actions in the

non-PWM modes refer to Table 20 on page 90. For fast PWM mode, refer to Table 21 on page 90, and for phase correct PWM refer to Table 22 on page 90.

A change of the COM0x1:0 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 FOC0x strobe bits.

14.6 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM02:0) and Compare Output mode (COM0x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (See “Compare Match Output Unit” on page 83.).

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 88.

14.6.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM02:0 = 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 (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 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 TOV0 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.

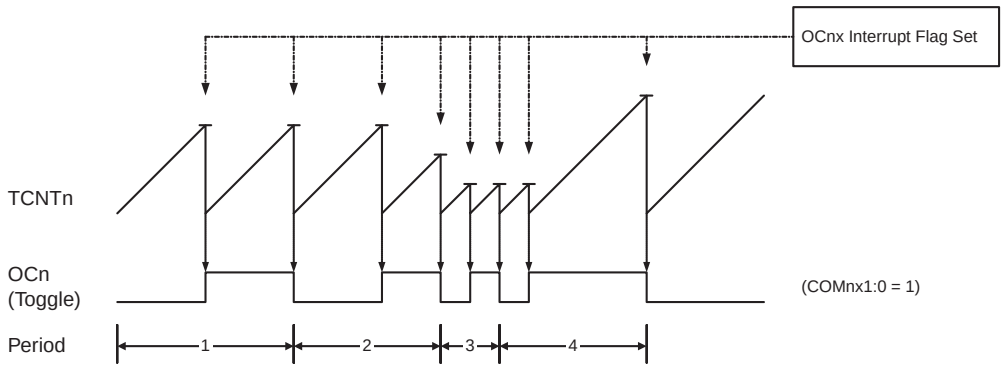
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

14.6.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM02:0 = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A 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 Figure 14-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 14-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A 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 OCR0A is lower than the current value of TCNT0, 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 OC0A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM0A1:0 = 1). The OC0A 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_{OC0} = $f_{clk_I/O}/2$ when OCR0A 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 prescale factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

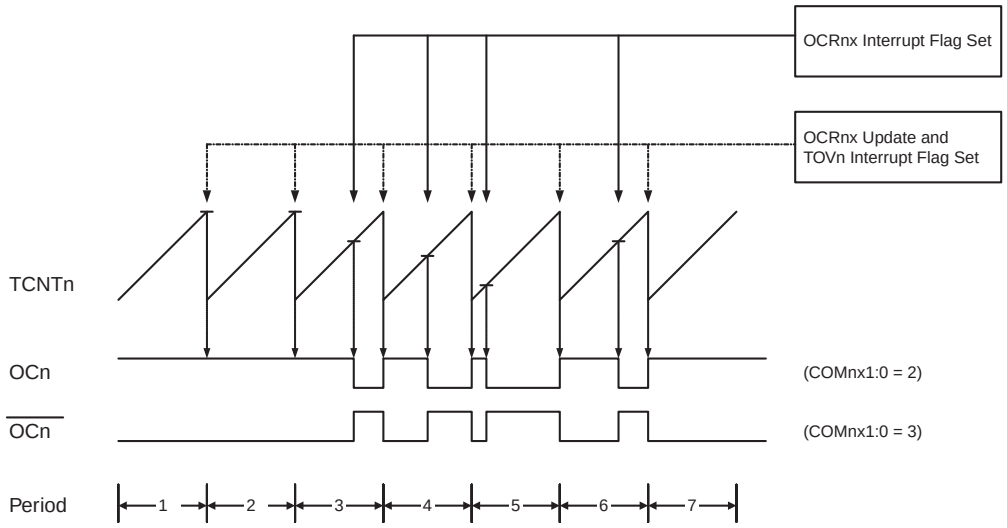
14.6.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 14-6. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and

inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 14-6. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) 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 OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A1:0 bits to one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 24 on page 91). The actual OC0x 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 OC0x Register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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 prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

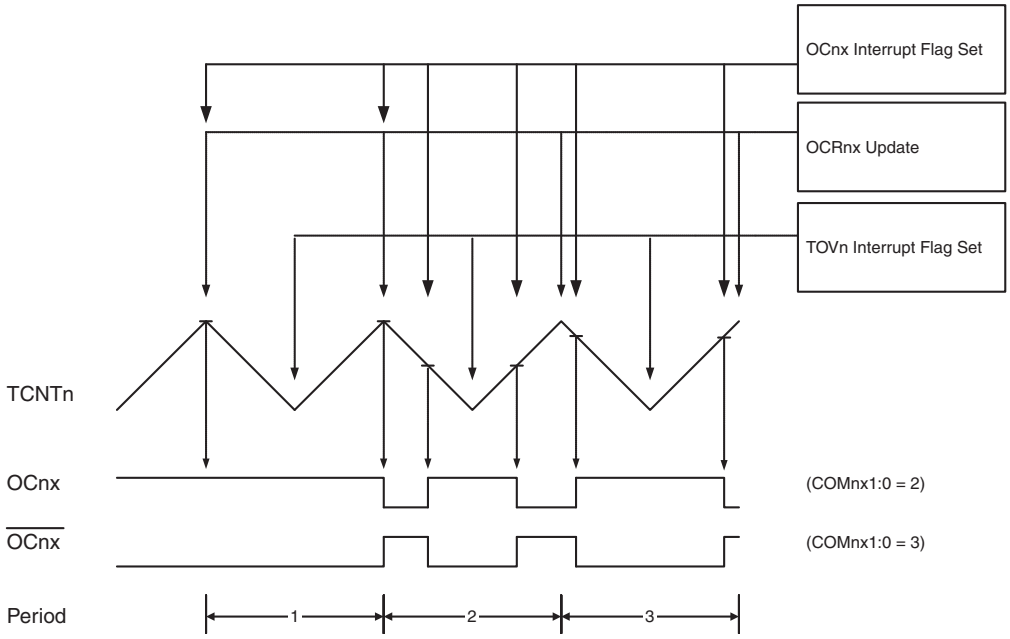
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

14.6.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM02:0 = 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 = 1, and OCR0A when WGM2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x 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 TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 14-7. The TCNT0 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 TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 14-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) 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 OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 25 on page 91). The actual OC0x 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 OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x Register at compare

match between OCR0x and TCNT0 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 prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A 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 14-7 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.

- OCRnx changes its value from MAX, like in Figure 14-7. When the OCR0A 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 OCnx 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 OCRnx, and for that reason misses the Compare Match and hence the OCnx change that would have happened on the way up.

14.7 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set. Figure 14-8 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 14-8. Timer/Counter Timing Diagram, no Prescaling

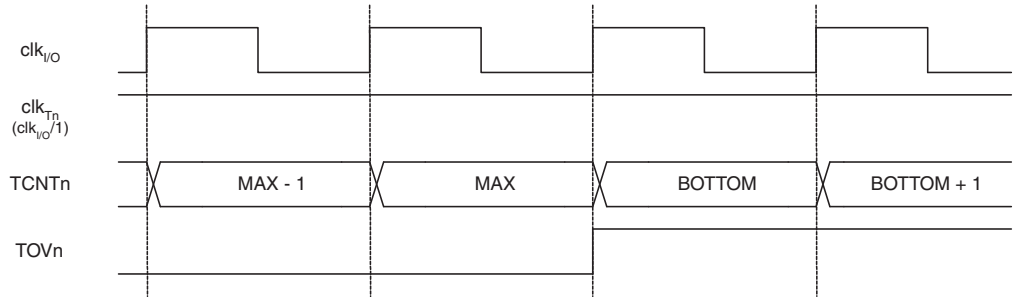


Figure 14-9 shows the same timing data, but with the prescaler enabled.

Figure 14-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

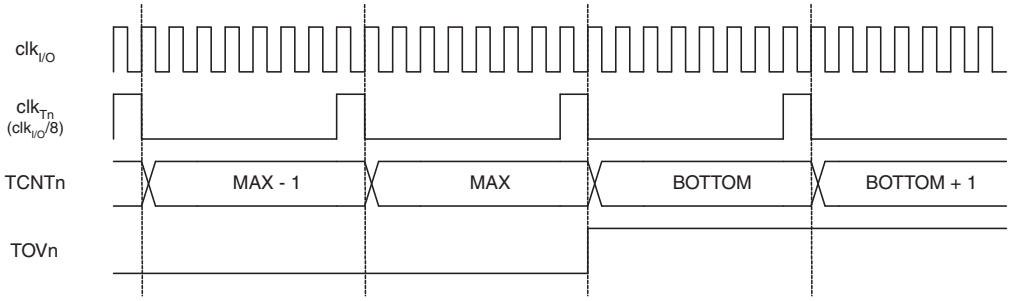


Figure 14-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 14-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk_I/O}/8$)

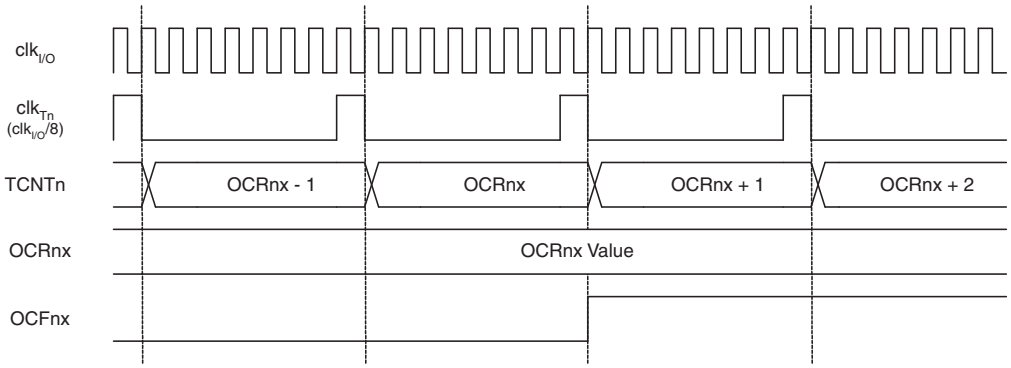
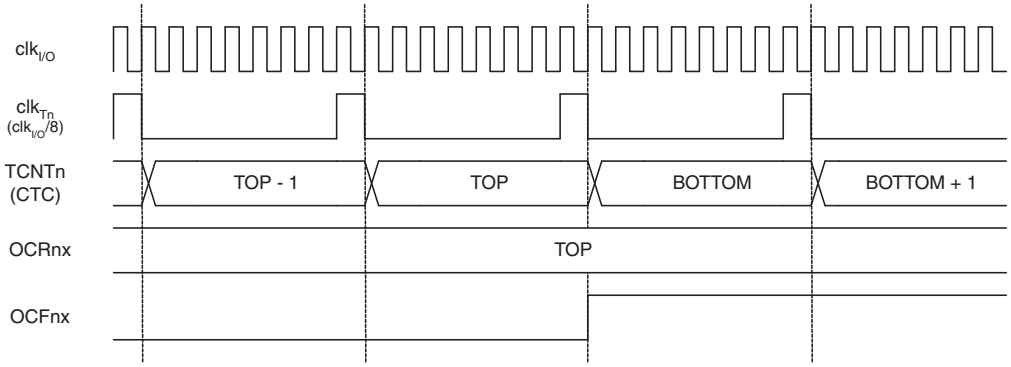


Figure 14-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 14-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk_I/O}/8$)



14.8 8-bit Timer/Counter Register Description

14.8.1 Timer/Counter Control Register A – TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A 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 OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 20 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 20. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Table 21 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 21. Compare Output Mode, Fast PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match, set OC0A at TOP
1	1	Set OC0A on Compare Match, clear OC0A at TOP

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 85 for more details.

Table 22 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 22. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 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 111 for more details.

- **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 23 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 23. 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 24 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 24. 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 “Fast PWM Mode” on page 85 for more details.

Table 25 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 25. 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 “Phase Correct PWM Mode” on page 87 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 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 26. 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 “Modes of Operation” on page 84).

Table 26. Waveform Generation Mode Bit Description

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

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

14.8.2 Timer/Counter Control Register B – TCCR0B

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A 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 TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bit 6 – FOC0B: Force Output Compare B**

The FOC0B 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 TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0B output is changed according to its COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a

strobe. Therefore it is the value present in the COM0B1:0 bits that determines the effect of the forced compare.

A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the “Timer/Counter Control Register A – TCCR0A” on page 89.

- **Bits 2:0 – CS02:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 27. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

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.

14.8.3 Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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 TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

14.8.4 Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A[7:0]								OCR0A
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 (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

14.8.5 Output Compare Register B – OCR0B

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
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 (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0B pin.

14.8.6 Timer/Counter Interrupt Mask Register – TIMSK0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter Interrupt Flag Register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

14.8.7 Timer/Counter 0 Interrupt Flag Register – TIFR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **Bit 2 – OCF0B: Timer/Counter 0 Output Compare B Match Flag**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

- **Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 26, “Waveform Generation Mode Bit Description” on page 92.

15. 16-bit Timer/Counter1 with PWM

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- **True 16-bit Design (i.e., Allows 16-bit PWM)**
- **Two independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

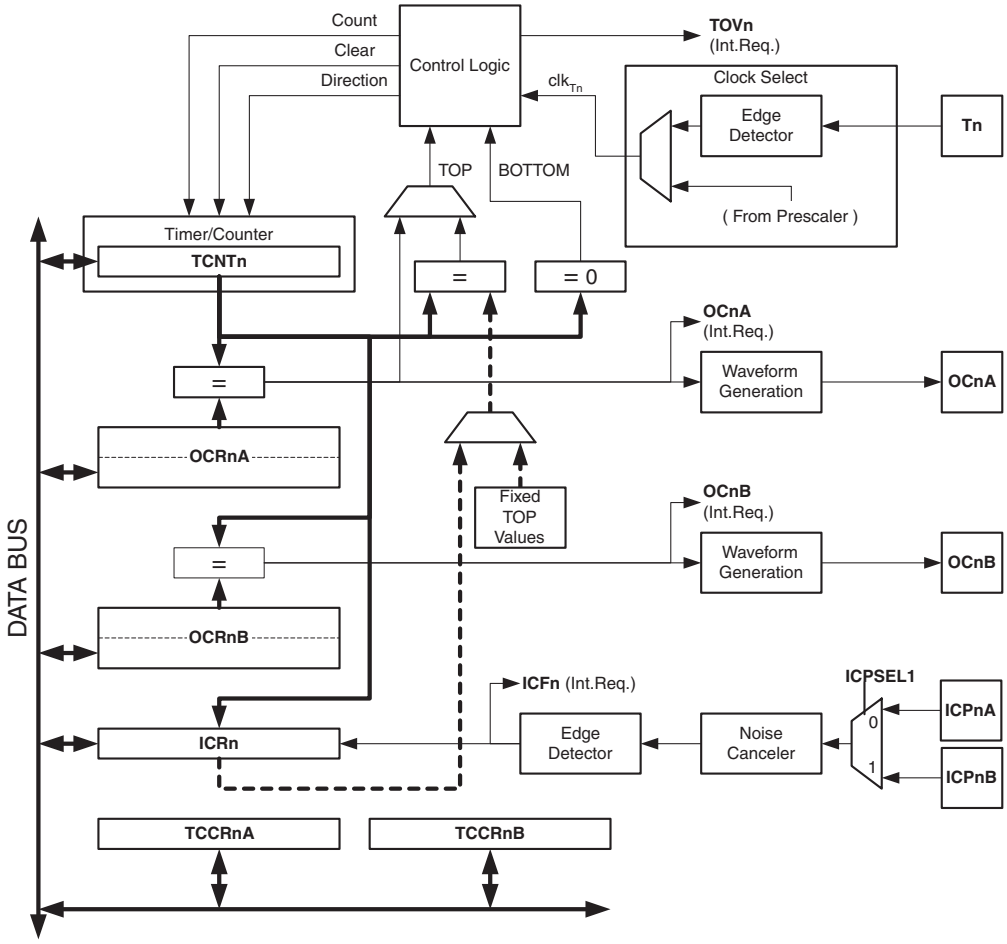
15.1 Overview

Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare unit channel. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 15-1. For the actual placement of I/O pins, refer to “Pin Descriptions” on page 4. 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 “16-bit Timer/Counter Register Description” on page 116.

The PRTIM1 bit in “Power Reduction Register” on page 39 must be written to zero to enable Timer/Counter1 module.

Figure 15-1. 16-bit Timer/Counter Block Diagram[1]



Note: 1. Refer to Table on page 4 for Timer/Counter1 pin placement and description.

15.1.1 Registers

The *Timer/Counter* (TCNTn), *Output Compare Registers* (OCRnx), and *Input Capture Register* (ICRn) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 98. The *Timer/Counter Control Registers* (TCCRnx) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the *Timer Interrupt Flag Register* (TIFRn). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSKn). TIFRn and TIMSKn are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The Clock Select logic block controls which clock source and edge 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_{Tn}).

The double buffered Output Compare Registers (OCRnx) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OCnx). See "Output Compare Units" on page 104. The compare match event will also set the Compare Match Flag (OCFnx) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICPn). The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCRnA Register, the ICRn Register, or by a set of fixed values. When using OCRnA as TOP value in a PWM mode, the OCRnA Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICRn Register can be used as an alternative, freeing the OCRnA to be used as PWM output.

15.1.2 Definitions

The following definitions are used extensively throughout the section:

Table 28.

BOTTOM	The counter reaches the *BOTTOM* when it becomes 0x0000.
MAX	The counter reaches its *MAX*imum when it becomes 0xFFFF (decimal 65535).
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 one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCRnA or ICRn Register. The assignment is dependent of the mode of operation.

15.2 Accessing 16-bit Registers

The TCNTn, OCRnx, and ICRn are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCRnx 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCRnx and ICRn Registers. Note that when using “C”, the compiler handles the 16-bit access.

TABLE 3.

Assembly Code Examples[1]

```
...
; Set TCNTn to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNTnH,r17
out TCNTnL,r16
; Read TCNTn into r17:r16
in  r16,TCNTnL
in  r17,TCNTnH
...
```

C Code Examples[1]

```
unsigned int i;
...
/* Set TCNTn to 0x01FF */
TCNTn = 0x1FF;
/* Read TCNTn into i */
i = TCNTn;
...
```

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

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

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

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

TABLE 4.

Assembly Code Example[1]

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

C Code Example[1]

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

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

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

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

TABLE 5.

Assembly Code Example[1]

```
TIM16_WriteTCNTn:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNTn to r17:r16
  out TCNTnH,r17
  out TCNTnL,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example[1]

```
void TIM16_WriteTCNTn( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNTn to i */
  TCNTn = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

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

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNTn.

15.2.1 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

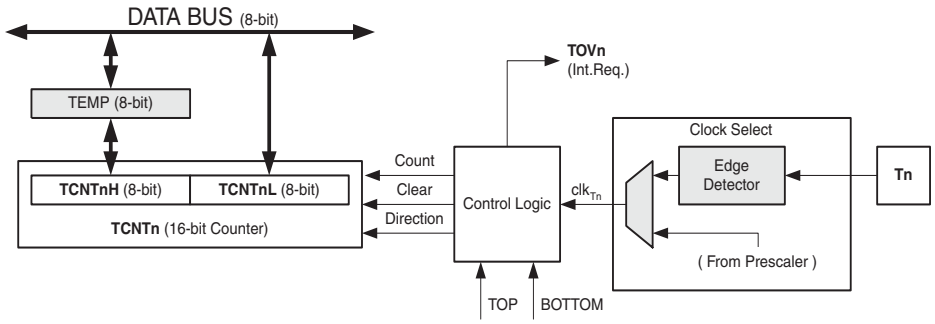
15.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CSn2:0) bits located in the *Timer/Counter control Register B* (TCCRnB). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 76.

15.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 15-2 shows a block diagram of the counter and its surroundings.

Figure 15-2. Counter Unit Block Diagram



Signal description (internal signals):

Count Increment or decrement TCNTn by 1.

Direction Select between increment and decrement.

Clear Clear TCNTn (set all bits to zero).

clk_{Tn} Timer/Counter clock.

TOP Signalize that TCNTn has reached maximum value.

BOTTOM Signalize that TCNTn has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNTnH) containing the upper eight bits of the counter, and *Counter Low* (TCNTnL) containing the lower eight bits. The TCNTnH Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNTnH I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNTnH value when the TCNTnL is read, and TCNTnH is updated with the temporary register value when TCNTnL is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNTn Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* (clk_{Tn}). The clk_{Tn} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CSn2:0). When no clock source is selected (CSn2:0 = 0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, independent of whether clk_{Tn} 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 *Waveform Generation mode* bits (WGMn3:0) located in the *Timer/Counter Control Registers* A and B (TCCRnA and TCCRnB). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OCnx. For more details about advanced counting sequences and waveform generation, see “16-bit Timer/Counter1 with PWM” on page 96.

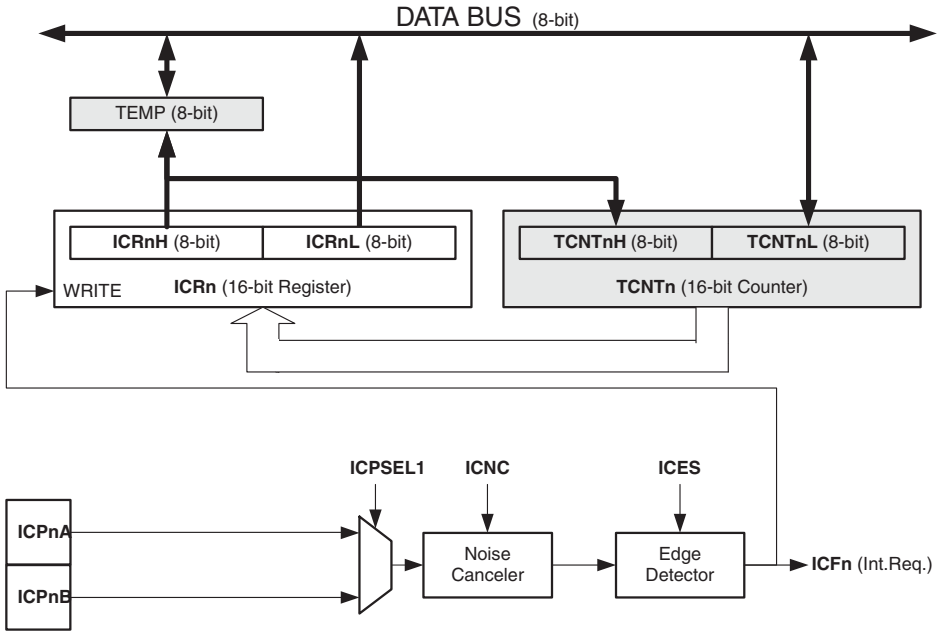
The Timer/Counter Overflow Flag (TOVn) is set according to the mode of operation selected by the WGMn3:0 bits. TOVn can be used for generating a CPU interrupt.

15.5 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICPn pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 15-3. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

Figure 15-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICPn), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNTn) is written to the *Input Capture Register* (ICRn). The *Input Capture Flag* (ICFn) is set at the same system clock as the TCNTn value is copied into ICRn Register. If enabled (ICIEn = 1), the Input Capture Flag generates an Input Capture interrupt. The ICFn Flag is automatically cleared when the interrupt is executed. Alternatively the ICFn Flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the *Input Capture Register* (ICRn) is done by first reading the low byte (ICRnL) and then the high byte (ICRnH). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICRnH I/O location it will access the TEMP Register.

The ICRn Register can only be written when using a Waveform Generation mode that utilizes the ICRn Register for defining the counter’s TOP value. In these cases the *Waveform Generation mode* (WGMn3:0) bits must be set before the TOP value can be written to the ICRn Register. When writing the ICRn Register the high byte must be written to the ICRnH I/O location before the low byte is written to ICRnL.

For more information on how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 98.

15.5.1 Input Capture Trigger Source

The trigger sources for the Input Capture unit arethe *Input Capture pin* (ICP1A & ICP1B).

Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

The *Input Capture pin* (ICPn) IS sampled using the same technique as for the Tn pin (Figure 13-1 on page 76). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICRn to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICPn pin.

15.5.2 Noise Canceler

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

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

15.5.3 Using the Input Capture Unit

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

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

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

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

15.6 Output Compare Units

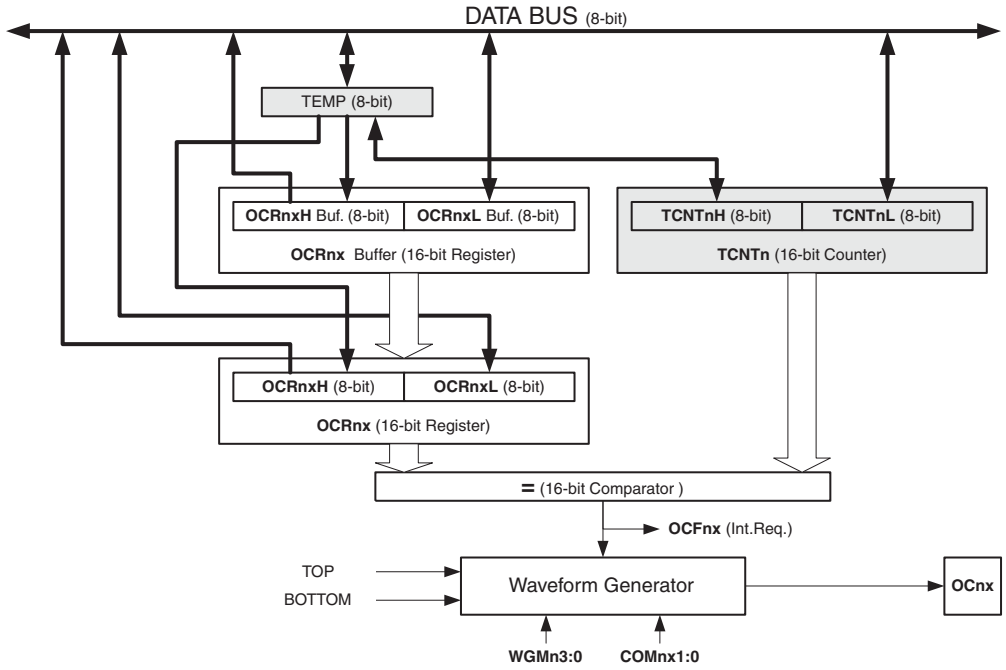
The 16-bit comparator continuously compares TCNTn with the *Output Compare Register* (OCRnx). If TCNT equals OCRnx the comparator signals a match. A match will set the *Output Compare Flag* (OCFnx) at the next “timer clock cycle”. If enabled (OCIEnx = 1), the Output Compare Flag generates an Output Compare interrupt. The OCFnx Flag is automatically cleared when the interrupt is executed. Alternatively the OCFnx Flag can be cleared by software by writ-

ing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGMn3:0) bits and *Compare Output mode* (COMnx1:0) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See “16-bit Timer/Counter1 with PWM” on page 96.)

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

Figure 15-4 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = n for Timer/Counter n), and the “x” indicates Output Compare unit (x). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

Figure 15-4. Output Compare Unit, Block Diagram



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

The OCRnx Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCRnx Buffer Register, and if double buffering is disabled the CPU will access the OCRnx directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCRnx Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCRnxH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be

updated by the value written. Then when the low byte (OCRnxL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCRnx buffer or OCRnx Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 98.

15.6.1 Force Output Compare

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

15.6.2 Compare Match Blocking by TCNTn Write

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

15.6.3 Using the Output Compare Unit

Since writing TCNTn in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNTn when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNTn equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is downcounting.

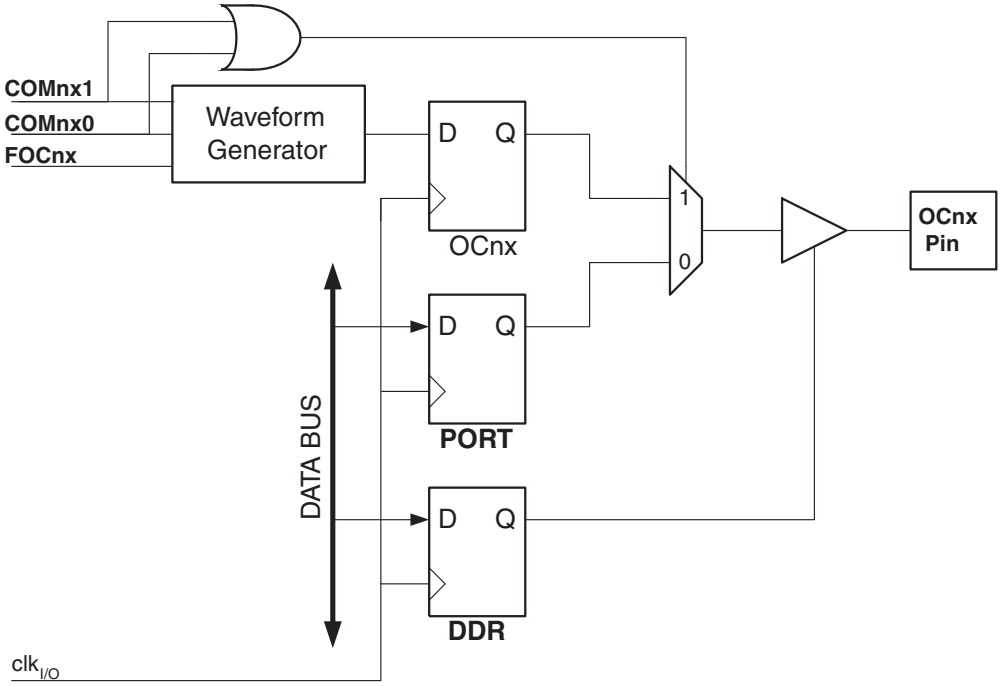
The setup of the OCnx should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCnx value is to use the Force Output Compare (FOCnx) strobe bits in Normal mode. The OCnx Register keeps its value even when changing between Waveform Generation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

Be aware that the COMnx1:0 bits are not double buffered together with the compare value. Changing the COMnx1:0 bits will take effect immediately.

15.7 Compare Match Output Unit

The *Compare Output mode* (COMnx1:0) bits have two functions. The Waveform Generator uses the COMnx1:0 bits for defining the Output Compare (OCnx) state at the next compare match. Secondly the COMnx1:0 bits control the OCnx pin output source. Figure 15-5 shows a simplified schematic of the logic affected by the COMnx1:0 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 COMnx1:0 bits are shown. When referring to the OCnx state, the reference is for the internal OCnx Register, not the OCnx pin. If a system reset occur, the OCnx Register is reset to “0”.

Figure 15-5. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OCnx) from the Waveform Generator if either of the COMnx1:0 bits are set. However, the OCnx 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 OCnx pin (DDR_OCnx) must be set as output before the OCnx value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 29, Table 30 and Table 31 for details.

The design of the Output Compare pin logic allows initialization of the OCnx state before the output is enabled. Note that some COMnx1:0 bit settings are reserved for certain modes of operation. See “16-bit Timer/Counter Register Description” on page 116.

The COMnx1:0 bits have no effect on the Input Capture unit.

15.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COMnx1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COMnx1:0 = 0 tells the Waveform Generator that no action on the OCnx Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 29 on page 117. For fast PWM mode refer to Table 30 on page 117, and for phase correct and phase and frequency correct PWM refer to Table 31 on page 117.

A change of the COMnx1:0 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 FOCnx strobe bits.

15.8 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGMn3:0) and *Compare Output mode* (COMnx1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COMnx1:0 bits control whether the PWM out-

put generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COMnx1:0 bits control whether the output should be set, cleared or toggle at a compare match (See “Compare Match Output Unit” on page 106.)

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 115.

15.8.1 Normal Mode

The simplest mode of operation is the *Normal mode* (WGMn3:0 = 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 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOVn) will be set in the same timer clock cycle as the TCNTn becomes zero. The TOVn Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn 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 Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

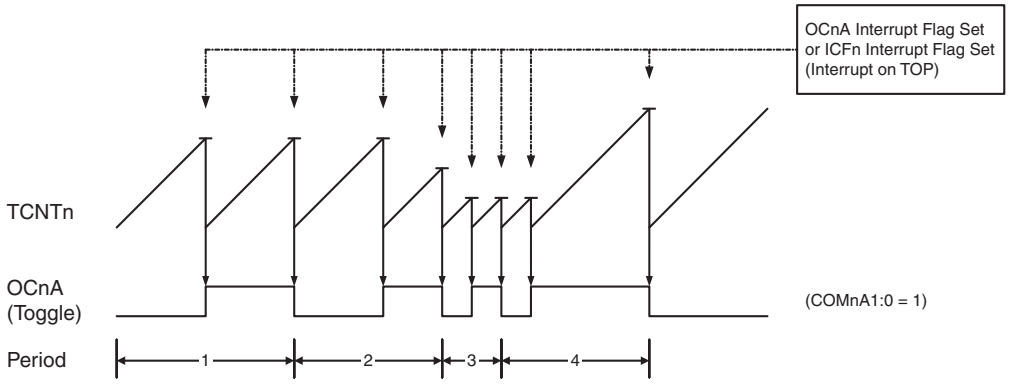
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

15.8.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGMn3:0 = 4 or 12), the OCRnA or ICRn Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches either the OCRnA (WGMn3:0 = 4) or the ICRn (WGMn3:0 = 12). The OCRnA or ICRn define 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 Figure 15-6. The counter value (TCNTn) increases until a compare match occurs with either OCRnA or ICRn, and then counter (TCNTn) is cleared.

Figure 15-6. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCFnA or ICFn Flag according to the register used to define the TOP value. If the

interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the 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 OCRnA or ICRn is lower than the current value of TCNTn, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCRnA for defining TOP (WGMn3:0 = 15) since the OCRnA then will be double buffered.

For generating a waveform output in CTC mode, the OCnA output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COMnA1:0 = 1). The OCnA value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OCnA = 1). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk_I/O}/2$ when OCRnA is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

As for the Normal mode of operation, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

15.8.3 Fast PWM Mode

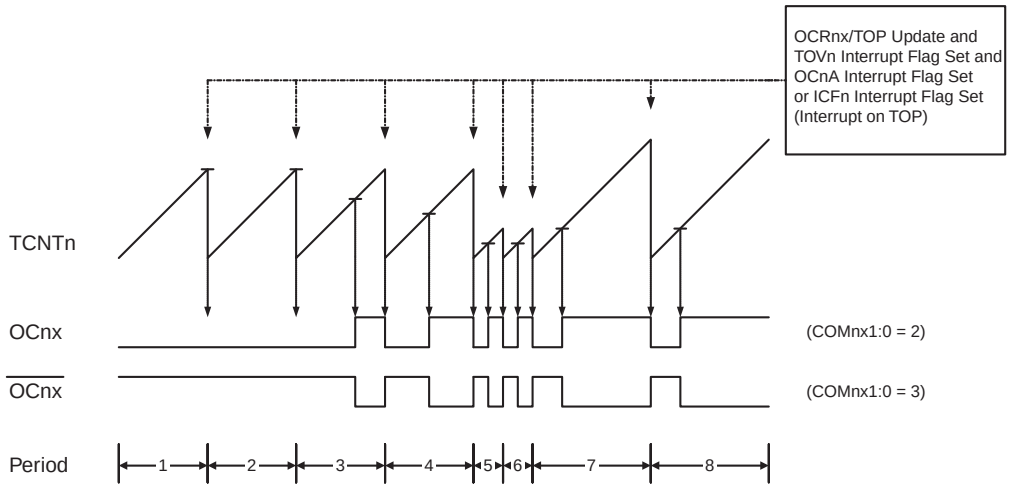
The *fast Pulse Width Modulation* or fast PWM mode (WGMn3:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is set on the compare match between TCNTn and OCRnx, and cleared at TOP. In inverting Compare Output mode output is cleared on compare match and set at TOP. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICRn or OCRnA. The minimum resolution allowed is 2-bit (ICRn or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICRn or OCRnA set to MAX). The PWM resolution in bits can be calculated by using the following equation:

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

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGMn3:0 = 5, 6, or 7), the value in ICRn (WGMn3:0 = 14), or the value in OCRnA (WGMn3:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 15-7. The figure shows fast PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 15-7. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches TOP. In addition the OCnA or ICFn Flag is set at the same timer clock cycle as TOVn is set when either OCRnA or ICRn is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

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 TCNTn and the OCRnx. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCRnx Registers are written.

The procedure for updating ICRn differs from updating OCRnA when used for defining the TOP value. The ICRn Register is not double buffered. This means that if ICRn is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICRn value written is lower than the current value of TCNTn. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCRnA Register however, is double buffered. This feature allows the OCRnA I/O location to be written anytime. When the OCRnA I/O location is written the value written will be put into the OCRnA Buffer Register. The OCRnA Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNTn matches TOP. The update is done at the same timer clock cycle as the TCNTn is cleared and the TOVn Flag is set.

Using the ICRn Register for defining TOP works well when using fixed TOP values. By using ICRn, the OCRnA Register is free to be used for generating a PWM output on OCnA. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCRnA as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (see Table on page 117). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn, and clearing (or setting) the OCnx Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot (1 + TOP)}$$

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

The extreme values for the OCRnx Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCRnx is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCRnx equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COMnx1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OCnA to toggle its logical level on each compare match (COMnA1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk_I/O}/2$ when OCRnA is set to zero (0x0000). This feature is similar to the OCnA toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

15.8.4 Phase Correct PWM Mode

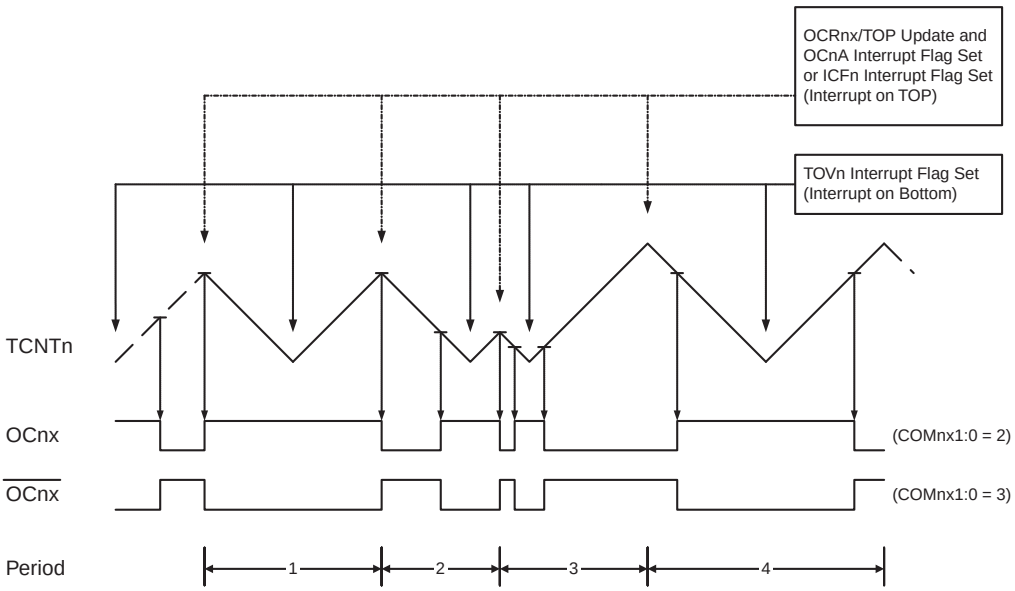
The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGMn3:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is cleared on the compare match between TCNTn and OCRnx 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.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICRn or OCRnA. The minimum resolution allowed is 2-bit (ICRn or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICRn or OCRnA set to MAX). The PWM resolution in bits can be calculated by using the following equation:

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

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGMn3:0 = 1, 2, or 3), the value in ICRn (WGMn3:0 = 10), or the value in OCRnA (WGMn3:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNTn value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 15-8. The figure shows phase correct PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn 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 TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 15-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOVn) is set each time the counter reaches BOTTOM. When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag is set accordingly at the same timer clock cycle as the OCRnx 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 TCNTn and the OCRnx. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCRnx Registers are written. As the third period shown in Figure 15-8 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 OCRnx Register. Since the OCRnx 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 OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (See Table on page 117). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn 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 OCRnx Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx 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.

15.8.5 Phase and Frequency Correct PWM Mode

The *phase and frequency correct Pulse Width Modulation,* or phase and frequency correct PWM mode (WGMn3:0 = 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 (OCnx) is cleared on the compare match between TCNTn and OCRnx while upcounting, and set on the compare match while downcounting. 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, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

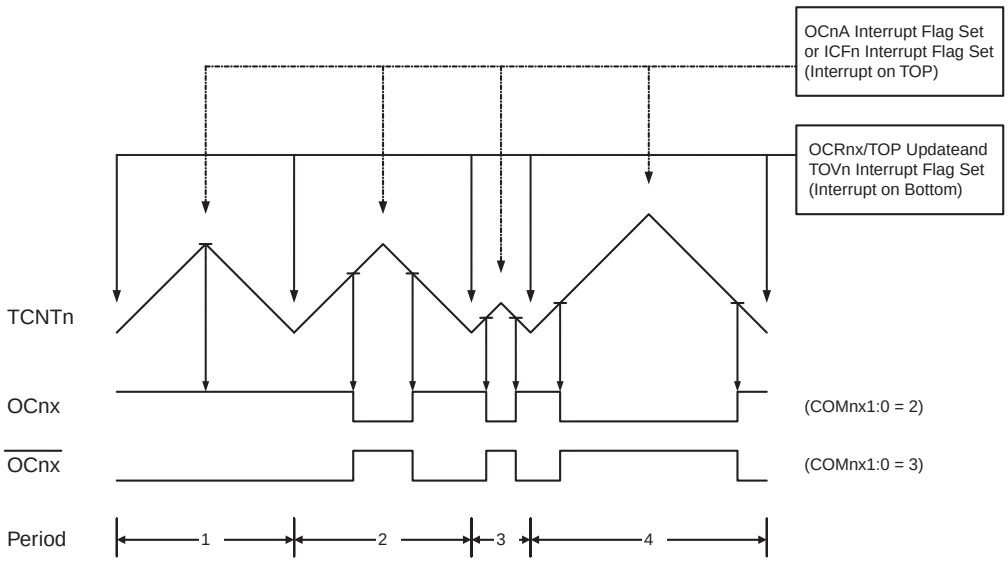
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCRnx Register is updated by the OCRnx Buffer Register, (see Figure 15-8 and Figure 15-9).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICRn or OCRnA. The minimum resolution allowed is 2-bit (ICRn or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICRn or OCRnA set to MAX). The PWM resolution in bits can be calculated using 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 ICRn (WGMn3:0 = 8), or the value in OCRnA (WGMn3:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNTn 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 on Figure 15-9. The figure shows phase and frequency correct PWM mode when OCRnA or ICRn is used to define TOP. The TCNTn 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 TCNTn slopes represent compare matches between OCRnx and TCNTn. The OCnx Interrupt Flag will be set when a compare match occurs.

Figure 15-9. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOVn) is set at the same timer clock cycle as the OCRnx Registers are updated with the double buffer value (at BOTTOM). When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn Flag set when TCNTn has reached TOP. 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 TCNTn and the OCRnx.

As Figure 15-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCRnx 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.

Using the ICRn Register for defining TOP works well when using fixed TOP values. By using ICRn, the OCRnA Register is free to be used for generating a PWM output on OCnA. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCRnA as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (See Table on page 117). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx Register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx Register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by 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 OCRnx Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx 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.

15.9 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 OCRnx Register is updated with the OCRnx buffer value (only for modes utilizing double buffering). Figure 15-10 shows a timing diagram for the setting of OCFnx.

Figure 15-10. Timer/Counter Timing Diagram, Setting of OCFnx, no Prescaling

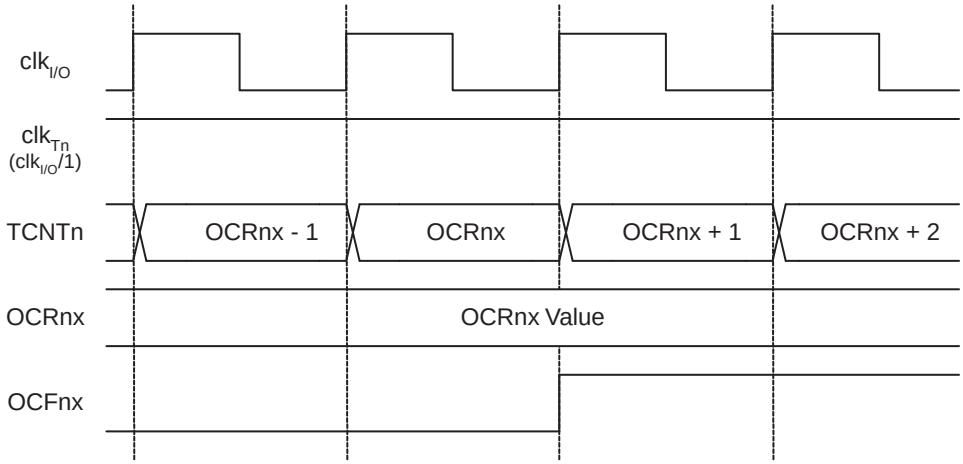


Figure 15-11 shows the same timing data, but with the prescaler enabled.

Figure 15-11. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk_I/O}/8$)

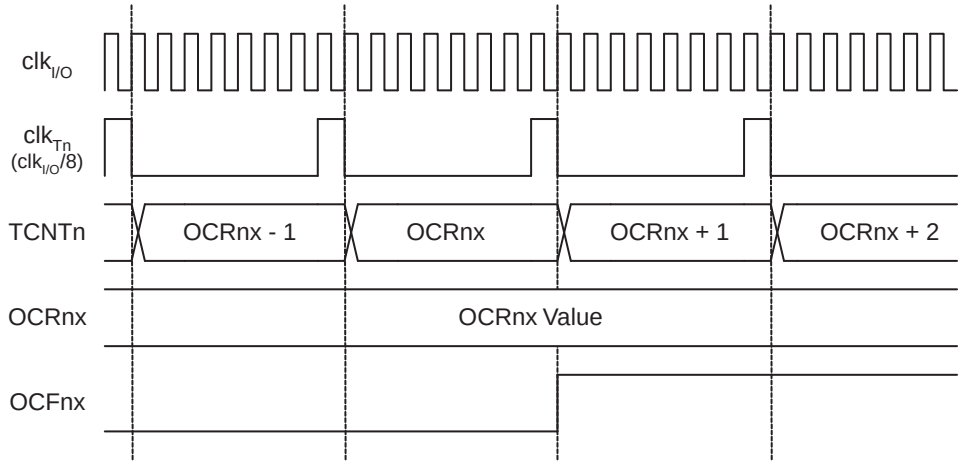


Figure 15-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCRnx Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOVn Flag at BOTTOM.

Figure 15-12. Timer/Counter Timing Diagram, no Prescaling

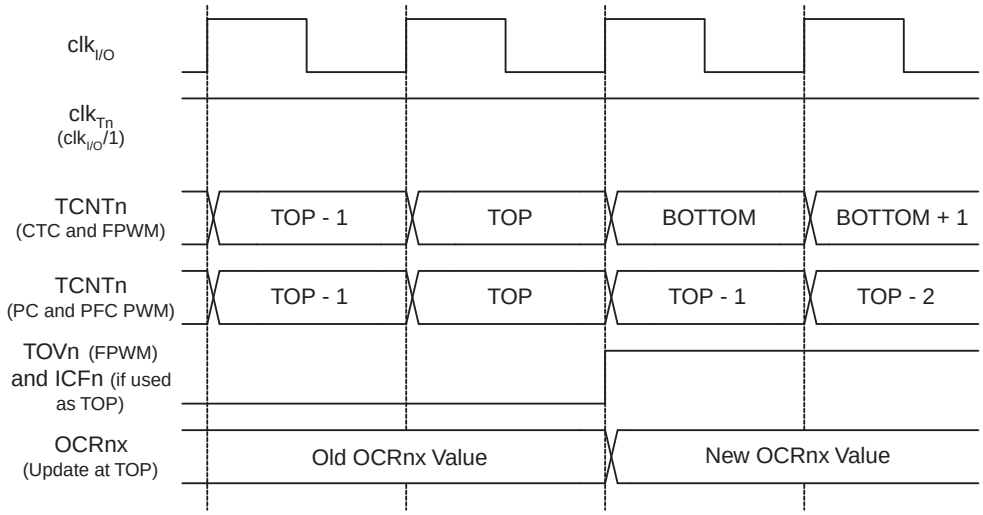
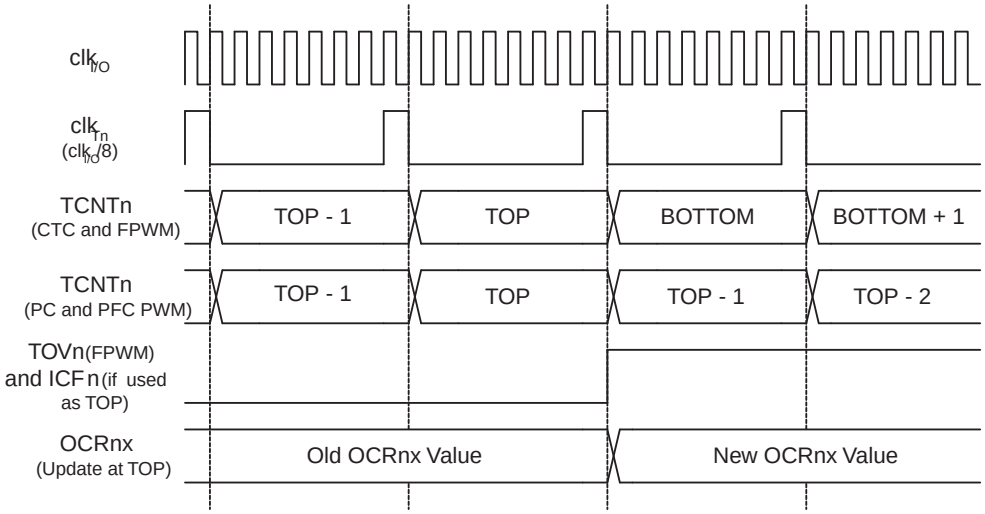


Figure 15-13 shows the same timing data, but with the prescaler enabled.

Figure 15-13. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)



15.10 16-bit Timer/Counter Register Description

15.10.1 Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COMnA1:0: Compare Output Mode for Channel A**
- **Bit 5:4 – COMnB1:0: Compare Output Mode for Channel B**

The COMnA1:0 and COMnB1:0 control the Output Compare pins (OCnA and OCnB respectively) behavior. If one or both of the COMnA1:0 bits are written to one, the OCnA output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COMnB1:0 bit are written to one, the OCnB 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 OCnA or OCnB pin must be set in order to enable the output driver.

When the OCnA or OCnB is connected to the pin, the function of the COMnx1:0 bits is dependent of the WGMn3:0 bits setting. Table 29 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to a Normal or a CTC mode (non-PWM).

Table 29. Compare Output Mode, non-PWM

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	Toggle OCnA/OCnB on Compare Match.
1	0	Clear OCnA/OCnB on Compare Match (Set output to low level).
1	1	Set OCnA/OCnB on Compare Match (Set output to high level).

Table 30 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the fast PWM mode.

Table 30. Compare Output Mode, Fast PWM(1)

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	WGMn3:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OCnA/OCnB on Compare Match, set OCnA/OCnB at TOP
1	1	Set OCnA/OCnB on Compare Match, clear OCnA/OCnB at TOP

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 109. for more details.

Table 31 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 31. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM(1)

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

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. See “Phase Correct PWM Mode” on page 111. for more details.

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

Combined with the WGMn3:2 bits found in the TCCRnB Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 32. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See “16-bit Timer/Counter1 with PWM” on page 96.).

Table 32. Waveform Generation Mode Bit Description[1]

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

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

15.10.2 Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	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 – ICNCn: Input Capture Noise Canceler**

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

- **Bit 6 – ICESn: Input Capture Edge Select**

This bit selects which edge on the Input Capture pin (ICPn) that is used to trigger a capture event. When the ICESn bit is written to zero, a falling (negative) edge is used as trigger, and when the ICESn bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICESn setting, the counter value is copied into the Input Capture Register (ICRn). The event will also set the Input Capture Flag (ICFn), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICRn is used as TOP value (see description of the WGMn3:0 bits located in the TCCRnA and the TCCRnB Register), the ICPn is disconnected and consequently the Input Capture function is disabled.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCRnB is written.

- **Bit 4:3 – WGMn3:2: Waveform Generation Mode**

See TCCRnA Register description.

- **Bit 2:0 – CSn2:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 15-10 and Figure 15-11.

Table 33. Clock Select Bit Description

CSn2	CSn1	CSn0	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 Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

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

15.10.3 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 – FOCnA: Force Output Compare for Channel A**
- **Bit 6 – FOCnB: Force Output Compare for Channel B**

The FOCnA/FOCnB bits are only active when the WGMn3:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCRnA is written when operating in a PWM mode. When writing a logical one to the FOCnA/FOCnB bit, an immediate compare match is forced on the Waveform Generation unit. The OCnA/OCnB output is changed according to its COMnx1:0 bits setting. Note that the

FOCnA/FOCnB bits are implemented as strobes. Therefore it is the value present in the COMnx1:0 bits that determine the effect of the forced compare.

A FOCnA/FOCnB strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCRnA as TOP.

The FOCnA/FOCnB bits are always read as zero.

15.10.4 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 (TCNTnH and TCNTnL, combined TCNTn) 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 “Accessing 16-bit Registers” on page 98.

Modifying the counter (TCNTn) while the counter is running introduces a risk of missing a compare match between TCNTn and one of the OCRnx Registers.

Writing to the TCNTn Register blocks (removes) the compare match on the following timer clock for all compare units.

15.10.5 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	

15.10.6 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 (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnx 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 “Accessing 16-bit Registers” on page 98.

15.10.7 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 (TCNTn) value each time an event occurs on the ICPn 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 “Accessing 16-bit Registers” on page 98.

15.10.8 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 AT90PWM1, 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 “Reset and Interrupt Vectors Placement in AT90PWM1(1)” on page 54) is executed when the ICF1 Flag, located in TIFR1, is set.

- **Bit 4, 3 – Res: Reserved Bits**

These bits are unused bits in the AT90PWM1, 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 “Reset and Interrupt Vectors Placement in AT90PWM1(1)” on page 54) 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 “Reset and Interrupt Vectors Placement in AT90PWM1(1)” on page 54) 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 “Reset and Interrupt Vectors Placement in AT90PWM1(1)” on page 54) is executed when the TOV1 Flag, located in TIFR1, is set.

15.10.9 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 AT90PWM1, 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 WGMn3:0 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 AT90PWM1, 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 WGMn3:0 bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 32 on page 118 for the TOV1 Flag behavior when using another WGMn3:0 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.

16. Power Stage Controller – (PSC0, PSC2)

The Power Stage Controller is a high performance waveform controller.

16.1 Features

- **PWM waveform generation function (2 complementary programmable outputs)**
- **Dead time control**
- **Standard mode up to 12 bit resolution**
- **Frequency Resolution Enhancement Mode (12 + 4 bits)**
- **Frequency up to 64 Mhz**
- **Conditional Waveform on External Events (Zero Crossing, Current Sensing ...)**
- **All on chip PSC synchronization**
- **ADC synchronization**
- **Overload protection function**
- **Abnormality protection function, emergency input to force all outputs to high impedance or in inactive state (fuse configurable)**
- **Center aligned and edge aligned modes synchronization**
- **Fast emergency stop by hardware**

16.2 Overview

Many register and bit references in this section are written in general form.

- A lower case “n” replaces the PSC number, in this case 0, 1 or 2. However, when using the register or bit defines in a program, the precise form must be used, i.e., PSOC1 for accessing PSC 0 Synchro and Output Configuration register and so on.
- A lower case “x” replaces the PSC part , in this case A or B. However, when using the register or bit defines in a program, the precise form must be used, i.e., PFRCnA for accessing PSC n Fault/Retrigger n A Control register and so on.

The purpose of a Power Stage Controller (PSC) is to control power modules on a board. It has two outputs on PSC0 and PSC1 and four outputs on PSC2.

These outputs can be used in various ways:

- “Two Ouputs” to drive a half bridge (lighting, DC motor ...)
- “One Output” to drive single power transistor (DC/DC converter, PFC, DC motor ...)
- “Four Outputs” in the case of PSC2 to drive a full bridge (lighting, DC motor ...)

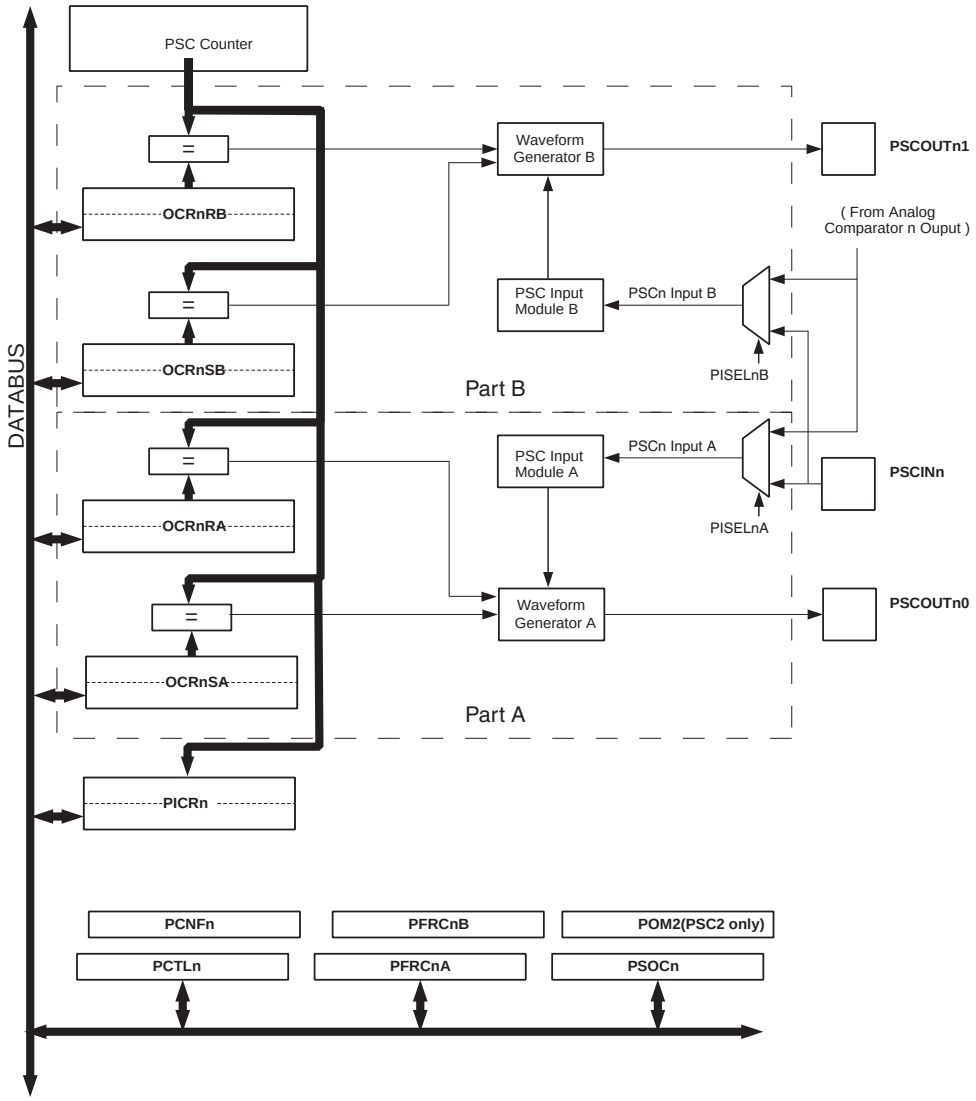
Each PSC has two inputs the purpose of which is to provide means to act directly on the generated waveforms:

- Current sensing regulation
- Zero crossing retriggering
- Demagnetization retriggering
- Fault input

The PSC can be chained and synchronized to provide a configuration to drive three half bridges. Thanks to this feature it is possible to generate a three phase waveforms for applications such as Asynchronous or BLDC motor drive.

16.3 PSC Description

Figure 16-1. Power Stage Controller 0



Note: n = 0, 1

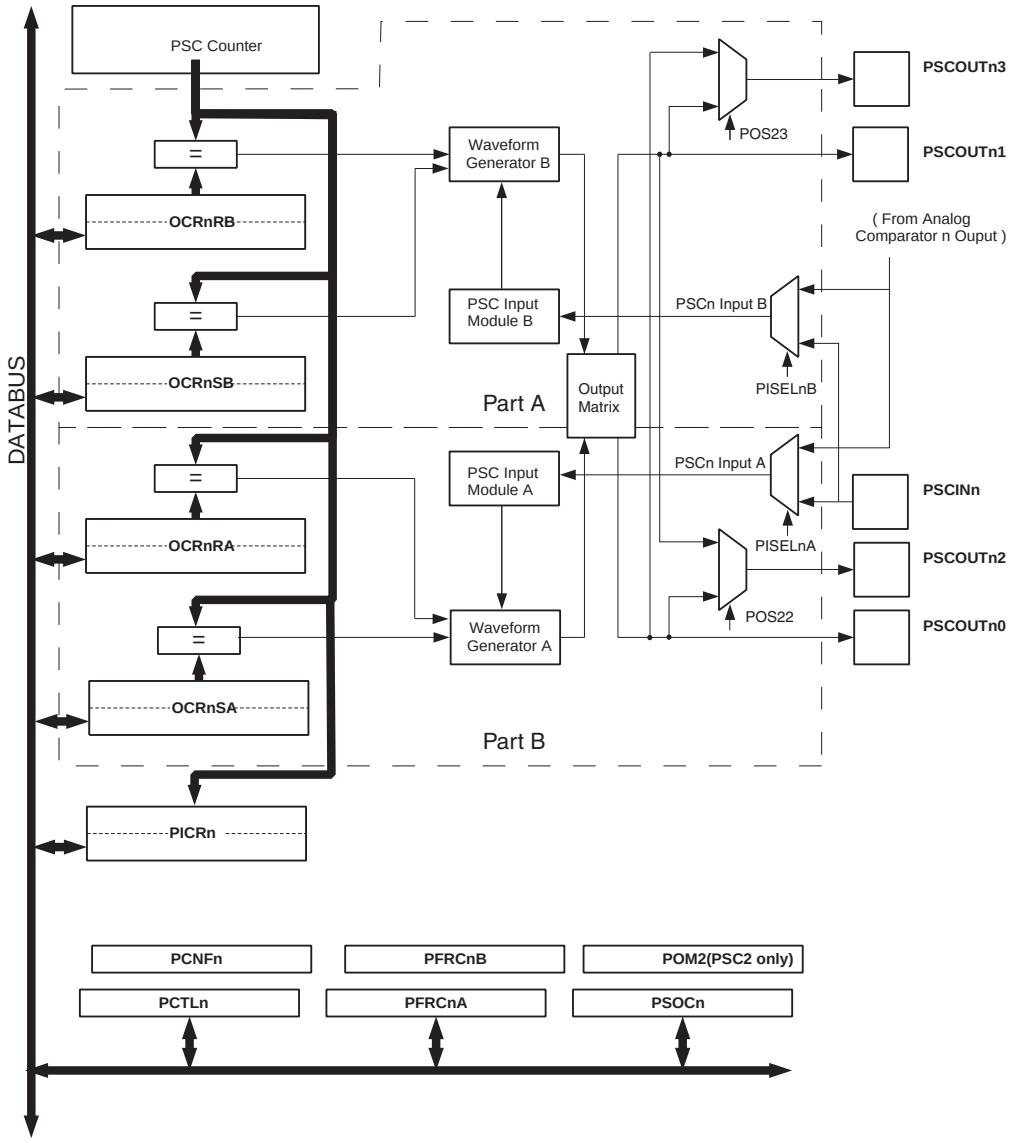
The principle of the PSC is based on the use of a counter (PSC counter). This counter is able to count up and count down from and to values stored in registers according to the selected running mode.

The PSC is seen as two symetrical entities. One part named part A which generates the output PSCOUTn0 and the second one named part B which generates the PSCOUTn1 output.

Each part A or B has its own PSC Input Module to manage selected input.

16.3.1 PSC2 Distinctive Feature

Figure 16-2. PSC2 Block Diagram



Note: n = 2

PSC2 has two supplementary outputs PSCOUT22 and PSCOUT23. Thanks to a first selector PSCOUT22 can duplicate PSCOUT20 or PSCOUT21. Thanks to a second selector PSCOUT23 can duplicate PSCOUT20 or PSCOUT21.

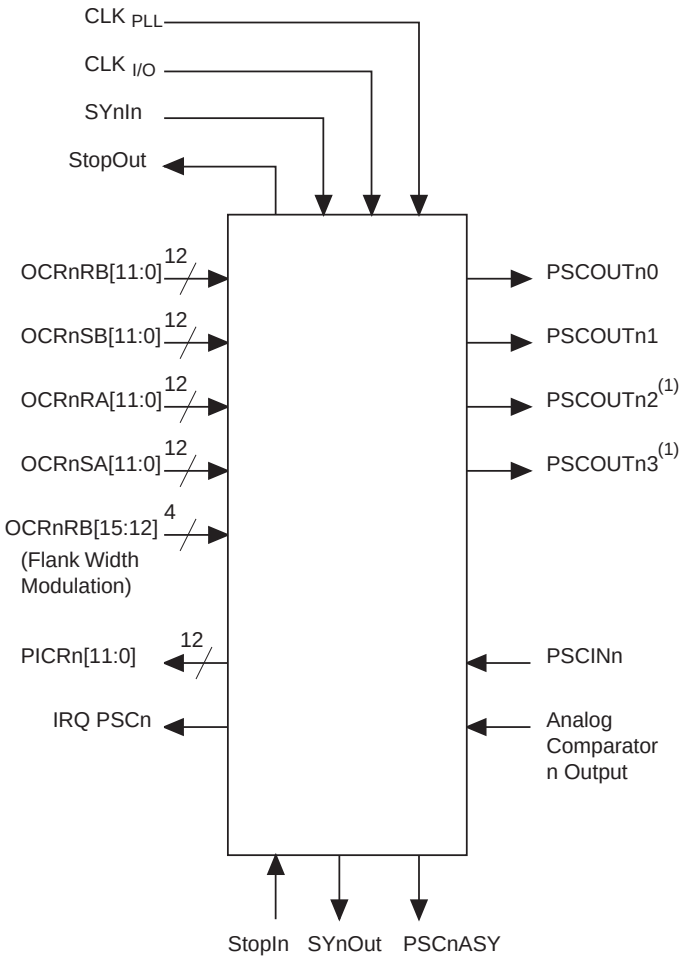
The Output Matrix is a kind of 2*2 look up table which gives the possibility to program the output values according to a PSC sequence (See “Output Matrix” on page 151.)

16.3.2 Output Polarity

The polarity “active high” or “active low” of the PSC outputs is programmable. All the timing diagrams in the following examples are given in the “active high” polarity.

16.4 Signal Description

Figure 16-3. PSC External Block View



Note: 1. available only for PSC2

2. n = 0, 1 or 2

16.4.1 Input Description

Table 34. Internal Inputs

Name	Description	Type Width
OCRnRB[11:0]	Compare Value which Reset Signal on Part B (PSCOUTn1)	Register 12 bits
OCRnSB[11:0]	Compare Value which Set Signal on Part B (PSCOUTn1)	Register 12 bits
OCRnRA[11:0]	Compare Value which Reset Signal on Part A (PSCOUTn0)	Register 12 bits
OCRnSA[11:0]	Compare Value which Set Signal on Part A (PSCOUTn0)	Register 12 bits
OCRnRB[15:12]	Frequency Resolution Enhancement value (Flank Width Modulation)	Register 4 bits
CLK I/O	Clock Input from I/O clock	Signal
CLK PLL	Clock Input from PLL	Signal
SYnIn	Synchronization In (from adjacent PSC)(1)	Signal
StopIn	Stop Input (for synchronized mode)	Signal

Note: 1. See Figure 16-38 on page 152

Table 35. Block Inputs

Name	Description	Type Width
PSCINn	Input 0 used for Retrigger or Fault functions	Signal
from A C	Input 1 used for Retrigger or Fault functions	Signal

16.4.2 Output Description

Table 36. Block Outputs

Name	Description	Type Width
PSCOUTn0	PSC n Output 0 (from part A of PSC)	Signal
PSCOUTn1	PSC n Output 1 (from part B of PSC)	Signal
PSCOUTn2 (PSC2 only)	PSC n Output 2 (from part A or part B of PSC)	Signal
PSCOUTn3(PSC2 only)	PSC n Output 3 (from part A or part B of PSC)	Signal

Table 37. Internal Outputs

Name	Description	Type Width
SYnOut	Synchronization Output(1)	Signal
PICRn [11:0]	PSC n Input Capture Register Counter value at retriggering event	Register 12 bits
IRQPSCn	PSC Interrupt Request : three souces, overflow, fault, and input capture	Signal
PSCnASY	ADC Synchronization (+ Amplifier Syncho.)(2)	Signal
StopOut	Stop Output (for synchronized mode)	

Note: 1. See Figure 16-38 on page 152
2. See “Analog Synchronization” on page 152.

16.5 Functional Description

16.5.1 Waveform Cycles

The waveform generated by PSC can be described as a sequence of two waveforms.

The first waveform is relative to PSCOUTn0 output and part A of PSC. The part of this waveform is sub-cycle A in the following figure.

The second waveform is relative to PSCOUTn1 output and part B of PSC. The part of this waveform is sub-cycle B in the following figure.

The complete waveform is ended with the end of sub-cycle B. It means at the end of waveform B.

Figure 16-4. Cycle Presentation in 1, 2 & 4 Ramp Mode

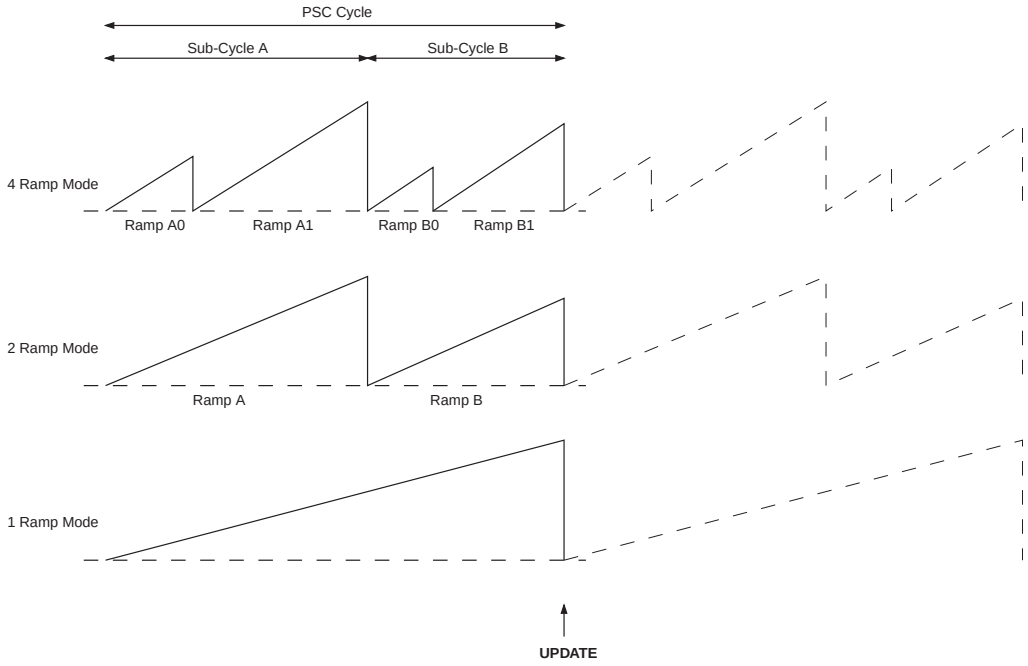
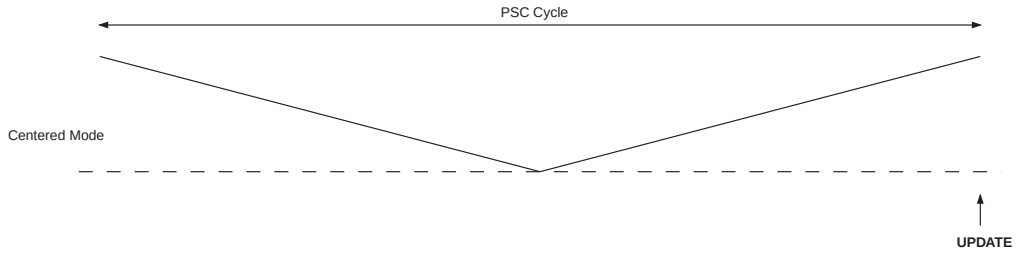


Figure 16-5. Cycle Presentation in Centered Mode



Ramps illustrate the output of the PSC counter included in the waveform generators. Centered Mode is like a one ramp mode which count down up and down.

Notice that the update of a new set of values is done regardless of ramp Mode at the top of the last ramp.

16.5.2 Running Mode Description

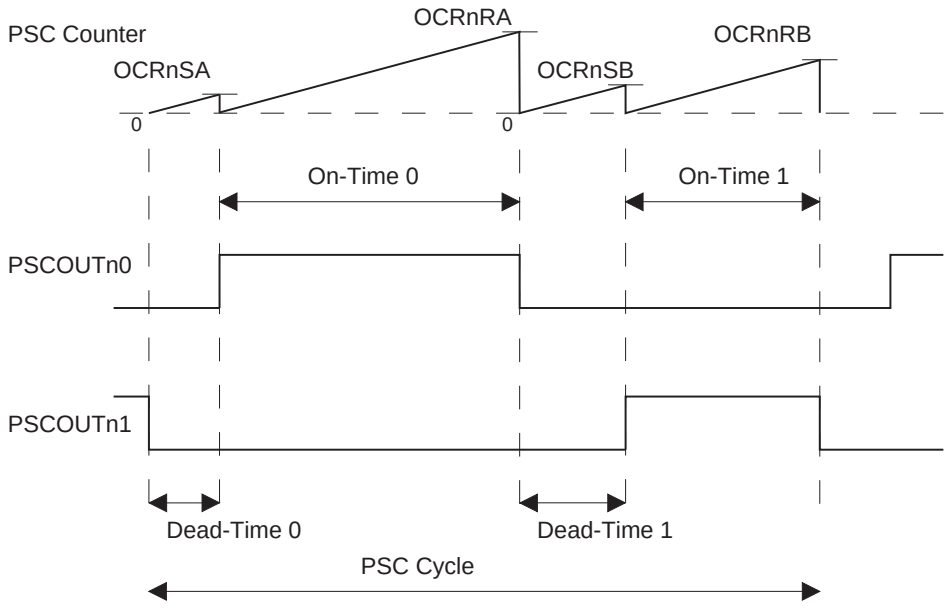
Waveforms and length of output signals are determined by Time Parameters (DT0, OT0, DT1, OT1) and by the running mode. Four modes are possible :

- Four Ramp mode
- Two Ramp mode
- One Ramp mode
- Center Aligned mode

16.5.2.1 Four Ramp Mode

In Four Ramp mode, each time in a cycle has its own definition

Figure 16-6. PSCn0 & PSCn1 Basic Waveforms in Four Ramp mode



The input clock of PSC is given by CLKPSC.

PSCOUTn0 and PSCOUTn1 signals are defined by On-Time 0, Dead-Time 0, On-Time 1 and Dead-Time 1 values with :

On-Time 0 = OCRnRAH/L * 1/Fclkpsc

On-Time 1 = OCRnRBH/L * 1/Fclkpsc

Dead-Time 0 = (OCRnSAH/L + 2) * 1/Fclkpsc

Dead-Time 1 = (OCRnSBH/L + 2) * 1/Fclkpsc

Note: Minimal value for Dead-Time 0 and Dead-Time 1 = 2 * 1/Fclkpsc

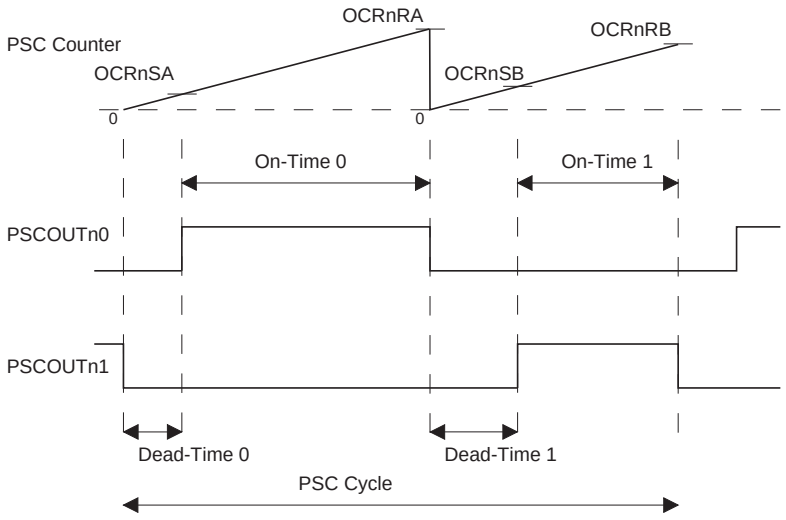
16.5.2.2 *Two Ramp Mode*

In Two Ramp mode, the whole cycle is divided in two moments

One moment for PSCn0 description with OT0 which gives the time of the whole moment

One moment for PSCn1 description with OT1 which gives the time of the whole moment

Figure 16-7. PSCn0 & PSCn1 Basic Waveforms in Two Ramp mode



PSCOUTn0 and PSCOUTn1 signals are defined by On-Time 0, Dead-Time 0, On-Time 1 and Dead-Time 1 values with :

On-Time 0 = (OCRnRAH/L - OCRnSAH/L) * 1/Fclkpsc

On-Time 1 = (OCRnRBH/L - OCRnSBH/L) * 1/Fclkpsc

Dead-Time 0 = (OCRnSAH/L + 1) * 1/Fclkpsc

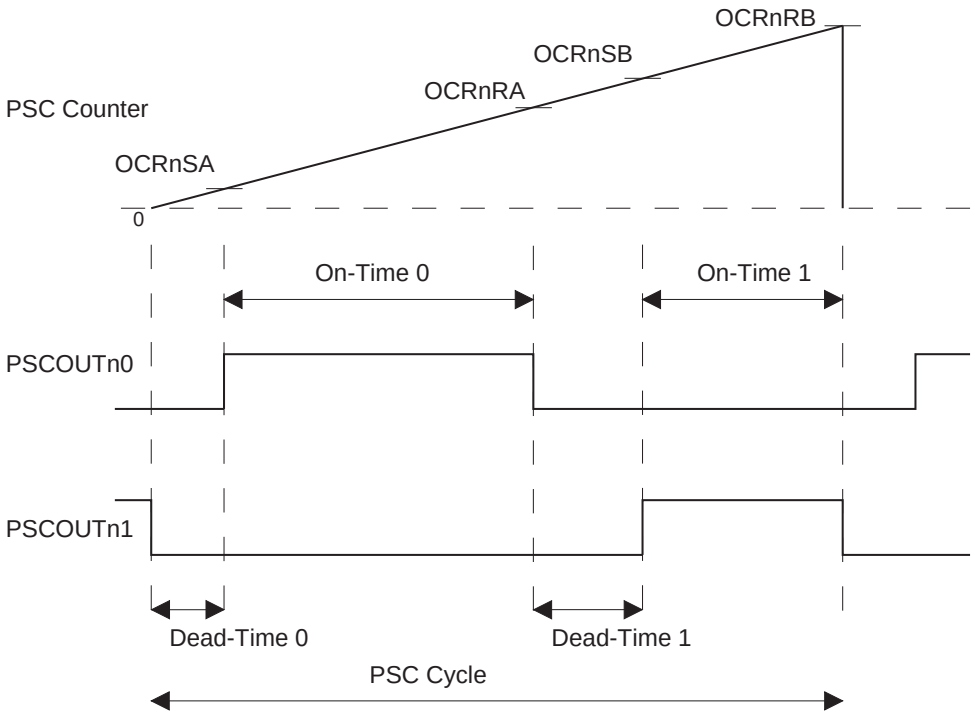
Dead-Time 1 = (OCRnSBH/L + 1) * 1/Fclkpsc

Note: Minimal value for Dead-Time 0 and Dead-Time 1 = 1/Fclkpsc

16.5.2.3 *One Ramp Mode*

In One Ramp mode, PSCOUTn0 and PSCOUTn1 outputs can overlap each other.

Figure 16-8. PSCn0 & PSCn1 Basic Waveforms in One Ramp mode



On-Time 0 = (OCRnRAH/L - OCRnSAH/L) * 1/Fclkpsc

On-Time 1 = (OCRnRBH/L - OCRnSBH/L) * 1/Fclkpsc

Dead-Time 0 = (OCRnSAH/L + 1) * 1/Fclkpsc

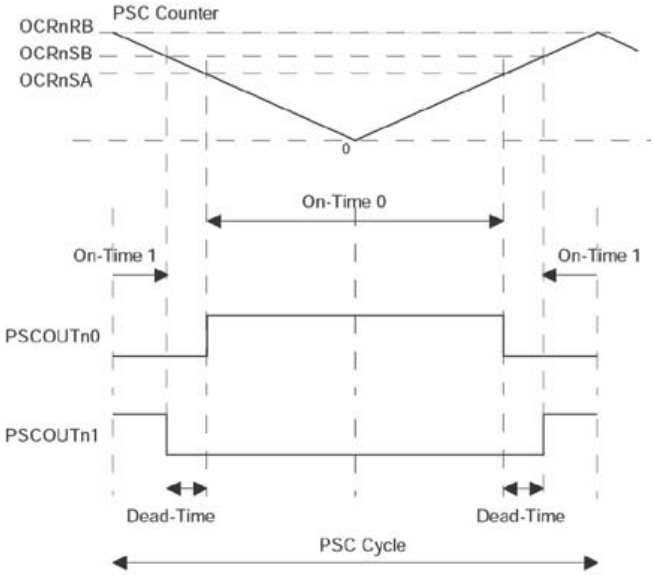
Dead-Time 1 = (OCRnSBH/L - OCRnRAH/L) * 1/Fclkpsc

Note: Minimal value for Dead-Time 0 = 1/Fclkpsc

16.5.2.4 Center Aligned Mode

In center aligned mode, the center of PSCn00 and PSCn01 signals are centered.

Figure 16-9. PSCn0 & PSCn1 Basic Waveforms in Center Aligned Mode



On-Time 0 = 2 * OCRnSAH/L * 1/Fclkpsc

On-Time 1 = 2 * (OCRnRBH/L - OCRnSBH/L + 1) * 1/Fclkpsc

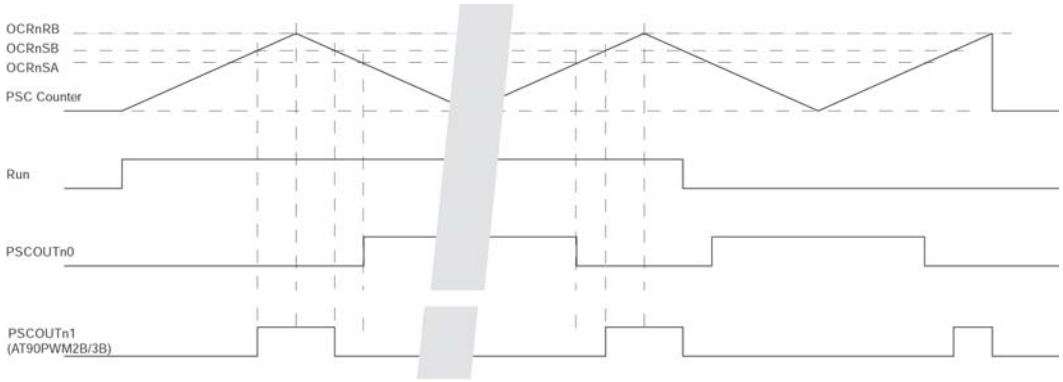
Dead-Time = (OCRnSBH/L - OCRnSAH/L) * 1/Fclkpsc

PSC Cycle = 2 * (OCRnRBH/L + 1) * 1/Fclkpsc

Note: Minimal value for PSC Cycle = 2 * 1/Fclkpsc

OCRnRAH/L is not used to control PSC Output waveform timing. Nevertheless, it can be useful to adjust ADC synchronization (See “Analog Synchronization” on page 152.).

Figure 16-10. Run and Stop Mechanism in Centered Mode



Note: See “PSC 0 Control Register – PCTL0” on page 158.(or PCTL2)

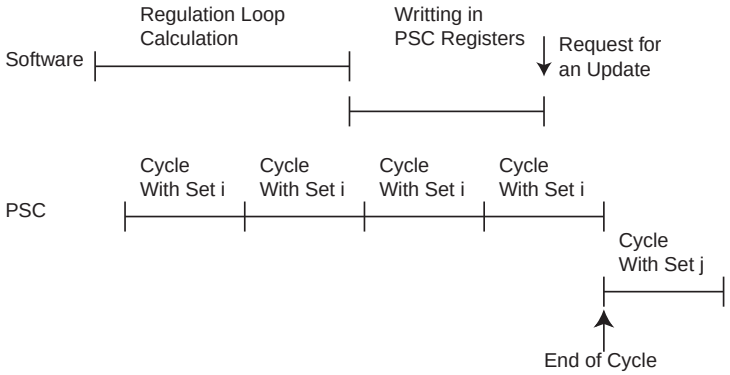
16.5.3 Fifty Percent Waveform Configuration

When PSCOUTn0 and PSCOUTn1 have the same characteristics, it's possible to configure the PSC in a Fifty Percent mode. When the PSC is in this configuration, it duplicates the OCRnSBH/L and OCRnRBH/L registers in OCRnSAH/L and OCRnRAH/L registers. So it is not necessary to program OCRnSAH/L and OCRnRAH/L registers.

16.6 Update of Values

To avoid unasynchronous and incoherent values in a cycle, if an update of one of several values is necessary, all values are updated at the same time at the end of the cycle by the PSC. The new set of values is calculated by sofware and the update is initiated by software.

Figure 16-11. Update at the end of complete PSC cycle.



The software can stop the cycle before the end to update the values and restart a new PSC cycle.

16.6.1 Value Update Synchronization

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

When AUTOLOCK configuration is selected, the update of the PSC internal registers will be done at the end of the PSC cycle if the Output Compare Register RB has been the last written. The AUTOLOCK configuration bit is taken into account at the end of the first PSC cycle.

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

The registers which update is synchronized thanks to LOCK and AUTOLOCK are PSOCn, POM2, OCRnSAH/L, OCRnRAH/L, OCRnSBH/L and OCRnRBH/L.

See these register's description starting on page 156.

When set, AUTOLOCK configuration bit prevails over LOCK configuration bit.

See "PSC 0 Configuration Register – PCNF0" on page 157.

16.7 Enhanced Resolution

Lamp Ballast applications need an enhanced resolution down to 50Hz. The method to improve the normal resolution is based on Flank Width Modulation (also called Fractional Divider). Cycles are grouped into frames of 16 cycles. Cycles are modulated by a sequence given by the

fractional divider number. The resulting output frequency is the average of the frequencies in the frame. The fractional divider (d) is given by OCRnRB[15:12].

The PSC output period is directly equal to the PSCOUTn0 On Time + Dead Time (OT0+DT0) and PSCOUTn1 On Time + DeadTime (OT1+DT1) values. These values are 12 bits numbers. The frequency adjustment can only be done in steps like the dedicated counters. The step width is defined as the frequency difference between two neighboring PSC frequencies:

$$\Delta f = |f1 - f2| = \left|\frac{f_{PLL}}{k} - \frac{f_{PLL}}{k+1}\right| = f_{PSC} \times \frac{1}{k(k+1)}$$

with k is the number of CLK_{PSC} period in a PSC cycle and is given by the following formula:

$$n = \frac{f_{PSC}}{f_{OP}}$$

with f_{OP} is the output operating frequency.

Exemple, in normal mode, with maximum operating frequency 160 kHz and f_{PLL} = 64 Mhz, k equals 400. The resulting resolution is Delta F equals 64MHz / 400 / 401 = 400 Hz.

In enhanced mode, the output frequency is the average of the frame formed by the 16 consecutive cycles.

$$f_{AVERAGE} = \frac{16-d}{16} \times f_{b1} + \frac{d}{16} \times f_{b2}$$

f_{b1} and f_{b2} are two neightboring base frequencies.

$$f_{AVERAGE} = \frac{16-d}{16} \times \frac{f_{PLL}}{n} + \frac{d}{16} \times \frac{f_{PLL}}{n+1}$$

Then the frequency resolution is divided by 16. In the example above, the resolution equals 25 Hz.

16.7.1 Frequency distribution

The frequency modulation is done by switching two frequencies in a 16 consecutive cycle frame. These two frequencies are f_{b1} and f_{b2} where f_{b1} is the nearest base frequency above the wanted frequency and f_{b2} is the nearest base frequency below the wanted frequency. The number of f_{b1} in the frame is (d-16) and the number of f_{b2} is d. The f_{b1} and f_{b2} frequencies are evenly distributed in the frame according to a predefined pattern. This pattern can be as given in the following table or by any other implementation which give an equivallent evenly distribution.

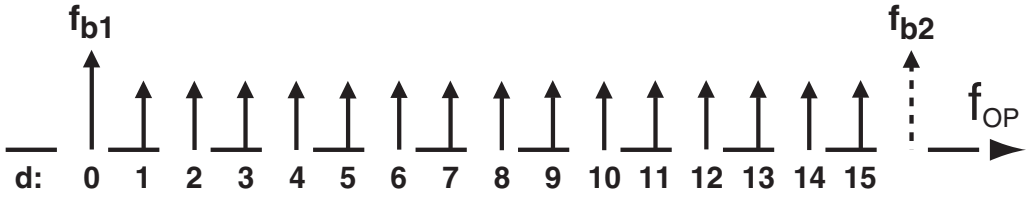
Table 38. Distribution of f_{b2} in the modulated frame

Distribution of fb2 in the modulated frame																
	PWM - cycle															
Fractional Divider (d)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1	X															
2	X								X							
3	X				X				X							
4	X				X				X				X			
5	X		X		X				X				X			
6	X		X		X				X		X		X			
7	X		X		X		X		X		X		X			
8	X		X		X		X		X		X		X		X	
9	X	X	X		X		X		X		X		X		X	
10	X	X	X		X		X		X	X	X		X		X	
11	X	X	X		X	X	X		X	X	X		X		X	
12	X	X	X		X	X	X		X	X	X		X	X	X	
13	X	X	X	X	X	X	X		X	X	X		X	X	X	
14	X	X	X	X	X	X	X		X	X	X	X	X	X	X	
15	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

While ‘X’ in the table, f_{b2} prime to f_{b1} in cycle corresponding cycle.

So for each row, a number of fb2 take place of fb1.

Figure 16-12. Resulting Frequency versus d.



16.7.2 Modes of Operation

16.7.2.1 Normal Mode

The simplest mode of operation is the normal mode. See Figure 16-6.

The active time of PSCOUTn0 is given by the OT0 value. The active time of PSCOUTn1 is given by the OT1 value. Both of them are 12 bit values. Thanks to DT0 & DT1 to ajust the dead time between PSCOUTn0 and PSCOUTn1 active signals.

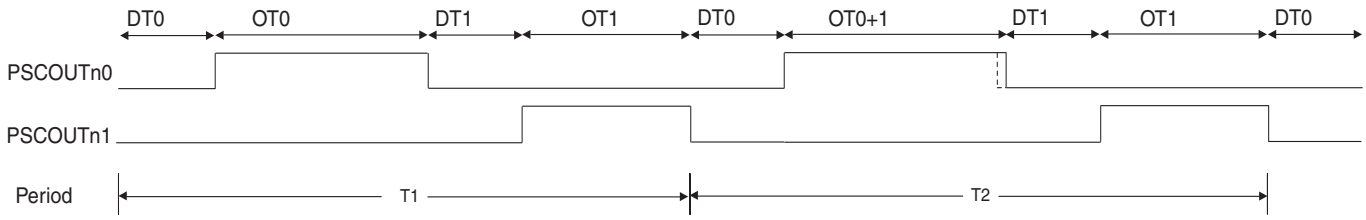
The waveform frequency is defined by the following equation:

$$f_{PSCn} = \frac{1}{PSCnCycle} = \frac{f_{CLK_PSCn}}{(OT0 + OT1 + DT0 + DT1)} = = 1$$

16.7.2.2 Enhanced Mode

The Enhanced Mode uses the previously described method to generate a high resolution frequency.

Figure 16-13. Enhanced Mode, Timing Diagram



The supplementary step in counting to generate f_{b2} is added on the PSCn0 signal while needed in the frame according to the fractional divider. SeeTable 38, “Distribution of fb2 in the modulated frame,” on page 135.

The waveform frequency is defined by the following equations:

$$f1_{PSCn} = \frac{1}{T_1} = \frac{f_{CLK_PSCn}}{(OT0 + OT1 + DT0 + DT1)}$$

$$f2_{PSCn} = \frac{1}{T_2} = \frac{f_{CLK_PSCn}}{(OT0 + OT1 + DT0 + DT1 + 1)}$$

$$f_{AVERAGE} = \frac{d}{16} \times f1_{PSCn} + \frac{16 - d}{16} \times f2_{PSCn}$$

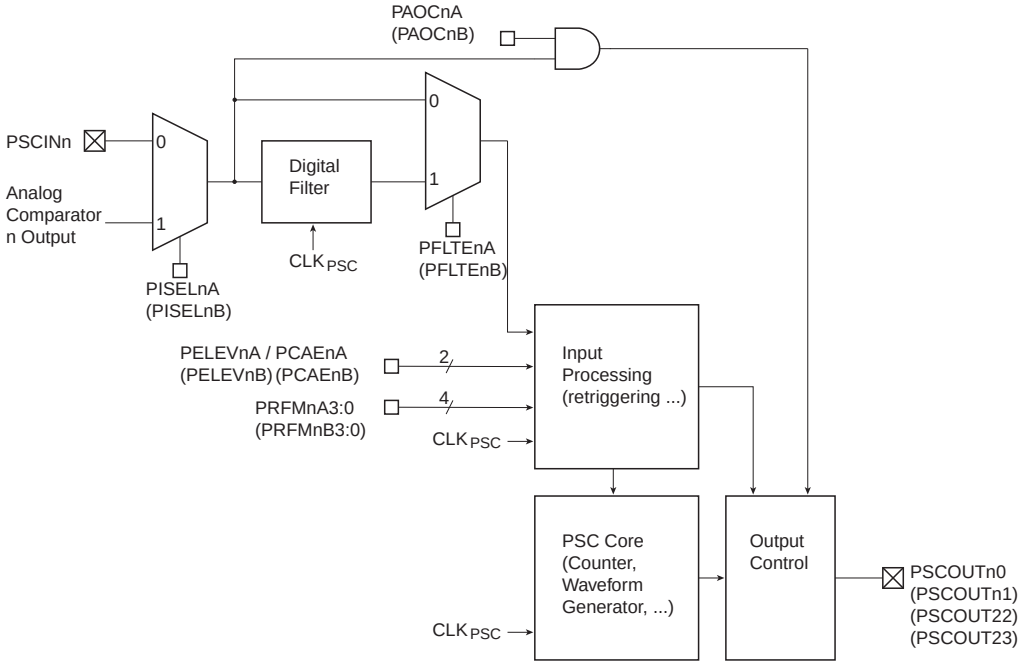
d is the fractionel divider factor.

16.8 PSC Inputs

Each part A or B of PSC has its own system to take into account one PSC input. According to PSC n Input A/B Control Register (see description 16.25.13page 161), PSCnIN0/1 input can act has a Retrigger or Fault input.

This system A or B is also configured by this PSC n Input A/B Control Register (PFRCnA/B).

Figure 16-14. PSC Input Module



16.8.1 PSC Retrigger Behaviour versus PSC running modes

In centered mode, Retrigger Inputs have no effect.

In two ramp or four ramp mode, Retrigger Inputs A or B cause the end of the corresponding cycle A or B and the beginning of the following cycle B or A.

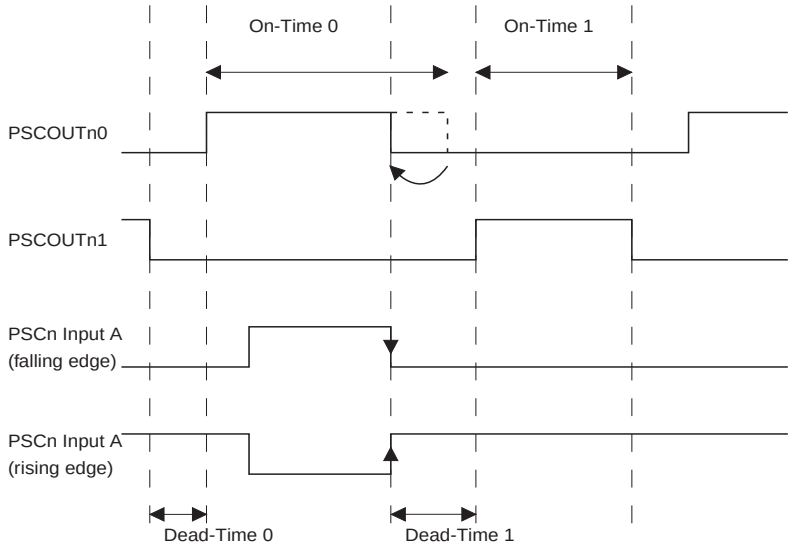
In one ramp mode, Retrigger Inputs A or B reset the current PSC counting to zero.

16.8.2 Retrigger PSCOUTn0 On External Event

PSCOUTn0 ouput can be resetted before end of On-Time 0 on the change on PSCn Input A. PSCn Input A can be configured to do not act or to act on level or edge modes. The polarity of PSCn Input A is configurable thanks to a sense control block. PSCn Input A can be the Output of the analog comparator or the PSCINn input.

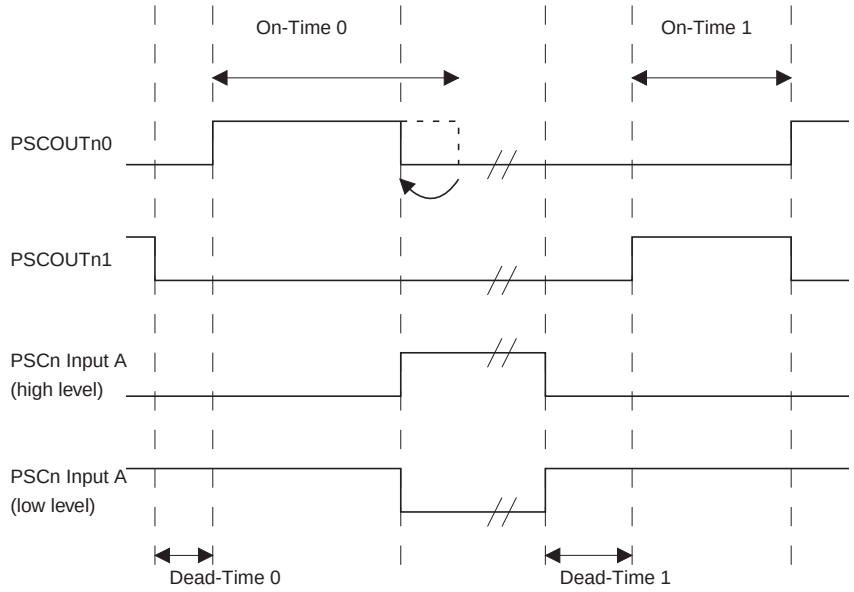
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 16-15. PSCOUTn0 retriggered by PSCn Input A (Edge Retriggering)



Note: This exemple is given in “Input Mode 8” in “2 or 4 ramp mode” See Figure 16-31. for details.

Figure 16-16. PSCOUTn0 retriggered by PSCn Input A (Level Acting)



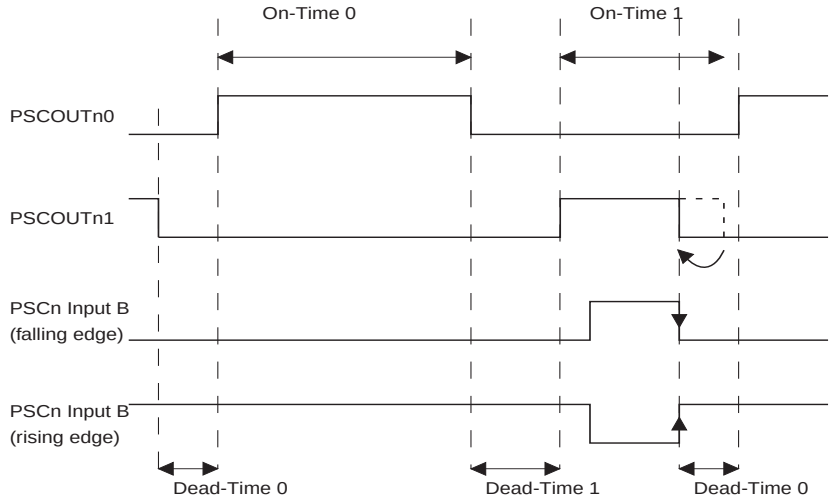
Note: This exemple is given in “Input Mode 1” in “2 or 4 ramp mode” See Figure 16-20. for details.

16.8.3 Retrigger PSCOUTn1 On External Event

PSCOUTn1 ouput can be resetted before end of On-Time 1 on the change on PSCn Input B. The polarity of PSCn Input B is configurable thanks to a sense control block. PSCn Input B can be configured to do not act or to act on level or edge modes. PSCn Input B can be the Output of the analog comparator or the PSCINn input.

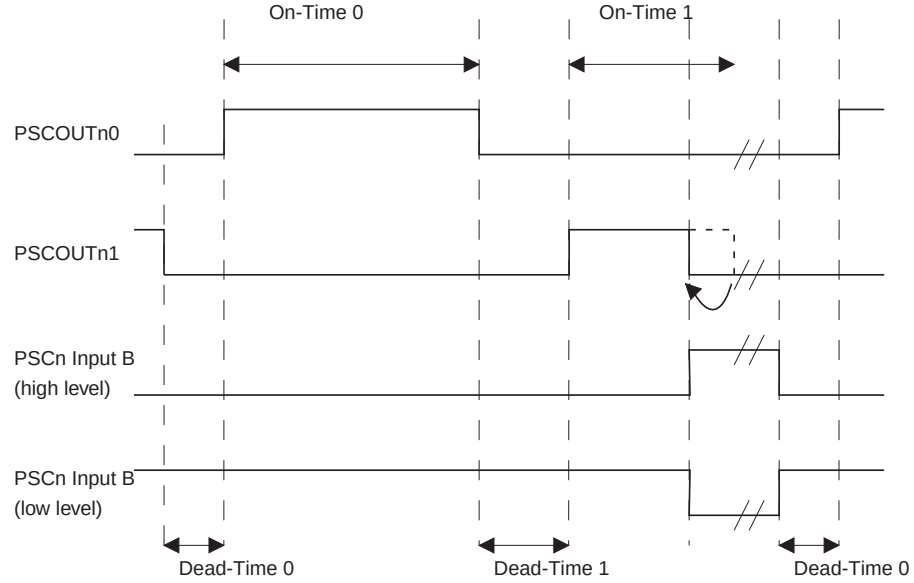
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 16-17. PSCOUTn1 retriggered by PSCn Input B (Edge Retriggering)



Note: This exemple is given in “Input Mode 8” in “2 or 4 ramp mode” See Figure 16-31. for details.

Figure 16-18. PSCOUTn1 retriggered by PSCn Input B (Level Acting)

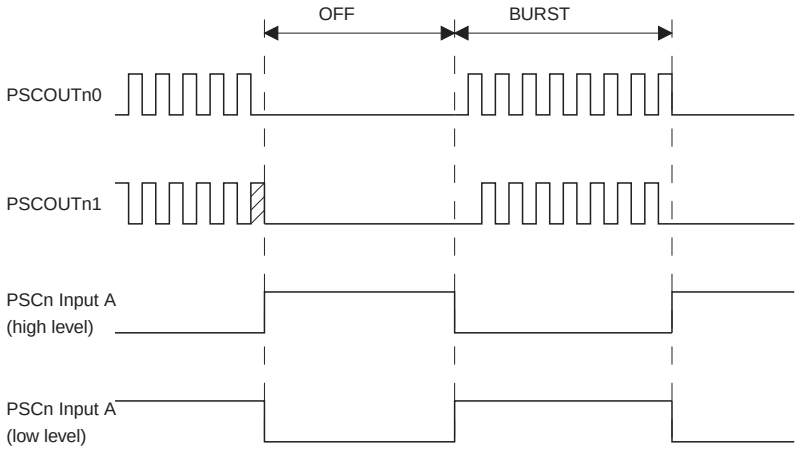


Note: This exemple is given in “Input Mode 1” in “2 or 4 ramp mode” See Figure 16-20. for details.

16.8.3.1 Burst Generation

Note: On level mode, it’s possible to use PSC to generate burst by using Input Mode 3 or Mode 4 (See Figure 16-24. and Figure 16-25. for details.)

Figure 16-19. Burst Generation



16.8.4 PSC Input Configuration

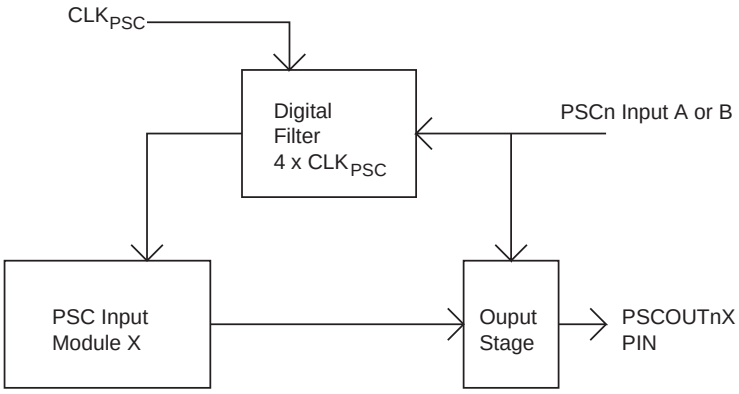
The PSC Input Configuration is done by programming bits in configuration registers.

16.8.4.1 Filter Enable

If the “Filter Enable” bit is set, a digital filter of 4 cycles is inserted before evaluation of the signal. The disable of this function is mainly needed for prescaled PSC clock sources, where the noise cancellation gives too high latency.

Important: If the digital filter is active, the level sensitivity is true also with a disturbed PSC clock to deactivate the outputs (emergency protection of external component). Likewise when used as fault input, PSCn Input A or Input B have to go through PSC to act on PSCOUTn0/1/2/3 output. This way needs that CLK_{PSC} is running. So thanks to PSC Asynchronous Output Control bit (PAOCnA/B), PSCnIN0/1 input can desactivate directly the PSC output. Notice that in this case, input is still taken into account as usually by Input Module System as soon as CLK_{PSC} is running.

PSC Input Filterring



16.8.4.2 Signal Polarity

One can select the active edge (edge modes) or the active level (level modes) See PELEVnx bit description in Section “PSC n Input A Control Register – PFRCnA”, page 16116.25.13.

If PELEVnx bit set, the significant edge of PSCn Input A or B is rising (edge modes) or the active level is high (level modes) and vice versa for unset/falling/low

- In 2- or 4-ramp mode, PSCn Input A is taken into account only during Dead-Time0 and On-Time0 period (respectively Dead-Time1 and On-Time1 for PSCn Input B).

- In 1-ramp-mode PSC Input A or PSC Input B act on the whole ramp.

16.8.4.3 Input Mode Operation

Thanks to 4 configuration bits (PRFM3:0), it's possible to define the mode of the PSC input. All

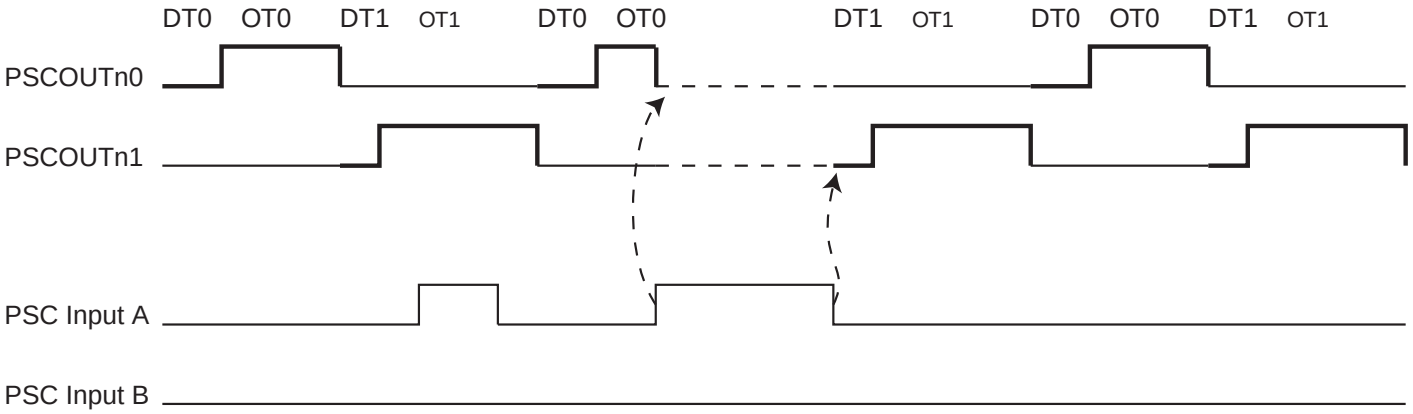
Table 39. PSC Input Mode Operation

	PRFM3:0	Description
0	0000b	PSCn Input has no action on PSC output
1	0001b	16.9See "PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait" on page 142.
2	0010b	See "PSC Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait" on page 143.
3	0011b	See "PSC Input Mode 3: Stop signal, Execute Opposite while Fault active" on page 144.
4	0100b	See "PSC Input Mode 4: Deactivate outputs without changing timing." on page 144.
5	0101b	See "PSC Input Mode 5: Stop signal and Insert Dead-Time" on page 145.
6	0110b	See "PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait." on page 146.
7	0111b	See "PSC Input Mode 7: Halt PSC and Wait for Software Action" on page 146.
8	1000b	See "PSC Input Mode 8: Edge Retrigger PSC" on page 147.
9	1001b	See "PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC" on page 148.
10	1010b	Reserved : Do not use
11	1011b	
12	1100b	
13	1101b	
14	1110b	See "PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Dis-activate Output" on page 149.
15	1111b	Reserved : Do not use

Notice: All following examples are given with rising edge or high level active inputs.

16.9 PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait

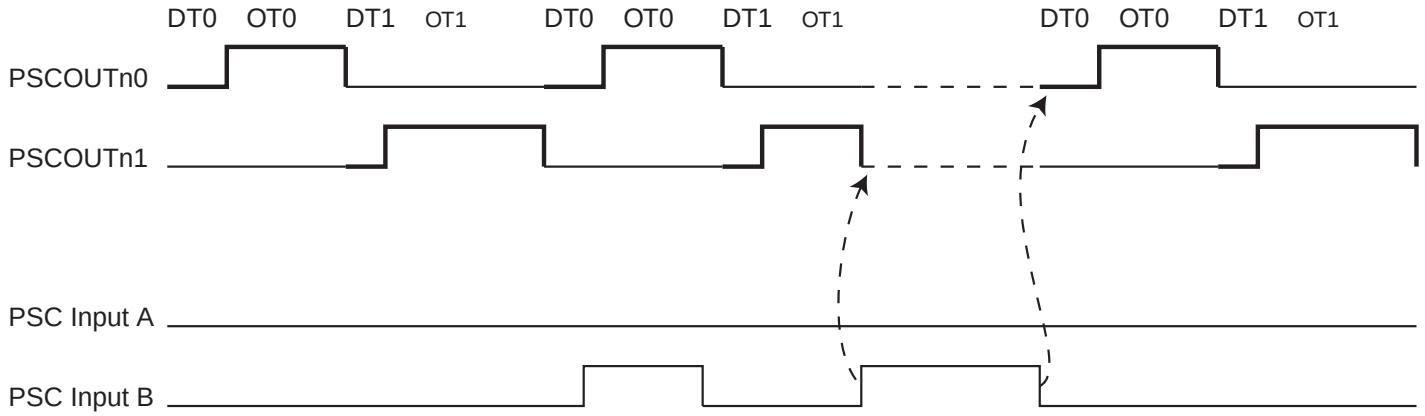
Figure 16-20. PSCn behaviour versus PSCn Input A in Fault Mode 1



PSC Input A is taken into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSC Input A event occurs, PSC releases PSCOUTn0, waits for PSC Input A inactive state and then jumps and executes DT1 plus OT1.

Figure 16-21. PSCn behaviour versus PSCn Input B in Fault Mode 1

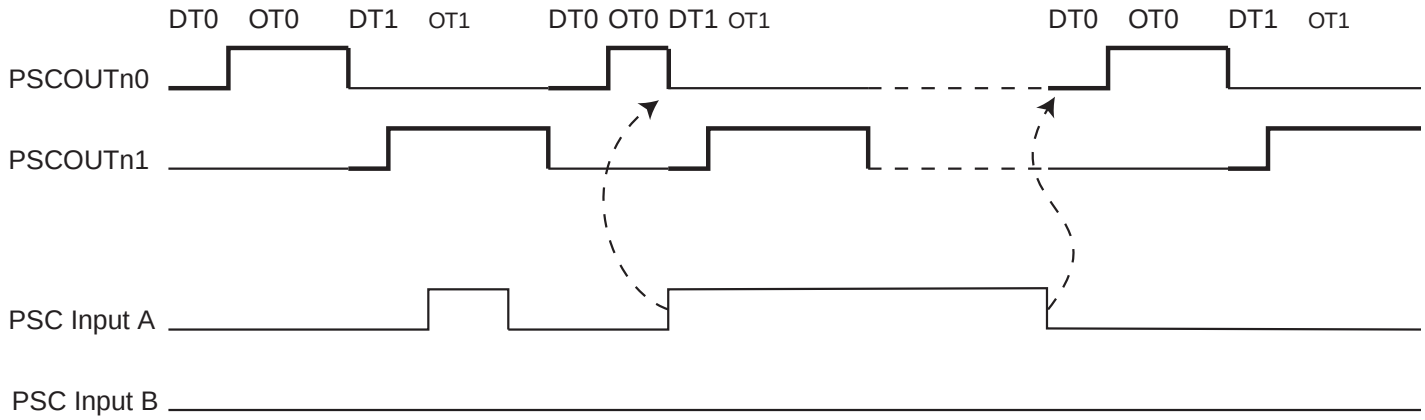


PSC Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSC Input B event occurs, PSC releases PSCOUTn1, waits for PSC Input B inactive state and then jumps and executes DT0 plus OT0.

16.10 PSC Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait

Figure 16-22. PSCn behaviour versus PSCn Input A in Fault Mode 2

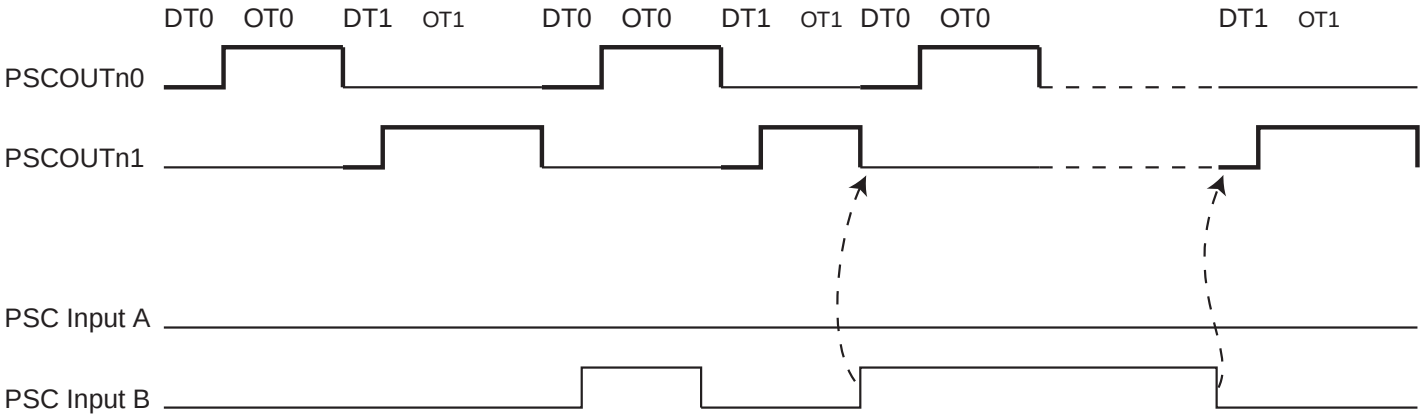


PSC Input A is take into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSCn Input A event occurs, PSC releases PSCOUTn0, jumps and executes DT1 plus OT1 and then waits for PSC Input A inactive state.

Even if PSC Input A is released during DT1 or OT1, DT1 plus OT1 sub-cycle is always completely executed.

Figure 16-23. PSCn behaviour versus PSCn Input B in Fault Mode 2



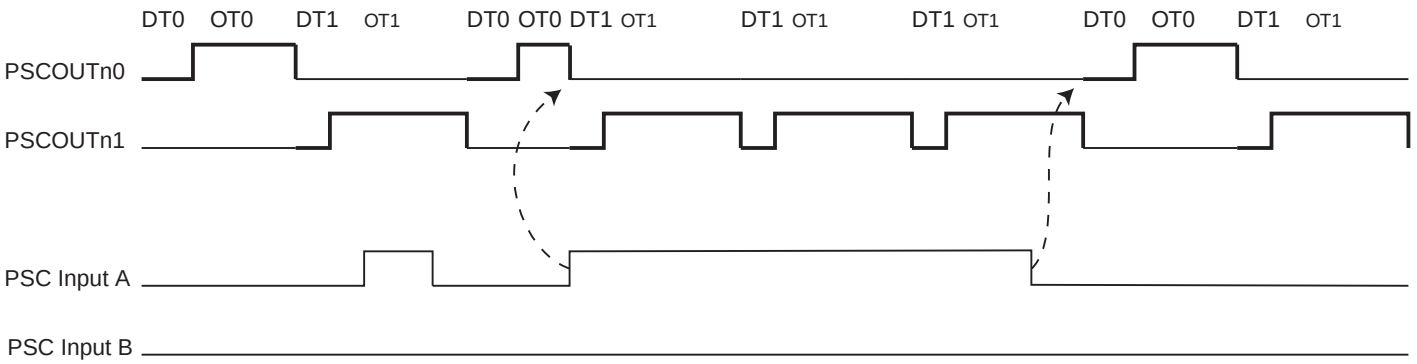
PSC Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSC Input B event occurs, PSC releases PSCOUTn1, jumps and executes DT0 plus OT0 and then waits for PSC Input B inactive state.

Even if PSC Input B is released during DT0 or OT0, DT0 plus OT0 sub-cycle is always completely executed.

16.11 PSC Input Mode 3: Stop signal, Execute Opposite while Fault active

Figure 16-24. PSCn behaviour versus PSCn Input A in Mode 3

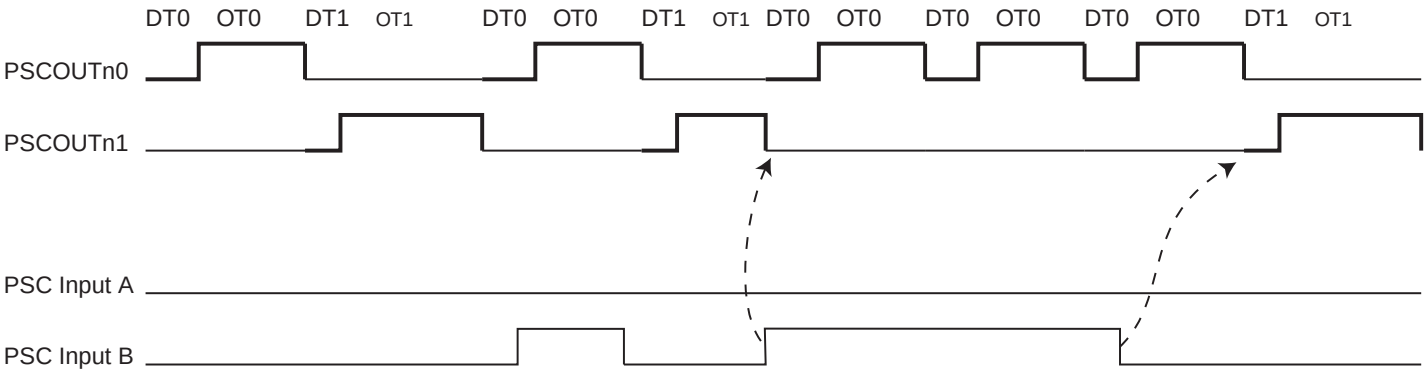


PSC Input A is taken into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSC Input A event occurs, PSC releases PSCOUTn0, jumps and executes DT1 plus OT1 plus DT0 while PSC Input A is in active state.

Even if PSC Input A is released during DT1 or OT1, DT1 plus OT1 sub-cycle is always completely executed.

Figure 16-25. PSCn behaviour versus PSCn Input B in Mode 3



PSC Input B is taken into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSC Input B event occurs, PSC releases PSCnOUT1, jumps and executes DT0 plus OT0 plus DT1 while PSC Input B is in active state.

Even if PSC Input B is released during DT0 or OT0, DT0 plus OT0 sub-cycle is always completely executed.

16.12 PSC Input Mode 4: Deactivate outputs without changing timing.

Figure 16-26. PSC behaviour versus PSCn Input A or Input B in Mode 4

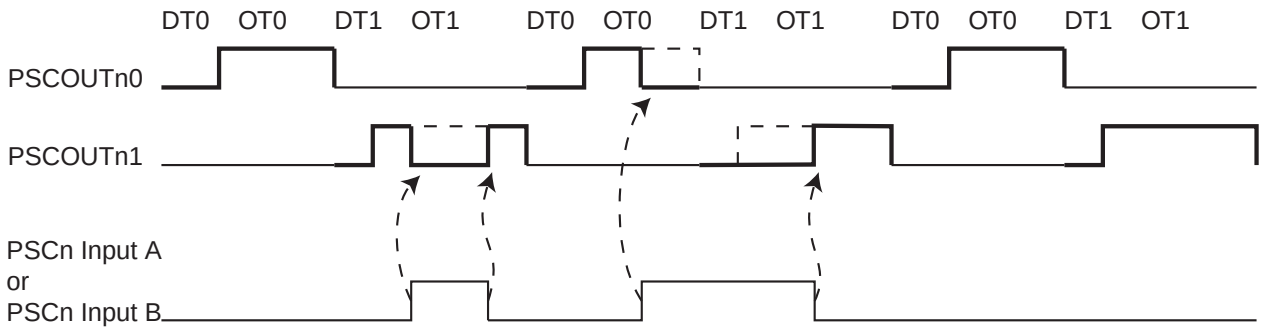
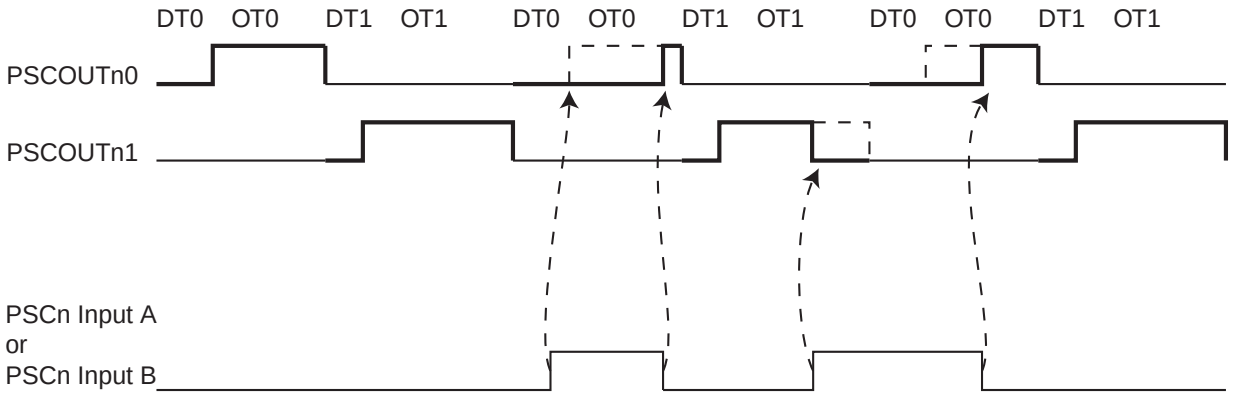


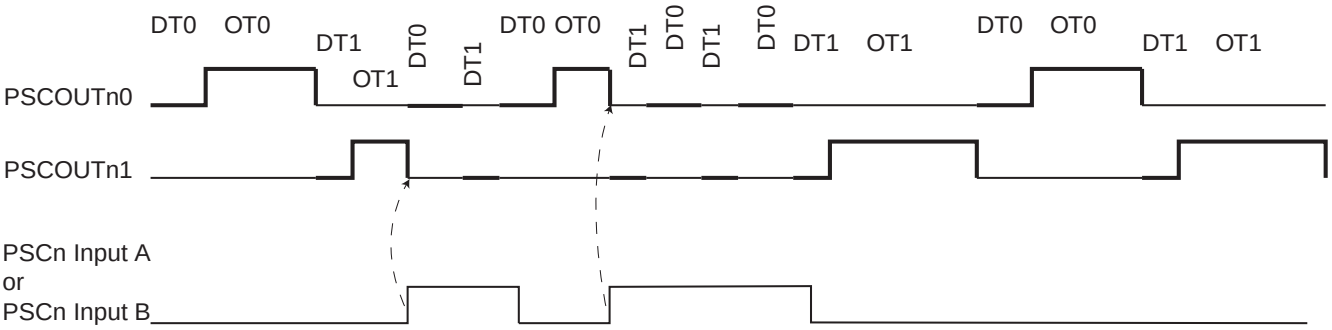
Figure 16-27. PSC behaviour versus PSCn Input A or Input B in Fault Mode 4



PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

16.13 PSC Input Mode 5: Stop signal and Insert Dead-Time

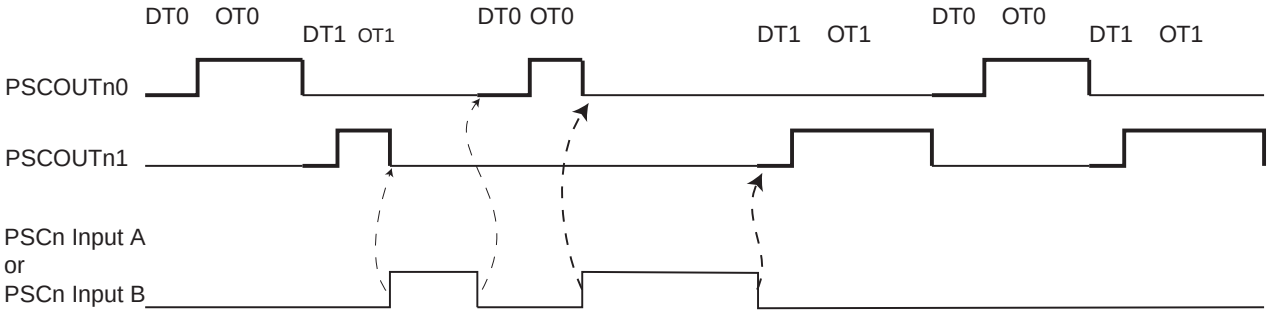
Figure 16-28. PSC behaviour versus PSCn Input A in Fault Mode 5



Used in Fault mode 5, PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

16.14 PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait.

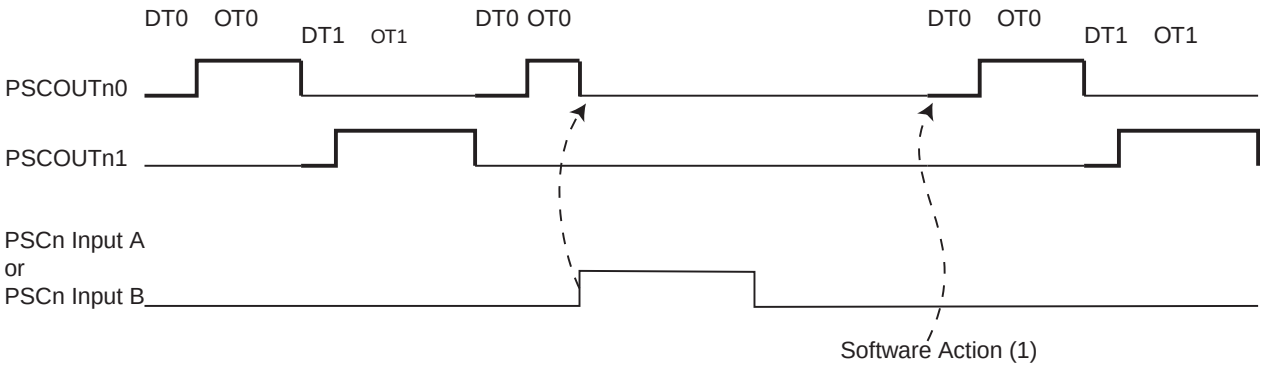
Figure 16-29. PSC behaviour versus PSCn Input A in Fault Mode 6



Used in Fault mode 6, PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

16.15 PSC Input Mode 7: Halt PSC and Wait for Software Action

Figure 16-30. PSC behaviour versus PSCn Input A in Fault Mode 7

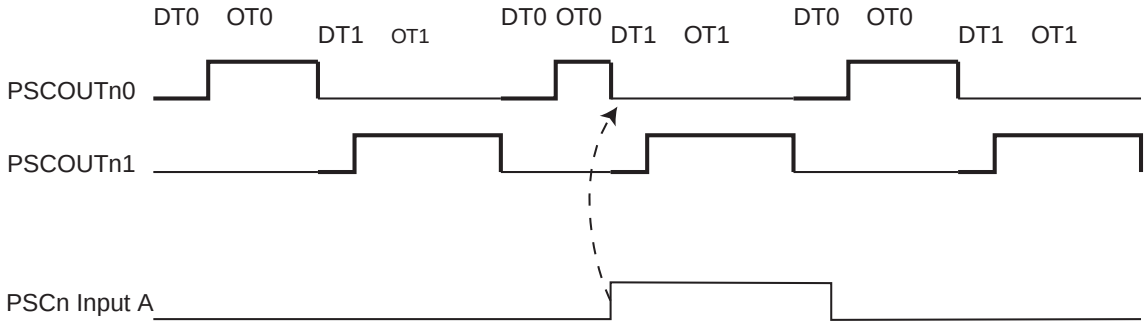


Note: 1. Software action is the setting of the PRUNn bit in PCTLn register.

Used in Fault mode 7, PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

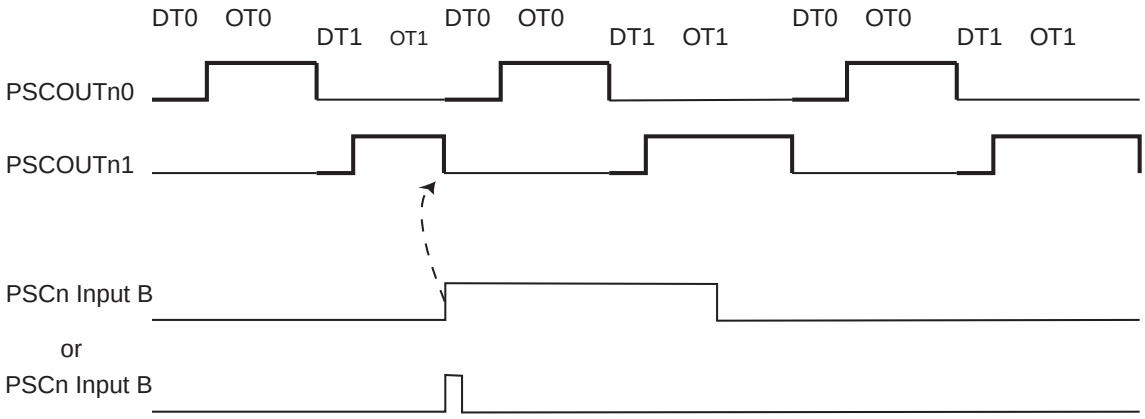
16.16 PSC Input Mode 8: Edge Retrigger PSC

Figure 16-31. PSC behaviour versus PSCn Input A in Mode 8



The output frequency is modulated by the occurence of significative edge of retriggering input.

Figure 16-32. PSC behaviour versus PSCn Input B in Mode 8

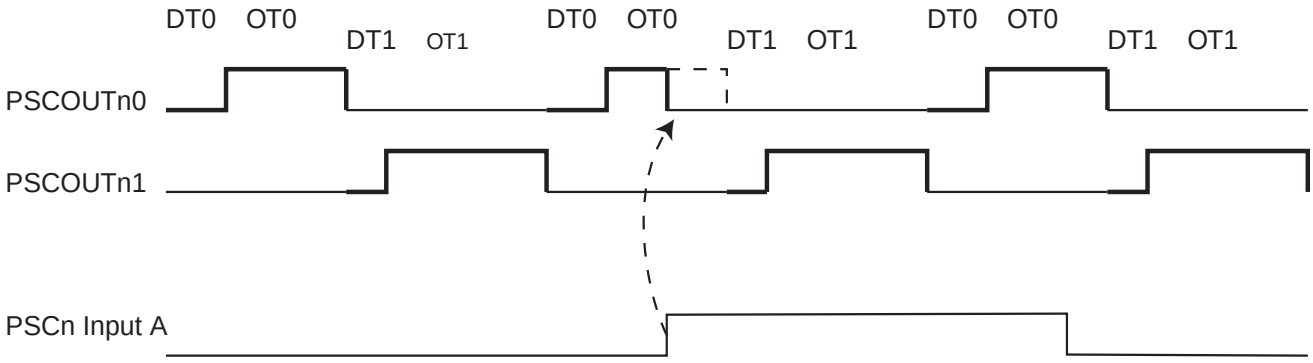


The output frequency is modulated by the occurence of significative edge of retriggering input.

The retrigger event is taken into account only if it occurs during the corresponding On-Time.

16.17 PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC

Figure 16-33. PSC behaviour versus PSCn Input A in Mode 9

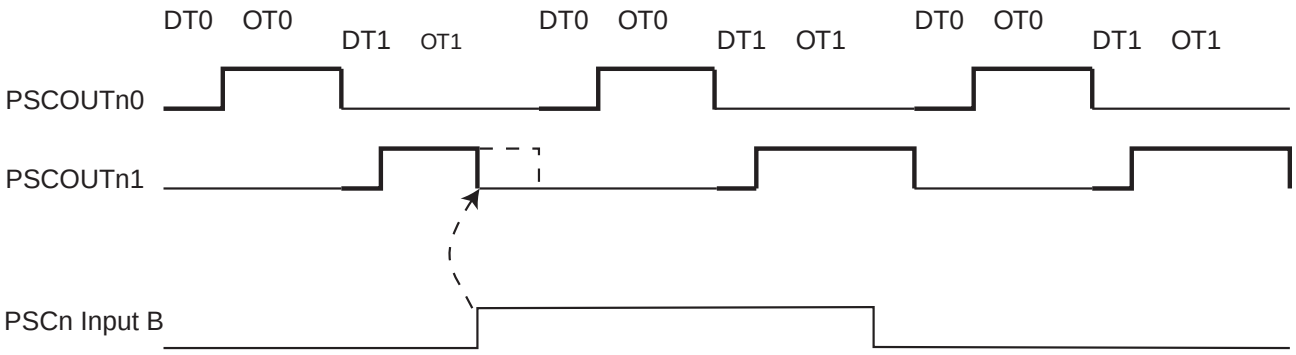


The output frequency is not modified by the occurence of significative edge of retriggering input.

Only the output is disactivated when significative edge on retriggering input occurs.

Note: In this mode the output of the PSC becomes active during the next ramp even if the Retrigger/Fault input is actve. Only the significative edge of Retrigger/Fault input is taken into account.

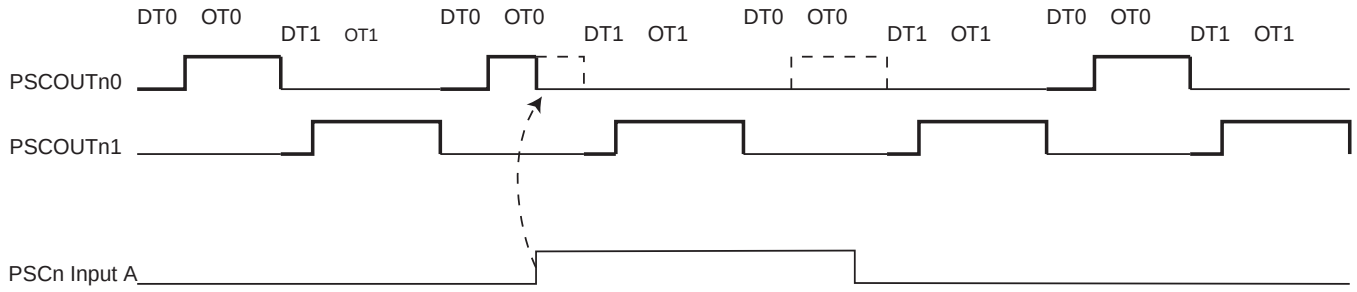
Figure 16-34. PSC behaviour versus PSCn Input B in Mode 9



The retrigger event is taken into account only if it occurs during the corresponding On-Time.

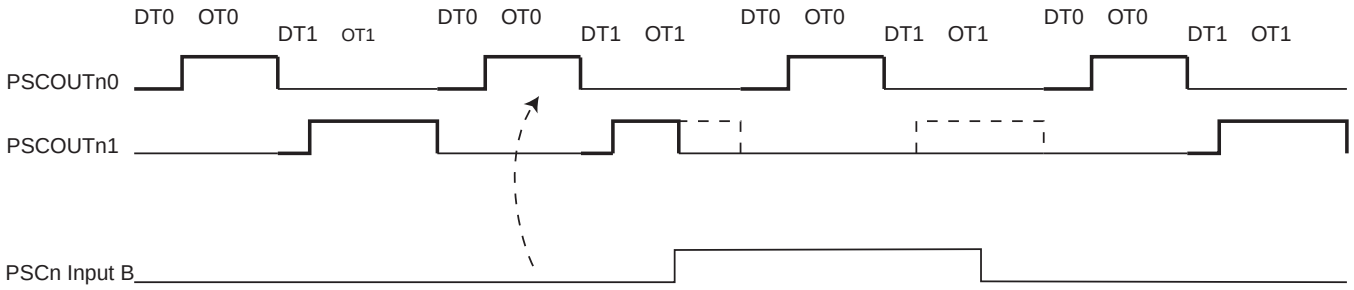
16.18 PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Disactivate Output

Figure 16-35. PSC behaviour versus PSCn Input A in Mode 14



The output frequency is not modified by the occurence of significative edge of retriggering input.

Figure 16-36. PSC behaviour versus PSCn Input B in Mode 14



The output is disactivated while retriggering input is active.

The output of the PSC is set to an inactive state and the corresponding ramp is not aborted. The output stays in an inactive state while the Retrigger/Fault input is actve. The PSC runs at constant frequency.

16.18.1 Available Input Mode according to Running Mode

Some Input Modes are not consistent with some Running Modes. So the table below gives the input modes which are valid according to running modes..

Table 40. Available Input Modes according to Running Modes

Input Mode Number :	1 Ramp Mode	2 Ramp Mode	4 Ramp Mode	Centered Mode
1	Valid	Valid	Valid	Do not use
2	Do not use	Valid	Valid	Do not use
3	Do not use	Valid	Valid	Do not use
4	Valid	Valid	Valid	Valid
5	Do not use	Valid	Valid	Do not use
6	Do not use	Valid	Valid	Do not use
7	Valid	Valid	Valid	Valid
8	Valid	Valid	Valid	Do not use
9	Valid	Valid	Valid	Do not use
10	Do not use			
11				
12				
13				
14	Valid	Valid	Valid	Do not use
15	Do not use			

16.18.2 Event Capture

The PSC can capture the value of time (PSC counter) when a retrigger event or fault event occurs on PSC inputs. This value can be read by sofware in PICRnH/L register.

16.18.3 Using the Input Capture Unit

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

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

16.19 PSC2 Outputs

16.19.1 Output Matrix

PSC2 has an output matrix which allow in 4 ramp mode to program a value of PSCOUT20 and PSCOUT21 binary value for each ramp.

Table 41. Output Matrix versus ramp number

	Ramp 0	Ramp 1	Ramp 2	Ramp 3
PSCOUT20	**POMV2A0**	**POMV2A1**	**POMV2A2**	**POMV2A3**
PSCOUT21	**POMV2B0**	**POMV2B1**	**POMV2B2**	**POMV2B3**

PSCOUT2m takes the value given in Table 41. during all corresponding ramp. Thanks to the Output Matrix it is possible to generate all kind of PSCOUT20/PSCOUT21 combination.

When Output Matrix is used, the PSC n Output Polarity POPn has no action on the outputs.

16.19.2 PSCOUT22 & PSCOUT23 Selectors

PSC 2 has two supplementary outputs PSCOUT22 and PSCOUT23.

According to POS22 and POS23 bits in PSOC2 register, PSCOUT22 and PSCOUT23 duplicate PSCOUT20 and PSCOU21.

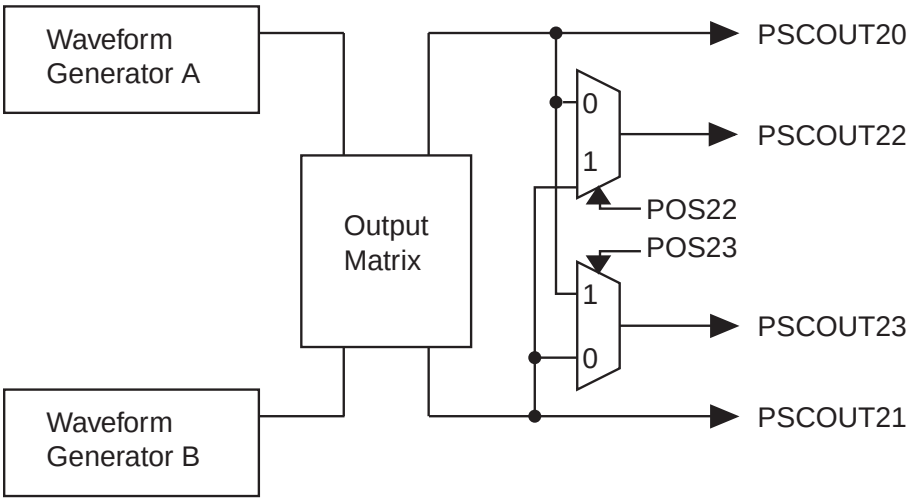
If POS22 bit in PSOC2 register is clear, PSCOUT22 duplicates PSCOUT20.

If POS22 bit in PSOC2 register is set, PSCOUT22 duplicates PSCOUT21.

If POS23 bit in PSOC2 register is clear, PSCOUT23 duplicates PSCOUT21.

If POS23 bit in PSOC2 register is set, PSCOUT23 duplicates PSCOUT20.

Figure 16-37. PSCOUT22 and PSCOUT23 Outptuts



16.20 Analog Synchronization

PSC generates a signal to synchronize the sample and hold; synchronisation is mandatory for measurements.

This signal can be selected between all falling or rising edge of PSCn0 or PSCn1 outputs.

In center aligned mode, OCRnRAH/L is not used, so it can be used to specified the synchronization of the ADC. It this case, it's minimum value is 1.

16.21 Interrupt Handling

As each PSC can be dedicated for one function, each PSC has its own interrupt system (vector ...)

List of interrupt sources:

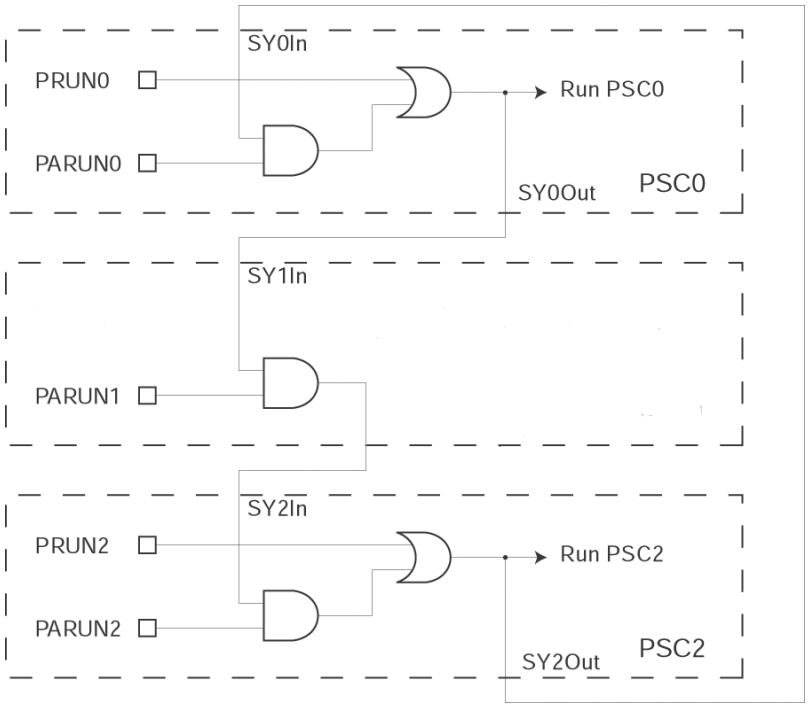
- Counter reload (end of On Time 1)
- PSC Input event (active edge or at the beginning of level configured event)
- PSC Mutual Synchronization Error

16.22 PSC Synchronization

2 or 3 PSC can be synchronized together. In this case, two waveform alignments are possible:

- The waveforms are center aligned in the Center Aligned mode if master and slaves are all with the same PSC period (which is the natural use).
- The waveforms are edge aligned in the 1, 2 or 4 ramp mode

Figure 16-38. PSC Run Synchronization



If the PSCm has its PARUNn bit set, then it can start at the same time than PSCn-1.

PRUNn and PARUNn bits are located in PCTLn register. See “PSC 0 Control Register – PCTL0” on page 158. See “PSC 1 Control Register – PCTL1” on page 160. See “PSC 2 Control Register – PCTL2” on page 160.

Note : Do not set the PARUNn bits on the three PSC at the same time.

Thanks to this feature, we can for example configure two PSC in slave mode (PARUNn = 1 / PRUNn = 0) and one PSC in master mode (PARUNm = 0 / PRUNm = 0). This PSC master can start all PSC at the same moment (PRUNm = 1).

16.22.1 Fault events in Autorun mode

To complete this master/slave mechanism, fault event (input mode 7) is propagated from PSCn-1 to PSCn and from PSCn to PSCn-1.

A PSC which propagate a Run signal to the following PSC stops this PSC when the Run signal is deactivate.

According to the architecture of the PSC synchronization which build a “daisy-chain on the PSC run signal” beetwen the three PSC, only the fault event (mode 7) which is able to “stop” the PSC through the PRUN bits is transmited along this daisy-chain.

A PSC which receive its Run signal from the previous PSC transmits its fault signal (if enabled) to this previous PSC. So a slave PSC propagates its fault events when they are configured and enabled.

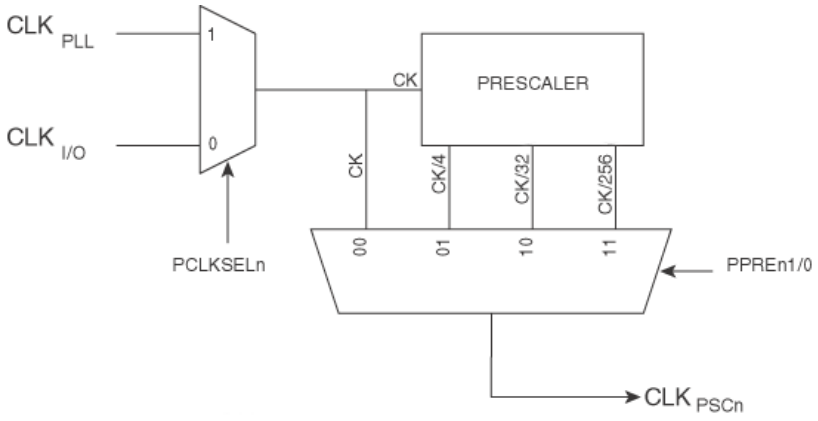
16.23 PSC Clock Sources

PSC must be able to generate high frequency with enhanced resolution.

Each PSC has two clock inputs:

- CLK PLL from the PLL
- CLK I/O

Figure 16-39. Clock selection



PCLKSELn bit in PSC n Configuration register (PCNFn) is used to select the clock source.

PPREn1/0 bits in PSC n Control Register (PCTLn) are used to select the divide factor of the clock.

Table 42. Output Clock versus Selection and Prescaler

PCLKSELn	PPREn1	PPREn0	CLKPSCn output AT90PWM1
0	0	0	CLK I/O
0	0	1	CLK I/O / 4
0	1	0	CLK I/O / 32
0	1	1	CLK I/O / 256
1	0	0	CLK PLL
1	0	1	CLK PLL / 4
1	1	0	CLK PLL / 32
1	1	1	CLK PLL / 256

16.24 Interrupts

This section describes the specifics of the interrupt handling as performed in AT90PWM1.

16.24.1 List of Interrupt Vector

Each PSC provides 2 interrupt vectors

- **PSCn EC (End of Cycle)**: When enabled and when a match with OCRnRB occurs
- **PSCn CAPT (Capture Event)**: When enabled and one of the two following events occurs : retrigger, capture of the PSC counter or Synchro Error.

16.26.216.26.2See PSCn Interrupt Mask Register page 164 and PSCn Interrupt Flag Register page 164.

16.24.2 PSC Interrupt Vectors in AT90PWM1

Table 43. PSC Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
-	-	-	-
2	0x0001	PSC2 CAPT	PSC2 Capture Event or Synchronization Error
3	0x0002	PSC2 EC	PSC2 End Cycle
6	0x0005	PSC0 CAPT	PSC0 Capture Event or Synchronization Error
7	0x0006	PSC0 EC	PSC0 End Cycle
-	-	-	-

16.25 PSC Register Definition

Registers are explained for PSC0. They are identical for PSC1. For PSC2 only different registers are described.

16.25.1 PSC 0 Synchro and Output Configuration – PSOC0

Bit	7	6	5	4	3	2	1	0	
	-	-	PSYNC01	PSYNC00	-	POEN0B	-	POEN0A	PSOC0
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	

16.25.2 PSC 2 Synchro and Output Configuration – PSOC2

Bit	7	6	5	4	3	2	1	0	
	POS23	POS22	PSYNC21	PSYNC20	POEN2D	POEN2B	POEN2C	POEN2A	PSOC2
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 – POS23 : PSCOUT23 Selection (PSC2 only)**

When this bit is clear, PSCOUT23 outputs the waveform generated by Waveform Generator B.

When this bit is set, PSCOUT23 outputs the waveform generated by Waveform Generator A.

- **Bit 6 – POS22 : PSCOUT22 Selection (PSC2 only)**

When this bit is clear, PSCOUT22 outputs the waveform generated by Waveform Generator A.

When this bit is set, PSCOUT22 outputs the waveform generated by Waveform Generator B.

- **Bit 5:4 – PSYNCn1:0: Synchronization Out for ADC Selection**

Select the polarity and signal source for generating a signal which will be sent to the ADC for synchronization.

Table 44. Synchronization Source Description in One/Two/Four Ramp Modes

PSYNCn1	PSYNCn0	Description
0	0	Send signal on leading edge of PSCOUTn0 (match with OCRnSA)
0	1	Send signal on trailing edge of PSCOUTn0 (match with OCRnRA or fault/retrigger on part A)
1	0	Send signal on leading edge of PSCOUTn1 (match with OCRnSB)
1	1	Send signal on trailing edge of PSCOUTn1 (match with OCRnRB or fault/retrigger on part B)

Table 45. Synchronization Source Description in Centered Mode

PSYNCn1	PSYNCn0	Description
0	0	Send signal on match with OCRnRA (during counting down of PSC). The min value of OCRnRA must be 1.
0	1	Send signal on match with OCRnRA (during counting up of PSC). The min value of OCRnRA must be 1.
1	0	no synchronization signal
1	1	no synchronization signal

- **Bit 3 – POEN2D : PSCOUT23 Output Enable (PSC2 only)**

When this bit is clear, second I/O pin affected to PSCOUT23 acts as a standard port.

When this bit is set, second I/O pin affected to PSCOUT23 is connected to the PSC waveform generator B output and is set and clear according to the PSC operation.

- **Bit 2 – POENnB: PSC n OUT Part B Output Enable**

When this bit is clear, I/O pin affected to PSCOUTn1 acts as a standard port.

When this bit is set, I/O pin affected to PSCOUTn1 is connected to the PSC waveform generator B output and is set and clear according to the PSC operation.

- **Bit 1 – POEN2C : PSCOUT22 Output Enable (PSC2 only)**

When this bit is clear, second I/O pin affected to PSCOUT22 acts as a standard port.

When this bit is set, second I/O pin affected to PSCOUT22 is connected to the PSC waveform generator A output and is set and clear according to the PSC operation.

- **Bit 0 – POENnA: PSC n OUT Part A Output Enable**

When this bit is clear, I/O pin affected to PSCOUTn0 acts as a standard port.

When this bit is set, I/O pin affected to PSCOUTn0 is connected to the PSC waveform generator A output and is set and clear according to the PSC operation.

16.25.3 Output Compare SA Register – OCRnSAH and OCRnSAL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCRnSA[11:8]				OCRnSAH
	OCRnSA[7:0]								OCRnSAL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

16.25.4 Output Compare RA Register – OCRnRAH and OCRnRAL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCRnRA[11:8]				OCRnRAH
	OCRnRA[7:0]								OCRnRAL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

16.25.5 Output Compare SB Register – OCRnSBH and OCRnSBL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCRnSB[11:8]				OCRnSBH
	OCRnSB[7:0]								OCRnSBL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

16.25.6 Output Compare RB Register – OCRnRBH and OCRnRBL

Bit	7	6	5	4	3	2	1	0	
	OCRnRB[15:12]				OCRnRB[11:8]				OCRnRBH
	OCRnRB[7:0]								OCRnRBL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

Note : n = 0 to 2 according to PSC number.

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the associated pin.

The Output Compare Registers RB contains also a 4-bit value that is used for the flank width modulation.

The Output Compare Registers are 16bit and 12-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.

16.25.7 PSC 0 Configuration Register – PCNF0

Bit	7	6	5	4	3	2	1	0	
	PFIFTY0	PALOCK0	PLOCK0	PMODE01	PMODE00	POP0	PCLKSEL0	-	PCNF0
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	

16.25.8 PSC 1 Configuration Register – PCNF1

Bit	7	6	5	4	3	2	1	0	
	PFIFTY1	PALOCK1	PLOCK1	PMODE11	PMODE10	POP1	PCLKSEL1	-	PCNF1
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	

16.25.9 PSC 2 Configuration Register – PCNF2

Bit	7	6	5	4	3	2	1	0	
	PFIFTY2	PALOCK2	PLOCK2	PMODE21	PMODE20	POP2	PCLKSEL2	POME2	PCNF2
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 PSC n Configuration Register is used to configure the running mode of the PSC.

- **Bit 7 - PFIFTYn: PSC n Fifty**

Writing this bit to one, set the PSC in a fifty percent mode where only OCRnRBH/L and OCRnSBH/L are used. They are duplicated in OCRnRAH/L and OCRnSAH/L during the update of OCRnRBH/L. This feature is useful to perform fifty percent waveforms.

- **Bit 6 - PALOCKn: PSC n Autolock**

When this bit is set, the Output Compare Registers RA, SA, SB, the Output Matrix POM2 and the PSC Output Configuration PSOCn can be written without disturbing the PSC cycles. The update of the PSC internal registers will be done at the end of the PSC cycle if the Output Compare Register RB has been the last written.

When set, this bit prevails over LOCK (bit 5)

- **Bit 5 – PLOCKn: PSC n Lock**

When this bit is set, the Output Compare Registers RA, RB, SA, SB, the Output Matrix POM2 and the PSC Output Configuration PSOCn can be written without disturbing the PSC cycles. The update of the PSC internal registers will be done if the LOCK bit is released to zero.

- **Bit 4:3 – PMODEn1: 0: PSC n Mode**

Select the mode of PSC.

Table 46. PSC n Mode Selection

PMODEn1	PMODEn0	Description
0	0	One Ramp Mode
0	1	Two Ramp Mode
1	0	Four Ramp Mode
1	1	Center Aligned Mode

- **Bit 2 – POPn: PSC n Output Polarity**

If this bit is cleared, the PSC outputs are active Low.

If this bit is set, the PSC outputs are active High.

- **Bit 1 – PCLKSELn: PSC n Input Clock Select**

This bit is used to select between CLKPF or CLKPS clocks.

Set this bit to select the fast clock input (CLKPF).

Clear this bit to select the slow clock input (CLKPS).

- **Bit 0 – POME2: PSC 2 Output Matrix Enable (PSC2 only)**

Set this bit to enable the Output Matrix feature on PSC2 outputs. See “PSC2 Outputs” on page 151.

When Output Matrix is used, the PSC n Output Polarity POPn has no action on the outputs.

16.25.10 PSC 0 Control Register – PCTL0

Bit	7	6	5	4	3	2	1	0	
	PPRE01	PPRE00	PBFM0	PAOC0B	PAOC0A	PARUN0	PCCYC0	PRUN0	PCTL0
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 7:6 – PPRE01:0 : PSC 0 Prescaler Select**

This two bits select the PSC input clock division factor. All generated waveform will be modified by this factor.

Table 47. PSC 0 Prescaler Selection

PPRE01	PPRE00	Description
0	0	No divider on PSC input clock
0	1	Divide the PSC input clock by 4
1	0	Divide the PSC input clock by 16
1	1	Divide the PSC clock by 64

- **Bit 5 – PBFM0 : Balance Flank Width Modulation**

When this bit is clear, Flank Width Modulation operates on On-Time 1 only.

When this bit is set, Flank Width Modulation operates on On-Time 0 and On-Time 1.

- **Bit 4 – PAOC0B : PSC 0 Asynchronous Output Control B**

When this bit is set, Fault input selected to block B can act directly to PSCOUT01 output. See Section “PSC Input Configuration”, page 140.

- **Bit 3 – PAOC0A : PSC 0 Asynchronous Output Control A**

When this bit is set, Fault input selected to block A can act directly to PSCOUT00 output. See Section “PSC Input Configuration”, page 140.

- **Bit 2 – PARUN0 : PSC 0 Autorun**

When this bit is set, the PSC 0 starts with PSC2. That means that PSC 0 starts :

- when PRUN2 bit in PCTL2 is set,
- or when PARUN2 bit in PCTL2 is set and PRUN1 bit in PCTL1 register is set.

Thanks to this bit, 2 or 3 PSCs can be synchronized (motor control for example)

- **Bit 1 – PCCYC0 : PSC 0 Complete Cycle**

When this bit is set, the PSC 0 completes the entire waveform cycle before halt operation requested by clearing PRUN0. This bit is not relevant in slave mode (PARUN0 = 1).

- **Bit 0 – PRUN0 : PSC 0 Run**

Writing this bit to one starts the PSC 0.

When set, this bit prevails over PARUN0 bit.

16.25.11 PSC 1 Control Register – PCTL1

Bit						2		
						PARUN1		
Read/Write						R/W		
Initial Value						0		

- **Bit 2 – PARUN1 : PSC 1 Autorun**

When this bit is set, the PSC 1 starts with PSC0. That means that PSC 1 starts :

- when PRUN0 bit in PCTL0 register is set,
- or when PARUN0 bit in PCTL0 is set and PRUN2 bit in PCTL2 register is set.

Thanks to this bit, 2 or 3 PSCs can be synchronized (motor control for example.

16.25.12 PSC 2 Control Register – PCTL2

Bit	7	6	5	4	3	2	1	0	
	PPRE21	PPRE20	PBFM2	PAOC2B	PAOC2A	PARUN2	PCCYC2	PRUN2	PCTL2
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 7:6 – PPRE21:0 : PSC 2 Prescaler Select**

This two bits select the PSC input clock division factor.All generated waveform will be modified by this factor.

Table 48. PSC 2 Prescaler Selection

PPRE21	PPRE20	Description
0	0	No divider on PSC input clock
0	1	Divide the PSC input clock by 4
1	0	Divide the PSC input clock by 16
1	1	Divide the PSC clock by 64

- **Bit 5 – PBFM2 : Balance Flank Width Modulation**

When this bit is clear, Flank Width Modulation operates on On-Time 1 only.

When this bit is set, Flank Width Modulation operates on On-Time 0 and On-Time 1.

- **Bit 4 – PAOC2B : PSC 2 Asynchronous Output Control B**

When this bit is set, Fault input selected to block B can act directly to PSCOUT21 and PSCOUT23 outputs. See Section “PSC Clock Sources”, page 153.

- **Bit 3 – PAOC2A : PSC 2 Asynchronous Output Control A**

When this bit is set, Fault input selected to block A can act directly to PSCOUT20 and PSCOUT22 outputs. See Section “PSC Clock Sources”, page 153.

- **Bit 2 – PARUN2 : PSC 2 Autorun**

When this bit is set, the PSC 2 starts with PSC1. That means that PSC 2 starts :

- when PRUN1 bit in PCTL1 register is set,
- or when PARUN1 bit in PCTL1 is set and PRUN0 bit in PCTL0 register is set.

- **Bit 1 – PCCYC2 : PSC 2 Complete Cycle**

When this bit is set, the PSC 2 completes the entire waveform cycle before halt operation requested by clearing PRUN2. This bit is not relevant in slave mode (PARUN2 = 1).

- **Bit 0 – PRUN2 : PSC 2 Run**

Writing this bit to one starts the PSC 2.

When set, this bit prevails over PARUN2 bit.

16.25.13 PSC n Input A Control Register – PFRCnA

Bit	7	6	5	4	3	2	1	0	
	PCAEnA	PISELnA	PELEVnA	PFLTEnA	PRFMnA3	PRFMnA2	PRFMnA1	PRFMnA0	PFRCnA
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	

16.25.14 PSC n Input B Control Register – PFRCnB

Bit	7	6	5	4	3	2	1	0	
	PCAEnB	PISELnB	PELEVnB	PFLTEnB	PRFMnB3	PRFMnB2	PRFMnB1	PRFMnB0	PFRCnB
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 Control Registers are used to configure the 2 PSC's Retrigger/Fault block A & B. The 2 blocks are identical, so they are configured on the same way.

- **Bit 7 – PCAEnx : PSC n Capture Enable Input Part x**

Writing this bit to one enables the capture function when external event occurs on input selected as input for Part x (see PISELnx bit in the same register).

- **Bit 6 – PISELnx : PSC n Input Select for Part x**

Clear this bit to select PSCINn as input of Fault/Retrigger block x.

Set this bit to select Comparator n Output as input of Fault/Retrigger block x.

- **Bit 5 –PELEVnx : PSC n Edge Level Selector of Input Part x**

When this bit is clear, the falling edge or low level of selected input generates the significative event for retrigger or fault function .

When this bit is set, the rising edge or high level of selected input generates the significative event for retrigger or fault function.

- **Bit 4 – PFLTEnx : PSC n Filter Enable on Input Part x**

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

- **Bit 3:0 – PRFMnx3:0: PSC n Fault Mode**

These four bits define the mode of operation of the Fault or Retrigger functions.

(see PSC Functional Specification for more explanations)

Table 49. Level Sensitivity and Fault Mode Operation

PRFMnx3:0	Description
0000b	No action, PSC Input is ignored
0001b	PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait
0010b	PSC Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait
0011b	PSC Input Mode 3: Stop signal, Execute Opposite while Fault active
0100b	PSC Input Mode 4: Deactivate outputs without changing timing.
0101b	PSC Input Mode 5: Stop signal and Insert Dead-Time
0110b	PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait.
0111b	PSC Input Mode 7: Halt PSC and Wait for Software Action
1000b	PSC Input Mode 8: Edge Retrigger PSC
1001b	PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC
1010b	Reserved (do not use)
1011b	
1100b	
1101b	
1110b	PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Disactivate Output
1111b	Reserved (do not use)

16.25.15 PSC 0 Input Capture Register – PICR0H and PICR0L

Bit	7	6	5	4	3	2	1	0	
	PCST0	–	–	–	PICR0[11:8]				PICR0H
	PICR0[7:0]								PICR0L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

16.25.16 PSC 2 Input Capture Register – PICR2H and PICR2L

Bit	7	6	5	4	3	2	1	0	
	PCST2	–	–	–	PICR2[11:8]				PICR2H
	PICR2[7:0]								PICR2L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – PCSTn : PSC Capture Software Trig bit**

Set this bit to trigger off a capture of the PSC counter. When reading, if this bit is set it means that the capture operation was triggered by PCSTn setting otherwise it means that the capture operation was triggered by a PSC input.

The Input Capture is updated with the PSC counter value each time an event occurs on the enabled PSC input pin (or optionally on the Analog Comparator output) if the capture function is enabled (bit PCAEnx in PFRCnx register is set).

The Input Capture Register is 12-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 or 12-bit registers.

16.26 PSC2 Specific Register

16.26.1 PSC 2 Output Matrix – POM2

Bit	7	6	5	4	3	2	1	0	
	POMV2B3	POMV2B2	POMV2B1	POMV2B0	POMV2A3	POMV2A2	POMV2A1	POMV2A0	POM2
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 – POMV2B3: Output Matrix Output B Ramp 3**

This bit gives the state of the PSCOUT21 (and/or PSCOUT23) during ramp 3

- **Bit 6 – POMV2B2: Output Matrix Output B Ramp 2**

This bit gives the state of the PSCOUT21 (and/or PSCOUT23) during ramp 2

- **Bit 5 – POMV2B1: Output Matrix Output B Ramp 1**

This bit gives the state of the PSCOUT21 (and/or PSCOUT23) during ramp 1

- **Bit 4 – POMV2B0: Output Matrix Output B Ramp 0**

This bit gives the state of the PSCOUT21 (and/or PSCOUT23) during ramp 0

- **Bit 3 – POMV2A3: Output Matrix Output A Ramp 3**

This bit gives the state of the PSCOUT20 (and/or PSCOUT22) during ramp 3

- **Bit 2 – POMV2A2: Output Matrix Output A Ramp 2**

This bit gives the state of the PSCOUT20 (and/or PSCOUT22) during ramp 2

- **Bit 1 – POMV2A1: Output Matrix Output A Ramp 1**

This bit gives the state of the PSCOUT20 (and/or PSCOUT22) during ramp 1

- **Bit 0 – POMV2A0: Output Matrix Output A Ramp 0**

This bit gives the state of the PSCOUT20 (and/or PSCOUT22) during ramp 0

16.26.2 PSC0 Interrupt Mask Register – PIM0

Bit	7	6	5	4	3	2	1	0	
	-	-	PSEIE0	PEVE0B	PEVE0A	-	-	PEOPE0	PIM0
Read/Write	R	R	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

16.26.3 PSC2 Interrupt Mask Register – PIM2

Bit	7	6	5	4	3	2	1	0	
	-	-	PSEIE2	PEVE2B	PEVE2A	-	-	PEOPE2	PIM2
Read/Write	R	R	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5 – PSEIEn : PSC n Synchro Error Interrupt Enable**

When this bit is set, the PSEIn bit (if set) generate an interrupt.

- **Bit 4 – PEVEnB : PSC n External Event B Interrupt Enable**

When this bit is set, an external event which can generates a capture from Retrigger/Fault block B generates also an interrupt.

- **Bit 3 – PEVEnA : PSC n External Event A Interrupt Enable**

When this bit is set, an external event which can generates a capture from Retrigger/Fault block A generates also an interrupt.

- **Bit 0 – PEOPEn : PSC n End Of Cycle Interrupt Enable**

When this bit is set, an interrupt is generated when PSC reaches the end of the whole cycle.

16.26.4 PSC0 Interrupt Flag Register – PIFR0

Bit	7	6	5	4	3	2	1	0	
	POAC0B	POAC0A	PSEI0	PEV0B	PEV0A	PRN01	PRN00	PEOP2	PIFR0
Read/Write	R	R	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

16.26.5 PSC2 Interrupt Flag Register – PIFR2

Bit	7	6	5	4	3	2	1	0	
	POAC2B	POAC2A	PSEI2	PEV2B	PEV2A	PRN21	PRN20	PEOP2	PIFR2
Read/Write	R	R	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – POACnB : PSC n Output B Activity** (not implemented on AT90PWM1)

This bit is set by hardware each time the output PSCOUTn1 changes from 0 to 1 or from 1 to 0.

Must be cleared by software by writing a one to its location.

This feature is useful to detect that a PSC output doesn't change due to a freezen external input signal.

- **Bit 6 – POACnA : PSC n Output A Activity** (not implemented on AT90PWM1)

This bit is set by hardware each time the output PSCOUTn0 changes from 0 to 1 or from 1 to 0.

Must be cleared by software by writing a one to its location.

This feature is useful to detect that a PSC output doesn't change due to a freezen external input signal.

- **Bit 5 – PSEIn : PSC n Synchro Error Interrupt**

This bit is set by hardware when the update (or end of PSC cycle) of the PSCn configured in auto run (PARUNn = 1) does not occur at the same time than the PSCn-1 which has generated the input run signal. (For PSC0, PSCn-1 is PSC2).

Must be cleared by software by writing a one to its location.

This feature is useful to detect that a PSC doesn't run at the same speed or with the same phase than the PSC master.

- **Bit 4 – PEVnB : PSC n External Event B Interrupt**

This bit is set by hardware when an external event which can generates a capture or a retrigger from Retrigger/Fault block B occurs.

Must be cleared by software by writing a one to its location.

This bit can be read even if the corresponding interrupt is not enabled (PEVEnB bit = 0).

- **Bit 3 – PEVnA : PSC n External Event A Interrupt**

This bit is set by hardware when an external event which can generates a capture or a retrigger from Retrigger/Fault block A occurs.

Must be cleared by software by writing a one to its location.

This bit can be read even if the corresponding interrupt is not enabled (PEVEnA bit = 0).

- **Bit 2:1 – PRNn1:0 : PSC n Ramp Number**

Memorization of the ramp number when the last PEVnA or PEVnB occured**.**

Table 50. PSC n Ramp Number Description

PRNn1	PRNn0	Description
0	0	The last event which has generated an interrupt occured during ramp 1
0	1	The last event which has generated an interrupt occured during ramp 2
1	0	The last event which has generated an interrupt occured during ramp 3
1	1	The last event which has generated an interrupt occured during ramp 4

- **Bit 0 – PEOPn: End Of PSC n Interrupt**

This bit is set by hardware when PSC n achieves its whole cycle.

Must be cleared by software by writing a one to its location.

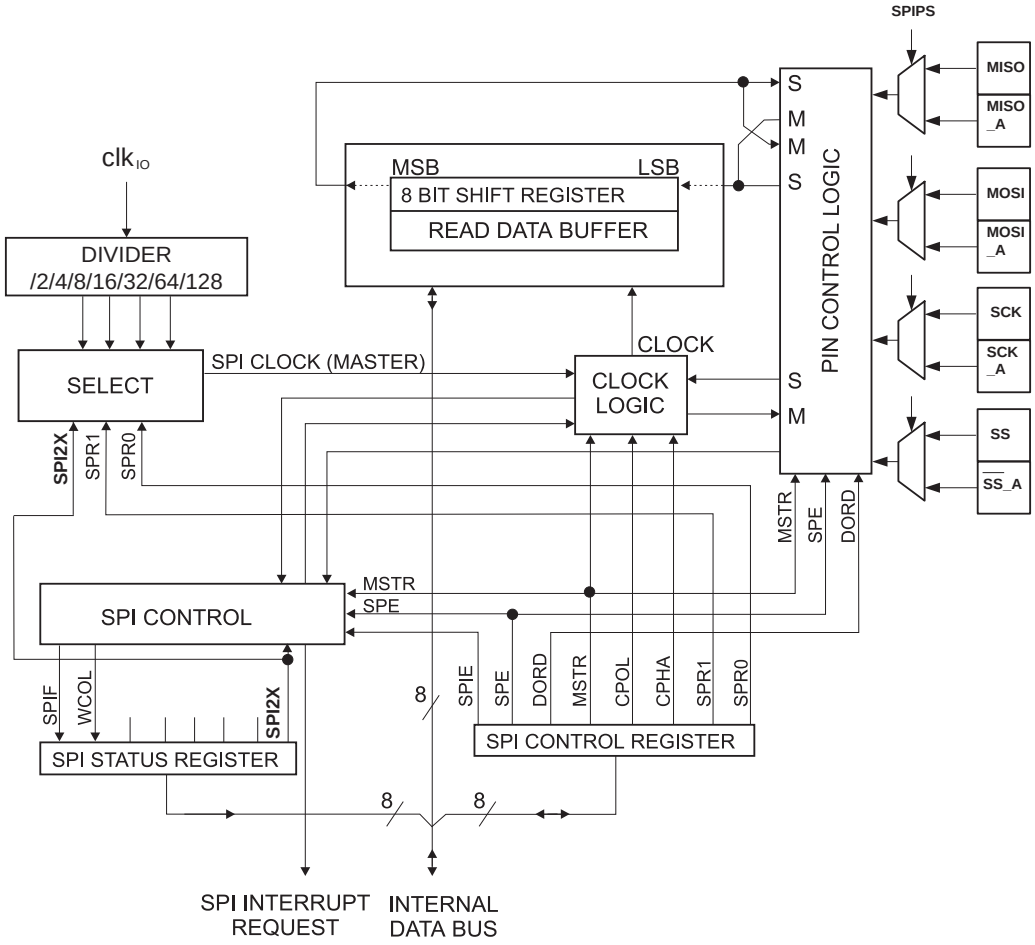
17. Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT90PWM1 and peripheral devices or between several AVR devices.
The AT90PWM1 SPI includes the following features:

17.1 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**

Figure 17-1. SPI Block Diagram(1)



Note: 1. Refer to Figure 3-1 on page 2, and Table 9 on page 65 for SPI pin placement.

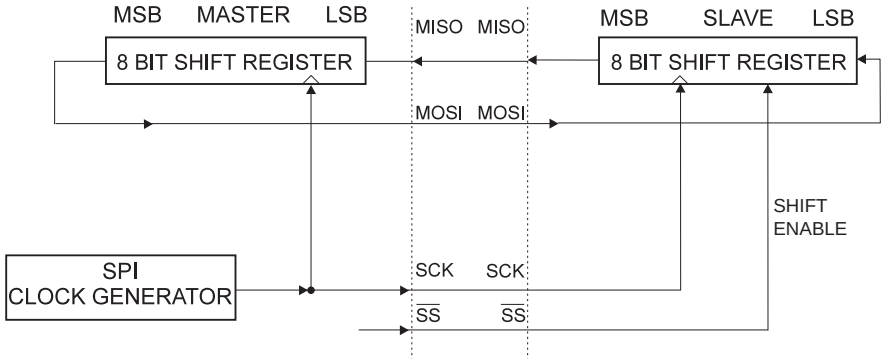
The interconnection between Master and Slave CPUs with SPI is shown in Figure 17-2. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare 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 $\overline{SS}$ 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 17-2. 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 frequency of the SPI clock should never exceed $f_{clkio}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 51. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 63.

Table 51. SPI Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input

Table 51. SPI Pin Overrides[1]

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

Note: 1. See “Alternate Functions of Port B” on page 65 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 PB2, replace DD_MOSI with DDB2 and DDR_SPI with DDRB.

TABLE 2.

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. The example code assumes that the part specific header file is included.

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

TABLE 2.

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. The example code assumes that the part specific header file is included.

17.2 $\overline{SS}$ Pin Functionality

17.2.1 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.

17.2.2 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.

17.2.3 MCU Control Register – MCUCR

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

- **Bit 7– SPIPS: SPI Pin Redirection**

Thanks to SPIPS (SPI Pin Select) in MCUCR Sfr, SPI pins can be redirected.

On 32 pins packages, SPIPS has the following action:

- When the SPIPS bit is written to zero, the SPI signals are directed on pins MISO,MOSI, SCK and SS.
- When the SPIPS bit is written to one,the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A.

On 24 pins package, SPIPS has the following action:

- When the SPIPS bit is written to zero, the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A.

– When the SPIPS bit is written to one,the SPI signals are directed on pins MISO,MOSI, SCK and SS.

Note that programming port are always located on alternate SPI port.

17.2.4 SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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 – 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.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to 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 $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become 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 Figure 17-3 and Figure 17-4 for an example. The CPOL functionality is summarized below:

Table 52. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **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 Figure 17-3 and Figure 17-4 for an example. The CPOL functionality is summarized below:

Table 53. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: 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 clk_{IO} frequency f_{clkio} is shown in the following table:

Table 54. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{clkio}/4$
0	0	1	$f_{clkio}/16$
0	1	0	$f_{clkio}/64$
0	1	1	$f_{clkio}/128$
1	0	0	$f_{clkio}/2$
1	0	1	$f_{clkio}/8$
1	1	0	$f_{clkio}/32$
1	1	1	$f_{clkio}/64$

17.2.5 SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
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. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF flag. 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, 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 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM1 and will always read as zero.

- **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 (see Table 54). 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 $f_{clkio}/4$ or lower.

The SPI interface on the AT90PWM1 is also used for program memory and EEPROM down-loading or uploading. See Serial Programming Algorithm234 for serial programming and verification.

17.2.6 SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

- **Bits 7:0 - SPD7:0: SPI Data**

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.

17.3 Data Modes

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 17-3 and Figure 17-4. 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 by summarizing Table 52 and Table 53, as done below:

Table 55. 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 17-3. SPI Transfer Format with CPHA = 0

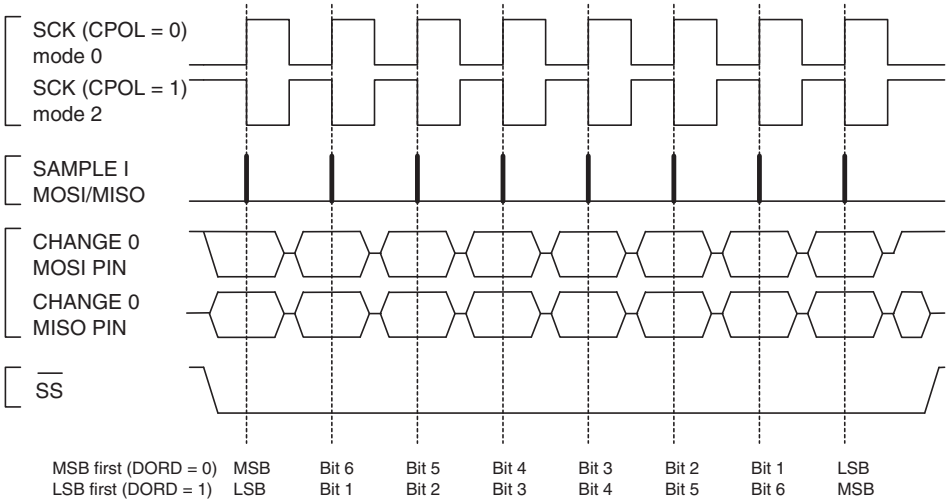
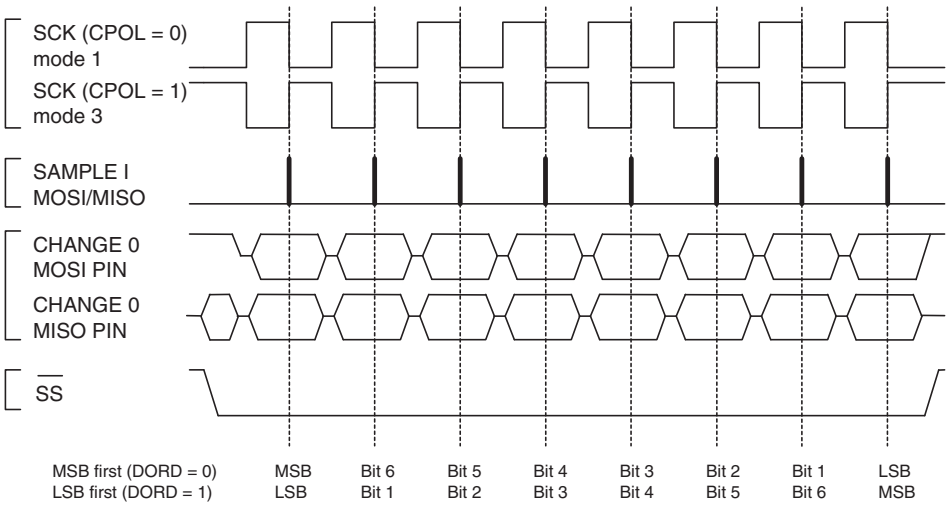


Figure 17-4. SPI Transfer Format with CPHA = 1



18. Analog Comparator

The Analog Comparator compares the input values on the positive pin ACMPx and negative pin ACMPM.

18.1 Overview

The AT90PWM1 features three fast analog comparators.

Each comparator has a dedicated input on the positive input, and the negative input can be configured as:

- a steady value among the 4 internal reference levels defined by the Vref selected thanks to the REFS1:0 bits in ADMUX register.
- a value generated from the internal DAC
- an external analog input ACMPM.

When the voltage on the positive ACMPn pin is higher than the voltage selected by the ACnM multiplexer on the negative input, the Analog Comparator output, ACnO, is set.

The comparator is a clocked comparator. A new comparison is done on the falling edge of $CLK_{I/O}$ or $CLK_{I/O}/2$ (Depending on ACCKDIV fit of ACSR register, See “Analog Comparator Status Register – ACSR” on page 179.).

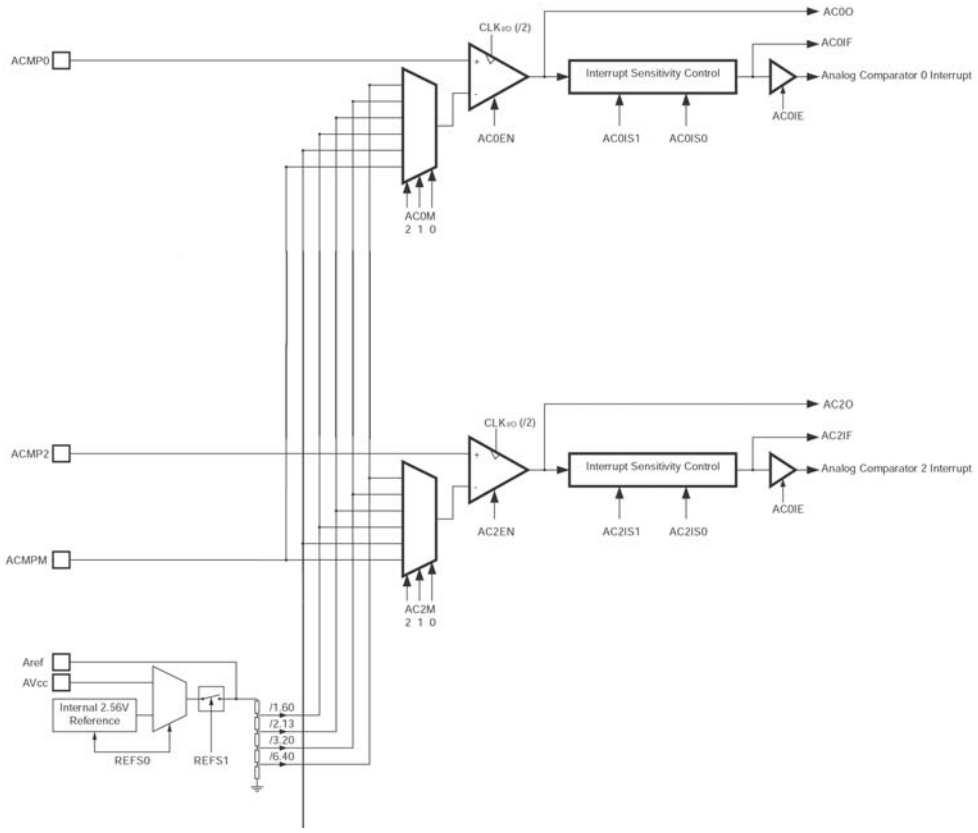
Each comparator can trigger a separate interrupt, exclusive to the Analog Comparator. In addition, the user can select Interrupt triggering on comparator output rise, fall or toggle.

The interrupt flags can also be used to synchronize ADC or DAC conversions.

Moreover, the comparator’s output of the comparator 1 can be set to trigger the Timer/Counter1 Input Capture function.

A block diagram of the three comparators and their surrounding logic is shown in Figure 18-1.

Figure 18-1. Analog Comparator Block Diagram[1][2]



Notes:
1. ADC multiplexer output: see Table 63 on page 195.
2. Refer to Figure 3-1 on page 2 and for Analog Comparator pin placement.
3. The voltage on Vref is defined in 62 ”ADC Voltage Reference Selection” on page 194

18.2 Analog Comparator Register Description

Each analog comparator has its own control register.

A dedicated register has been designed to consign the outputs and the flags of the 3 analog comparators.

18.2.1 Analog Comparator 0 Control Register – AC0CON

Bit	7	6	5	4	3	2	1	0	
	AC0EN	**AC0IE**	**AC0IS1**	**AC0IS0**	**-**	**AC0M2**	**AC0M1**	**AC0M0**	**AC0CON**
Read/Write	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– AC0EN: Analog Comparator 0 Enable Bit**

Set this bit to enable the analog comparator 0.
Clear this bit to disable the analog comparator 0.

- **Bit 6– AC0IE: Analog Comparator 0 Interrupt Enable bit**

Set this bit to enable the analog comparator 0 interrupt.
Clear this bit to disable the analog comparator 0 interrupt.

- **Bit 5, 4– AC0IS1, AC0IS0: Analog Comparator 0 Interrupt Select bit**

These 2 bits determine the sensitivity of the interrupt trigger.
The different setting are shown in Table 56.

Table 56. Interrupt sensitivity selection

AC0IS1	AC0IS0	Description
0	0	Comparator Interrupt on output toggle
0	1	Reserved
1	0	Comparator interrupt on output falling edge
1	1	Comparator interrupt on output rising edge

- **Bit 2, 1, 0– AC0M2, AC0M1, AC0M0: Analog Comparator 0 Multiplexer register**

These 3 bits determine the input of the negative input of the analog comparator.
The different setting are shown in Table 57.

Table 57. Analog Comparator 0 negative input selection

AC0M2	AC0M1	AC0M0	Description
0	0	0	“Vref”/6.40
0	0	1	“Vref”/3.20
0	1	0	“Vref”/2.13
0	1	1	“Vref”/1.60
1	0	0	Analog Comparator Negative Input (ACMPM pin)
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

18.2.2 Analog Comparator 2 Control Register – AC2CON

Bit	7	6	5	4	3	2	1	0	
	AC2EN	AC2IE	AC2IS1	AC2IS0		AC2M2	AC2M1	AC2M0	AC2CON
Read/Write	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– AC2EN: Analog Comparator 2 Enable Bit**

Set this bit to enable the analog comparator 2.
Clear this bit to disable the analog comparator 2.

- **Bit 6– AC2IE: Analog Comparator 2 Interrupt Enable bit**

Set this bit to enable the analog comparator 2 interrupt.
Clear this bit to disable the analog comparator 2 interrupt.

- **Bit 5, 4– AC2IS1, AC2IS0: Analog Comparator 2 Interrupt Select bit**

These 2 bits determine the sensitivity of the interrupt trigger.
The different setting are shown in Table 56.

Table 58. Interrupt sensitivity selection

AC2IS1	AC2IS0	Description
0	0	Comparator Interrupt on output toggle
0	1	Reserved
1	0	Comparator interrupt on output falling edge
1	1	Comparator interrupt on output rising edge

- **Bit 2, 1, 0– AC2M2, AC2M1, AC2M0: Analog Comparator 2 Multiplexer register**

These 3 bits determine the input of the negative input of the analog comparator.
The different setting are shown in Table 59.

Table 59. Analog Comparator 2 negative input selection

AC2M2	AC2M1	AC2M0	Description
0	0	0	“Vref”/6.40
0	0	1	“Vref”/3.20
0	1	0	“Vref”/2.13
0	1	1	“Vref”/1.60
1	0	0	Analog Comparator Negative Input (ACMPM pin)
1	0	1	DAC result
1	1	0	Reserved
1	1	1	Reserved

18.2.3 Analog Comparator Status Register – ACSR

Bit	7	6		4	3	2		0	
	ACCKDIV	AC2IF		AC0IF	-	AC2O		AC0O	ACSR
Read/Write	R/W	R/W		R/W	-	R		R	
Initial Value	0	0		0	0	0		0	

- **Bit 7– ACCKDIV: Analog Comparator Clock Divider**

The analog comparators can work with a clock up to 8MHz@3V and 16MHz@5V.
Set this bit in case the clock frequency of the microcontroller is higher than 8 MHz to insert a divider by 2 between the clock of the microcontroller and the clock of the analog comparators.
Clear this bit to have the same clock frequency for the microcontroller and the analog comparators.

- **Bit 6– AC2IF: Analog Comparator 2 Interrupt Flag Bit**

This bit is set by hardware when comparator 2 output event triggers off the interrupt mode defined by AC2IS1 and AC2IS0 bits in AC2CON register.
This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC2IE in AC2CON register is set. Anyway, this bit is cleared by writing a logical one on it.
This bit can also be used to synchronize ADC or DAC conversions.

- **Bit 5– AC0IF: Analog Comparator 0 Interrupt Flag Bit**

This bit is set by hardware when comparator 0 output event triggers off the interrupt mode defined by AC0IS1 and AC0IS0 bits in AC0CON register.

This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC0IE in AC0CON register is set. Anyway, this bit is cleared by writing a logical one on it. This bit can also be used to synchronize ADC or DAC conversions.

- **Bit 2– AC2O: Analog Comparator 2 Output Bit**

AC2O bit is directly the output of the Analog comparator 2.
Set when the output of the comparator is high.
Cleared when the output comparator is low.

- **Bit 0– AC0O: Analog Comparator 0 Output Bit**

AC0O bit is directly the output of the Analog comparator 0.
Set when the output of the comparator is high.
Cleared when the output comparator is low.

18.2.4 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D ACMPM	ADC2D ACMP2D	ADC1D	ADC0D	DIDR0
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 3:2 – ACMPM and ACMP2D: ACMPM and ACMP2 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding Analog 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 one of these pins 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.

18.2.5 Digital Input Disable Register 1– DIDR1

Bit	7	6	5	4	3	2	1	0	
	-	-	ACMP0D	AMP0PD	AMP0ND				DIDR1
Read/Write	-	-	R/W	R/W	R/W				
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5, 2: ACMP0D and ACMP1 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding analog 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 one of these pins 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.

19. Analog to Digital Converter - ADC

19.1 Features

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **± 2 LSB Absolute Accuracy**
- **8- 250 µs Conversion Time**
- **Up to 120 kSPS at Maximum Resolution**
- **11 Multiplexed Single Ended Input Channels**
- **Two Differential input channels with accurate (5%) programmable gain 5, 10, 20 and 40**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{CC} ADC Input Voltage Range**
- **Selectable 2.56 V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The AT90PWM1 features a 10-bit successive approximation ADC. The ADC is connected to an 15-channel Analog Multiplexer which allows eleven single-ended input. The single-ended voltage inputs refer to 0V (GND).

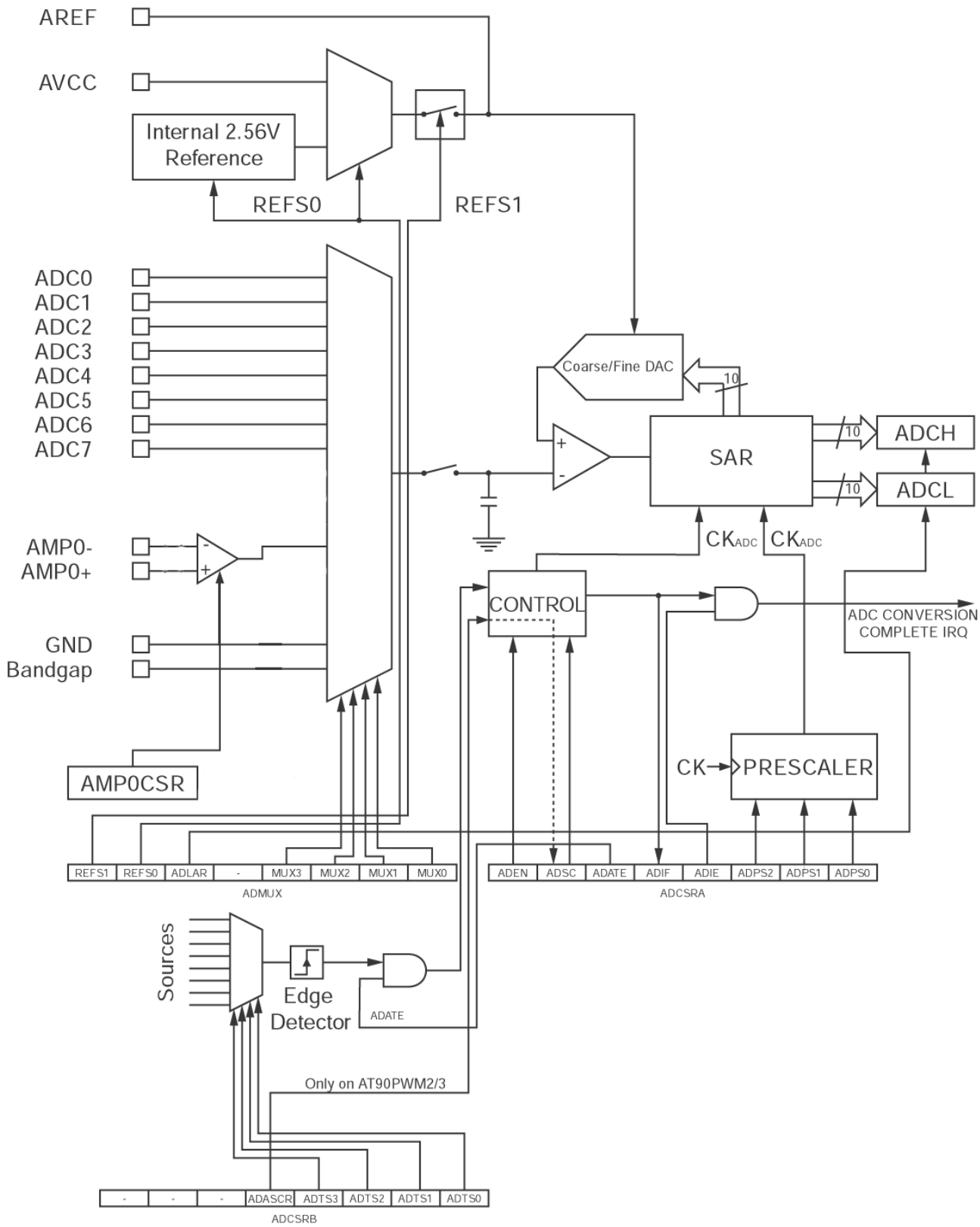
The device also supports 2 differential voltage input combinations which are equipped with a programmable gain stage, providing amplification steps of 14dB (5x), 20 dB (10x), 26 dB (20x), or 32dB (40x) on the differential input voltage before the A/D conversion. On the amplified channels, 8-bit resolution 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 19-1.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ± 0.3V from V_{CC}. See the paragraph “ADC Noise Canceler” on page 188 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Figure 19-1. Analog to Digital Converter Block Schematic



19.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AV_{CC} or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference is set by the REFS1 and REFS0 bits in ADMUX register, whatever the ADC is enabled or not. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off 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 completed 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.

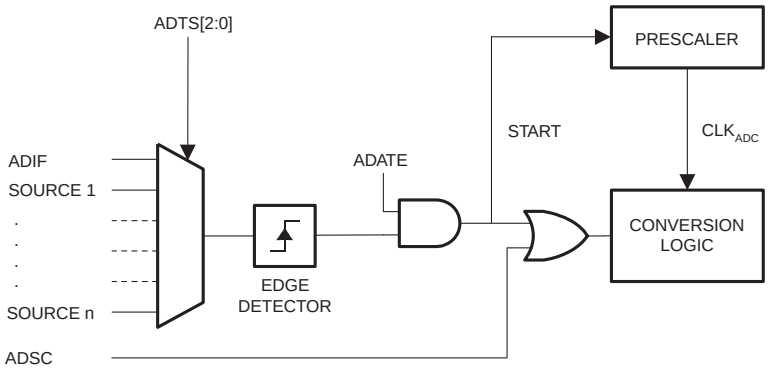
The ADC has its own interrupt which can be triggered when a conversion completes. The ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

19.3 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 is still 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 19-2. ADC Auto Trigger Logic

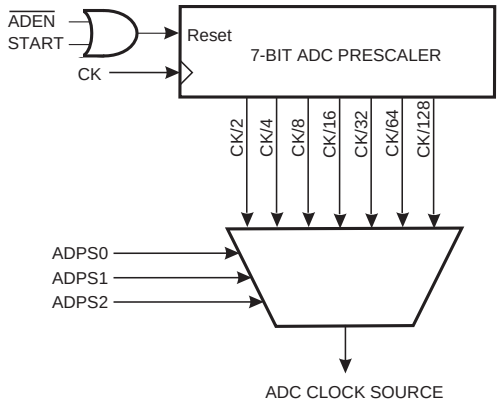


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the 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. The free running mode is not allowed on the amplified channels.

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

19.4 Prescaling and Conversion Timing

Figure 19-3. ADC Prescaler



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

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

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See “Changing Channel or Reference Selection” on page 186 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

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

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

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 60.

Figure 19-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

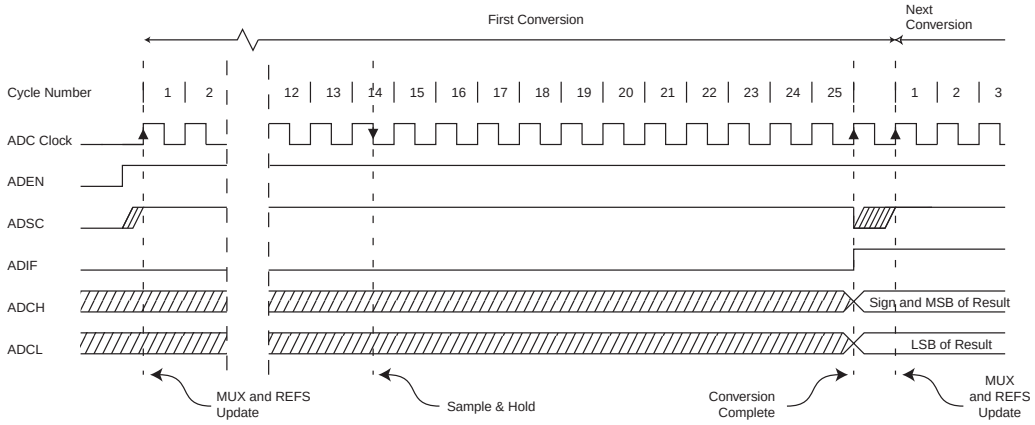


Figure 19-5. ADC Timing Diagram, Single Conversion

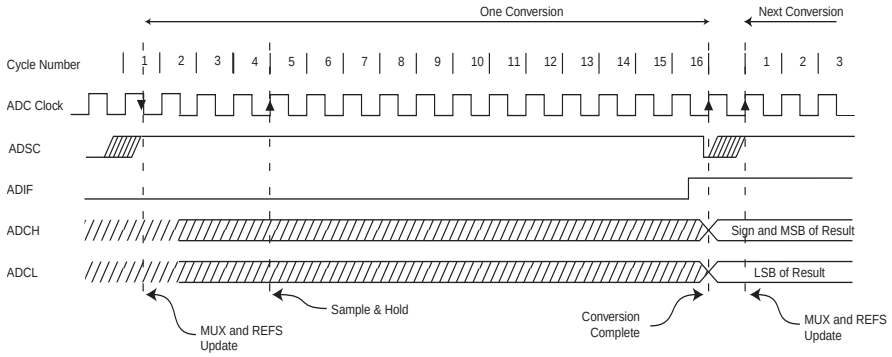


Figure 19-6. ADC Timing Diagram, Auto Triggered Conversion

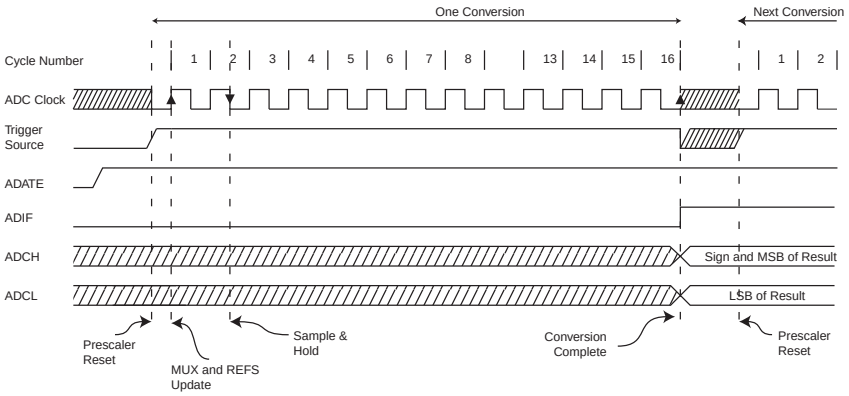


Figure 19-7. ADC Timing Diagram, Free Running Conversion

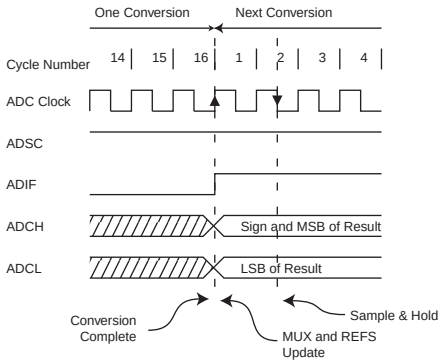


Table 60. ADC Conversion Time

Condition	First Conversion	Normal Conversion, Single Ended	Auto Triggered Conversion
Sample & Hold (Cycles from Start of Conversion)	13.5	3.5	2
Conversion Time (Cycles)	25	15.5	16

19.5 Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX 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 until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If Auto Triggering is used, the exact time of the triggering event can be indeterministic. 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 is 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 conversion, minimum one ADC clock cycle after the trigger event.
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 ADC conversion.

In order to start a conversion on an amplified channel, there is a dedicated ADASCR bit in ADCSRB register which wait for the next amplifier trigger event before really starting the conversion by an hardware setting of the ADSC bit in ADCSRA register.

19.5.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

- In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.
- In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.
- In Free Running mode, because the amplifier clear the ADSC bit at the end of an amplified conversion, it is not possible to use the free running mode, unless ADSC bit is set again by soft at the end of each conversion.

19.5.2 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 in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedant voltmeter. Note that V_{REF} is a high impedant source, and only a capacitive load should be connected in a system.

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. If no external voltage is applied to the AREF pin, the user may switch between AV_{CC} and 2.56V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

If differential channels are used, the selected reference should not be closer to AV_{CC} than indicated in Table 23-5 on page 245.

19.6 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

a. Make sure that the ADATE bit is reset.

b. 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.

c. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.

d. If no other interrupts occur before the ADC 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 ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC 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.

If the ADC is enabled in such sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

19.6.1 Analog Input Circuitry

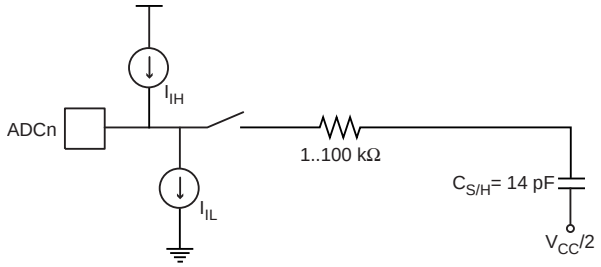
The analog input circuitry for single ended channels is illustrated in Figure 19-8. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred kΩ or less is recommended.

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.

Figure 19-8. Analog Input Circuitry

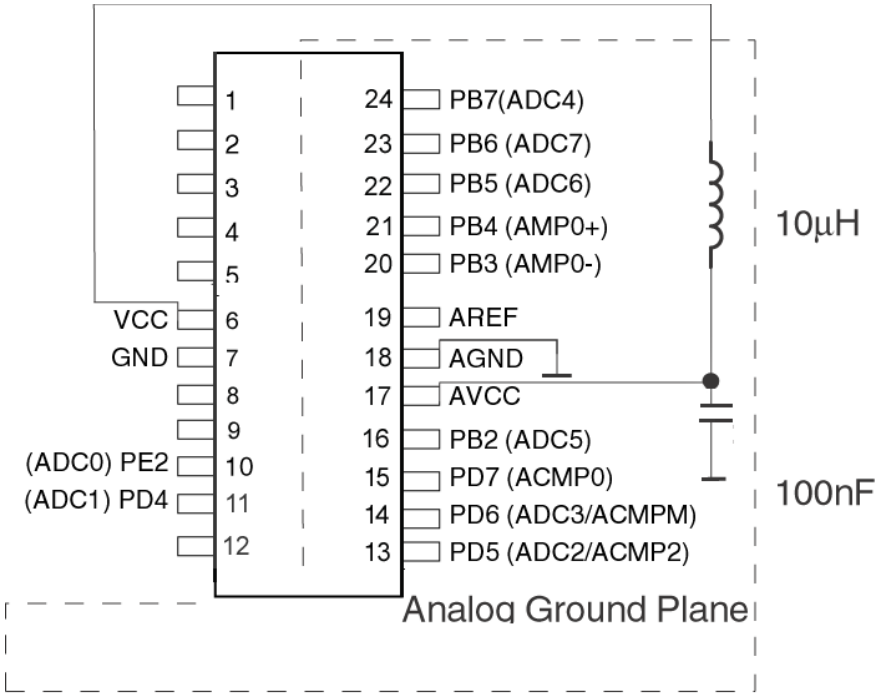


19.6.2 Analog Noise Canceling Techniques

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 analog ground plane, and keep them well away from high-speed switching digital tracks.
2. The AV_{CC} pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 19-9.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 19-9. ADC Power Connections



19.6.3 Offset Compensation Schemes

The gain stage 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 shortening both differential inputs using the AMPxIS bit with both inputs unconnected. (See “Amplifier 0 Control and Status register – AMP0CSR” on page 202.). This offset residue can be then subtracted in software from the measurement results. Using this kind of software based offset correction, offset on any channel can be reduced below one LSB.

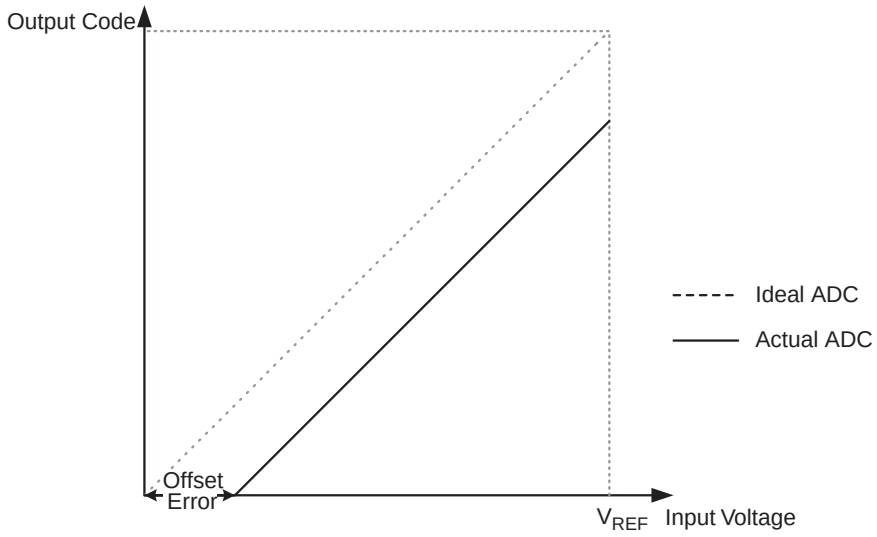
19.6.4 ADC Accuracy Definitions

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:

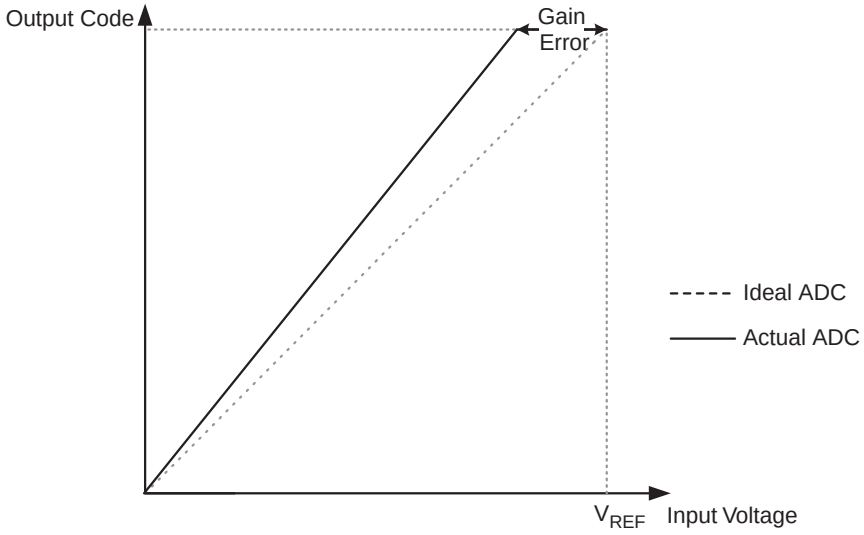
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 19-10. Offset Error



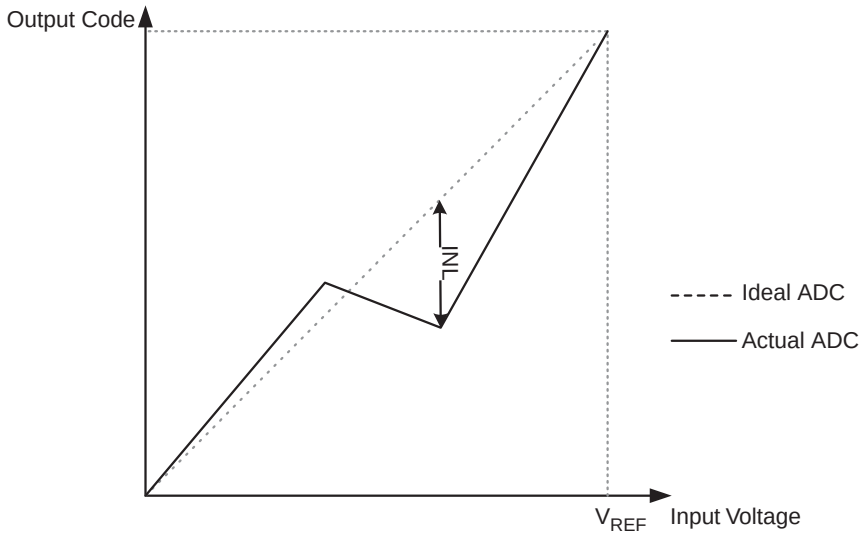
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 19-11. Gain Error



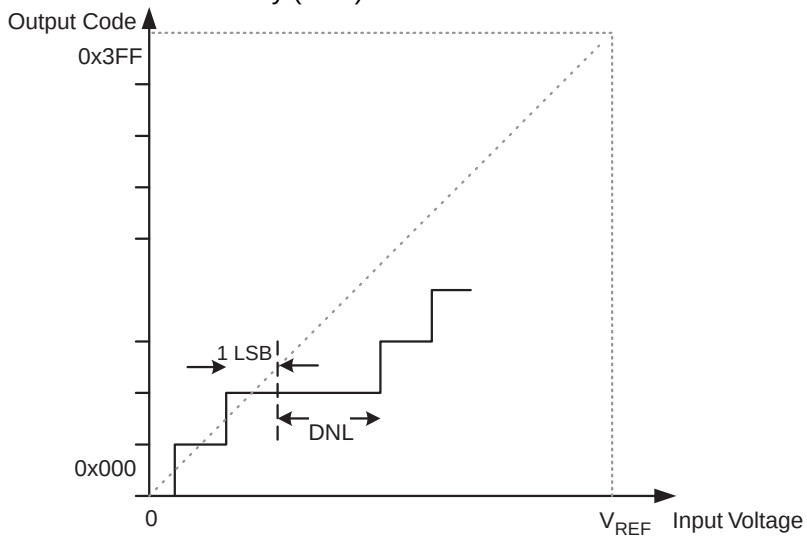
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 19-12. 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 19-13. 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. 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.

19.7 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:

$$ADC = \frac{V_{IN} \cdot 1023}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 62 on page 194 and Table 63 on page 195). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage.

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, GAIN the selected gain factor 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 19-14 shows the decoding of the differential input range.

Table 82 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a reference voltage of V_{REF}.

Figure 19-14. Differential Measurement Range

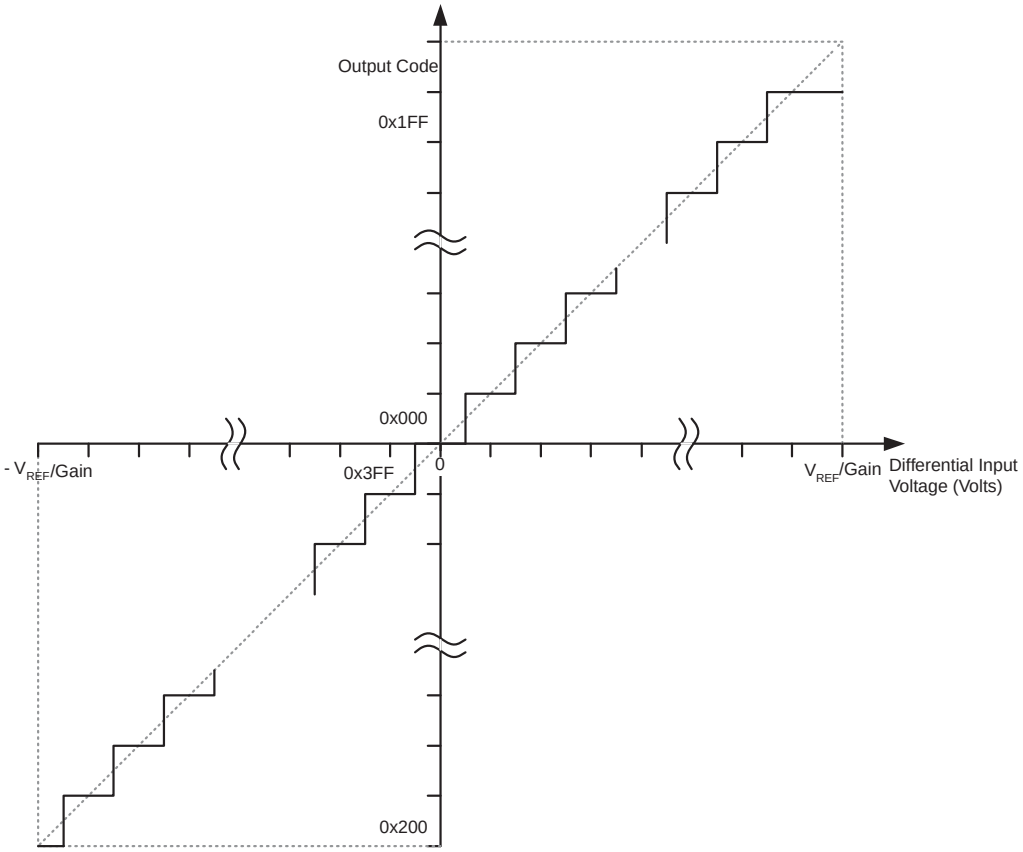


Table 61. Correlation Between Input Voltage and Output Codes

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 1:

– ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)
– Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.
– ADCR = 512 * 10 * (300 - 500) / 2560 = -400 = 0x270

– ADCL will thus read 0x00, and ADCH will read 0x9C.
Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

Example 2:

– ADMUX = 0xFB (ADC3 - ADC2, 1x gain, 2.56V reference, left adjusted result)
– Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.
– ADCR = 512 * 1 * (300 - 500) / 2560 = -41 = 0x029.
– ADCL will thus read 0x40, and ADCH will read 0x0A.
Writing zero to ADLAR right adjusts the result: ADCL = 0x00, ADCH = 0x29.

19.8 ADC Register Description

The ADC of the AT90PWM1 is controlled through 3 different registers. The ADCSRA and The ADCSRB registers which are the ADC Control and Status registers, and the ADMUX which allows to select the Vref source and the channel to be converted.

The conversion result is stored on ADCH and ADCL register which contain respectively the most significant bits and the less significant bits.

19.8.1 ADC Multiplexer Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	**REFS0**	**ADLAR**	**-**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	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, 6 – REFS1, 0: ADC Vref Selection Bits**

These 2 bits determine the voltage reference for the ADC.
The different setting are shown in Table 62.

Table 62. ADC Voltage Reference Selection

REFS1	REFS0	Description
0	0	External Vref on AREF pin, Internal Vref is switched off
0	1	AVcc with external capacitor connected on the AREF pin
1	0	Reserved
1	1	Internal 2.56V Reference voltage with external capacitor connected on the AREF pin

If these bits are changed during a conversion, the change will not take effect until this conversion is complete (it means while the ADIF bit in ADCSRA register is set).
In case the internal Vref is selected, it is turned ON as soon as an analog feature needed it is set.

- **Bit 5 – ADLAR: ADC Left Adjust Result**

Set this bit to left adjust the ADC result.
Clear it to right adjust the ADC result.
The ADLAR bit affects the configuration of the ADC result data registers. Changing this bit affects the ADC data registers immediately regardless of any on going conversion. For a complete description of this bit, see Section “ADC Result Data Registers – ADCH and ADCL”, page 198.

- **Bit 3, 2, 1, 0 – MUX3, MUX2, MUX1, MUX0: ADC Channel Selection Bits**

These 4 bits determine which analog inputs are connected to the ADC input. The different setting are shown in Table 63.

Table 63. ADC Input Channel Selection

MUX3	MUX2	MUX1	MUX0	Description
0	0	0	0	ADC0
0	0	0	1	ADC1
0	0	1	0	ADC2
0	0	1	1	ADC3
0	1	0	0	ADC4
0	1	0	1	ADC5
0	1	1	0	ADC6
0	1	1	1	ADC7
1	0	0	0	Reserved
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	AMP0
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Bandgap
1	1	1	1	GND

If these bits are changed during a conversion, the change will not take effect until this conversion is complete (it means while the ADIF bit in ADCSRA register is set).

19.8.2 ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable Bit**

Set this bit to enable the ADC.
Clear this bit to disable the ADC.
Clearing this bit while a conversion is running will take effect at the end of the conversion.

- **Bit 6– ADSC: ADC Start Conversion Bit**

Set this bit to start a conversion in single conversion mode or to start the first conversion in free running mode.
Cleared by hardware when the conversion is complete. Writing this bit to zero has no effect.
The first conversion performs the initialization of the ADC.

In order to start a conversion on an amplified channel with the AT90PWM1, use the ADCS bit in ADCSRA register.

- **Bit 5 – ADATE: ADC Auto trigger Enable Bit**

Set this bit to enable the auto triggering mode of the ADC.
Clear it to return in single conversion mode.
In auto trigger mode the trigger source is selected by the ADTS bits in the ADCSRB register. See Table 19-1 on page 197.

- **Bit 4– ADIF: ADC Interrupt Flag**

Set by hardware as soon as a conversion is complete and the Data register are updated with the conversion result.
Cleared by hardware when executing the corresponding interrupt handling vector.
Alternatively, ADIF can be cleared by writing it to logical one.

- **Bit 3– ADIE: ADC Interrupt Enable Bit**

Set this bit to activate the ADC end of conversion interrupt.
Clear it to disable the ADC end of conversion interrupt.

- **Bit 2, 1, 0– ADPS2, ADPS1, ADPS0: ADC Prescaler Selection Bits**

These 3 bits determine the division factor between the system clock frequency and input clock of the ADC.
The different setting are shown in Table 64.

Table 64. ADC Prescaler Selection

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

19.8.3 ADC Control and Status Register B– ADCSRB

Bit	7	6	5	4	3	2	1	0	
	ADHSM	-	-	ADASCR	ADTS3	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADHSM: ADC High Speed Mode**

Writing this bit to one enables the ADC High Speed mode. Set this bit if you wish to convert with an ADC clock frequency higher than 200KHz.

- **Bit 3, 2, 1, 0– ADTS3:ADTS0: ADC Auto Trigger Source Selection Bits**

These bits are only necessary in case the ADC works in auto trigger mode. It means if ADATE bit in ADCSRA register is set.

In accordance with the Table 19-1, these 3 bits select the interrupt event which will generate the trigger of the start of conversion. The start of conversion will be generated by the rising edge of the selected interrupt flag whether the interrupt is enabled or not. In case of trig on PSCnASY

event, there is no flag. So in this case a conversion will start each time the trig event appears and the previous conversion is completed

Table 19-1. ADC Auto Trigger Source Selection **for non amplified conversions**

ADTS3	ADTS2	ADTS1	ADTS0	Description
0	0	0	0	Free Running Mode
0	0	0	1	Analog Comparator 0
0	0	1	0	External Interrupt Request 0
0	0	1	1	Timer/Counter0 Compare Match
0	1	0	0	Timer/Counter0 Overflow
0	1	0	1	Timer/Counter1 Compare Match B
0	1	1	0	Timer/Counter1 Overflow
0	1	1	1	Timer/Counter1 Capture Event
1	0	0	0	PSC0ASY Event (1)
1	0	0	1	Reserved
1	0	1	0	PSC2ASY Event(1)
1	0	1	1	Analog comparator 1
1	1	0	0	Analog comparator 2
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

1. For trigger on any PSC event, if the PSC uses the PLL clock, the core must use PLL/4 clock source.

Table 19-2. ADC Auto Trigger Source Selection **for amplified conversions**

ADTS3	ADTS2	ADTS1	ADTS0	Description
0	0	0	0	Free Running Mode
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	Reserved
0	1	0	0	Reserved
0	1	0	1	Reserved
0	1	1	0	Reserved
0	1	1	1	Reserved
1	0	0	0	PSC0ASY Event (1)
1	0	0	1	Reserved
1	0	1	0	PSC2ASY Event(1)
1	0	1	1	Reserved

Table 19-2. ADC Auto Trigger Source Selection **for amplified conversions**

ADTS3	ADTS2	ADTS1	ADTS0	Description
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

1. For trigger on any PSC event, if the PSC uses the PLL clock, the core must use PLL/4 clock source.

19.8.4 ADC Result Data Registers – ADCH and ADCL

When an ADC conversion is complete, the conversion results are stored in these two result data registers.

When the ADCL register is read, the two ADC result data registers can't be updated until the ADCH register has also been read.
Consequently, in 10-bit configuration, the ADCL register must be read first before the ADCH.
Nevertheless, to work easily with only 8-bit precision, there is the possibility to left adjust the result thanks to the ADLAR bit in the ADCSRA register. Like this, it is sufficient to only read ADCH to have the conversion result.

19.8.4.1 ADLAR = 0

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
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	

19.8.4.2 ADLAR = 1

Bit	7	6	5	4	3	2	1	0	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
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	

19.8.5 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D ACMPM	ADC2D ACMP2D	ADC1D	ADC0D	DIDR0
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 – ADC7D..ADC0D: ACMP2:1 and ADC7:0 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.

19.8.6 Digital Input Disable Register 1– DIDR1

Bit	7	6	5	4	3	2	1	0	
	-	-	ACMP0D	AMP0PD	AMP0ND				DIDR1
Read/Write	-	-	R/W	R/W	R/W				
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5:0 – ACMP0D, AMP0+D, AMP0-D, ADC10D..ADC8D: ACMP0, AMP1:0 and ADC10:8 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 an analog 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.

19.9 Amplifier

The AT90PWM1 features one differential amplified channel with programmable 5, 10, 20, and 40 gain stage.

On AT90PWM1, the amplifier has been improved in order to speed-up the conversion time.The proposed improvement takes advantage of the amplifier characteristics to ensure a conversion in less time.

In order to have a better understanding of the functioning of the amplifier synchronization, a timing diagram example is shown Figure 19-16 for AT90PWM1.

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

Figure 19-15. Amplifier synchronization timing diagram f
With change on analog input signal

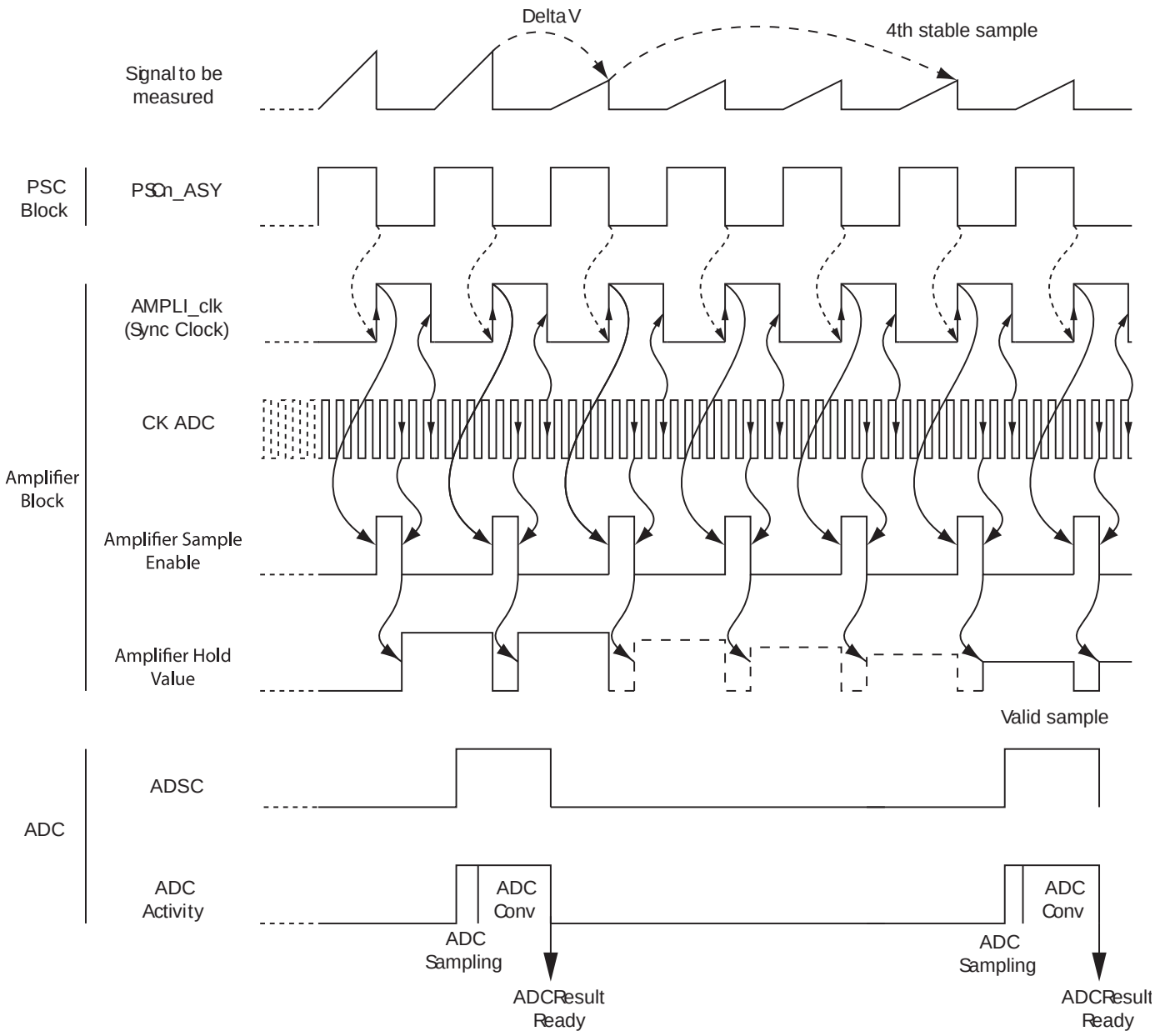
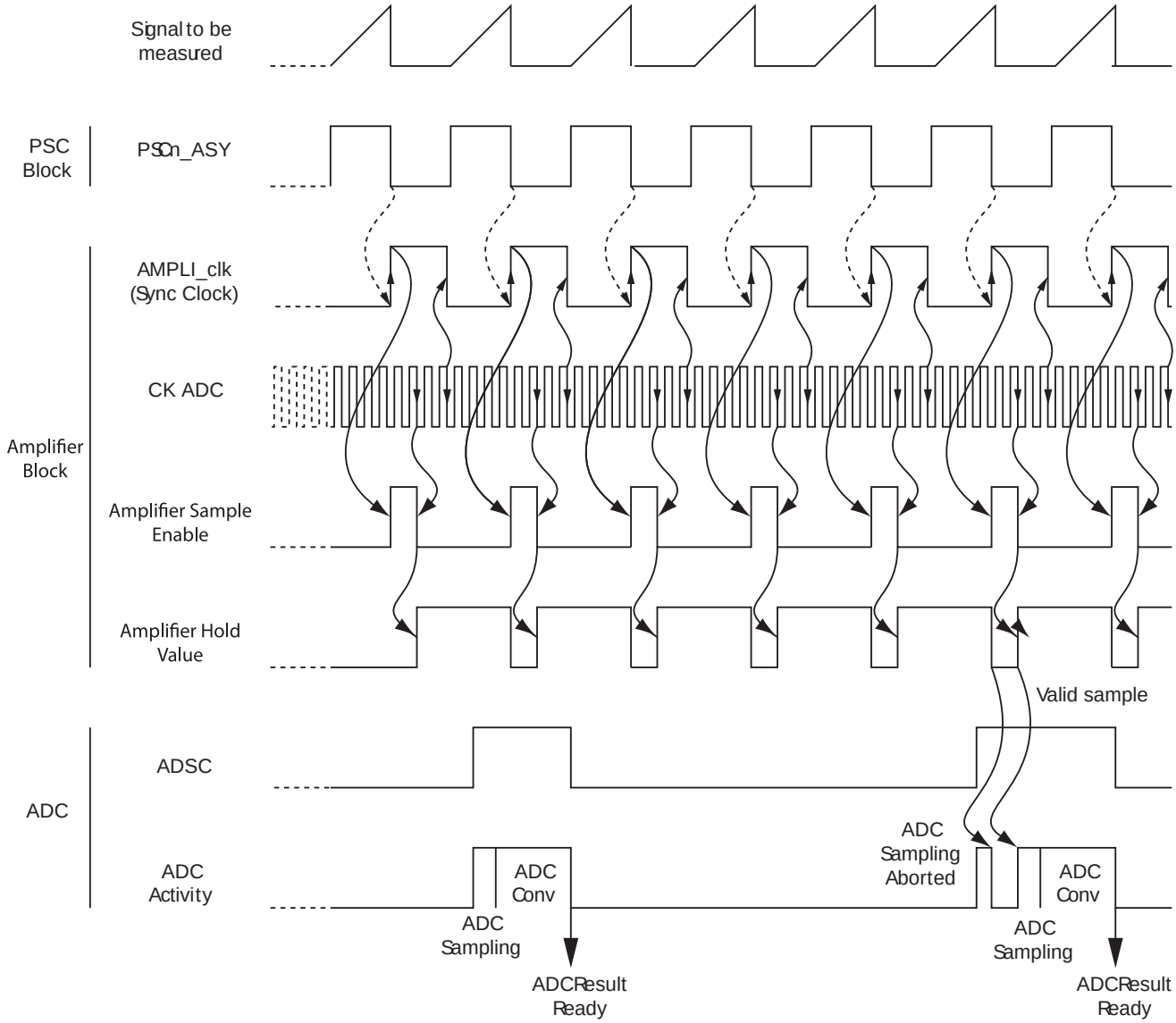
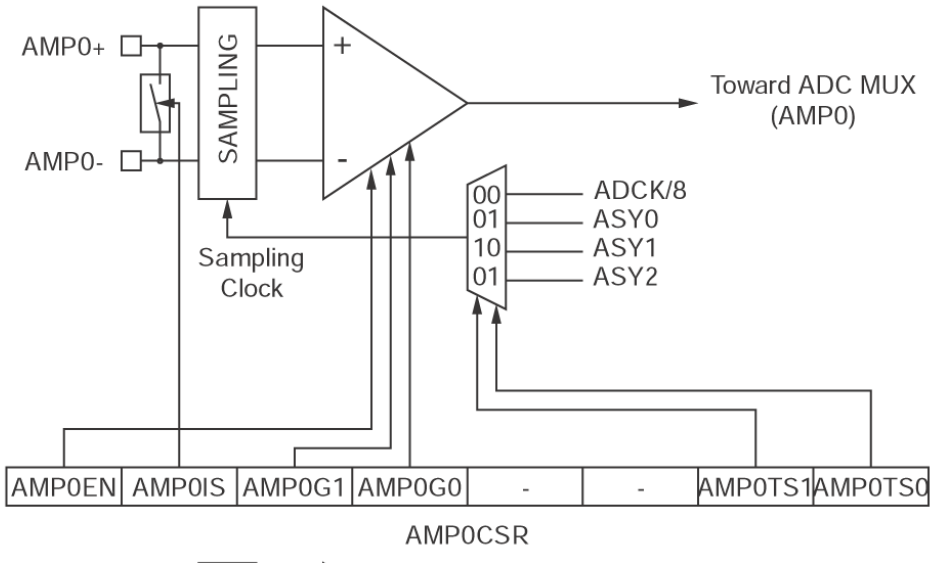


Figure 19-16. Amplifier synchronization timing diagram
ADSC is set when the amplifier output is changing due to the amplifier clock switch.



The block diagram of the amplifier is shown on Figure 19-17.

Figure 19-17. Amplifiers block diagram



19.10 Amplifier Control Registers

The configuration of the amplifiers are controlled via two dedicated registers AMP0CSR and AMP1CSR. Then the start of conversion is done via the ADC control and status registers.

The conversion result is stored on ADCH and ADCL register which contain respectively the most significant bits and the less significant bits.

19.10.1 Amplifier 0 Control and Status register – AMP0CSR

Bit	7	6	5	4	3	2	1	0	
	AMP0EN	**AMP0IS**	**AMP0G1**	**AMP0G0**	**-**	**-**	**AMP0TS1**	**AMP0TS0**	**AMP0CSR**
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – AMP0EN: Amplifier 0 Enable Bit**

Set this bit to enable the Amplifier 0.
Clear this bit to disable the Amplifier 0.
Clearing this bit while a conversion is running will take effect at the end of the conversion.

Warning: Always clear AMP0TS1:0 when clearing AMP0EN.

- **Bit 6– AMP0IS: Amplifier 0 Input Shunt**

Set this bit to short-circuit the Amplifier 0 input.
Clear this bit to normally use the Amplifier 0.

- **Bit 5, 4– AMP0G1, 0: Amplifier 0 Gain Selection Bits**

These 2 bits determine the gain of the amplifier 0.
The different setting are shown in Table 19-3.

Table 19-3. Amplifier 0 Gain Selection

AMP0G1	AMP0G0	Description
0	0	Gain 5
0	1	Gain 10
1	0	Gain 20
1	1	Gain 40

To ensure an accurate result, after the gain value has been changed, the amplifier input needs to have a quite stable input value during at least 4 Amplifier synchronization clock periods.

- **Bit 1, 0– AMP0TS1, AMP0TS0: Amplifier 0 Trigger Source Selection Bits**

In accordance with the Table 19-4, these 2 bits select the event which will generate the trigger for the amplifier 0. This trigger source is necessary to start the conversion on the amplified channel.

Table 19-4. AMP0 Auto Trigger Source Selection

AMP0TS1	AMP0TS0	Description
0	0	Auto synchronization on ADC Clock/8
0	1	Trig on PSC0ASY
1	0	Reserved
1	1	Trig on PSC2ASY

20. debugWIRE On-chip Debug System

20.1 Features

- **Complete Program Flow Control**
- **Emulates All On-chip Functions, Both Digital and Analog, except RESET Pin**
- **Real-time Operation**
- **Symbolic Debugging Support (Both at C and Assembler Source Level, or for Other HLLs)**
- **Unlimited Number of Program Break Points (Using Software Break Points)**
- **Non-intrusive Operation**
- **Electrical Characteristics Identical to Real Device**
- **Automatic Configuration System**
- **High-Speed Operation**
- **Programming of Non-volatile Memories**

20.2 Overview

The debugWIRE On-chip debug system uses a One-wire, bi-directional interface to control the program flow, execute AVR instructions in the CPU and to program the different non-volatile memories.

20.3 Physical Interface

When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 20-1. The debugWIRE Setup

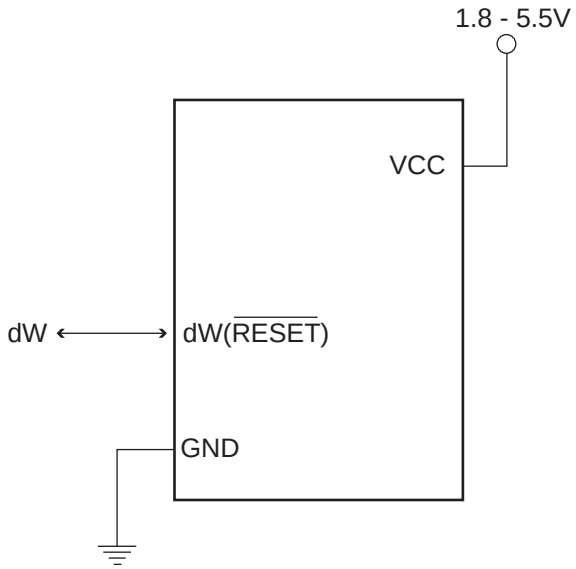


Figure 20-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

20.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio® will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

20.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (AVR Studio).

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

20.6 debugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

20.6.1 debugWire Data Register – DWDR

Bit	7	6	5	4	3	2	1	0	
	DWDR[7:0]								DWDR
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 DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

21. Boot Loader Support – Read-While-Write Self-Programming

In AT90PWM1, 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 (program) that 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, and it can also erase itself 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.

21.1 Boot Loader 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 83 on page 225) used during programming. The page organization does not affect normal operation.

21.2 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 21-2). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 70 on page 218 and Figure 21-2. These two sections can have different level of protection since they have different sets of Lock bits.

21.2.1 Application Section

The Application section is the section 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), see Table 66 on page 210. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

21.2.2 BLS – Boot Loader Section

While the Application section is used for storing the application code, the The Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. 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), see Table 67 on page 210.

21.3 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 which 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 71 on page 218 and Figure 21-2 on page 209. The main difference between the two sections is:

- 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 which

section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.

21.3.1 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 a call/jmp/lpm 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 “Store Program Memory Control and Status Register – SPMCSR” on page 211. for details on how to clear RWWSB.

21.3.2 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 65. Read-While-Write Features

Which Section does the Z-pointer Address During the Programming?	Which Section Can be Read During Programming?	Is the CPU Halted?	Read-While-Write Supported?
RWW Section	NRWW Section	No	Yes
NRWW Section	None	Yes	No

Figure 21-1. Read-While-Write vs. No Read-While-Write

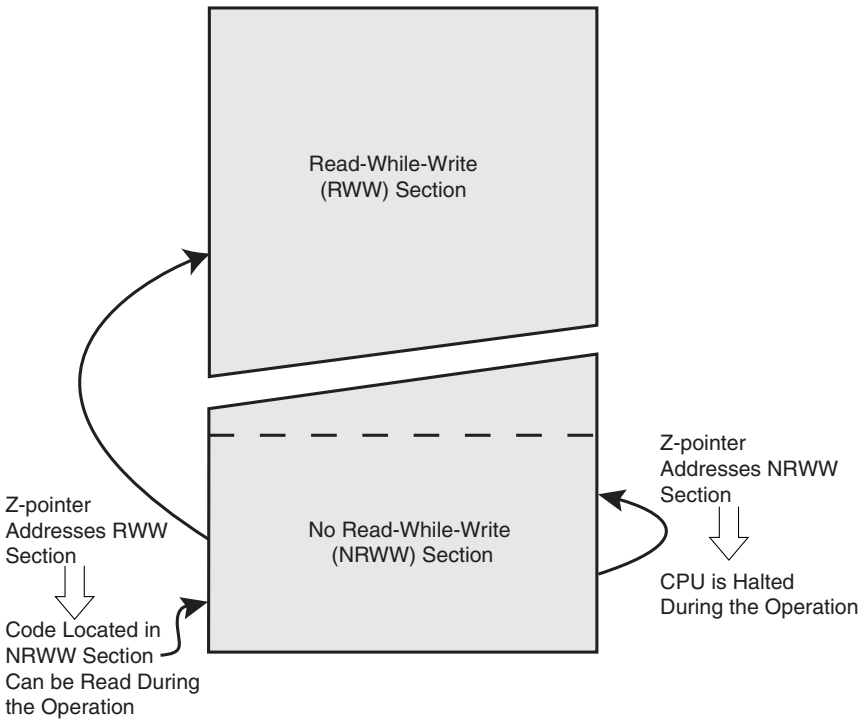
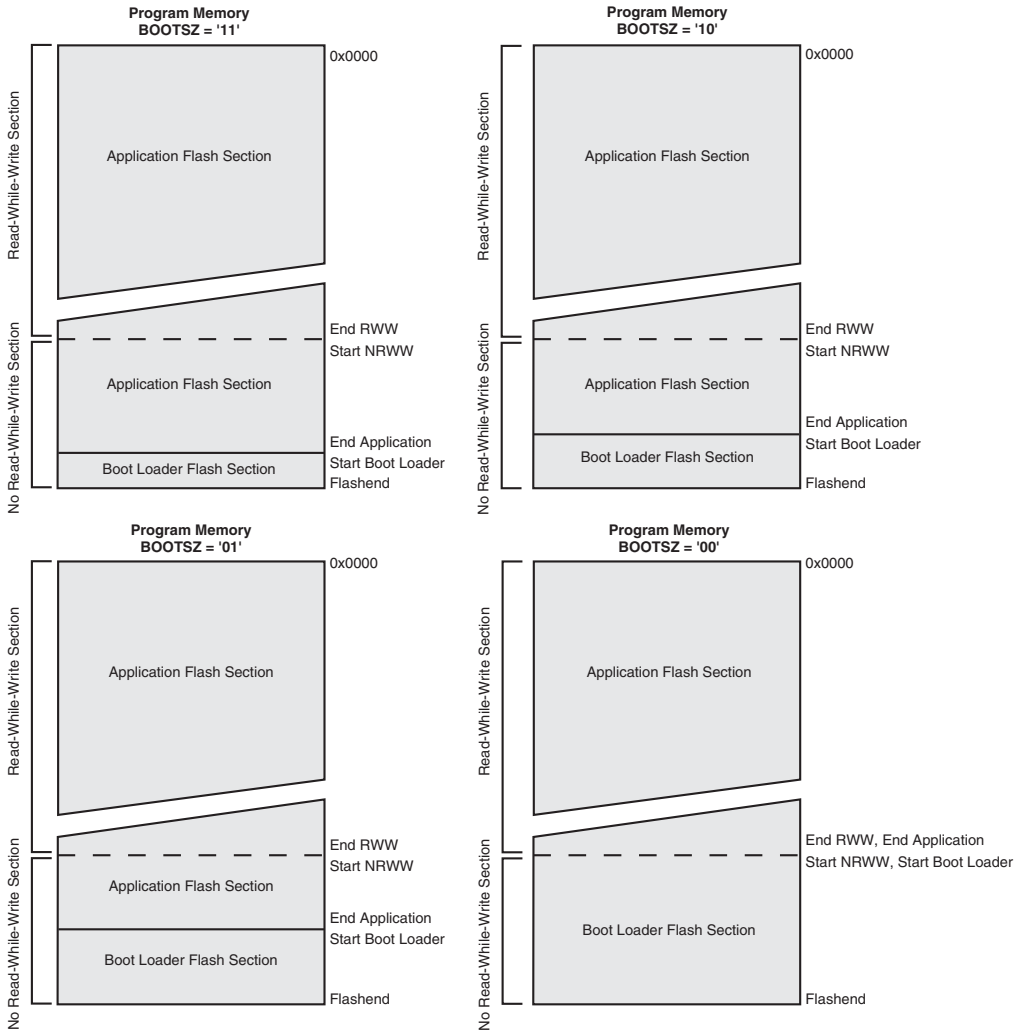


Figure 21-2. Memory Sections



Note: 1. The parameters in the figure above are given in Table 70 on page 218.

21.4 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 66 and Table 67 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 LPM/SPM, if it is attempted.

Table 66. Boot Lock Bit0 Protection Modes (Application Section)[1]

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

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

Table 67. Boot Lock Bit1 Protection Modes (Boot Loader Section)[1]

BLB1 Mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or 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 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	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.

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

21.5 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 68. Boot Reset Fuse[1]

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 70 on page 218)

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

21.5.1 Store Program Memory Control and Status Register – SPMCSR

The Store Program Memory Control and Status Register contains the control bits needed to control the Boot Loader operations.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

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

- **Bit 6 – RWWSB: Read-While-Write Section Busy**

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

- **Bit 5 – Res: Reserved Bit**

This bit is a reserved bit in the AT90PWM1 and always read as zero.

- **Bit 4 – RWWSRE: Read-While-Write Section Read Enable**

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

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

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

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCSR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See “Reading the Fuse and Lock Bits from Software” on page 215 for details.

- **Bit 2 – PGWRT: Page Write**

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

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The

data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation if the NRWW section is addressed.

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

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

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

21.6 Addressing the Flash During Self-Programming

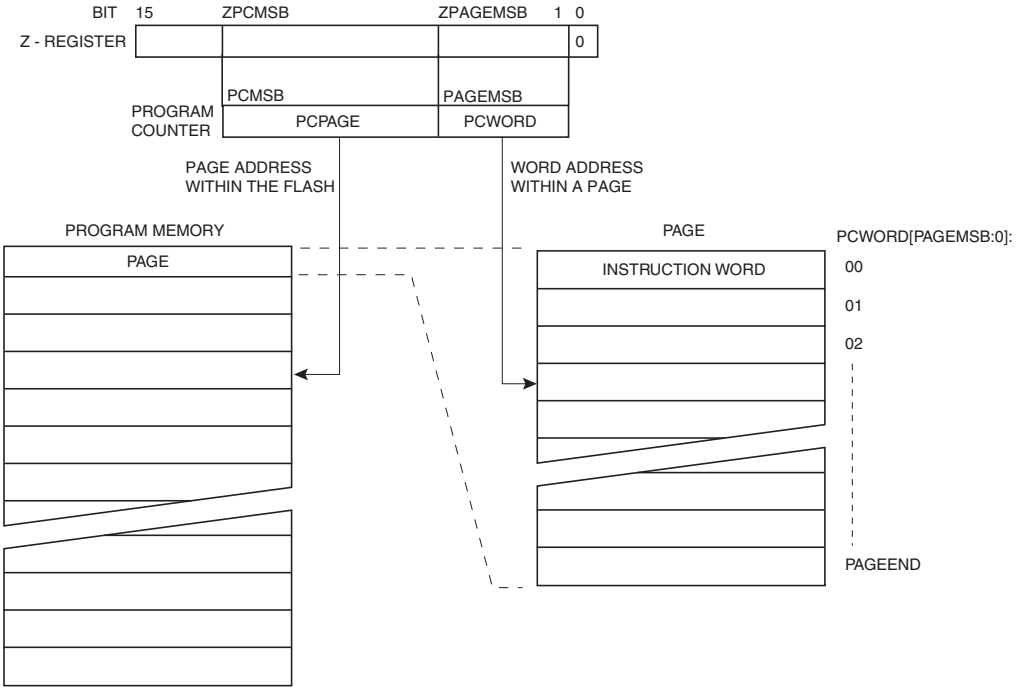
The Z-pointer is used to address the SPM commands.

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 83 on page 225), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is1 shown in Figure 21-3. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

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

Figure 21-3. Addressing the Flash During SPM[1]



Note: 1. The different variables used in Figure 21-3 are listed in Table 72 on page 219.

21.7 Self-Programming the Flash

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

Alternative 1, fill the buffer before a Page Erase

- Fill temporary page buffer
- Perform a Page Erase
- Perform a Page Write

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. See “Simple Assembly Code Example for a Boot Loader” on page 216 for an assembly code example.

21.7.1 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.

21.7.2 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 “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 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.

21.7.3 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.

21.7.4 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 XXXXXXXX.

21.7.5 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.

21.7.6 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 XXXXXXX, 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 216 for an example.

21.7.7 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 66 and Table 67 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 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 lO_{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.

21.7.8 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the 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.

21.7.9 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 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 LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SPMEN are cleared, 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 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 below. Refer to Table 76 on page 221 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, when reading the Fuse High byte, load 0x0003 in the Z-pointer. When an 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 77 on page 222 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

When reading the Extended Fuse byte, load 0x0002 in the Z-pointer. When an 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 76 on page 221 for detailed description and mapping of the Extended Fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	EFB3	EFB2	EFB1	EFB0

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

21.7.10 Preventing Flash Corruption

During periods of low V_{CC}, 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 the 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_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
3. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

21.7.11 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 69 shows the typical programming time for Flash accesses from the CPU.

Table 69. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms

21.7.12 Simple Assembly Code Example for a Boot Loader

```
;-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.
.equ PAGESIZEB = PAGESIZE*2   ;PAGESIZEB is page size 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:
  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<<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
```

21.7.13 Boot Loader Parameters

In Table 70 through Table 72, the parameters used in the description of the self programming are given.

Table 70. Boot Size Configuration

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	128 words	4	0x000 - 0xF7F	0xF80 - 0xFFF	0xF7F	0xF80
1	0	256 words	8	0x000 - 0xEFF	0xF00 - 0xFFF	0xEFF	0xF00
0	1	512 words	16	0x000 - 0xDFF	0xE00 - 0xFFF	0xDFF	0xE00
0	0	1024 words	32	0x000 - 0xBFF	0xC00 - 0xFFF	0xBFF	0xC00

Note: The different BOOTSZ Fuse configurations are shown in Figure 21-2.

Table 71. Read-While-Write Limit

Section	Pages	Address
Read-While-Write section (RWW)	96	0x000 - 0xBFF
No Read-While-Write section (NRWW)	32	0xC00 - 0xFFF

For details about these two section, see “NRWW – No Read-While-Write Section” on page 207 and “RWW – Read-While-Write Section” on page 207

Table 72. Explanation of Different Variables used in Figure 21-3 and the Mapping to the Z-pointer

Variable		Corresponding Z-value[1]	Description
PCMSB	11		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		Most significant bit which is used to address the words within one page (32 words in a page requires 5 bits PC [4:0]).
ZPCMSB		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z5	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[11:5]	Z12:Z6	Program counter page address: Page select, for page erase and page write
PCWORD	PC[4:0]	Z5:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z13: always ignored
Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See “Addressing the Flash During Self-Programming” on page 212 for details about the use of Z-pointer during Self-Programming.

22. Memory Programming

22.1 Program And Data Memory Lock Bits

The AT90PWM1 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 74. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 73. Lock Bit Byte[1]

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

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

Table 74. 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]

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

Table 75. Lock Bit Protection Modes[1][2]. Only ATmega88/168.

BLB0 Mode	**BLB02**	**BLB01**	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
BLB1 Mode	**BLB12**	**BLB11**	
1	1	1	No restrictions for SPM or 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 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	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 unprogrammed, “0” means programmed

22.2 Fuse Bits

The AT90PWM1 has three Fuse bytes. Table 76 - Table 78 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 76. Extended Fuse Byte

Extended Fuse Byte	Bit No	Description	Default Value
PSC2RB	7	PSC2 Reset Behaviour	1
PSC0RB	5	PSC0 Reset Behaviour	1
PSCRV	4	PSCOUT Reset Value	1
–	3	–	1
BOOTSZ1	2	Select Boot Size (see Table 113 for details)	0 (programmed)(1)
BOOTSZ0	1	Select Boot Size (see Table 113 for details)	0 (programmed)(1)
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Note: 1. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 79 on page 224 for details.

22.3 PSC Output Behaviour During Reset

For external component safety reason, the state of PSC outputs during Reset can be programmed by fuses PSCRV, PSC0RB, PSC1RB & PSC2RB.

These fuses are located in the Extended Fuse Byte (see Table 76)

PSCRV gives the state low or high which will be forced on PSC outputs selected by PSC0RB, PSC2RB fuses.

If PSCRV fuse equals 0 (programmed), the selected PSC outputs will be forced to low state. If PSCRV fuse equals 1 (unprogrammed), the selected PSC outputs will be forced to high state.

If PSC0RB fuse equals 1 (unprogrammed), PSCOUT00 & PSCOUT01 keep a standard port behaviour. If PSC0RB fuse equals 0 (programmed), PSCOUT00 & PSCOUT01 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT00 & PSCOUT01 keep the forced state until PSOC0 register is written..

If PSC2RB fuse equals 1 (unprogrammed), PSCOUT20, PSCOUT21, PSCOUT22 & PSCOUT23 keep a standard port behaviour. If PSC1RB fuse equals 0 (programmed), PSCOUT20, PSCOUT21, PSCOUT22 & PSCOUT23 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT20, PSCOUT21, PSCOUT22 & PSCOUT23 keep the forced state until PSOC2 register is written.

Table 77. Fuse High Byte

High Fuse Byte	Bit No	Description	Default Value
RSTDISBL[1]	7	External Reset Disable	1 (unprogrammed)
DWEN	6	debugWIRE Enable	1 (unprogrammed)
SPIEN[2]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI programming enabled)
WDTON[3]	4	Watchdog Timer Always On	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed), EEPROM not reserved
BODLEVEL2[4]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1[4]	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[4]	0	Brown-out Detector trigger level	1 (unprogrammed)

Notes: 1. See "Alternate Port Functions" on page 63 for description of RSTDISBL Fuse.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. See "Watchdog Timer Configuration" on page 51 for details.
4. See Table 9-2 on page 45 for BODLEVEL Fuse decoding.

Table 78. Fuse Low Byte

Low Fuse Byte	Bit No	Description	Default Value
CKDIV8[4]	7	Divide clock by 8	0 (programmed)
CKOUT[3]	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[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 (unprogrammed)[2]
CKSEL0	0	Select Clock source	0 (programmed)[2]

Note: 1. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 7-10 on page 34 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 8 MHz. See Table 7-10 on page 34 for details.
3. The CKOUT Fuse allows the system clock to be output on PORTB0. See "Clock Output Buffer" on page 35 for details.
4. See "System Clock Prescaler" on page 35 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.

22.3.1 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.

22.4 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 **AT90PWM1** the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x93 (indicates 8KB Flash memory).
3. 0x002: 0x83 (indicates AT90PWM1 device when 0x001 is 0x93).

22.5 Calibration Byte

The AT90PWM1 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.

22.6 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 AT90PWM1. Pulses are assumed to be at least 250 ns unless otherwise noted.

22.6.1 Signal Names

In this section, some pins of the AT90PWM1 are referenced by signal names describing their functionality during parallel programming, see Figure 22-1 and Table 79. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 81.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 82.

Figure 22-1. Parallel Programming

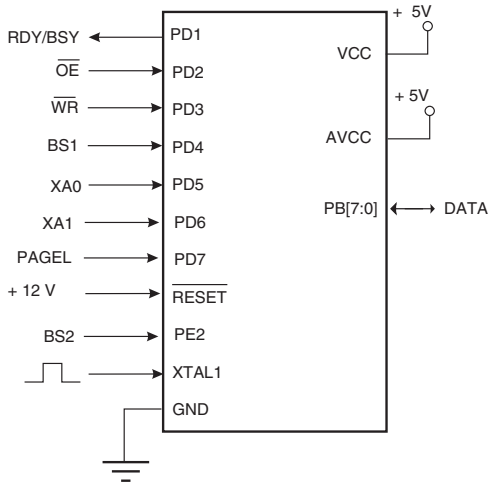


Table 79. Pin Name Mapping

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 (“0” selects Low byte, “1” selects High byte)
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 (“0” selects Low byte, “1” selects 2’nd High byte)
DATA	PB[7:0]	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is $\overline{low}$)

Table 80. 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 81. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by 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 82. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash

Table 82. Command Byte Bit Coding

Command Byte	Command Executed
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

Table 83. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
AT90PWM1	4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11

Table 84. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
AT90PWM1	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

22.7 Serial Programming Pin Mapping

Table 85. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI_A	PD3	I	Serial Data in
MISO_A	PD2	O	Serial Data out
SCK_A	PD4	I	Serial Clock

22.8 Parallel Programming

22.8.1 Enter Programming Mode

The following algorithm puts the device in Parallel (High-voltage) > Programming mode:

1. Set Prog_enable pins listed in Table 80. to “0000”, RESET pin to “0” and Vcc to 0V.
2. Apply 4.5 - 5.5V between VCC and GND. Ensure that Vcc reaches at least 1.8V within the next 20µs.
3. Wait 20 - 60µs, and apply 11.5 - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Wait at least 300µs before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing RESET pin to 0V.

If the rise time of the Vcc is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set Prog_enable pins listed in Table 80. to “0000”, RESET pin to “0” and Vcc to 0V.
2. Apply 4.5 - 5.5V between VCC and GND.
3. Monitor Vcc, and as soon as Vcc reaches 0.9 - 1.1V, apply 11.5 - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Wait until Vcc actually reaches 4.5 -5.5V before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing RESET pin to 0V.

22.8.2 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.

22.8.3 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 EEPRPOM 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.

22.8.4 Programming the Flash

The Flash is organized in pages, see Table 83 on page 225. 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 22-3 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 22-2 on page 228. Note that if less than eight bits are required to address words in the page (pagesize < 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

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

H. Program Page

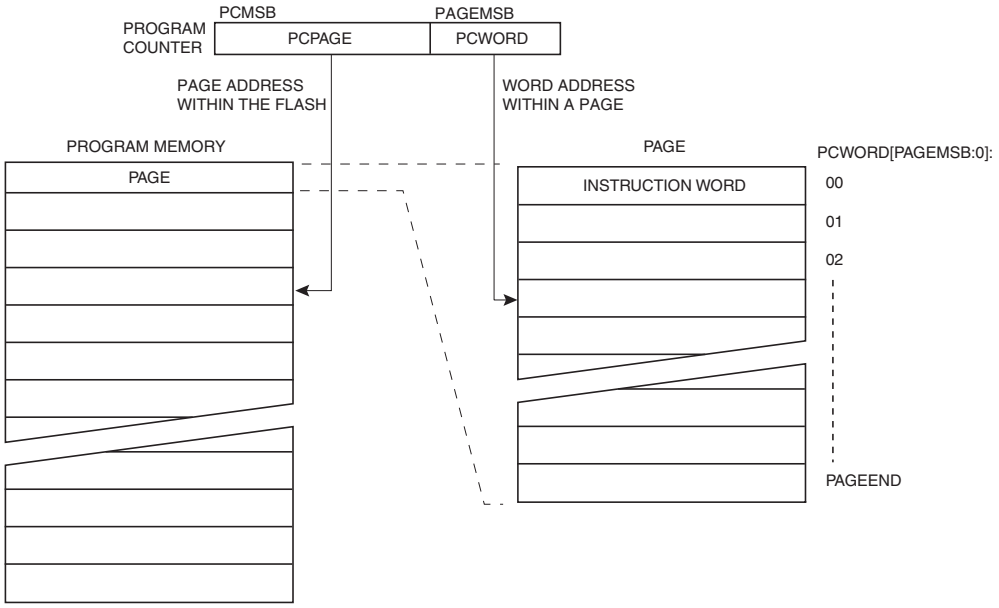
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (See Figure 22-3 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

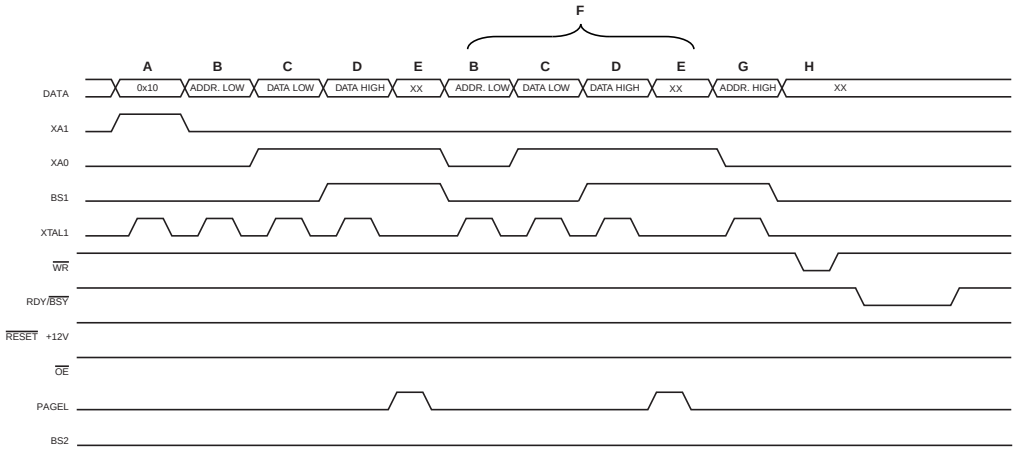
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 22-2. Addressing the Flash Which is Organized in Pages[1]



Note: 1. PCPAGE and PCWORD are listed in Table 83 on page 225.

Figure 22-3. Programming the Flash Waveforms[1]



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

22.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 84 on page 225. 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 226 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).

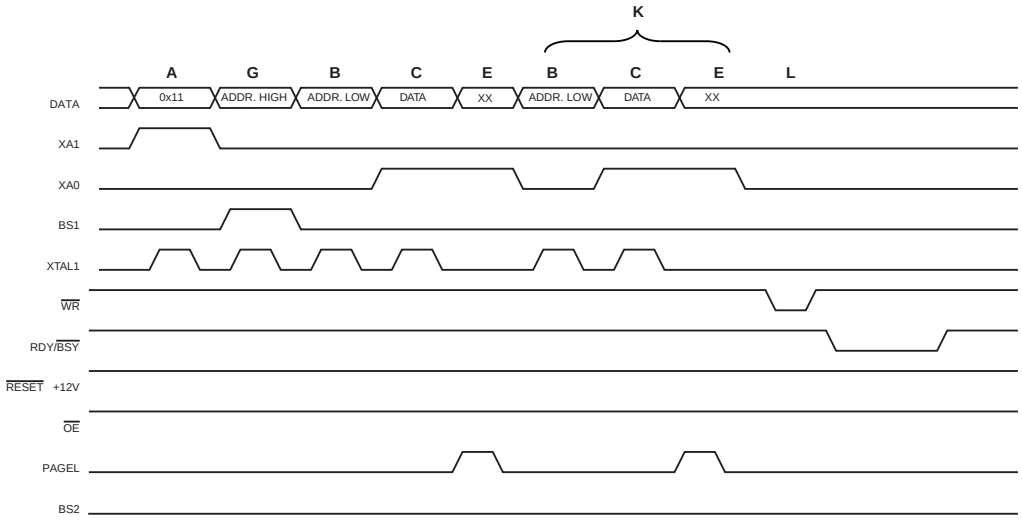
5. E: Latch data (give PAGEL a positive pulse).

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 22-4 for signal waveforms).

Figure 22-4. Programming the EEPROM Waveforms



22.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 226 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”.

22.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 226 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”.

22.8.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 226 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.

22.8.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 226 for details on Command and Data loading):

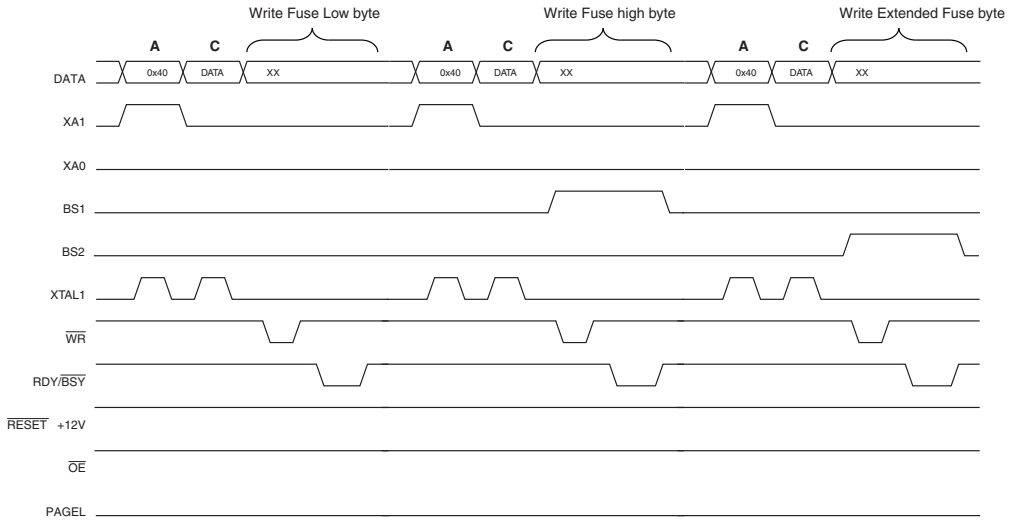
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

22.8.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 226 for details on Command and Data loading):

1. 1. A: Load Command “0100 0000”.
2. 2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. 3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. 4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. 5. Set BS2 to “0”. This selects low data byte.

Figure 22-5. Programming the FUSES Waveforms



22.8.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 226 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 programmed (LB1 and LB2 is 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.

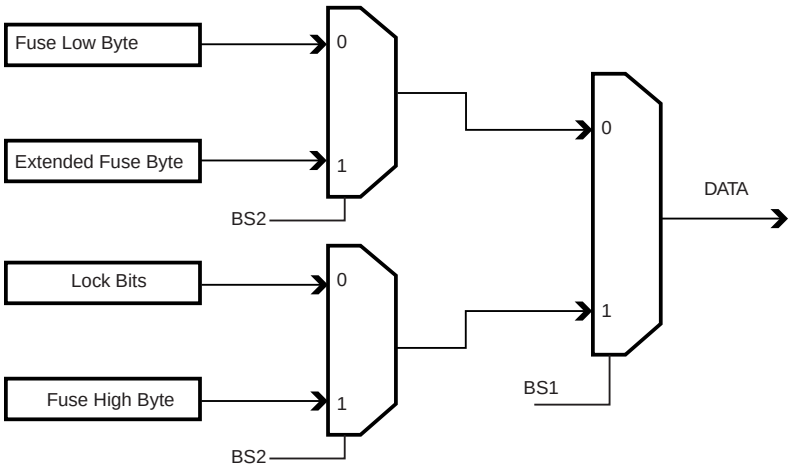
The Lock bits can only be cleared by executing Chip Erase.

22.8.12 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 226 for details on Command loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 22-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



22.8.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to “Programming the Flash” on page 226 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

22.8.14 Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to “Programming the Flash” on page 226 for details on Command and Address 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”.

22.8.15 Parallel Programming Characteristics

Figure 22-7. Parallel Programming Timing, Including some General Timing Requirements

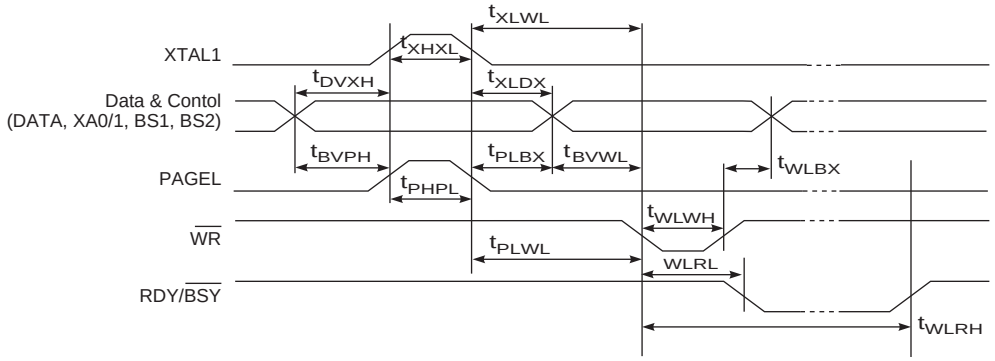
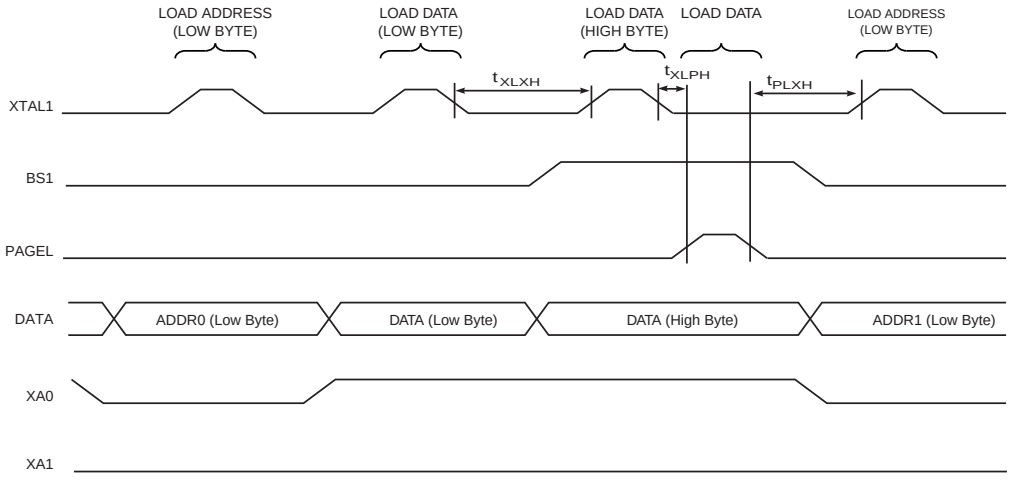
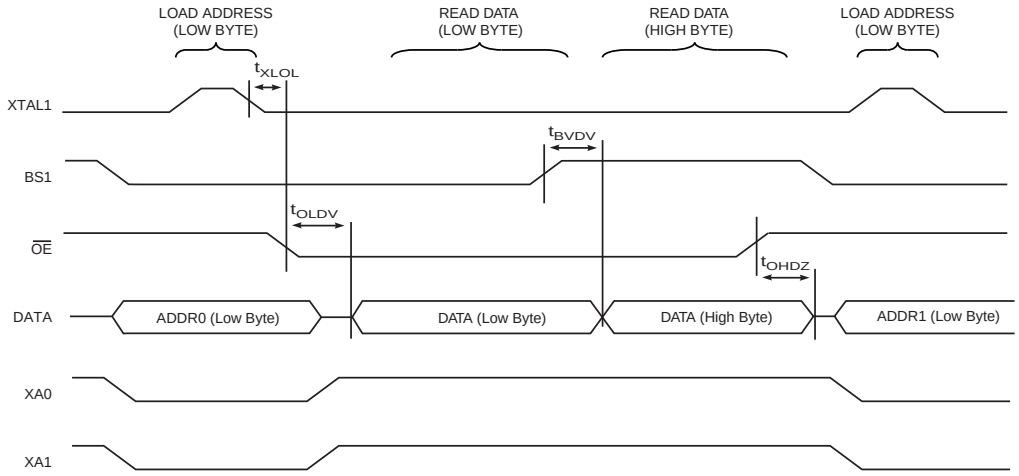


Figure 22-8. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 22-7 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 22-9. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 22-7 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 86. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to XTAL1 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}	BS1 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]	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns

Table 86. Parallel Programming Characteristics, V_{CC} = 5V ± 10% (Continued)

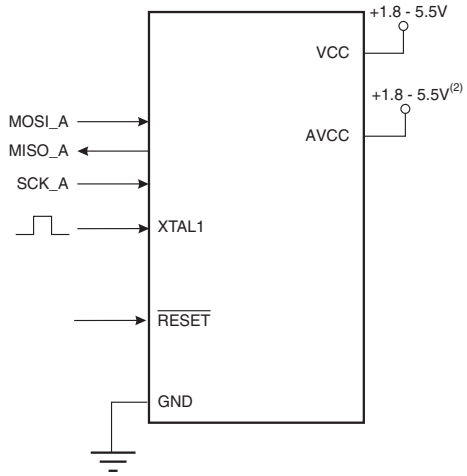
Symbol	Parameter	Min	Typ	Max	Units
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

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.

22.9 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 85 on page 225, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 22-10. Serial Programming and Verify(1)



Notes: 1. If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.
2. V_{CC} - 0.3V < AVCC < V_{CC} + 0.3V, however, AVCC should always be within 1.8 - 5.5V

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

22.9.1 Serial Programming Algorithm

When writing serial data to the AT90PWM1, data is clocked on the rising edge of SCK.

When reading data from the AT90PWM1, data is clocked on the falling edge of SCK. See Figure 22-11 for timing details.

To program and verify the AT90PWM1 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 88):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ 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, $\overline{\text{RESET}}$ 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 MOSI.
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 $\overline{\text{RESET}}$ 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 6 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 8 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 87.) 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 87.) 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 MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

22.9.2 Data Polling Flash

When a page is being programmed into the Flash, reading an address location within the page being programmed will give the value 0xFF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the Flash will not work for the value 0xFF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. See Table 87 for t_{WD_FLASH} value.

22.9.3 Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value 0xFF. At the time the device is ready for

a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value 0xFF, but the user should have the following in mind: As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. This does not apply if the EEPROM is re-programmed without chip erasing the device. In this case, data polling cannot be used for the value 0xFF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See Table 87 for t_{WD_EEPROM} value.

Table 87. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	3.6 ms
t_{WD_ERASE}	9.0 ms

Figure 22-11. Serial Programming Waveforms

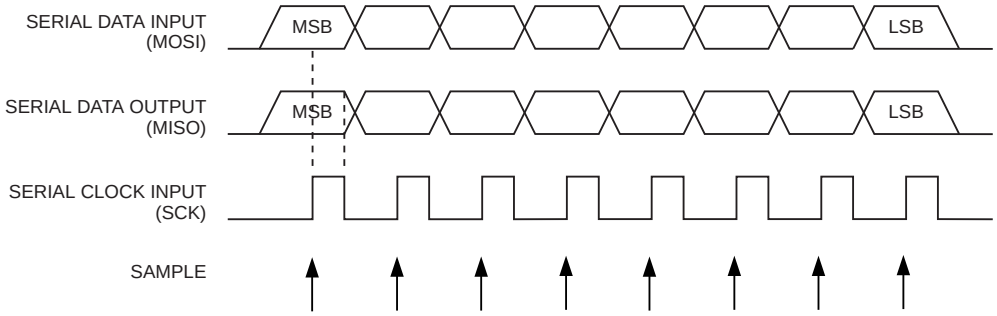


Table 88. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program Memory	0010 **H**000	000**a aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Load Program Memory Page	0100 **H**000	000x xxxx	xx**bb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program Memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
Write Program Memory Page	0100 1100	000**a aaaa**	**bb**xx xxxx	xxxx xxxx	Write Program Memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	000x xx**aa**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	000x xx**aa**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.

Table 88. Serial Programming Instruction Set (Continued)

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Load EEPROM Memory Page (page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM Memory Page (page access)	1100 0010	00xx xx**aa**	**bbbb bb00**	xxxx xxxx	Write EEPROM page at address **a**:**b**.
Read Lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. “0” = programmed, “1” = unprogrammed. See Table 73 on page 219 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = “0” to program Lock bits. See Table 73 on page 219 for details.
Read Signature Byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See **Table XXX on page XXX** for details.
Write Fuse High bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 77 on page 222 for details.
Write Extended Fuse Bits	1010 1100	1010 0100	xxxx xxxx	**xxxx xxii**	Set bits = “0” to program, “1” to unprogram. See Table 76 on page 221 for details.
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed. See **Table XXX on page XXX** for details.
Read Fuse High bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse High bits. “0” = programmed, “1” = unprogrammed. See Table 77 on page 222 for details.
Read Extended Fuse Bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read Extended Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 76 on page 221 for details.
Read Calibration Byte	0011 1000	000x xxxx	0000 0000	**oooo oooo**	Read Calibration Byte
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = “1”, a programming operation is still busy. Wait until this bit returns to “0” before applying another command.

Note: **a** = address high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don’t care

22.9.4 SPI Serial Programming Characteristics

For characteristics of the SPI module see “SPI Serial Programming Characteristics” on page 237.

23. Electrical Characteristics(1)

23.1 Absolute Maximum Ratings*

Operating Temperature..... -40°C to +105°C Storage Temperature ..... -65°C to +150°C Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground .....-1.0V to V_{CC}+0.5V Voltage on $\overline{\text{RESET}}$ with respect to Ground.....-1.0V to +13.0V Maximum Operating Voltage ..... 6.0V DC Current per I/O Pin ..... 40.0 mA DC Current V_{CC} and GND Pins..... 200.0 mA	*NOTICE: 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 other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note: 1. Electrical Characteristics for this product have not yet been finalized. Please consider all values listed herein as preliminary and non-contractual.

23.2 DC Characteristics

T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage	Port B, C & D and XTAL1, XTAL2 pins as I/O	-0.5		$0.2V_{CC}$[1]	V
V_{IH}	Input High Voltage	Port B, C & D and XTAL1, XTAL2 pins as I/O	$0.6V_{CC}$[2]		V_{CC}+0.5	V
V_{IL1}	Input Low Voltage	XTAL1 pin , External Clock Selected	-0.5		$0.1V_{CC}$[1]	V
V_{IH1}	Input High Voltage	XTAL1 pin , External Clock Selected	$0.7V_{CC}$[2]		V_{CC}+0.5	V
V_{IL2}	Input Low Voltage	$\overline{\text{RESET}}$ pin	-0.5		$0.2V_{CC}$[1]	V
V_{IH2}	Input High Voltage	$\overline{\text{RESET}}$ pin	$0.9V_{CC}$[2]		V_{CC}+0.5	V
V_{IL3}	Input Low Voltage	$\overline{\text{RESET}}$ pin as I/O	-0.5		$0.2V_{CC}$[1]	V
V_{IH3}	Input High Voltage	$\overline{\text{RESET}}$ pin as I/O	$0.8V_{CC}$[2]		V_{CC}+0.5	V
V_{OL}	Output Low Voltage[3] (Port B, C & D and XTAL1, XTAL2 pins as I/O)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	Output High Voltage[4] (Port B, C & D and XTAL1, XTAL2 pins as I/O)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.4			V V
V_{OL3}	Output Low Voltage[3] ($\overline{\text{RESET}}$ pin as I/O)	I_{OL} = 2.1 mA, V_{CC} = 5V I_{OL} = 0.8 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH3}	Output High Voltage[4] ($\overline{\text{RESET}}$ pin as I/O)	I_{OH} = -0.6 mA, V_{CC} = 5V I_{OH} = -0.4 mA, V_{CC} = 3V	3.8 2.2			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		30		200	kΩ
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ

T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{CC}	Power Supply Current	Active 8 MHz, V_{CC} = 3V, RC osc, PRR = 0xFF		3.8	7	mA
		Active 16 MHz, V_{CC} = 5V, Ext Clock, PRR = 0xFF		14	24	mA
		Idle 8 MHz, V_{CC} = 3V, RC Osc		1.5	3	mA
		Idle 16 MHz, V_{CC} = 5V, Ext Clock		5.5	10	mA
	Power-down mode(5)	WDT enabled, V_{CC} = 3V t0 < 90°C		5	15	µA
		WDT enabled, V_{CC} = 3V t0 < 105°C		9	20	µA
		WDT disabled, V_{CC} = 3V t0 < 90°C		2	3	µA
		WDT disabled, V_{CC} = 3V t0 < 105°C		5	10	µA
V_{ACIO}	Analog Comparator Input Offset Voltage AT90PWM2/3	V_{CC} = 5V, V_{in} = 3V		20	50	mV
V_{hysr}	Analog Comparator Hysteresis Voltage AT90PWM2B/3B	V_{CC} = 5V, V_{in} = 3V Rising Edge Falling Edge	 33 34	 46 62	 71 110	 mV mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA
t_{ACID}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 5.0V		(6) (6)		ns

Note:
1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   SO32, SO24 and TQFN Package:
   1] The sum of all IOL, for all ports, should not exceed 400 mA.
   2] The sum of all IOL, for ports B6 - B7, C0 - C1, D0 - D3, E0 should not exceed 100 mA.
   3] The sum of all IOL, for ports B0 - B1, C2 - C3, D4, E1 - E2 should not exceed 100 mA.
   4] The sum of all IOL, for ports B3 - B5, C6 - C7 should not exceed 100 mA.
   5] The sum of all IOL, for ports B2, C4 - C5, D5 - D7 should not exceed 100 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   SO32, SO24 and TQFN Package:
   1] The sum of all IOH, for all ports, should not exceed 400 mA.
   2] The sum of all IOH, for ports B6 - B7, C0 - C1, D0 - D3, E0 should not exceed 150 mA.
   3] The sum of all IOH, for ports B0 - B1, C2 - C3, D4, E1 - E2 should not exceed 150 mA.
   4] The sum of all IOH, for ports B3 - B5, C6 - C7 should not exceed 150 mA.
   5] The sum of all IOH, for ports B2, C4 - C5, D5 - D7 should not exceed 150 mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

5. Minimum V_{CC} for Power-down is 2.5V.
6. The Analog Comparator Propogation Delay equals 1 comparator clock plus 30 nS. See “Analog Comparator” on page 176. for comparator clock definition.

23.3 External Clock Drive Characteristics

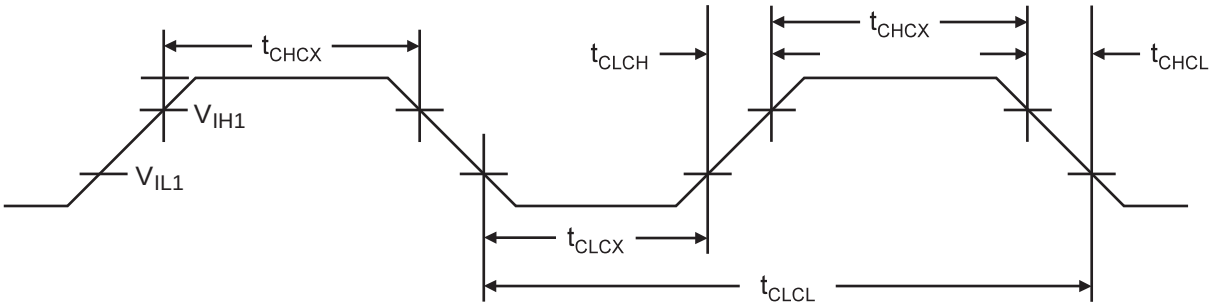
23.3.1 Calibrated Internal RC Oscillator Accuracy

Table 23-1. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{CC}	Temperature	Calibration Accuracy
Factory Calibration	8.0 MHz	3V	25°C	±10%
User Calibration	7.3 - 8.1 MHz	2.7V - 5.5V	-40°C - 85°C	±1%

23.3.2 External Clock Drive Waveforms

Figure 23-1. External Clock Drive Waveforms



23.3.3 External Clock Drive

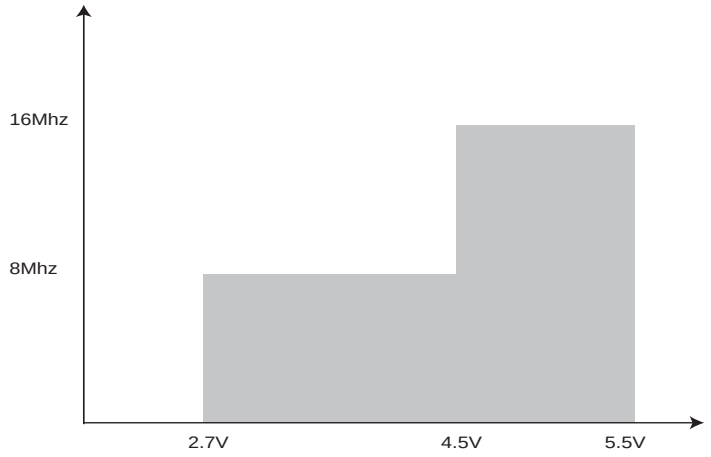
Table 23-2. External Clock Drive

		V_{CC}=1.8-5.5V		V_{CC}=2.7-5.5V		V_{CC}=4.5-5.5V		
Symbol	**Parameter**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	10	0	20	MHz
t_{CLCL}	Clock Period	250		100		50		ns
t_{CHCX}	High Time	100		40		20		ns
t_{CLCX}	Low Time	100		40		20		ns
t_{CLCH}	Rise Time		2.0		1.6		0.5	µs
t_{CHCL}	Fall Time		2.0		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

23.4 Maximum Speed vs. V_{CC}

Maximum frequency is depending on $V_{CC.}$ As shown in Figure 23-2 , the Maximum Frequency equals 8Mhz when V_{CC} is contained between 2.7V and 4.5V and equals 16Mhz when V_{CC} is contained between 4.5V and 5.5V.

Figure 23-2. Maximum Frequency vs. V_{CC}, AT90PWM1



23.5 PLL Characteristics

.

Table 23-3. PLL Characteristics - V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Min.	Typ.	Max.	Units
PLL_{IF}	Input Frequency	0.5	1	2	MHz
PLL_{F}	PLL Factor		64		
PLL_{LT}	Lock-in Time			64	µS

Note: While connected to external clock or external oscillator, PLL Input Frequency must be selected to provide outputs with frequency in accordance with driven parts of the circuit (CPU core, PSC...)

23.6 SPI Timing Characteristics

See Figure 23-3 and Figure 23-4 for details.

Table 23-4. SPI Timing Parameters

	Description	Mode	Min.	Typ.	Max.	
1	SCK period	Master		See Table 54		ns
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t_{sck}		
7	SCK to out	Master		10		
8	SCK to out high	Master		10		
9	SS low to out	Slave		15		
10	SCK period	Slave	4 • t_{ck}			
11	SCK high/low (1)	Slave	2 • t_{ck}			
12	Rise/Fall time	Slave			1.6	
13	Setup	Slave	10			
14	Hold	Slave	t_{ck}			
15	SCK to out	Slave		15		
16	SCK to $\overline{SS}$ high	Slave	20			
17	$\overline{SS}$ high to tri-state	Slave		10		
18	SS low to SCK	Slave	2 • t_{ck}			

Note: In SPI Programming mode the minimum SCK high/low period is:
- 2 t_{CLCL} for f_{CK} < 12 MHz
- 3 t_{CLCL} for f_{CK} >12 MHz

Figure 23-3. SPI Interface Timing Requirements (Master Mode)

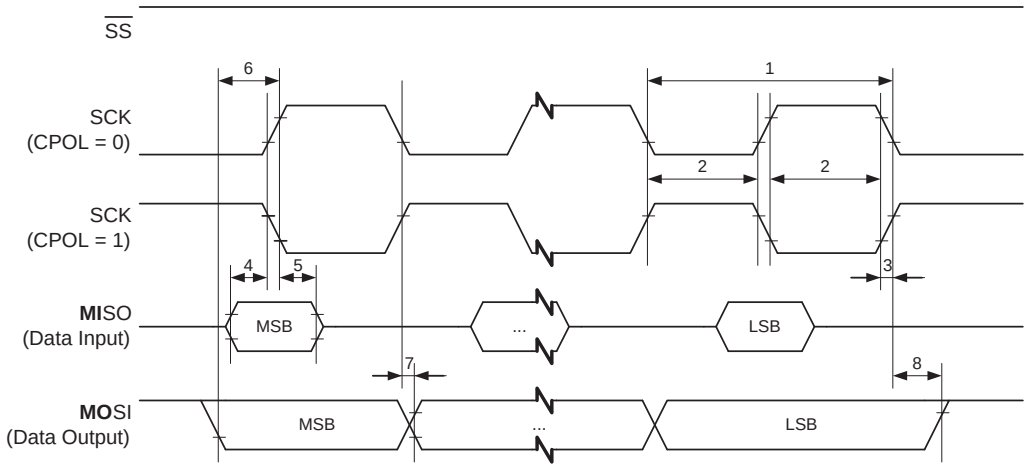
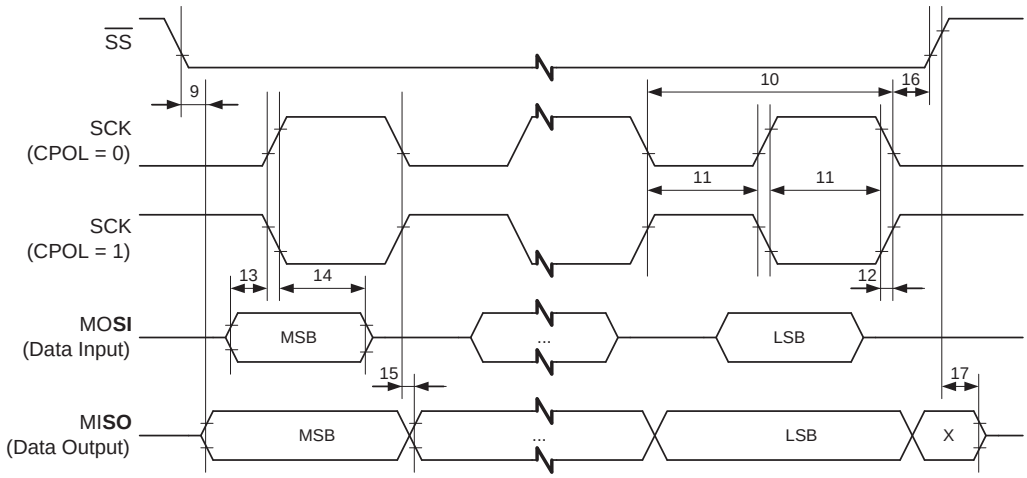


Figure 23-4. SPI Interface Timing Requirements (Slave Mode)



23.7 ADC Characteristics

Table 23-5. ADC Characteristics - T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution	Single Ended Conversion		10		Bits
		Differential Conversion		8		Bits
	Absolute accuracy	Single Ended Conversion V_{REF} = 2.56V ADC clock = 500 kHz		2.5	3	LSB
		Single Ended Conversion V_{REF} = 2.56V ADC clock = 1MHz		6 (*)	7	LSB
		Single Ended Conversion V_{REF} = 2.56V ADC clock = 2MHz			20	LSB
		Differential Conversion V_{REF} = 2.56V ADC clock = 500 kHz		2	3	LSB
		Differential Conversion V_{REF} = 2.56V ADC clock = 1MHz		3 (1)	4	LSB
	Integral Non-linearity	Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 2.56V ADC clock = 1MHz		1.1	1.5	LSB
		Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 2.56V ADC clock = 500 kHz		0.6	1	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 2.56V ADC clock = 1MHz		1.5	2	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 2.56V ADC clock = 500 kHz		1	1.5	LSB
	Differential Non-linearity	Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 1MHz		0.4	0.6	LSB
		Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 500 kHz		0.3	0.5	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 1MHz		0.5	0.8	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 500 kHz		0.4	0.8	LSB

Table 23-5. ADC Characteristics - T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted) (Continued)

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Zero Error (Offset)	Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 1MHz	-4		0	LSB
		Single Ended Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 500 kHz	-2		2.5	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 1MHz	-0.5		-0.5	LSB
		Differential Conversion V_{CC} = 4.5V, V_{REF} = 4V ADC clock = 500 kHz	-0.5		-0.5	LSB
	Conversion Time	Single Conversion	8		320	µs
	Clock Frequency		50		2000	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3		V_{CC} + 0.3	V
V_{REF}	Reference Voltage	Single Ended Conversion	2.0		AV_{CC}	V
		Differential Conversion	2.0		$AV_{CC - 0.2}$	V
V_{IN}	Input voltage	Single Ended Conversion	GND		V_{REF}	
		Differential Conversion	$-V_{REF}$/Gain		$+V_{REF}$/Gain	
	Input bandwidth	Single Ended Conversion		38.5		kHz
		Differential Conversion		4(2)		kHz
V_{INT}	Internal Voltage Reference			2.56		V
R_{REF}	Reference Input Resistance			30		kΩ
R_{AIN}	Analog Input Resistance			100		MΩ
I_{HSM}	Increased Current Consumption	High Speed Mode Single Ended Conversion			380	µA

Notes: 1. On AT90PWM2B/3B, this value will be close to the value at 500kHz.

2. 125KHz when input signal is synchronous with amplifier clock.

23.8 Parallel Programming Characteristics

Figure 23-5. Parallel Programming Timing, Including some General Timing Requirements

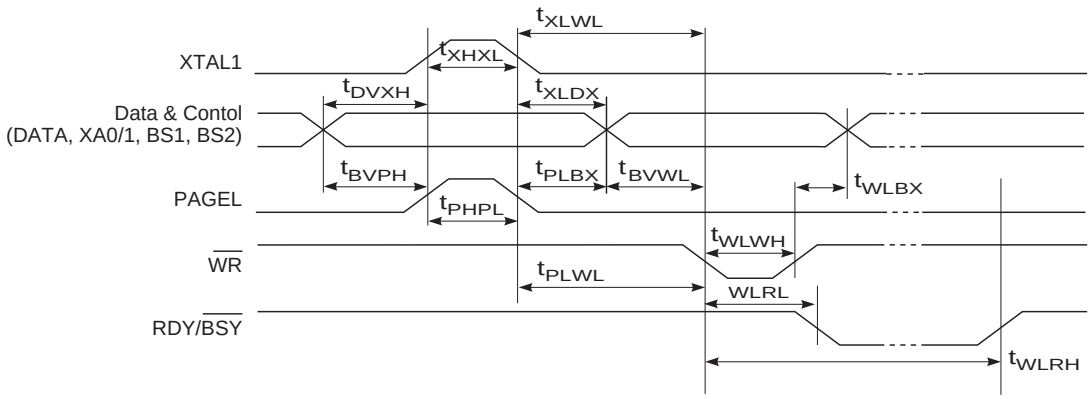
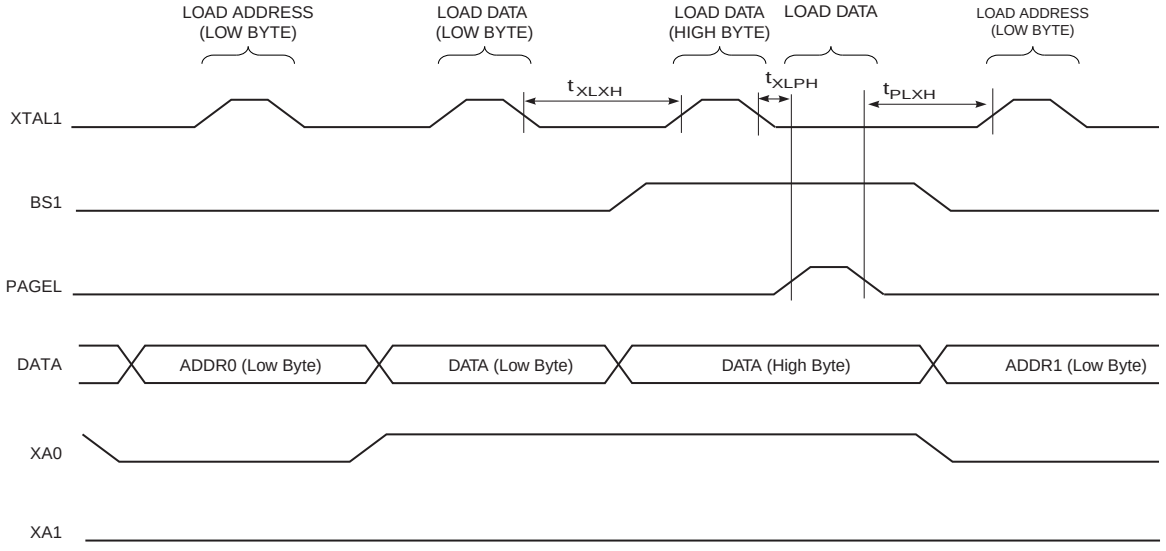
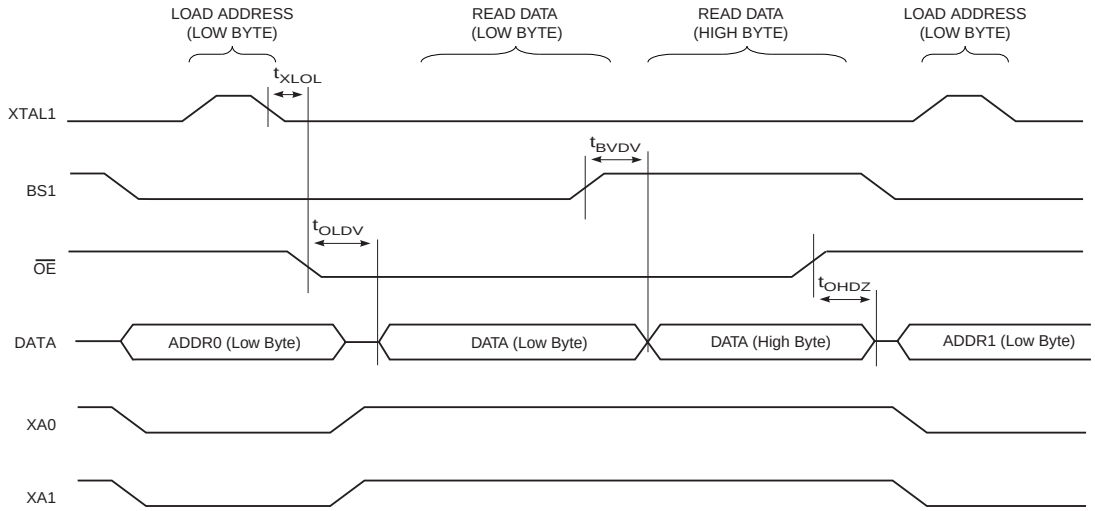


Figure 23-6. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 23-5 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 23-7. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[1]



Note: 1. ggThe timing requirements shown in Figure 23-5 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 23-6. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to XTAL1 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}	BS1 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		5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	7.5		10	ms

Table 23-6. Parallel Programming Characteristics, V_{CC} = 5V ± 10% (Continued)

Symbol	Parameter	Min.	Typ.	Max.	Units
t_{XLOL}	XTAL1 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.

24. AT90PWM1 Typical Characteristics

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

All Active- and Idle current consumption measurements are done with all bits in the PRR register set and thus, the corresponding I/O modules are turned off. Also the Analog Comparator is disabled during these measurements. Table 24-1 on page 254 and Table 24-2 on page 254 show the additional current consumption compared to I_{CC} Active and I_{CC} Idle for every I/O module controlled by the Power Reduction Register. See "Power Reduction Register" on page 37 for details.

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

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

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

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

The difference between current consumption in Power-down mode with Watchdog Timer enabled and Power-down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

24.1 Active Supply Current

Figure 24-1. Active Supply Current vs. Frequency (0.1 - 1.0 MHz)

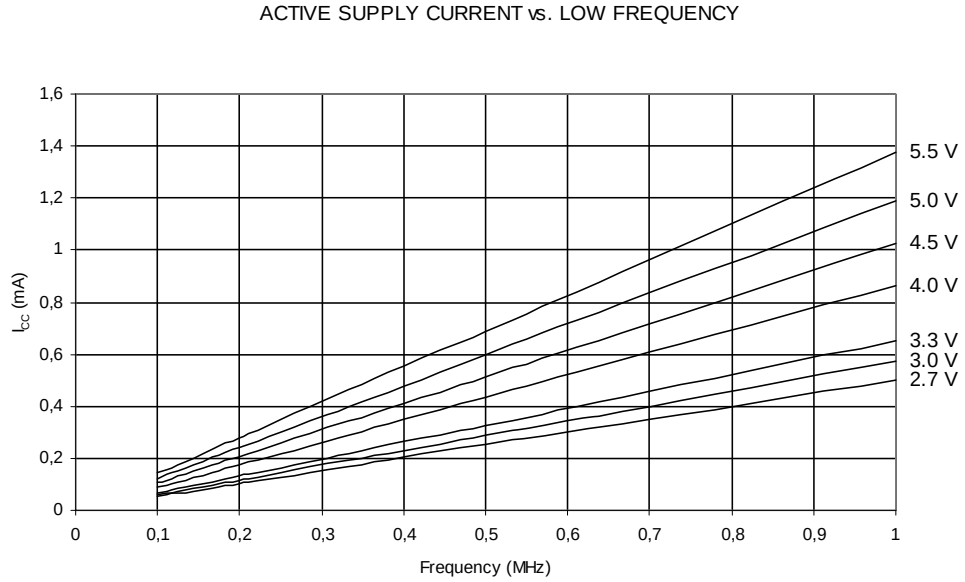


Figure 24-2. Active Supply Current vs. Frequency (1 - 24 MHz)

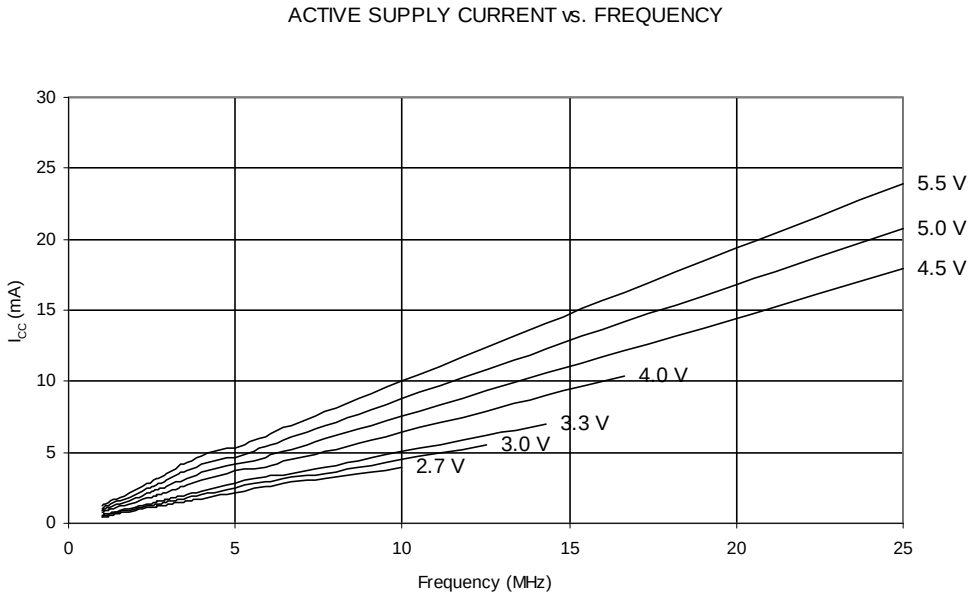


Figure 24-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

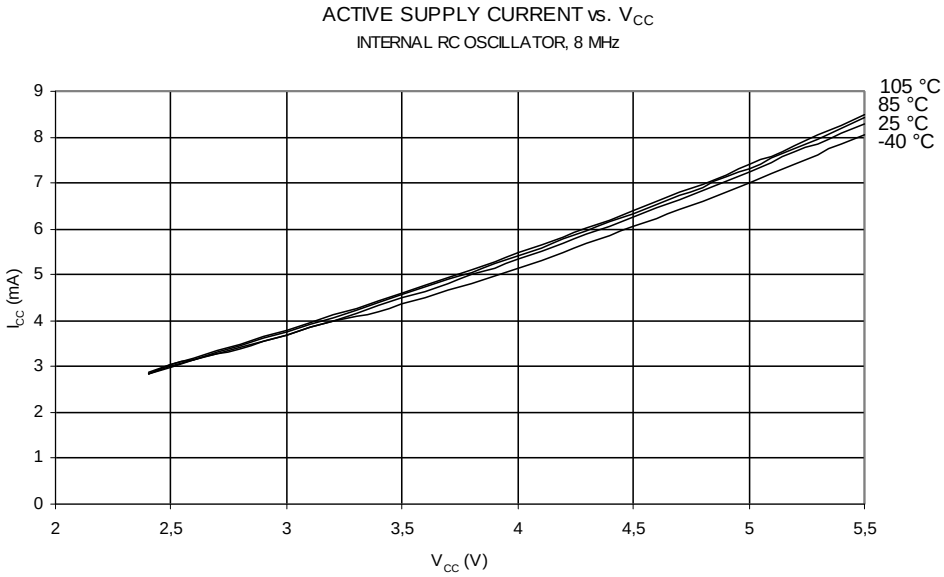
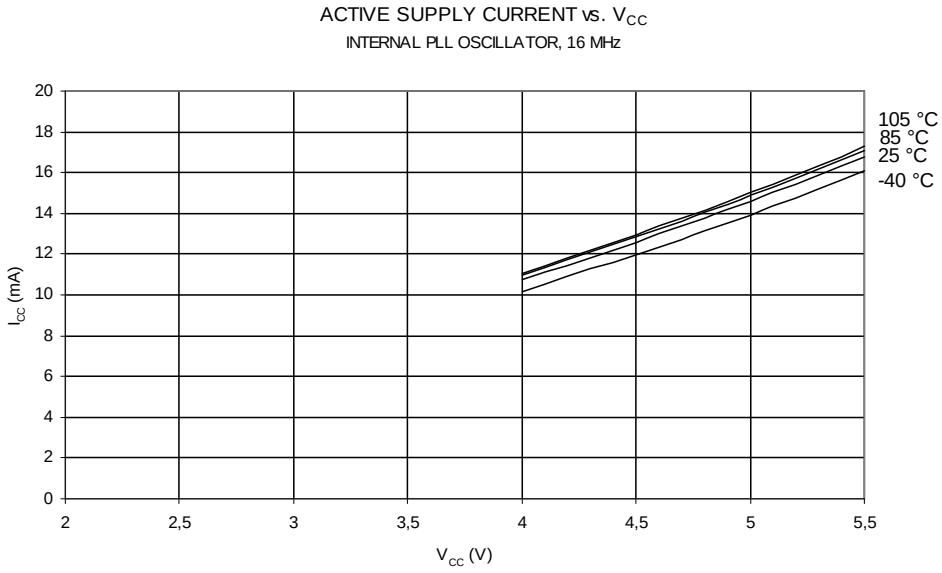


Figure 24-4. Active Supply Current vs. V_{CC} (Internal PLL Oscillator, 16 MHz)



24.2 Idle Supply Current

Figure 24-5. Idle Supply Current vs. Frequency (0.1 - 1.0 MHz)

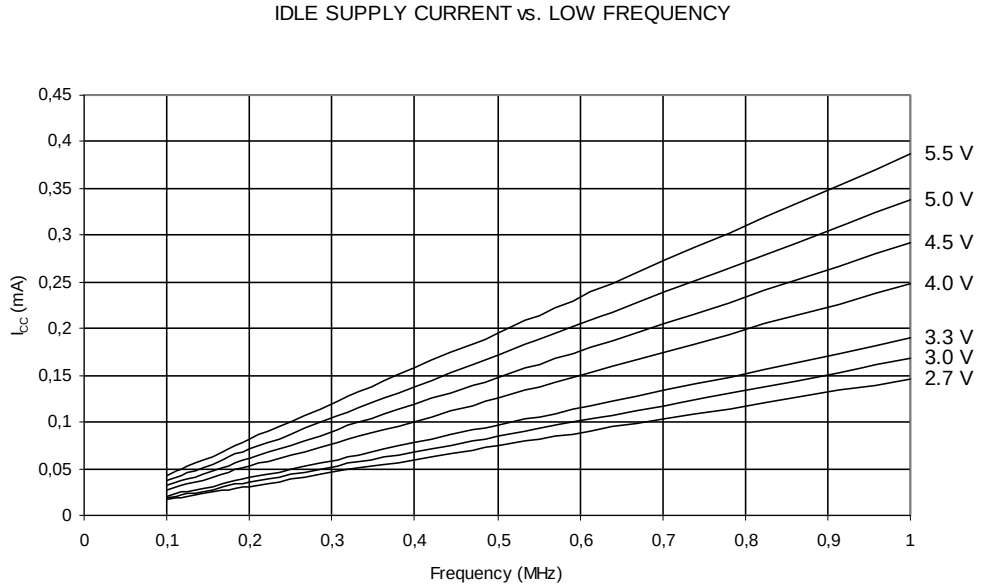


Figure 24-6. Idle Supply Current vs. Frequency (1 - 24 MHz)

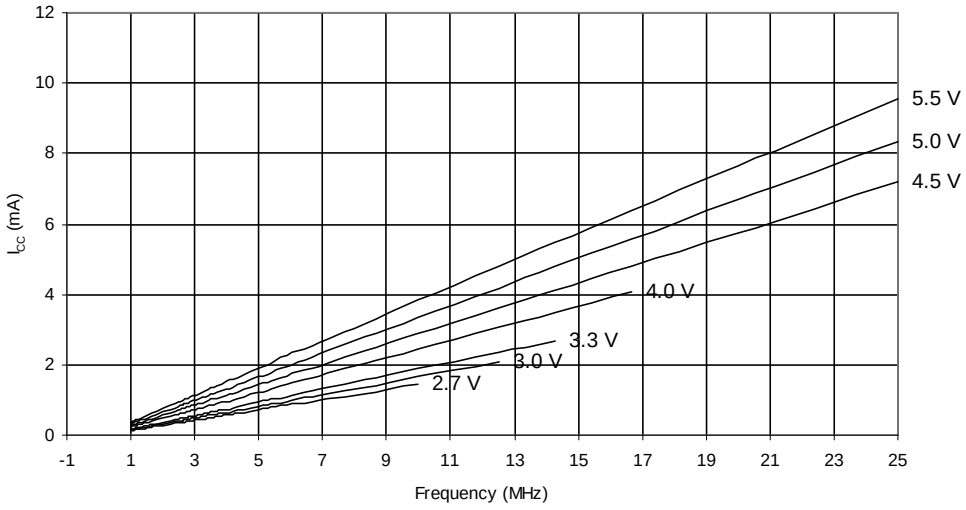


Figure 24-7. IIdle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

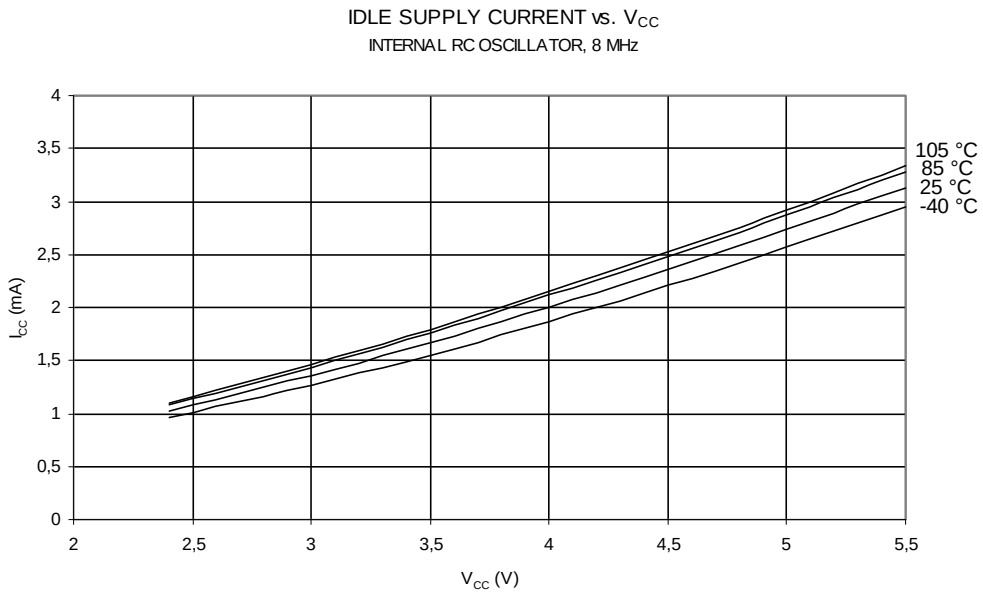
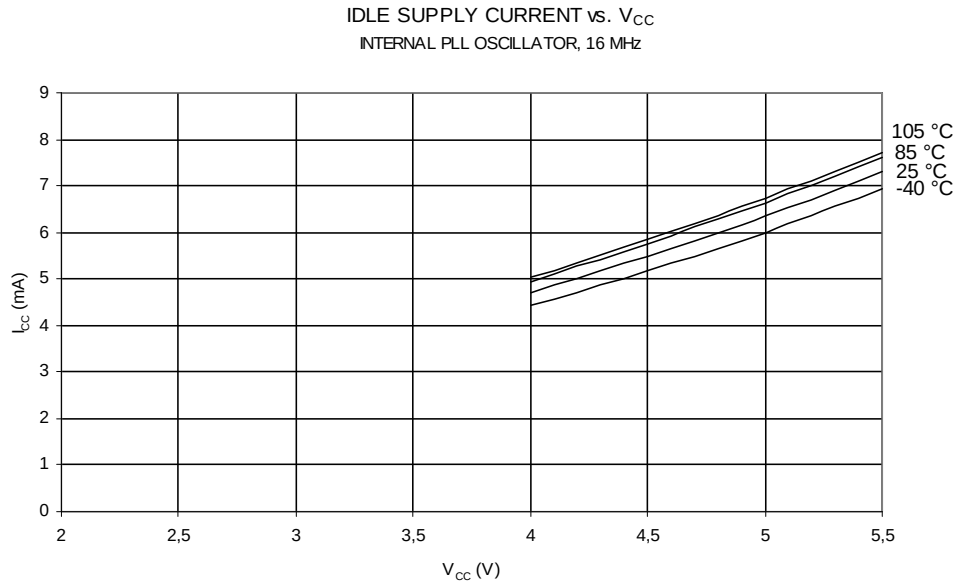


Figure 24-8. Idle Supply Current vs. V_{CC} (Internal PLL Oscillator, 16 MHz)



24.2.1 Using the Power Reduction Register

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules

are controlled by the Power Reduction Register. See "Power Reduction Register" on page 39 for details.

Table 24-1. Additional Current Consumption for the different I/O modules (absolute values)

PRR bit	Typical numbers	
	V_{CC} = 3V, F = 8MHz	V_{CC} = 5V, F = 16MHz
PRPSC2	350 uA	1.3 mA
PRPSC1	350 uA	1.3 mA
PRPSC0	350 uA	1.3 mA
PRTIM1	300 uA	1.15 mA
PRTIM0	200 uA	0.75 mA
PRSPI	250 uA	0.9 mA
PRUSART	550 uA	2 mA
PRADC	350 uA	1.3 mA

Table 24-2. Additional Current Consumption (percentage) in Active and Idle mode

PRR bit	Additional Current consumption compared to Active with external clock (see Figure 24-1 and Figure 24-2)	Additional Current consumption compared to Idle with external clock (see Figure 24-5 and Figure 24-6)
PRPSC2	10%	25%
PRPSC1	10%	25%
PRPSC0	10%	25%
PRTIM1	8.5%	22%
PRTIM0	4.3%	11%
PRSPI	5.3%	14%
PRUSART	15.6	36
PRADC	10.5%	25%

It is possible to calculate the typical current consumption based on the numbers from Table 24-2 for other V_{CC} and frequency settings than listed in Table 24-1.

24.2.1.1 Example 1

Calculate the expected current consumption in idle mode with USART, TIMER1, and SPI enabled at V_{CC} = 3.0V and F = 1MHz. From Table 24-2, third column, we see that we need to add 18% for the USART, 26% for the SPI, and 11% for the TIMER1 module. Reading from **Figure 24-5**, we find that the idle current consumption is ~0,17mA at V_{CC} = 3.0V and F = 1MHz. The total current consumption in idle mode with USART0, TIMER1, and SPI enabled, gives:

$$I_{CC\,total} \approx 0.17mA \bullet (1 + 0.36 + 0.22 + 0.14) \approx 0.29mA$$

24.2.1.2 Example 2

Same conditions as in example 1, but in active mode instead. From Table 24-2, second column we see that we need to add 3.3% for the USART, 4.8% for the SPI, and 2.0% for the TIMER1 module. Reading from **Figure 24-1**, we find that the active current consumption is ~0,6mA at V_{CC}

= 3.0V and F = 1MHz. The total current consumption in idle mode with USART, TIMER1, and SPI enabled, gives:

$$I_{CC\,total} \approx 0.6mA \bullet (1 + 0.156 + 0.085 + 0.053) \approx 0.77mA$$

24.2.1.3 *Example 3*

All I/O modules should be enabled. Calculate the expected current consumption in active mode at V_{CC} = 3.6V and F = 10MHz. We find the active current consumption without the I/O modules to be ~ 7.0mA (from **Figure 24-2**). Then, by using the numbers from Table 24-2 - second column, we find the total current consumption:

$$I_{CC\,total} \approx 7.0mA \bullet (1 + 0.1 + 0.1 + 0.1 + 0.085 + 0.043 + 0.053 + 0.156 + 0.105) \approx 12.2mA$$

24.2.2 Power consumption estimation with clock prescaling

Curves on Figure 24-1, Figure 24-2, Figure 24-5 and Figure 24-6 are measured with an external clock source and provides the power consumption of the microcontroller core at a given frequency.

The total power consumption of the microcontroller comes from the clock tree shown on Figure 7-1 on page 27, thus it is the sum of the following terms:

- **Oscillator**
- **PLL if used**
- **Clock prescaler**
- **Core**
- **Peripheral**
- **I/O**

When using the clock prescaler CLKPR to reduce the frequency (See “System Clock Prescaler” on page 35.), the power consumption of the oscillators and the prescaler can became the dominant factor of the active and idle power consumption.

Typical power consumption of the oscillators is given on Table 24-3.

Table 24-3. Power consumption of the clock system

Block	MHz	Typical numbers µA	
		V_{CC} = 3V	V_{CC} = 5V
Watch dog oscillator	0.128	5	5
XTAL Oscillator	4	60	60
	8	125	125
	16	250	250
RC Oscillator	8	50	50
PLL	64/16 (1)	700	700
Clock Prescaler	N (2)	N*30	N*50

1.The PLL generates a 64 MHz clock for the PSC and a 16 MHz clock for the CPU.
2.N is the frequency (in MHz) at the input of the clock prescaler

24.2.2.1 Example 1

Calculate the expected current consumption in idle mode with a 8 MHz Xtal oscillator feeding the PLL to generate a 16 MHz clock further divided by 16 using the CLKPR at V_{CC} = 3.0V. The system clock is thus 1 MHz. Peripherals and I/O are not active.

Reading from Table 24-3, we find that the Xtal oscillator current is 0.125 mA, the PLL current is 0.7 mA and the prescaler will need 16*30= 0.48 mA as it is clocked at 16 MHz.

Reading from Figure 24-5, we find that the idle current consumption of the core is ~0,17mA at V_{CC} = 3.0V and F = 1MHz.

The total current consumption in idle mode with the XTAL and the PLL gives:

$$I_{CCtotal} \approx 0.125 + 0.7 + 0.48 + 0.17 \approx 1.475 mA$$

24.2.2.2 Example 2

Calculate the expected current consumption in active mode with a 4 MHz Xtal oscillator clock further divided by 8 using the CLKPR at V_{CC} = 5.0V. The system clock is thus 0.5 MHz. Peripherals and I/O are not active.

Reading from Table 24-3, we find that the Xtal oscillator current is 0.06 mA and the prescaler will need 4*50= 0.2 mA as it is clocked at 4MHz.

Reading from Figure 24-1, we find that the idle current consumption of the core is ~0,6 mA at V_{CC} =5.0V and F = 0.5MHz.

The total typical current consumption in active mode with the 4MHz XTAL gives:

$$I_{CCtotal} \approx 0.06 + 0.2 + 0.6 \approx 0.86 mA$$

24.3 Power-Down Supply Current

Figure 24-9. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

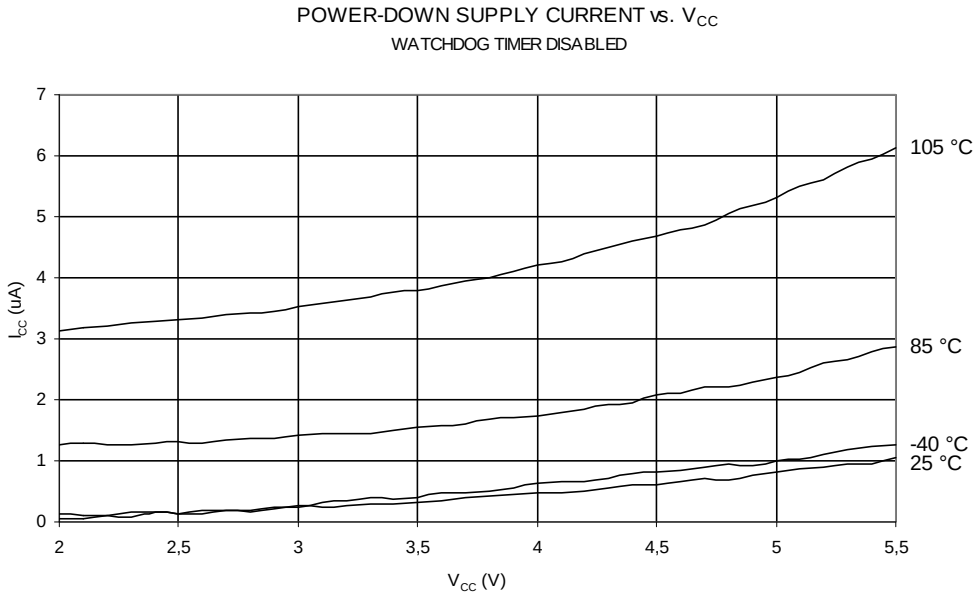
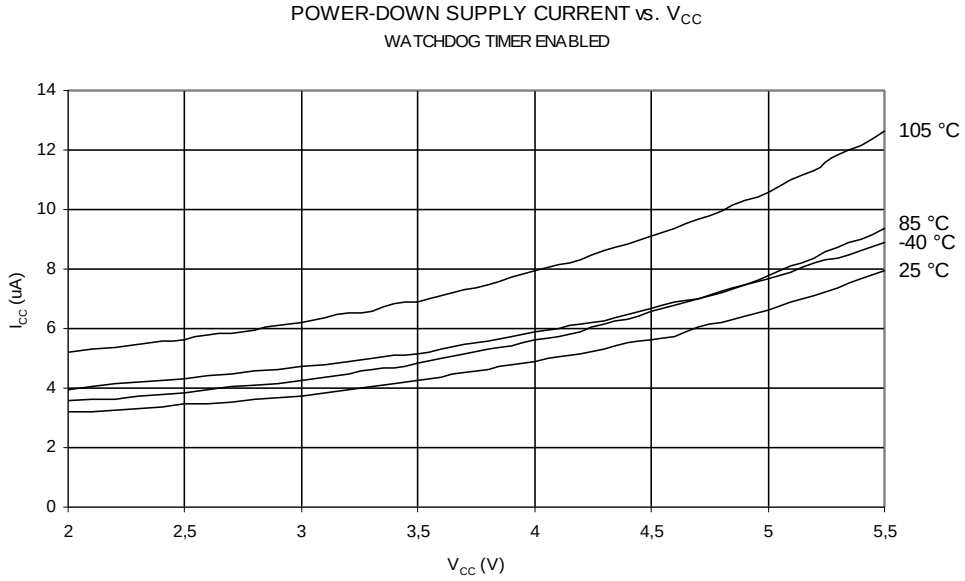


Figure 24-10. Power-Down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



24.4 Pin Pull-up

Figure 24-11. I/O Pin Pull-Up Resistor Current vs. Input Voltage (V_{CC} = 5V)

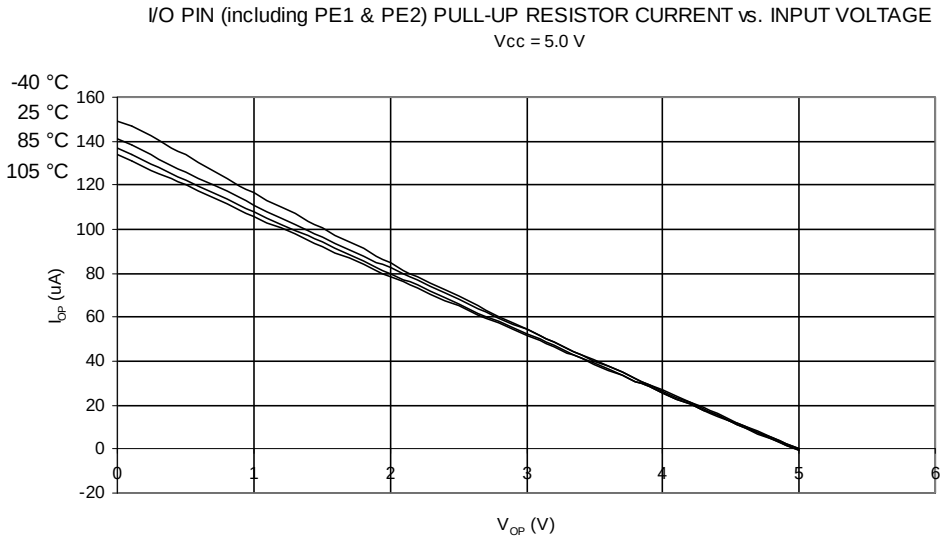


Figure 24-12. I/O Pin Pull-Up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

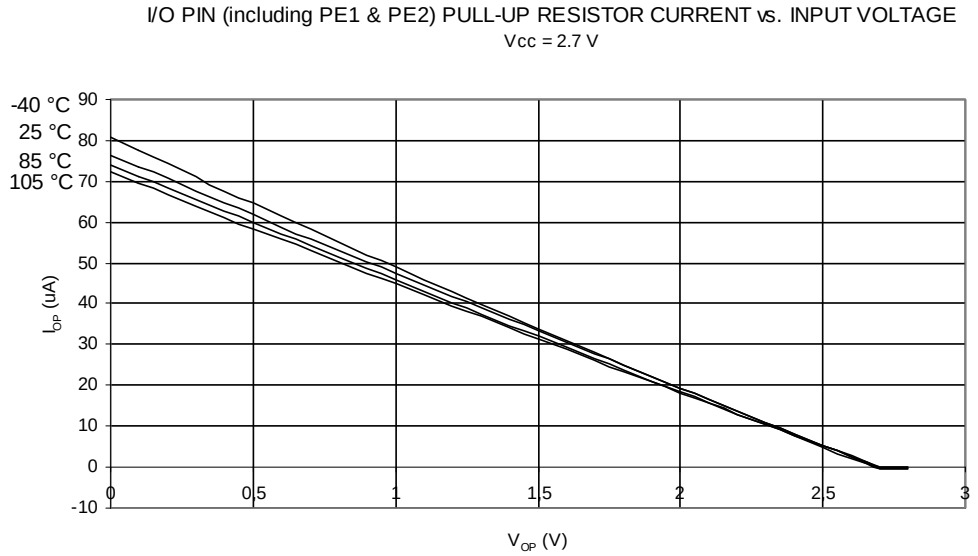


Figure 24-13. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)

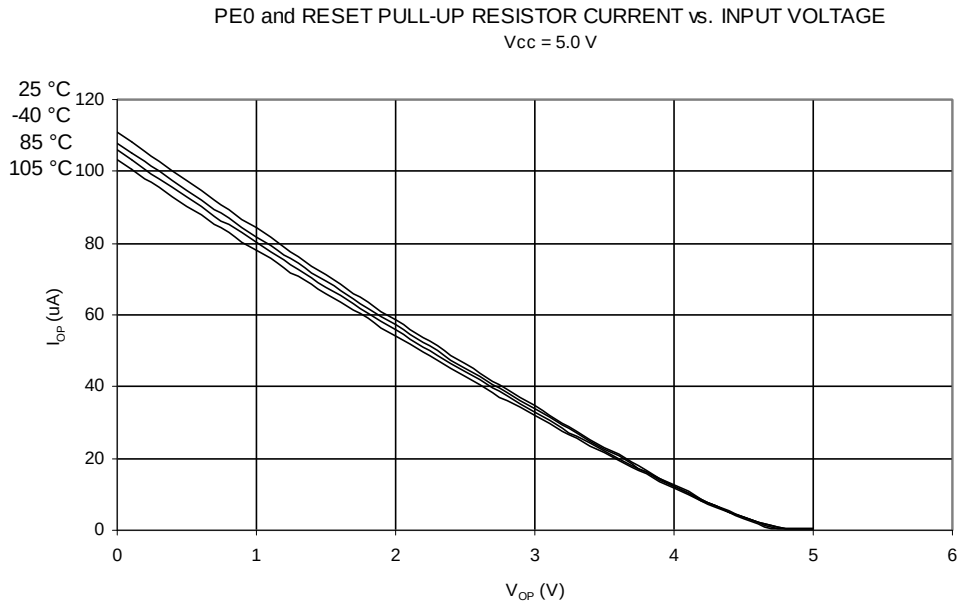
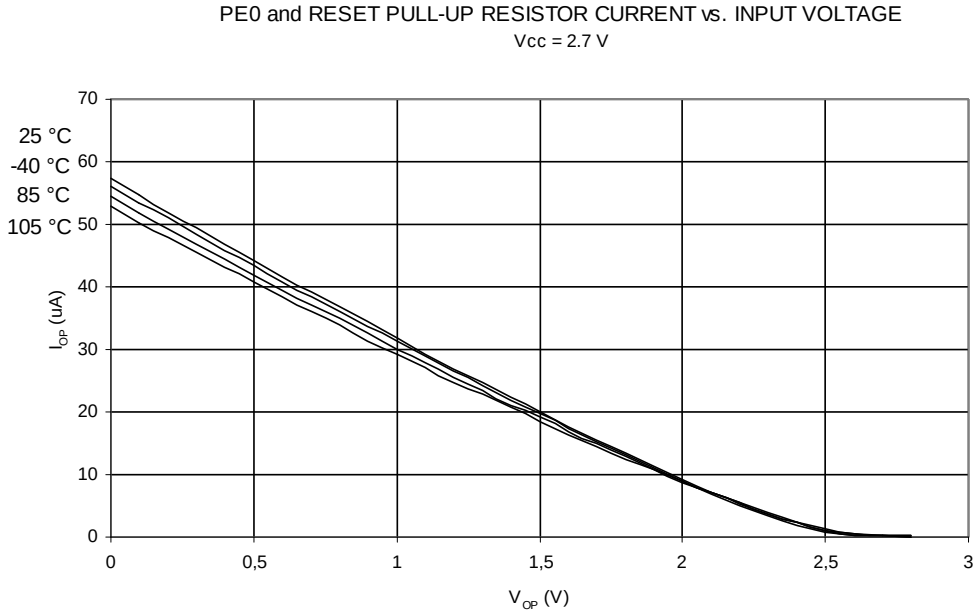


Figure 24-14. Reset Pull-Up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)



24.5 Pin Driver Strength

Figure 24-15. I/O Pin Source Current vs. Output Voltage (V_{CC} = 5V)

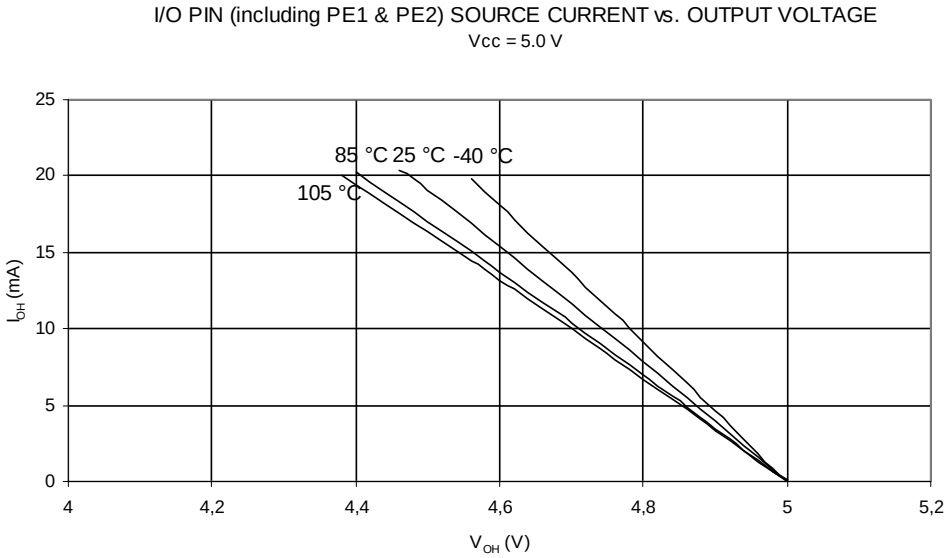


Figure 24-16. I/O Pin Source Current vs. Output Voltage (V_{CC} = 2.7V)

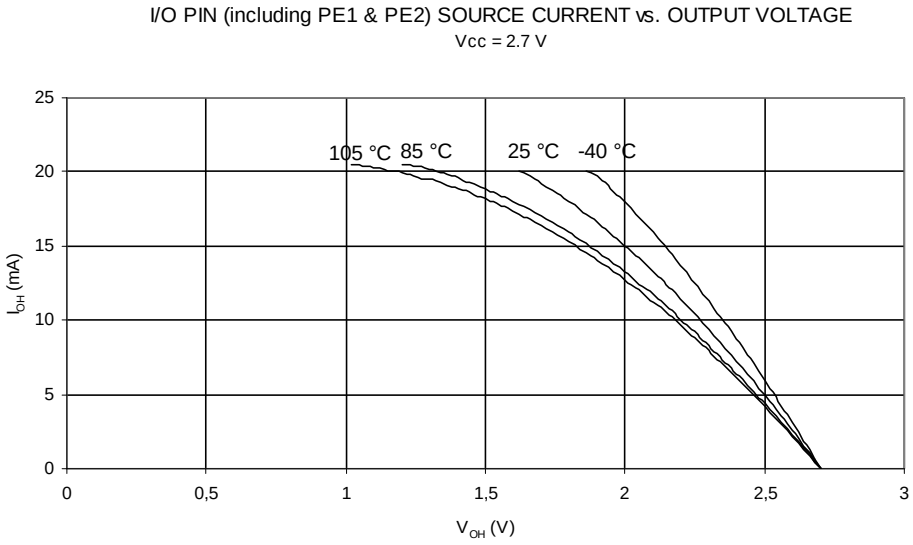


Figure 24-17. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

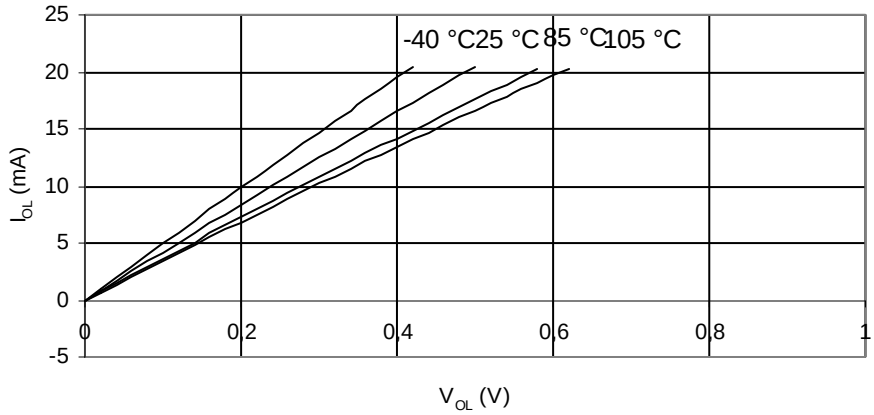
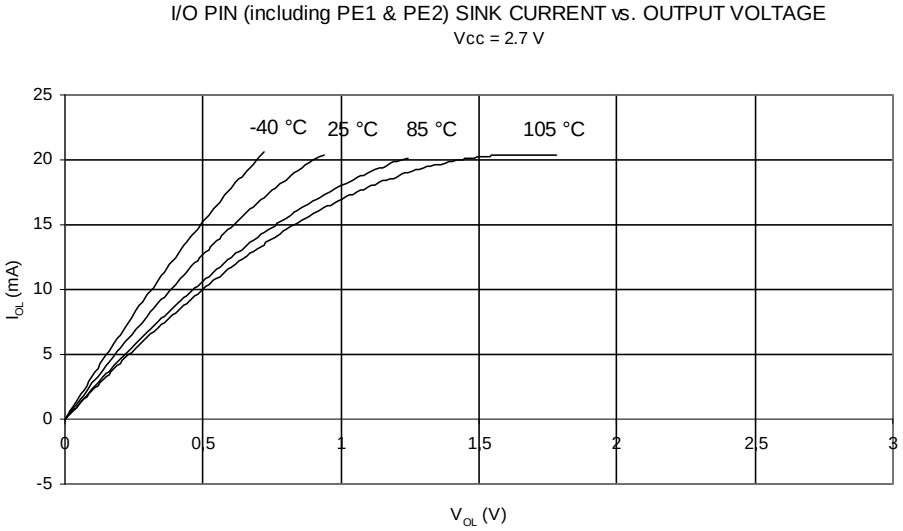


Figure 24-18. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)



24.6 Pin Thresholds and Hysteresis

Figure 24-19. I/O Pin Input Threshold Voltage vs. V_{CC} (VIH, I/O Pin Read As '1')

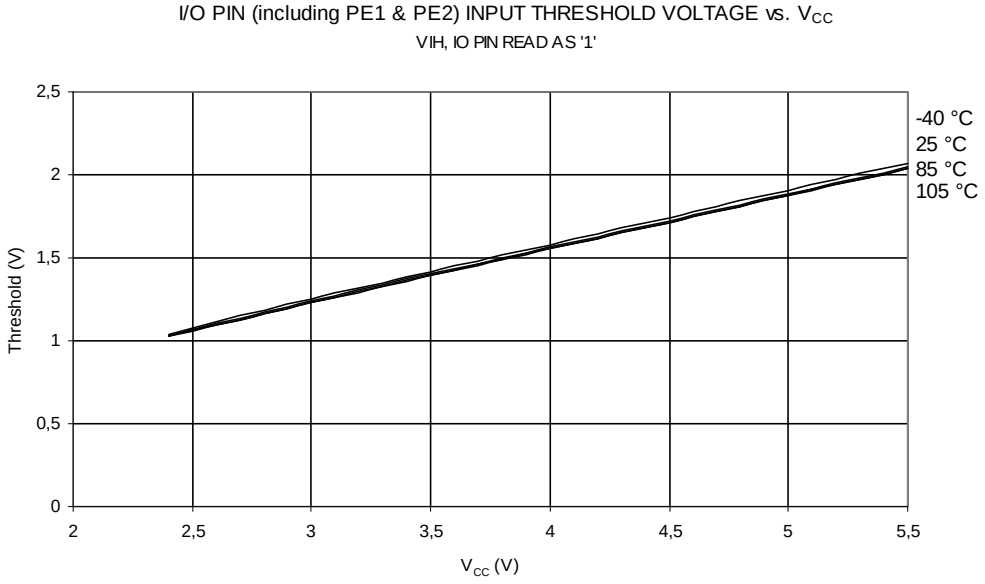


Figure 24-20. I/O Pin Input Threshold Voltage vs. V_{CC} (VIL, I/O Pin Read As '0')

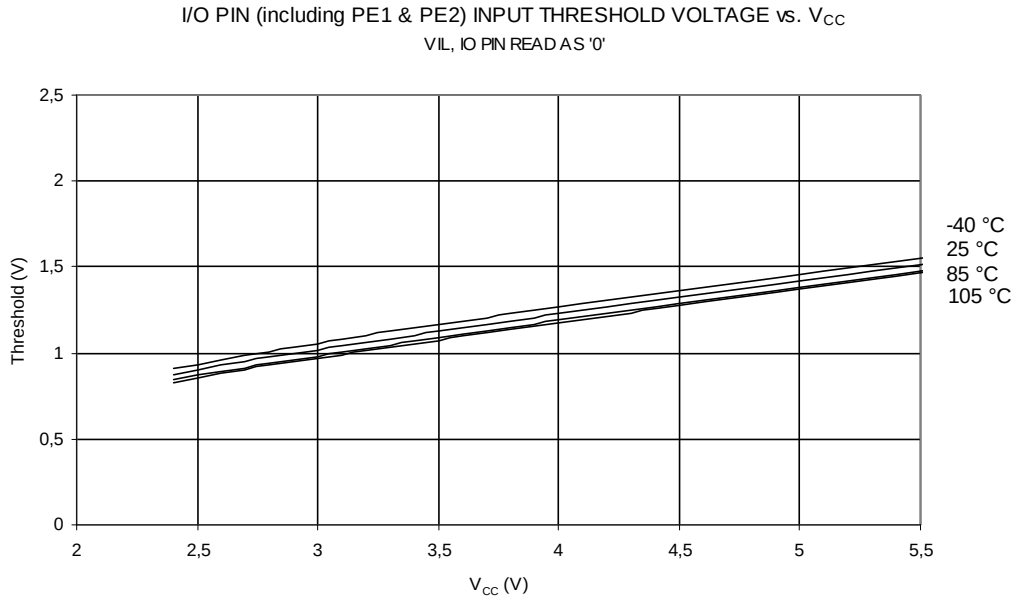


Figure 24-21. I/O Pin Input HysteresisVoltage vs. V_{CC}

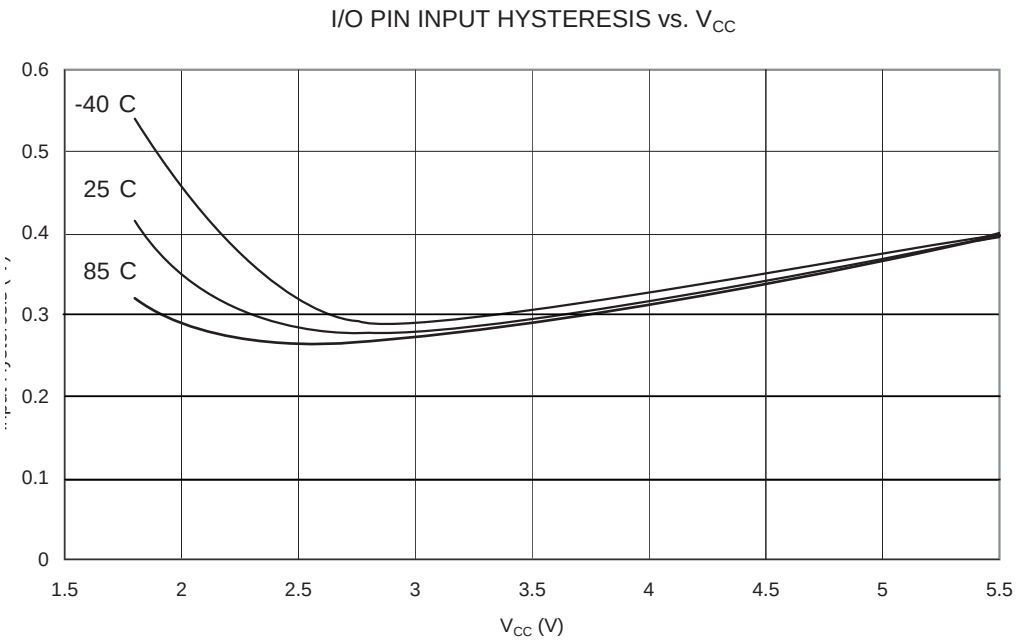


Figure 24-22. Reset Input Threshold Voltage vs. V_{CC} (VIH, Reset Pin Read As '1')

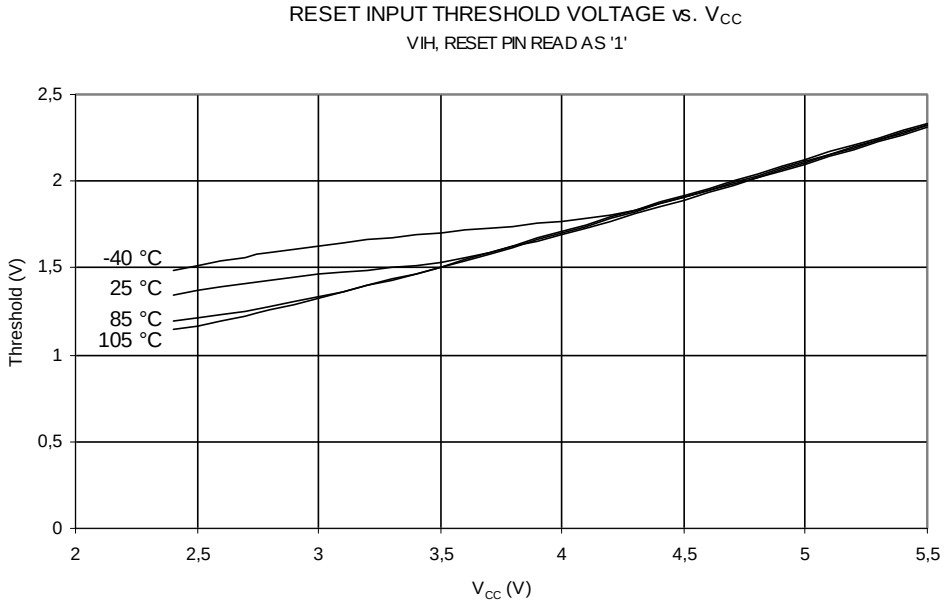


Figure 24-23. Reset Input Threshold Voltage vs. V_{CC} (VIL, Reset Pin Read As '0')

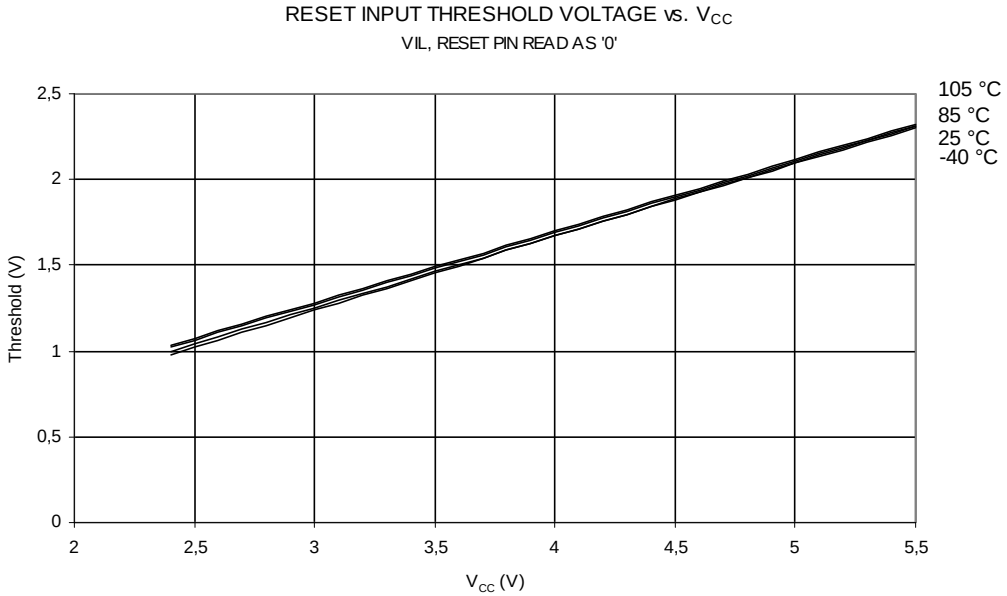


Figure 24-24. Reset Input Pin Hysteresis vs. V_{CC}

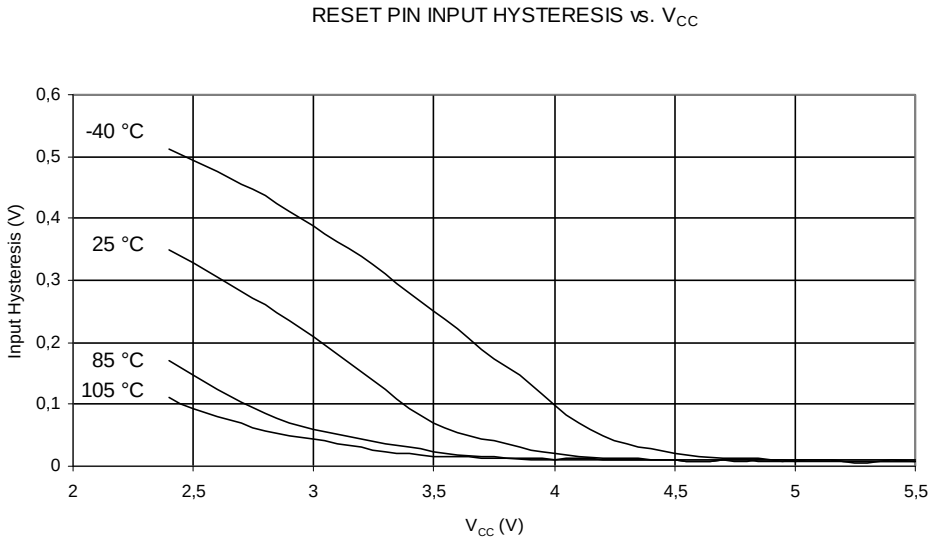


Figure 24-25. XTAL1 Input Threshold Voltage vs. V_{CC} (XTAL1 Pin Read As '1')

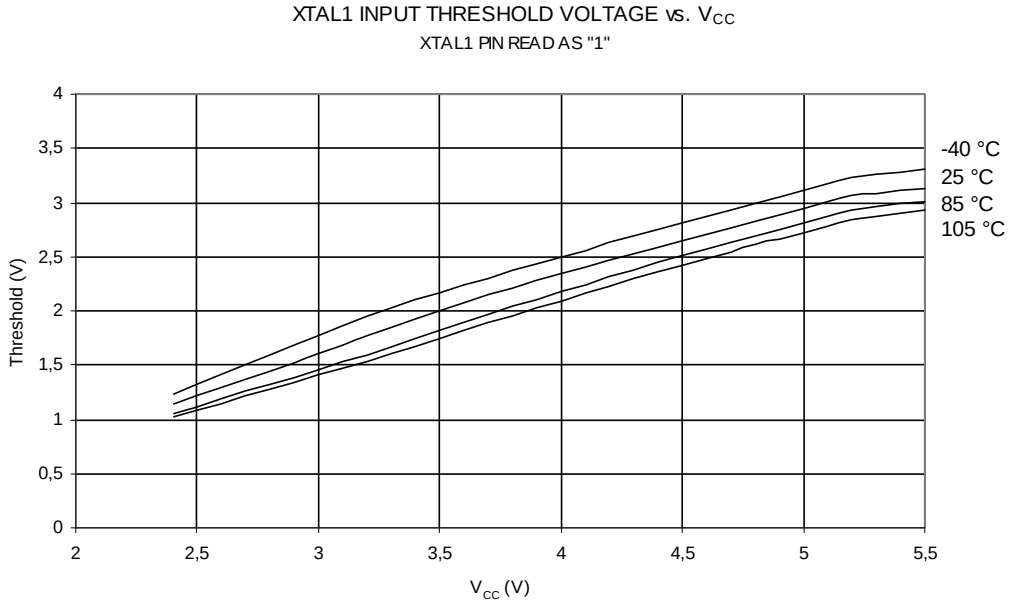


Figure 24-26. XTAL1 Input Threshold Voltage vs. V_{CC} (XTAL1 Pin Read As '0')

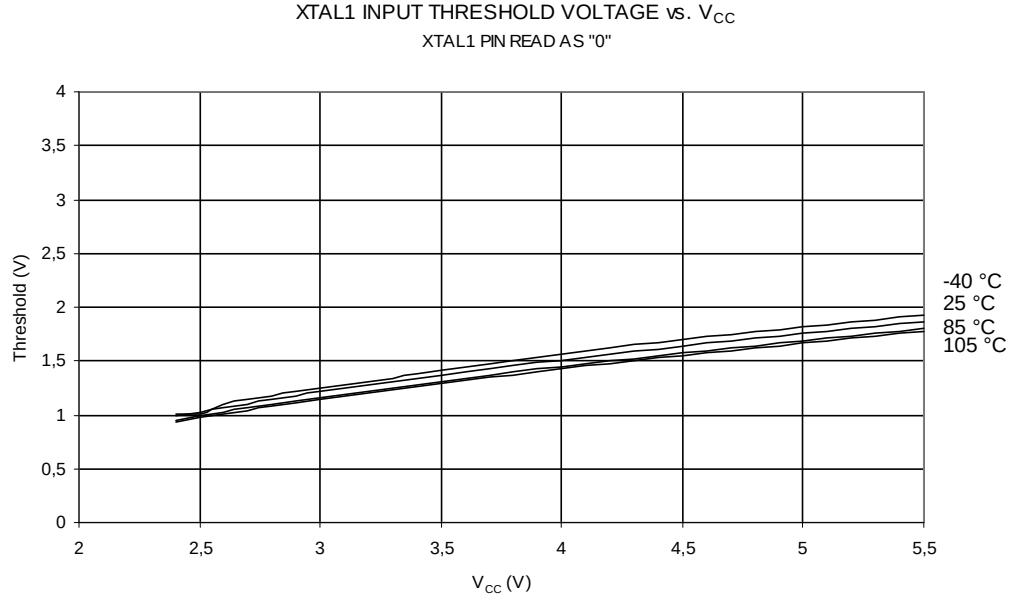


Figure 24-27. PE0 Input Threshold Voltage vs. V_{CC} (PE0 Pin Read As '1')

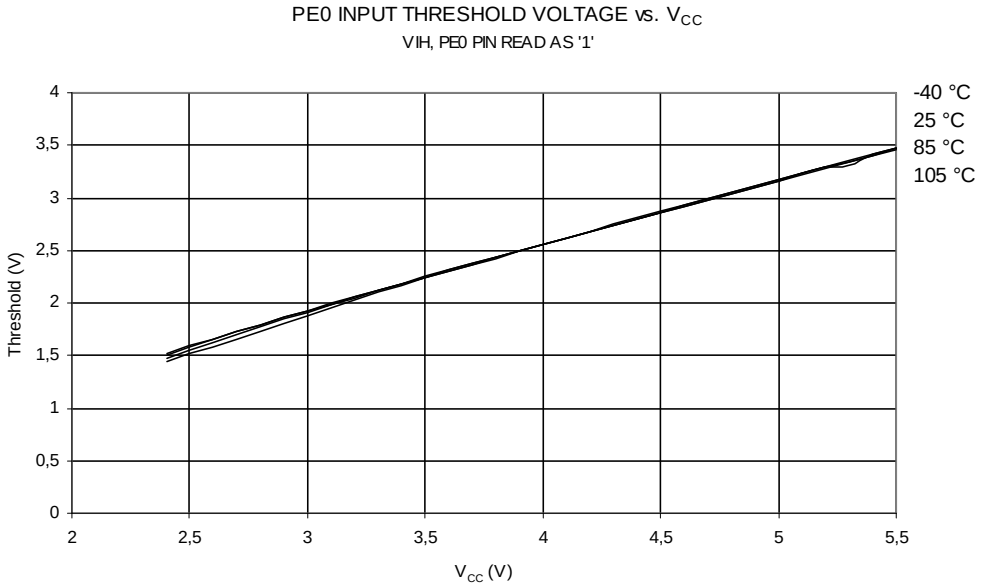
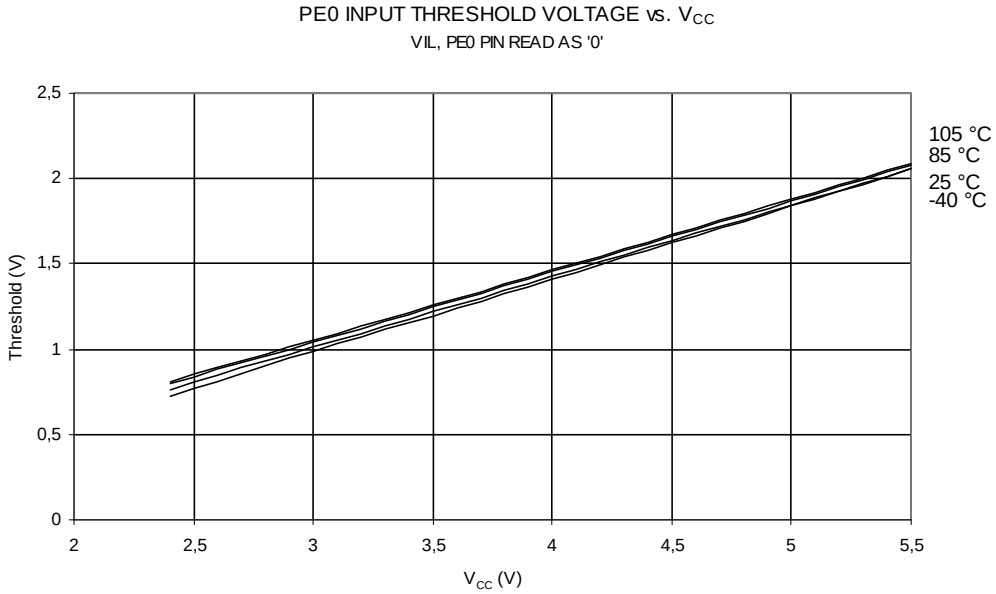


Figure 24-28. PE0 Input Threshold Voltage vs. V_{CC} (PE0 Pin Read As '0')



24.7 BOD Thresholds and Analog Comparator Offset

Figure 24-29. BOD Thresholds vs. Temperature (BODLEVEL Is 4.3V)

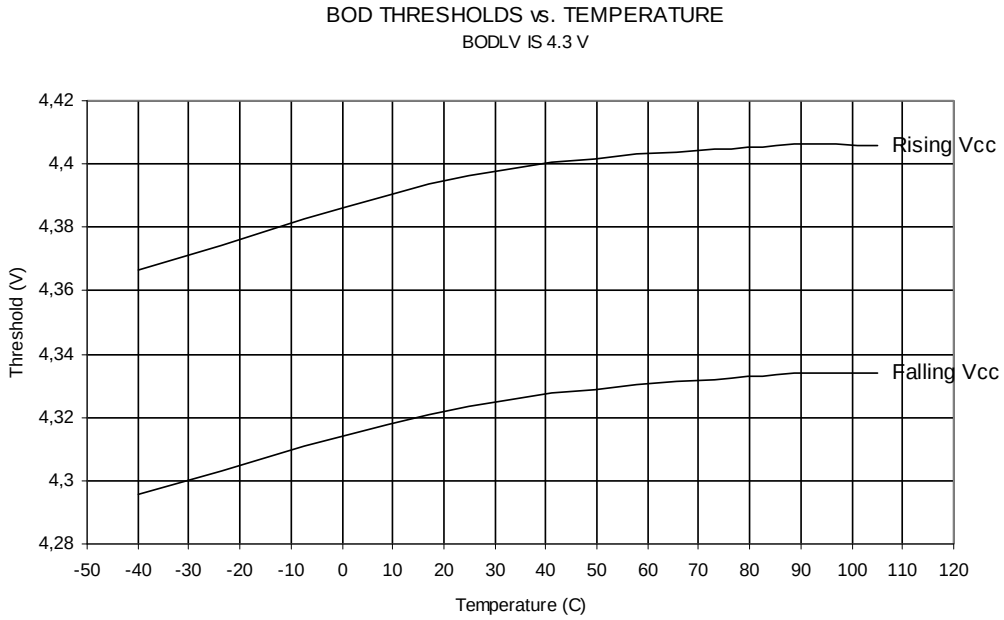


Figure 24-30. BOD Thresholds vs. Temperature (BODLEVEL Is 2.7V)

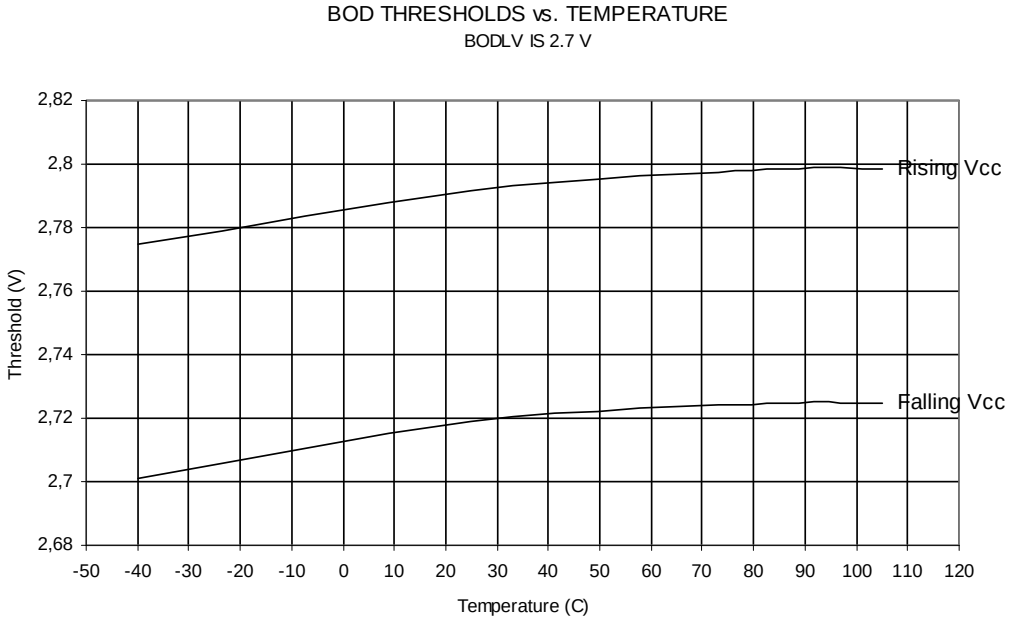
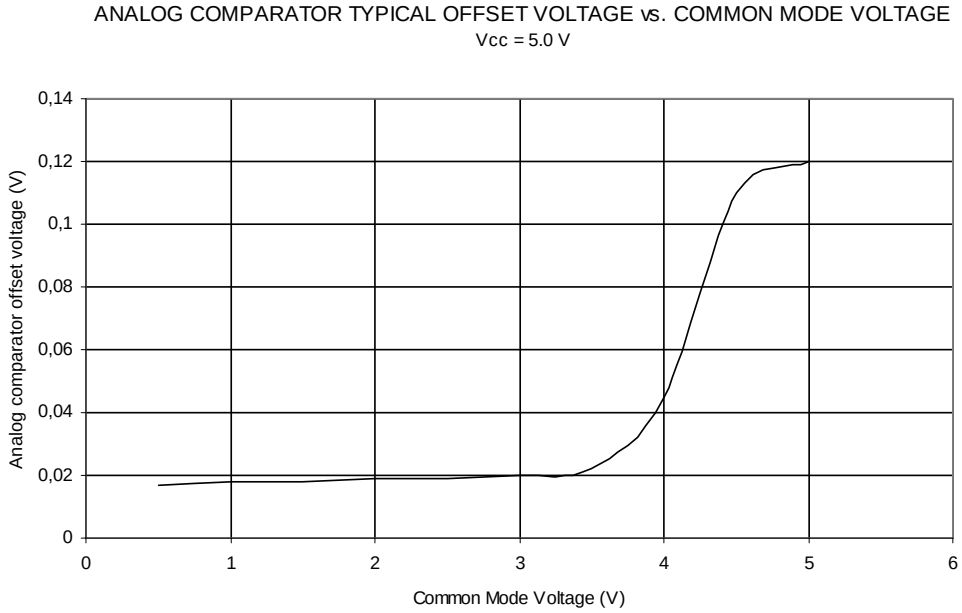
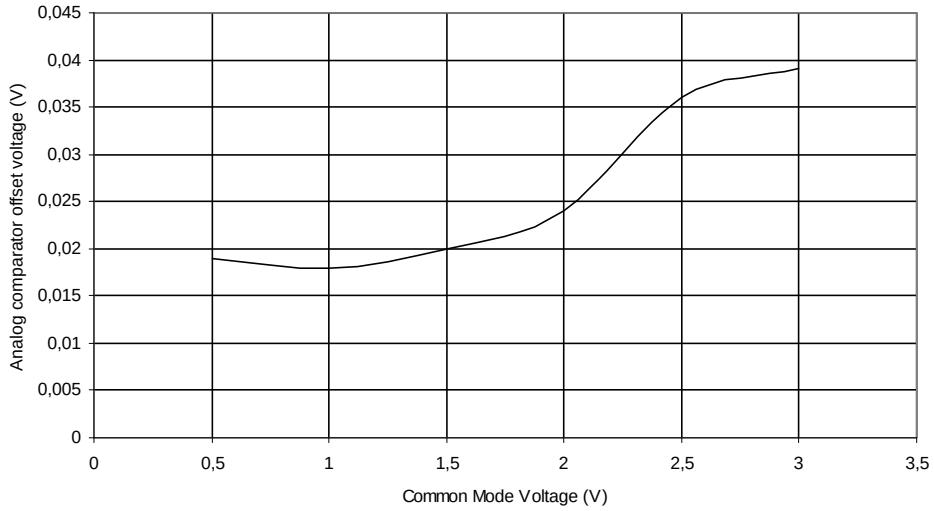


Figure 24-31. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC}=5V)



Note: corrected on AT90PWM2B/3B to allow almost full scale use.

Figure 24-32. Analog Comparator Offset Voltage vs. Common Mode Voltage (V_{CC}=3V)



Note: corrected on AT90PWM2B/3B to allow almost full scale use.

24.8 Analog Reference

Figure 24-33. AREF Voltage vs. V_{CC}

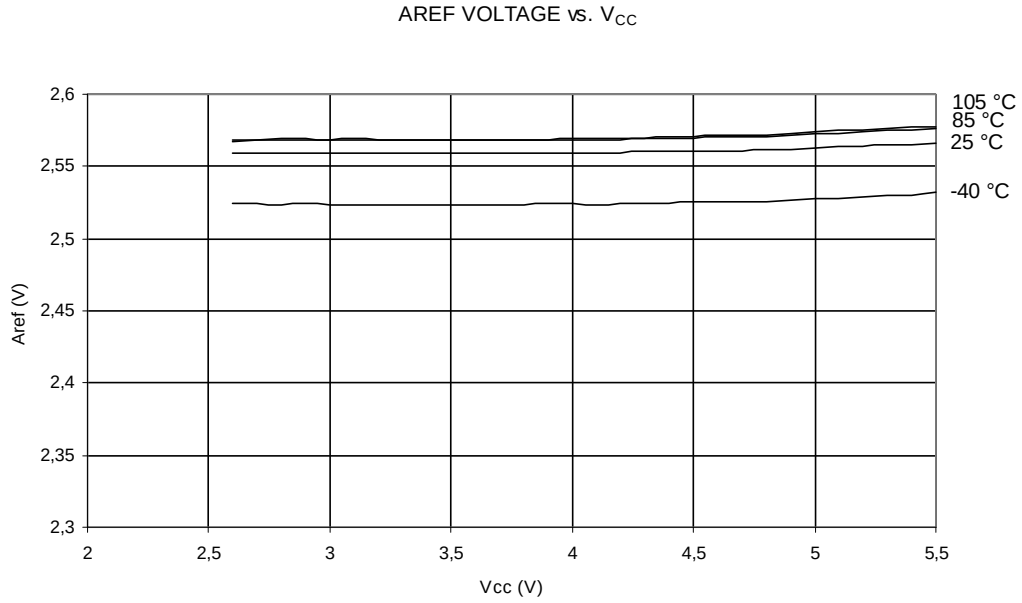
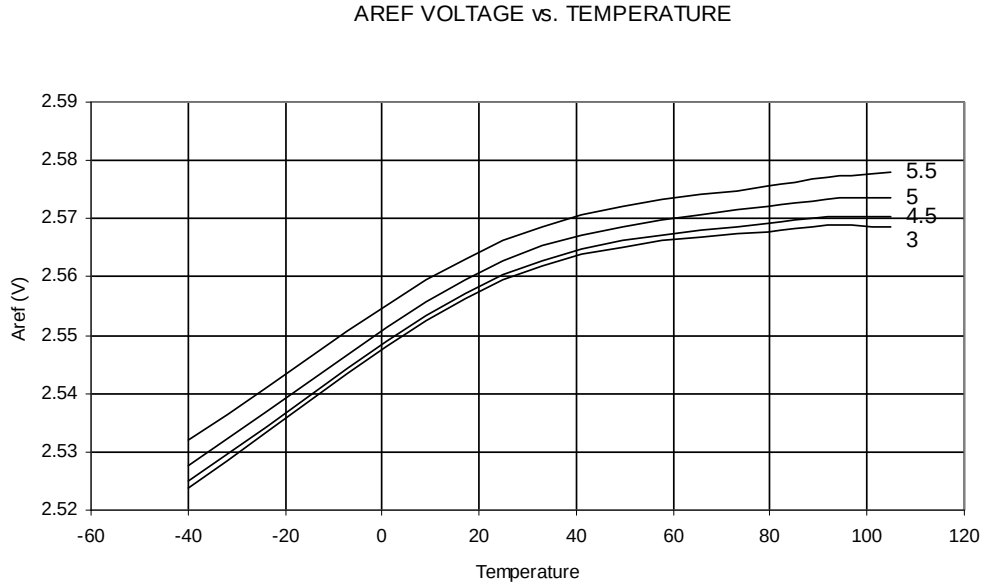


Figure 24-34. AREF Voltage vs. Temperature



24.9 Internal Oscillator Speed

Figure 24-35. Watchdog Oscillator Frequency vs. V_{CC}

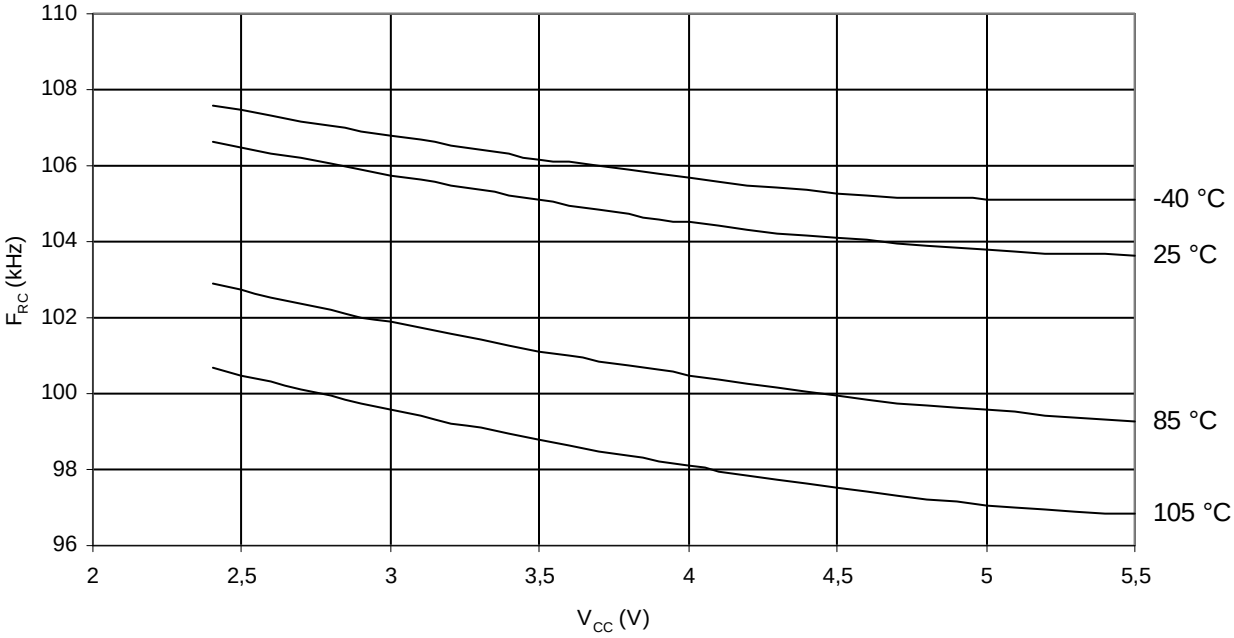


Figure 24-36. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

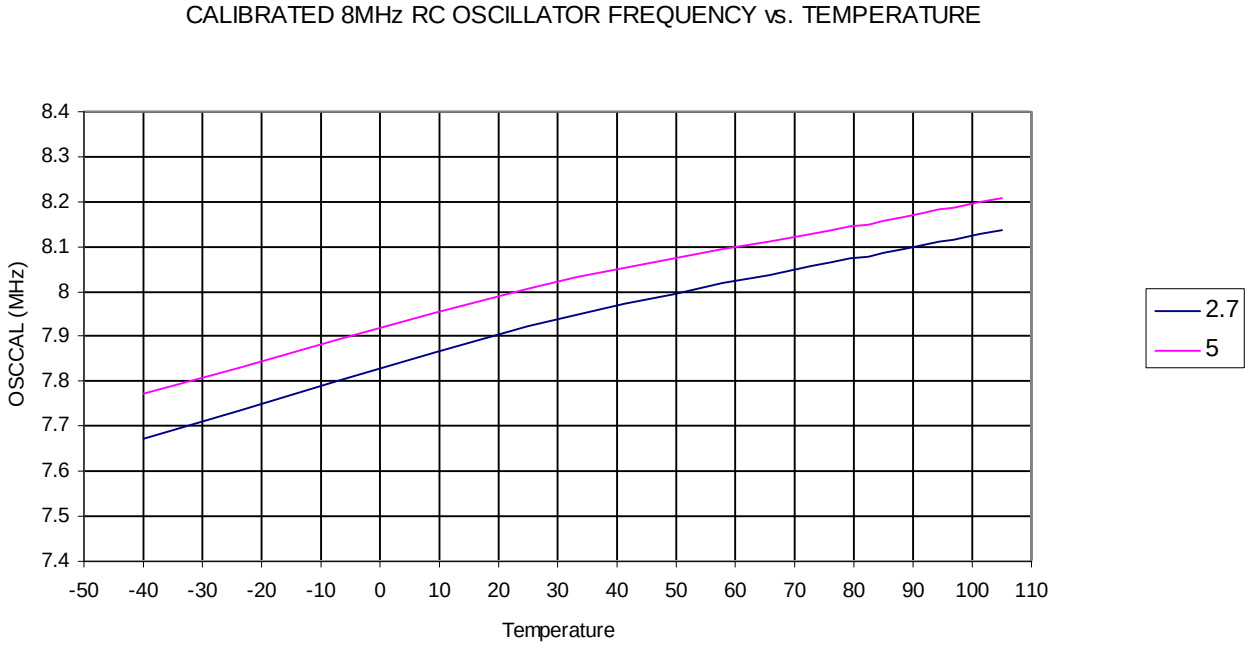


Figure 24-37. Calibrated 8 MHz RC Oscillator Frequency vs. V_{CC}

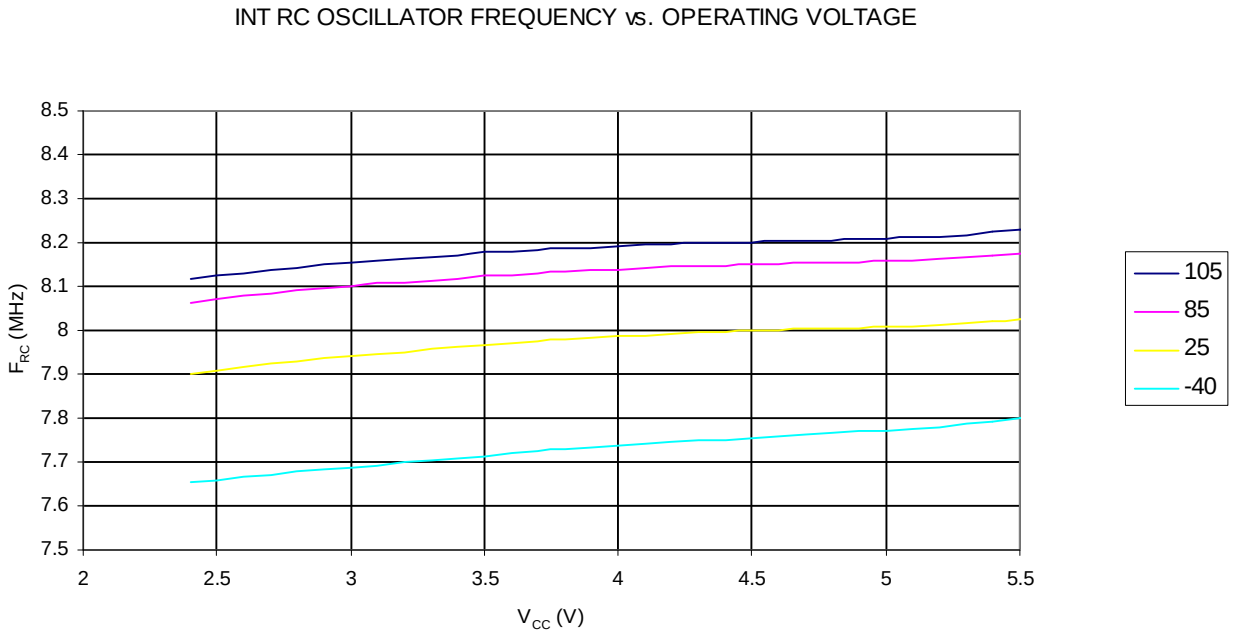
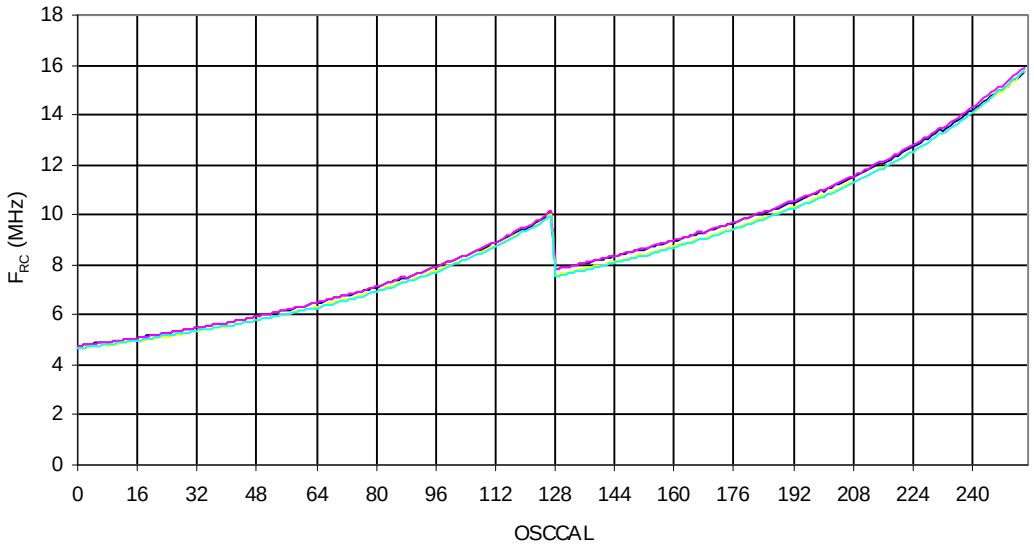


Figure 24-38. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value



24.10 Current Consumption of Peripheral Units

Figure 24-39. Brownout Detector Current vs. V_{CC}

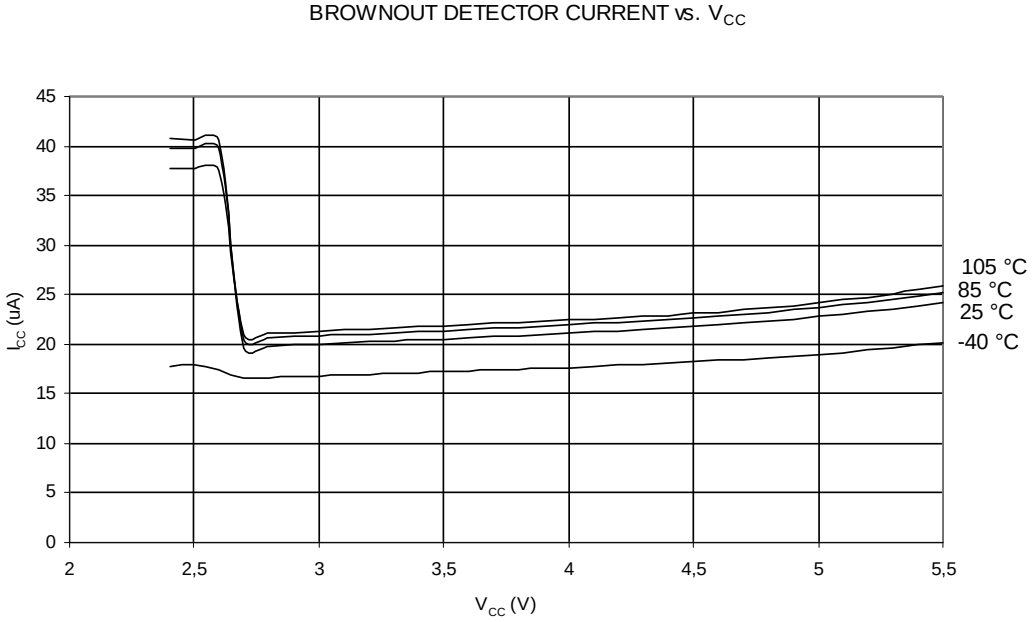


Figure 24-40. ADC Current vs. V_{CC} (ADC at 50 kHz)

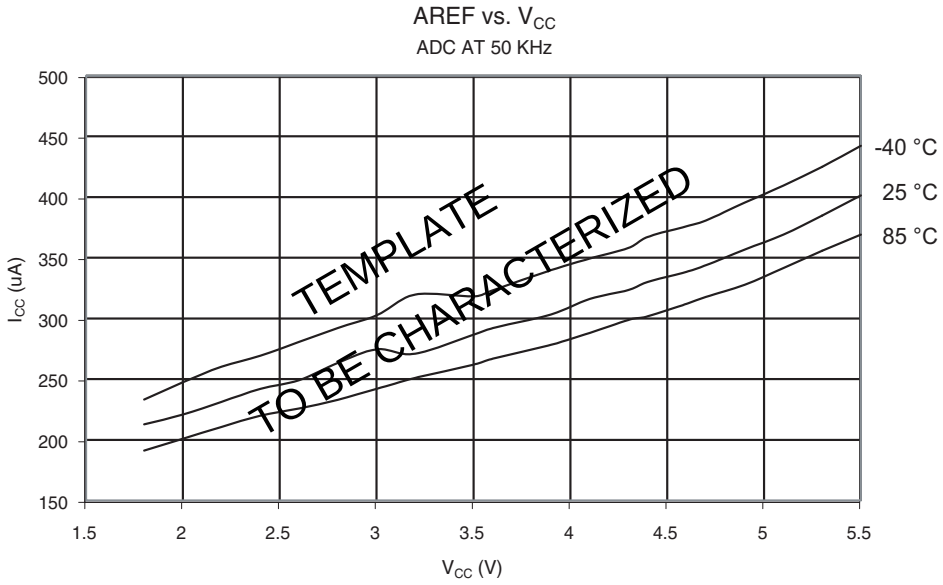


Figure 24-41. Aref Current vs. V_{CC} (ADC at 1 MHz)

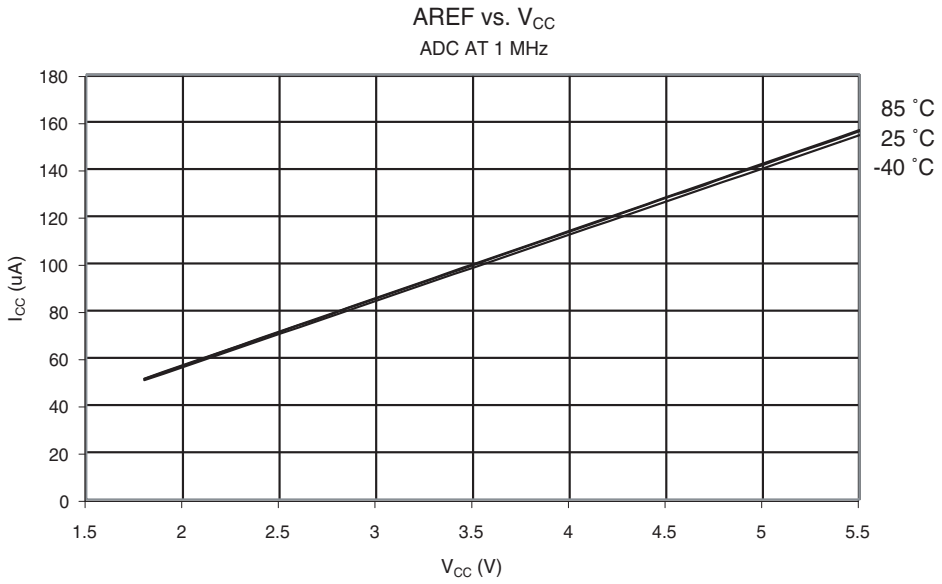


Figure 24-42. Analog Comparator Current vs. V_{CC}

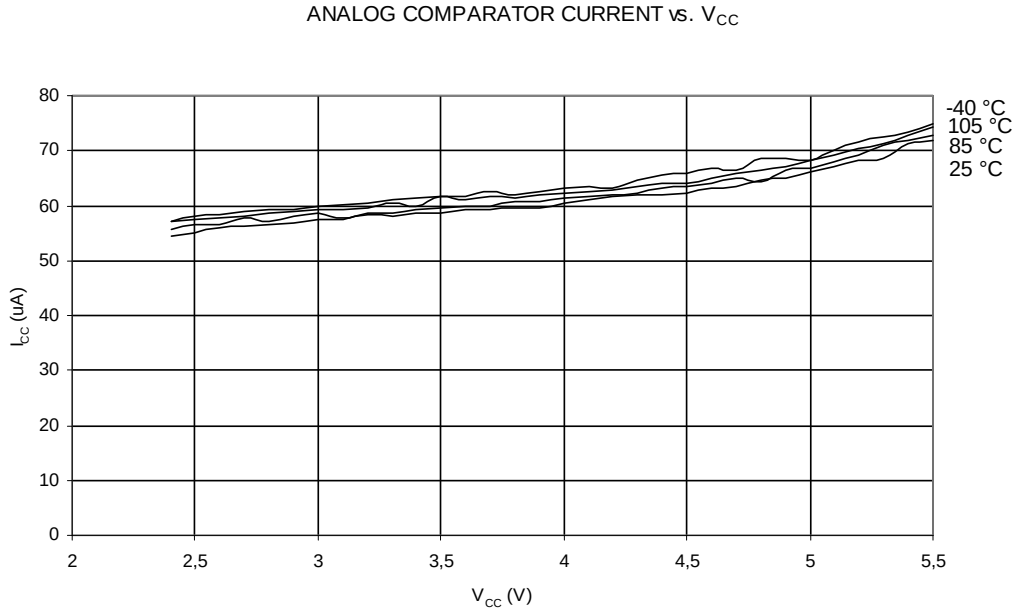
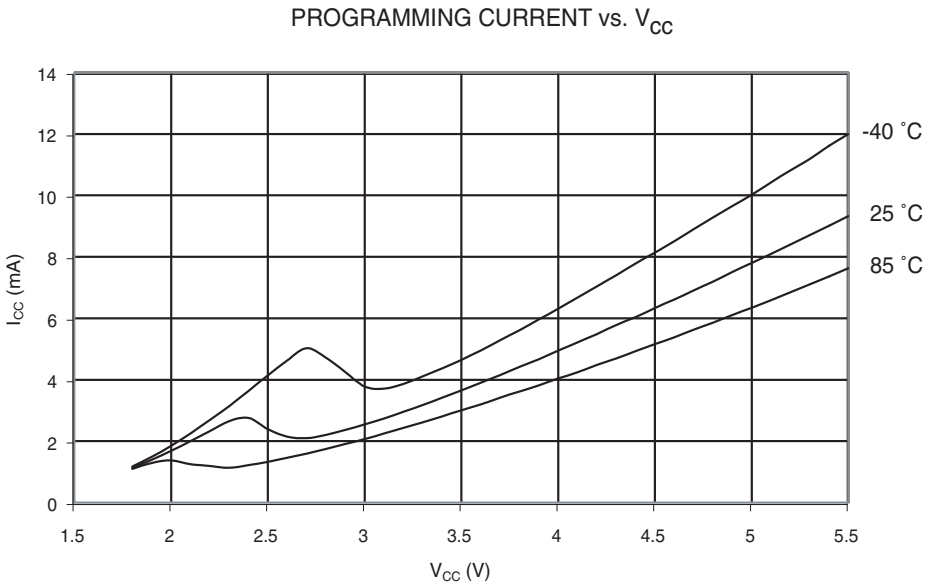


Figure 24-43. Programming Current vs. V_{CC}



24.11 Current Consumption in Reset and Reset Pulse width

Figure 24-44. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current through the Reset Pull-up)

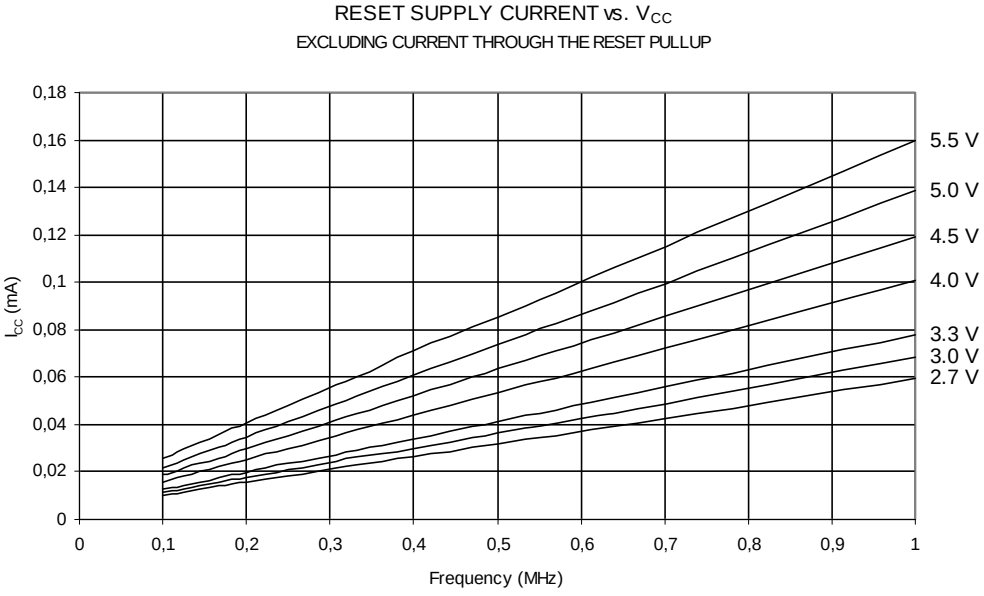


Figure 24-45. Reset Supply Current vs. V_{CC} (1 - 24 MHz, Excluding Current through the Reset Pull-up)

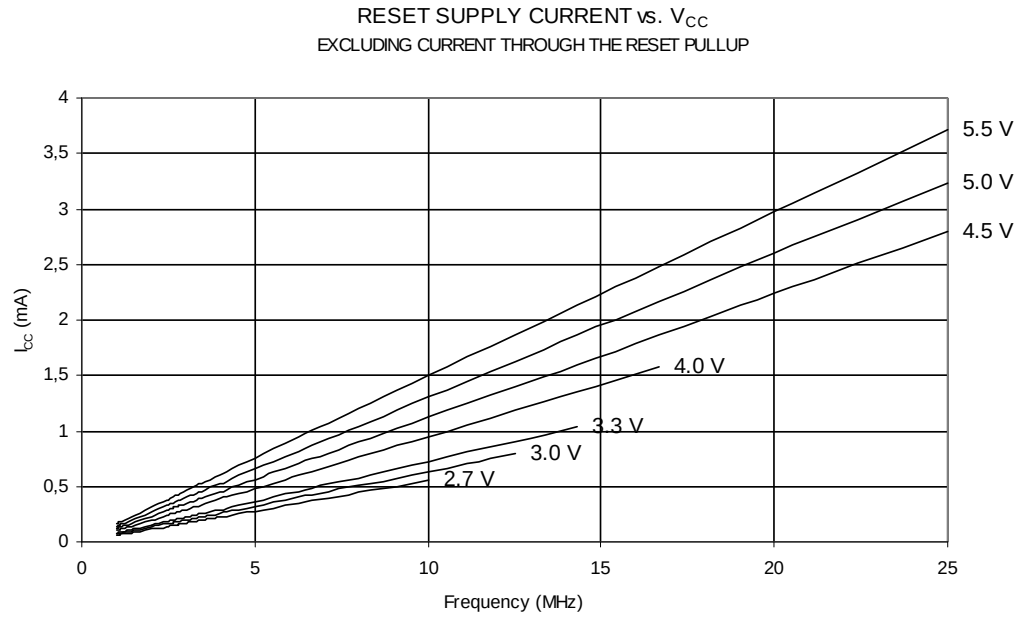


Figure 24-46. Reset Supply Current vs. V_{CC} (Clock Stopped, Excluding Current through the Reset Pull-up)

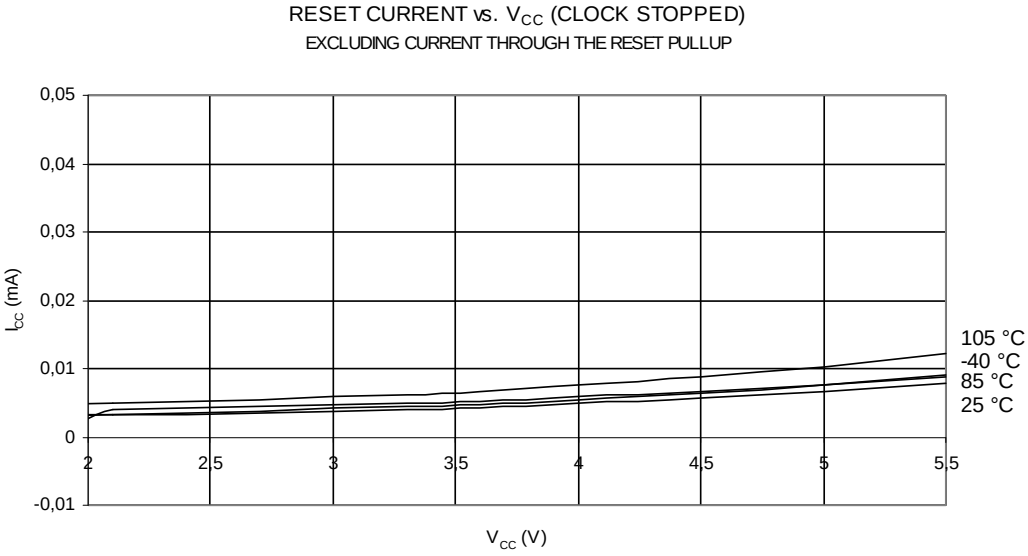
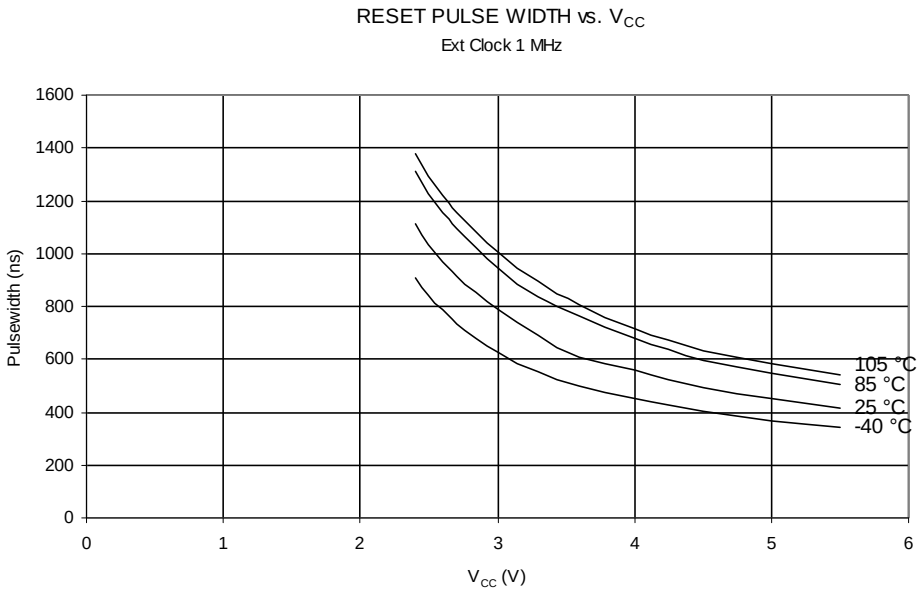


Figure 24-47. Reset Pulse Width vs. V_{CC}



25. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	PICR2H									page 162
(0xFE)	PICR2L									page 162
(0xFD)	PFRC2B	PCAE2B	PISEL2B	PELEV2B	PFLTE2B	PRFM2B3	PRFM2B2	PRFM2B1	PRFM2B0	page 161
(0xFC)	PFRC2A	PCAE2A	PISEL2A	PELEV2A	PFLTE2A	PRFM2A3	PRFM2A2	PRFM2A1	PRFM2A0	page 161
(0xFB)	PCTL2	PPRE21	PPRE20	PBFM2	PAOC2B	PAOC2A	PARUN2	PCCYC2	PRUN2	page 160
(0xFA)	PCNF2	PFIFTY2	PALOCK2	PLOCK2	PMODE21	PMODE20	POP2	PCLKSEL2	POME2	page 157
(0xF9)	OCR2RBH									page 157
(0xF8)	OCR2RBL									page 157
(0xF7)	OCR2SBH									page 157
(0xF6)	OCR2SBL									page 157
(0xF5)	OCR2RAH									page 156
(0xF4)	OCR2RAL									page 156
(0xF3)	OCR2SAH									page 156
(0xF2)	OCR2SAL									page 156
(0xF1)	POM2	POMV2B3	POMV2B2	POMV2B1	POMV2B0	POMV2A3	POMV2A2	POMV2A1	POMV2A0	page 163
(0xF0)	PSOC2	POS23	POS22	PSYNC21	PSYNC20	POEN2D	POEN2B	POEN2C	POEN2A	page 155
(0xEF)	PICR1H									
(0xEE)	PICR1L									
(0xED)	PFRC1B	PCAE1B	PISEL1B	PELEV1B	PFLTE1B	PRFM1B3	PRFM1B2	PRFM1B1	PRFM1B0	page 161
(0xEC)	PFRC1A	PCAE1A	PISEL1A	PELEV1A	PFLTE1A	PRFM1A3	PRFM1A2	PRFM1A1	PRFM1A0	page 161
(0xEB)	PCTL1								PRUN1	page 160
(0xEA)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE9)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE8)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE7)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE6)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE5)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE4)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE3)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE2)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE1)	**Reserved**	–	–	–	–	–	–	–	–	
(0xE0)	PSOC1	–	–	PSYNC11	PSYNC10	–	POEN1B	–	POEN1A	
(0xDF)	PICR0H									page 162
(0xDE)	PICR0L									page 162
(0xDD)	PFRC0B	PCAE0B	PISEL0B	PELEV0B	PFLTE0B	PRFM0B3	PRFM0B2	PRFM0B1	PRFM0B0	page 161
(0xDC)	PFRC0A	PCAE0A	PISEL0A	PELEV0A	PFLTE0A	PRFM0A3	PRFM0A2	PRFM0A1	PRFM0A0	page 161
(0xDB)	PCTL0	PPRE01	PPRE00	PBFM0	PAOC0B	PAOC0A	PARUN0	PCCYC0	PRUN0	page 158
(0xDA)	PCNF0	PFIFTY0	PALOCK0	PLOCK0	PMODE01	PMODE00	POP0	PCLKSEL0	-	page 157
(0xD9)	OCR0RBH									page 157
(0xD8)	OCR0RBL									page 157
(0xD7)	OCR0SBH									page 157
(0xD6)	OCR0SBL									page 157
(0xD5)	OCR0RAH									page 156
(0xD4)	OCR0RAL									page 156
(0xD3)	OCR0SAH									page 156
(0xD2)	OCR0SAL									page 156
(0xD1)	**Reserved**	–	–	–	–	–	–	–	–	
(0xD0)	PSOC0	–	–	PSYNC01	PSYNC00	–	POEN0B	–	POEN0A	page 155
(0xCF)	**Reserved**	–	–	–	–	–	–	–	–	
(0xCE)	**Reserved**	–	–	–	–	–	–	–	–	
(0xCD)	**Reserved**	–	–	–	–	–	–	–	–	
(0xCC)	**Reserved**	–	–	–	–	–	–	–	–	
(0xCB)	**Reserved**	–	–	–	–	–	–	–	–	
(0xCA)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC9)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC8)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC7)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC6)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC5)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC4)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC3)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC2)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC1)	**Reserved**	–	–	–	–	–	–	–	–	
(0xC0)	**Reserved**	–	–	–	–	–	–	–	–	
(0xBF)	**Reserved**	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBE)	**Reserved**	–	–	–	–	–	–	–	–	
(0xBD)	**Reserved**	–	–	–	–	–	–	–	–	
(0xBC)	**Reserved**	–	–	–	–	–	–	–	–	
(0xBB)	**Reserved**	–	–	–	–	–	–	–	–	
(0xBA)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB9)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB8)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB7)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB6)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB5)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB4)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB3)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB2)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB1)	**Reserved**	–	–	–	–	–	–	–	–	
(0xB0)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAF)	**AC2CON**	AC2EN	AC2IE	AC2IS1	AC2IS0	AC2SADE-	AC2M2	AC2M1	AC2M0	page 178
(0xAD)	**AC0CON**	AC0EN	AC0IE	AC0IS1	AC0IS0	-	AC0M2	AC0M1	AC0M0	page 177
(0xAC)	**Reserved**	–	–	–	–	–	–	–	–	page 258
(0xAB)	**Reserved**	–	–	–	–	–	–	–	–	page 258
(0xAA)	**Reserved**	–	–	–	–	–	–	–	–	page 257
(0xA9)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA8)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA7)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA6)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA5)	PIM2	-	-	PSEIE2	PEVE2B	PEVE2A	-	-	PEOPE2	page 164
(0xA4)	PIFR2	-	-	PSEI2	PEV2B	PEV2A	PRN21	PRN20	PEOP2	page 164
(0xA3)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA2)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA1)	PIM0	-	-	PSEIE0	PEVE0B	PEVE0A	-	-	PEOPE0	page 164
(0xA0)	PIFR0	-	-	PSEI0	PEV0B	PEV0A	PRN01	PRN00	PEOP0	page 164
(0x9F)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9E)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9D)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9C)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9B)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9A)	**Reserved**	–	–	–	–	–	–	–	–	
(0x99)	**Reserved**	–	–	–	–	–	–	–	–	
(0x98)	**Reserved**	–	–	–	–	–	–	–	–	
(0x97)	**Reserved**	–	–	–	–	–	–	–	–	
(0x96)	**Reserved**	–	–	–	–	–	–	–	–	
(0x95)	**Reserved**	–	–	–	–	–	–	–	–	
(0x94)	**Reserved**	–	–	–	–	–	–	–	–	
(0x93)	**Reserved**	–	–	–	–	–	–	–	–	
(0x92)	**Reserved**	–	–	–	–	–	–	–	–	
(0x91)	**Reserved**	–	–	–	–	–	–	–	–	
(0x90)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8F)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8E)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8D)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8C)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8B)	**OCR1BH**	OCR1B15	OCR1B14	OCR1B13	OCR1B12	OCR1B11	OCR1B10	OCR1B9	OCR1B8	page 120
(0x8A)	**OCR1BL**	OCR1B7	OCR1B6	OCR1B5	OCR1B4	OCR1B3	OCR1B2	OCR1B1	OCR1B0	page 120
(0x89)	**OCR1AH**	OCR1A15	OCR1A14	OCR1A13	OCR1A12	OCR1A11	OCR1A10	OCR1A9	OCR1A8	page 120
(0x88)	**OCR1AL**	OCR1A7	OCR1A6	OCR1A5	OCR1A4	OCR1A3	OCR1A2	OCR1A1	OCR1A0	page 120
(0x87)	**ICR1H**	ICR115	ICR114	ICR113	ICR112	ICR111	ICR110	ICR19	ICR18	page 121
(0x86)	**ICR1L**	ICR17	ICR16	ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	page 121
(0x85)	**TCNT1H**	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18	page 120
(0x84)	**TCNT1L**	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	page 120
(0x83)	**Reserved**	–	–	–	–	–	–	–	–	
(0x82)	**TCCR1C**	FOC1A	FOC1B	–	–	–	–	–	–	page 119
(0x81)	**TCCR1B**	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	page 118
(0x80)	**TCCR1A**	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	page 116
(0x7F)	**DIDR1**	–	–	ACMP0D	AMP0PD	AMP0ND	ADC10D/ACMP1D	ADC9D/AMP1PD	ADC8D/AMP1ND	page 199
(0x7E)	**DIDR0**	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D/ACMPMD	ADC2D/ACMP2D	ADC1D	ADC0D	page 199
(0x7D)	**Reserved**	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7C)	**ADMUX**	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	page 194
(0x7B)	**ADCSRB**	ADHSM	–	–	ADASCR	ADTS3	ADTS2	ADTS1	ADTS0	page 196
(0x7A)	**ADCSRA**	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 195
(0x79)	**ADCH**	- / ADC9	- / ADC8	- / ADC7	- / ADC6	- / ADC5	- / ADC4	ADC9 / ADC3	ADC8 / ADC2	page 198
(0x78)	**ADCL**	ADC7 / ADC1	ADC6 / ADC0	ADC5 / -	ADC4 / -	ADC3 / -	ADC2 / -	ADC1 / -	ADC0 /	page 198
(0x77)										
(0x76)	**AMP0CSR**	AMP0EN	-	AMP0G1	AMP0G0	-	AMP0TS2	AMP0TS1	AMP0TS0	page 202
(0x75)	**Reserved**	–	–	–	–	–	–	–	–	
(0x74)	**Reserved**	–	–	–	–	–	–	–	–	
(0x73)	**Reserved**	–	–	–	–	–	–	–	–	
(0x72)	**Reserved**	–	–	–	–	–	–	–	–	
(0x71)	**Reserved**	–	–	–	–	–	–	–	–	
(0x70)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6F)	**TIMSK1**	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	page 121
(0x6E)	**TIMSK0**	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	page 94
(0x6D)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6C)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6B)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6A)	**Reserved**	–	–	–	–	–	–	–	–	
(0x69)	**EICRA**	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	page 74
(0x68)	**Reserved**	–	–	–	–	–	–	–	–	
(0x67)	**Reserved**	–	–	–	–	–	–	–	–	
(0x66)	**OSCCAL**	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	page 31
(0x65)	**Reserved**	–	–	–	–	–	–	–	–	
(0x64)	**PRR**	PRPSC2	PRPSC1	PRPSC0	PRTIM1	PRTIM0	PRSPI	–	PRADC	page 39
(0x63)	**Reserved**	–	–	–	–	–	–	–	–	
(0x62)	**Reserved**	–	–	–	–	–	–	–	–	
(0x61)	**CLKPR**	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	page 35
(0x60)	**WDTCSR**	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	page 50
0x3F (0x5F)	**SREG**	I	T	H	S	V	N	Z	C	page 11
0x3E (0x5E)	**SPH**	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	page 13
0x3D (0x5D)	**SPL**	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 13
0x3C (0x5C)	**Reserved**	–	–	–	–	–	–	–	–	
0x3B (0x5B)	**Reserved**	–	–	–	–	–	–	–	–	
0x3A (0x5A)	**Reserved**	–	–	–	–	–	–	–	–	
0x39 (0x59)	**Reserved**	–	–	–	–	–	–	–	–	
0x38 (0x58)	**Reserved**	–	–	–	–	–	–	–	–	
0x37 (0x57)	**SPMCSR**	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	page 211
0x36 (0x56)	**Reserved**	–	–	–	–	–	–	–	–	
0x35 (0x55)	**MCUCR**	SPIPS	–	–	PUD	–	–	IVSEL	IVCE	page 56 & page 65
0x34 (0x54)	**MCUSR**	–	–	–	–	WDRF	BORF	EXTRF	PORF	page 46
0x33 (0x53)	**SMCR**	–	–	–	–	SM2	SM1	SM0	SE	page 37
0x32 (0x52)	MSMCR	Monitor Stop Mode Control Register								reserved
0x31 (0x51)	MONDR	Monitor Data Register								reserved
0x30 (0x50)	**ACSR**	ACCKDIV	AC2IF	–	AC0IF	–	AC2O	–	AC0O	page 179
0x2F (0x4F)	**Reserved**	–	–	–	–	–	–	–	–	
0x2E (0x4E)	**SPDR**	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	page 174
0x2D (0x4D)	**SPSR**	SPIF	WCOL	–	–	–	–	–	SPI2X	page 173
0x2C (0x4C)	**SPCR**	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 172
0x2B (0x4B)	**Reserved**	–	–	–	–	–	–	–	–	
0x2A (0x4A)	**Reserved**	–	–	–	–	–	–	–	–	
0x29 (0x49)	PLLCSR	-	-	-	-	-	PLLF	PLLE	PLOCK	page 33
0x28 (0x48)	**OCR0B**	OCR0B7	OCR0B6	OCR0B5	OCR0B4	OCR0B3	OCR0B2	OCR0B1	OCR0B0	page 94
0x27 (0x47)	**OCR0A**	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	page 93
0x26 (0x46)	**TCNT0**	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	page 93
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	page 92
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	page 89
0x23 (0x43)	**GTCCR**	TSM	ICPSEL1	–	–	–	–	–	PSRSYNC	page 77
0x22 (0x42)	**EEARH**	–	–	–	–	EEAR11	EEAR10	EEAR9	EEAR8	page 19
0x21 (0x41)	**EEARL**	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	page 19
0x20 (0x40)	**EEDR**	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	page 20
0x1F (0x3F)	**EECR**	–	–	–	–	EERIE	EEMWE	EEWE	EERE	page 20
0x1E (0x3E)	**GPIOR0**	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	page 25
0x1D (0x3D)	**EIMSK**	–	–	–	–	INT3	INT2	INT1	INT0	page 75
0x1C (0x3C)	**EIFR**	–	–	–	–	INTF3	INTF2	INTF1	INTF0	page 75
0x1B (0x3B)	**GPIOR3**	GPIOR37	GPIOR36	GPIOR35	GPIOR34	GPIOR33	GPIOR32	GPIOR31	GPIOR30	page 25

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1A (0x3A)	**GPIOR2**	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	page 25
0x19 (0x39)	**GPIOR1**	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	page 25
0x18 (0x38)	**Reserved**	–	–	–	–	–	–	–	–	
0x17 (0x37)	**Reserved**	–	–	–	–	–	–	–	–	
0x16 (0x36)	**TIFR1**	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	page 122
0x15 (0x35)	**TIFR0**	–	–	–	–	–	OCF0B	OCF0A	TOV0	page 94
0x14 (0x34)	**Reserved**	–	–	–	–	–	–	–	–	
0x13 (0x33)	**Reserved**	–	–	–	–	–	–	–	–	
0x12 (0x32)	**Reserved**	–	–	–	–	–	–	–	–	
0x11 (0x31)	**Reserved**	–	–	–	–	–	–	–	–	
0x10 (0x30)	**Reserved**	–	–	–	–	–	–	–	–	
0x0F (0x2F)	**Reserved**	–	–	–	–	–	–	–	–	
0x0E (0x2E)	**PORTE**	–	–	–	–	–	PORTE2	PORTE1	PORTE0	page 73
0x0D (0x2D)	**DDRE**	–	–	–	–	–	DDE2	DDE1	DDE0	page 73
0x0C (0x2C)	**PINE**	–	–	–	–	–	PINE2	PINE1	PINE0	page 73
0x0B (0x2B)	**PORTD**	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	page 73
0x0A (0x2A)	**DDRD**	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	page 73
0x09 (0x29)	**PIND**	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	page 73
0x08 (0x28)	–	–	–	–	–	–	–	–	–	–
0x07 (0x27)	–	–	–	–	–	–	–	–	–	–
0x06 (0x26)	–	–	–	–	–	–	–	–	–	–
0x05 (0x25)	**PORTB**	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 72
0x04 (0x24)	**DDRB**	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 72
0x03 (0x23)	**PINB**	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 73
0x02 (0x22)	**Reserved**	–	–	–	–	–	–	–	–	
0x01 (0x21)	**Reserved**	–	–	–	–	–	–	–	–	
0x00 (0x20)	**Reserved**	–	–	–	–	–	–	–	–	

Note:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The AT90PWM1 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.

26. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					

Mnemonics	Operands	Description	Operation	Flags	#Clocks
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

27. Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
16	2.7 - 5.5V	AT90PWM1-16SU	SO24	Extended (-40°C to 105°C)
16	2.7 - 5.5V	AT90PWM1-16MU	QFN32	Extended (-40°C to 105°C)

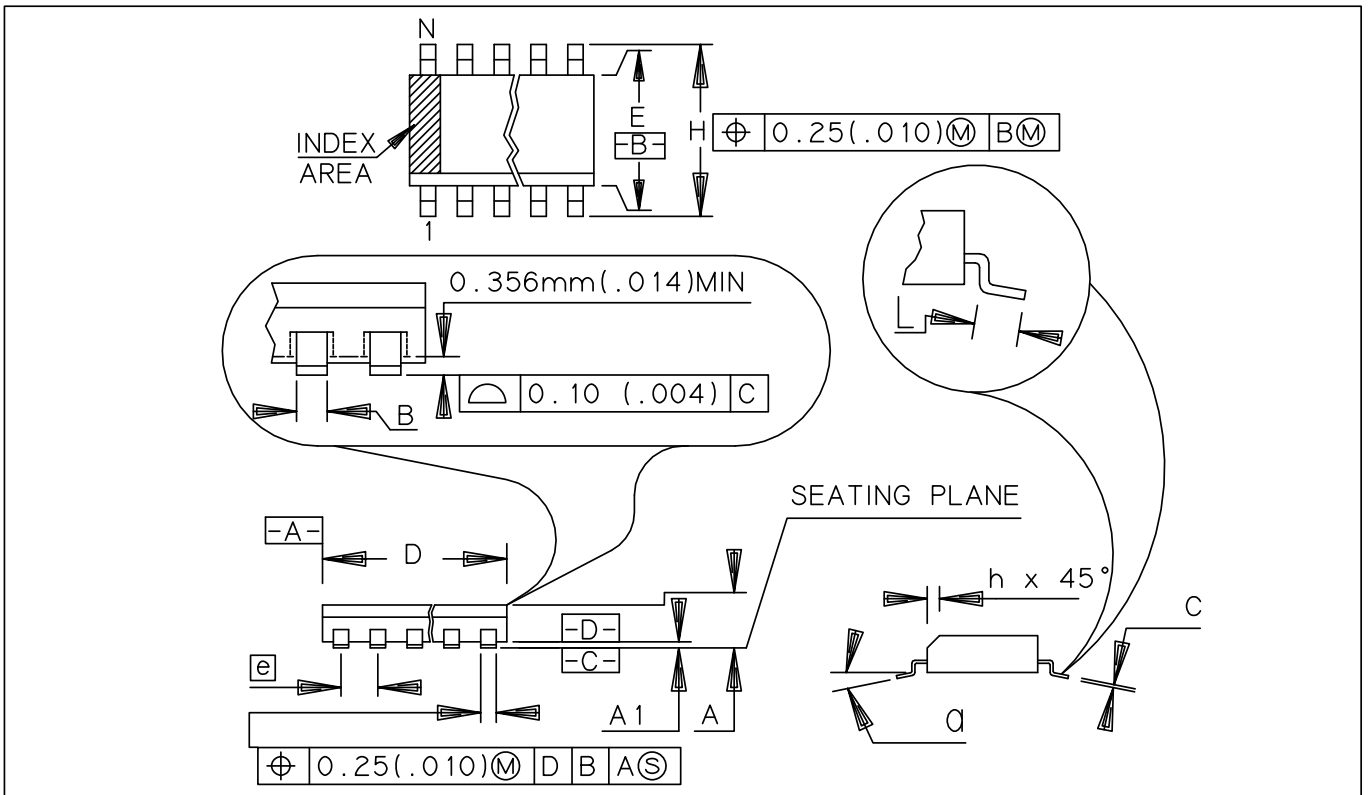
Note: All packages are Pb free, fully LHF

Note: This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

28. Package Information

Package Type	
SO24	24-Lead, 0.300” Body width, Plastic GullWing Small Outline Package (SOIC)
QFN32	32-Lead, Quad Flat No lead

28.1 SO24

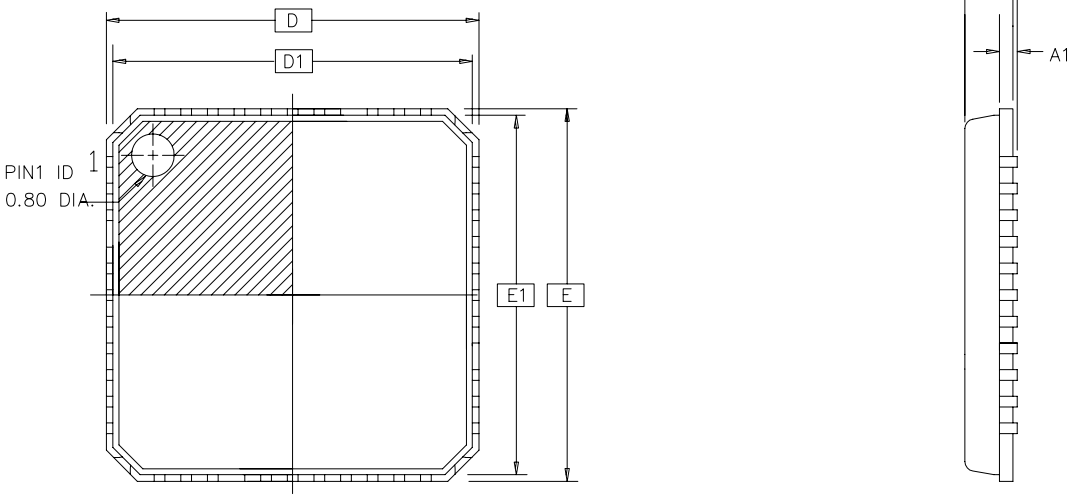


	MM		INCH	
A	2.35	2.65	.093	.104
A1	0.10	0.30	.004	.012
B	0.35	0.49	.014	.019
C	0.23	0.32	.009	.013
D	15.20	15.60	.599	.614
E	7.40	7.60	.291	.299
e	1.27	BSC	.050	BSC
H	10.00	10.65	.394	.419
h	0.25	0.75	.010	.029
L	0.40	1.27	.016	.050
N	24		24	
a	0°		8°	

07/27/07

Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	TITLE TD, 24 - Lead, 0.300" Body Width Plastic Gull Wing Small Outline Package (SOIC)	DRAWING No. TD	REV. A

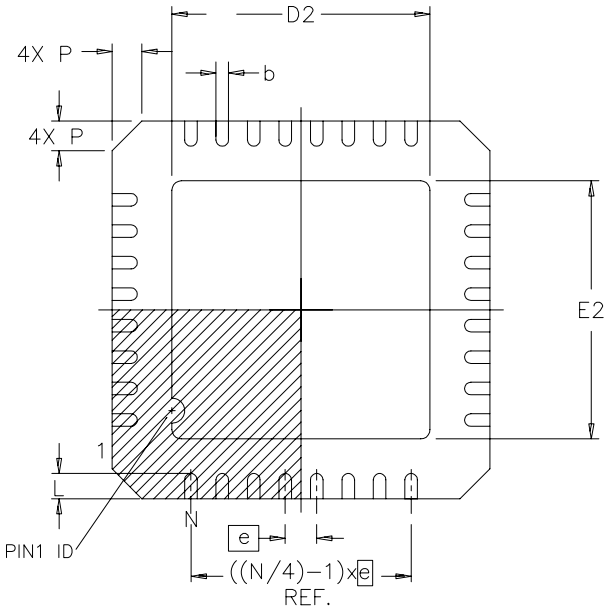
28.2 QFN32



TOP VIEW

SIDE VIEW

DRAWINGS NOT SCALED



BOTTOM VIEW

	MM			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.80	-	1.00	.032	-	.040
J	0.00	0.01	0.05	.000	.000	.002
A1		0.20 ref			.008 ref	
D/E		7.00 BSC			.276 BSC	
D1/E1		6.75 BSC			.266 BSC	
D2/E2	2.25	-	5.25	.090	-	.207
N	32					
P	0.24	0.42	0.60	.009	.016	.024
e		0.65 BSC			.026 BSC	
L	0.35	-	0.75	.014	-	.030
b	0.23	-	0.35	.009	-	.014

Compliant JEDEC Standard MO-220 variation VKKC

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 Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

29. Errata

- **PSC : Double End-Of-Cycle Interrupt Request in Centered Mode**
- **ADC : Conversion accuracy**
-

1. **PSC : Double End-Of-Cycle Interrupt Request in Centered Mode**

   In centered mode, after the “expected” End-Of-Cycle Interrupt, a second unexpected Interrupt occurs 1 PSC cycle after the previous interrupt.

   Work around:

   While CPU cycle is lower than PSC clock, the CPU sees only one interrupt request. For PSC clock period greater than CPU cycle, the second interrupt request must be cleared by software.

2. **ADC : Conversion accuracy**

   The conversion accuracy degrades when the ADC clock is 2 MHz.

   Work around:

   When a 10 bit conversion accuracy is required, use an ADC clock of 1 MHz or below.

   At 2 Mhz the ADC can be used as a 7 bits ADC.

30. Datasheet Revision History for AT90PWM1

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

30.1 Changes from 4378A to 4378B

1. PSC : the Balance Flank Width Modulation is done On-Time 1 rather than On-Time 0 (correction of figures)
2. Updated "Device Clocking Options Select AT90PWM1" on page 28
3. Updated "Start-up Times when the PLL is selected as system clock" on page 32
4. Updated "Calibrated Internal RC Oscillator" on page 30
5. Updated"ADC Noise Canceler" on page 188
6. Updated"ADC Auto Trigger Source Selection for non amplified conversions" on page 197
7. Updated"ADC Auto Trigger Source Selection for amplified conversions" on page 197
8. Updated"Amplifier" on page 199
9. Updated"Amplifier 0 Control and Status register – AMP0CSR" on page 202
10. Updated"AMP0 Auto Trigger Source Selection" on page 203
11. Updated temperature range in "DC Characteristics" on page 239
12. Updated Vhysr in "DC Characteristics" on page 239
13. Added "Calibrated Internal RC Oscillator Accuracy" on page 241
14. Updated "Maximum Speed vs. VCC" on page 242
15. Updated "ADC Characteristics" on page 245
16. Updated "Example 1" on page 254
17. Updated "Example 2" on page 254
18. Updated "Example 3" on page 255
19. Added "I/O Pin Input HysteresisVoltage vs. VCC" on page 263
20. Specify the "AREF Voltage vs. Temperature" on page 269
21. Updated Figure 24-36 on page 270
22. Updated Figure 24-37 on page 271
23. Updated Figure 24-38 on page 271
24. Updated "Ordering Information" on page 284
25. Updated "Package Information" on page 285

30.2 Changes from 4378B to 4378C

1. Minor corrections throughout the document.

Table of Contents

1 History 2

2 Disclaimer 2

3 Pin Configurations 2

3.1 Pin Descriptions 4

4 Overview 5

4.1 Block Diagram 6

4.2 Pin Descriptions 7

4.3 About Code Examples 8

5 AVR CPU Core 9

5.1 Introduction 9

5.2 Architectural Overview 9

5.3 ALU – Arithmetic Logic Unit 10

5.4 Status Register 11

5.5 General Purpose Register File 12

5.6 Stack Pointer 13

5.7 Instruction Execution Timing 13

5.8 Reset and Interrupt Handling 14

6 Memories 17

6.1 In-System Reprogrammable Flash Program Memory 17

6.2 SRAM Data Memory 17

6.3 EEPROM Data Memory 19

6.4 I/O Memory 25

6.5 General Purpose I/O Registers 25

7 System Clock 27

7.1 Clock Systems and their Distribution 27

7.2 Clock Sources 28

7.3 Default Clock Source 29

7.4 Low Power Crystal Oscillator 29

7.5 Calibrated Internal RC Oscillator 30

7.6 PLL 32

7.7 128 kHz Internal Oscillator 34

7.8 External Clock 34

7.9 Clock Output Buffer 35
7.10 System Clock Prescaler 35

8 Power Management and Sleep Modes 37

8.1 Idle Mode 38
8.2 ADC Noise Reduction Mode 38
8.3 Power-down Mode 38
8.4 Standby Mode 38
8.5 Power Reduction Register 39
8.6 Minimizing Power Consumption 40

9 System Control and Reset 42

9.1 Internal Voltage Reference 47
9.2 Watchdog Timer 48

10 Interrupts 53

10.1 Interrupt Vectors in AT90PWM1 53

11 I/O-Ports 58

11.1 Introduction 58
11.2 Ports as General Digital I/O 59
11.3 Alternate Port Functions 63
11.4 Register Description for I/O-Ports 72

12 External Interrupts 74

13 Timer/Counter0 and Timer/Counter1 Prescalers 76

14 8-bit Timer/Counter0 with PWM 79

14.1 Overview 79
14.2 Timer/Counter Clock Sources 80
14.3 Counter Unit 80
14.4 Output Compare Unit 81
14.5 Compare Match Output Unit 83
14.6 Modes of Operation 84
14.7 Timer/Counter Timing Diagrams 88
14.8 8-bit Timer/Counter Register Description 89

15 16-bit Timer/Counter1 with PWM 96

15.1 Overview 96
15.2 Accessing 16-bit Registers 98

15.3 Timer/Counter Clock Sources 101
15.4 Counter Unit 102
15.5 Input Capture Unit 103
15.6 Output Compare Units 104
15.7 Compare Match Output Unit 106
15.8 Modes of Operation 107
15.9 Timer/Counter Timing Diagrams 115
15.10 16-bit Timer/Counter Register Description 116

16 Power Stage Controller – (PSC0, PSC2) 123

16.1 Features 123
16.2 Overview 123
16.3 PSC Description 124
16.4 Signal Description 126
16.5 Functional Description 128
16.6 Update of Values 133
16.7 Enhanced Resolution 133
16.8 PSC Inputs 137
16.9 PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait 142
16.10 PSC Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait 143
16.11 PSC Input Mode 3: Stop signal, Execute Opposite while Fault active 144
16.12 PSC Input Mode 4: Deactivate outputs without changing timing. 144
16.13 PSC Input Mode 5: Stop signal and Insert Dead-Time 145
16.14 PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait. 146
16.15 PSC Input Mode 7: Halt PSC and Wait for Software Action 146
16.16 PSC Input Mode 8: Edge Retrigger PSC 147
16.17 PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC 148
16.18 PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Disactivate Output 149
16.19 PSC2 Outputs 151
16.20 Analog Synchronization 152
16.21 Interrupt Handling 152
16.22 PSC Synchronization 152
16.23 PSC Clock Sources 153
16.24 Interrupts 154
16.25 PSC Register Definition 154
16.26 PSC2 Specific Register 163

17 Serial Peripheral Interface – SPI 166

17.1 Features 166

17.2 $\overline{SS}$ Pin Functionality 171

17.3 Data Modes 174

18 Analog Comparator 176

18.1 Overview 176

18.2 Analog Comparator Register Description 177

19 Analog to Digital Converter - ADC 181

19.1 Features 181

19.2 Operation 183

19.3 Starting a Conversion 183

19.4 Prescaling and Conversion Timing 184

19.5 Changing Channel or Reference Selection 186

19.6 ADC Noise Canceler 188

19.7 ADC Conversion Result 192

19.8 ADC Register Description 194

19.9 Amplifier 199

19.10 Amplifier Control Registers 202

20 debugWIRE On-chip Debug System 204

20.1 Features 204

20.2 Overview 204

20.3 Physical Interface 204

20.4 Software Break Points 205

20.5 Limitations of debugWIRE 205

20.6 debugWIRE Related Register in I/O Memory 205

21 Boot Loader Support – Read-While-Write Self-Programming 205

21.1 Boot Loader Features 206

21.2 Application and Boot Loader Flash Sections 206

21.3 Read-While-Write and No Read-While-Write Flash Sections 206

21.4 Boot Loader Lock Bits 209

21.5 Entering the Boot Loader Program 210

21.6 Addressing the Flash During Self-Programming 212

21.7 Self-Programming the Flash 213

22 Memory Programming 219

22.1 Program And Data Memory Lock Bits 219
22.2 Fuse Bits 221
22.3 PSC Output Behaviour During Reset 221
22.4 Signature Bytes 223
22.5 Calibration Byte 223
22.6 Parallel Programming Parameters, Pin Mapping, and Commands 223
22.7 Serial Programming Pin Mapping 225
22.8 Parallel Programming 225
22.9 Serial Downloading 234

23 Electrical Characteristics(1) 238

23.1 Absolute Maximum Ratings* 238
23.2 DC Characteristics 239
23.3 External Clock Drive Characteristics 241
23.4 Maximum Speed vs. V_{CC} 242
23.5 PLL Characteristics 242
23.6 SPI Timing Characteristics 243
23.7 ADC Characteristics 245
23.8 Parallel Programming Characteristics 247

24 AT90PWM1 Typical Characteristics 249

24.1 Active Supply Current 250
24.2 Idle Supply Current 252
24.3 Power-Down Supply Current 257
24.4 Pin Pull-up 258
24.5 Pin Driver Strength 260
24.6 Pin Thresholds and Hysteresis 262
24.7 BOD Thresholds and Analog Comparator Offset 267
24.8 Analog Reference 269
24.9 Internal Oscillator Speed 270
24.10 Current Consumption of Peripheral Units 272
24.11 Current Consumption in Reset and Reset Pulse width 274

25 Register Summary 277

26 Instruction Set Summary 281

27 Ordering Information 284

28 Package Information 285

28.1 SO24 286
28.2 QFN32 287

29 Errata 289

30 Datasheet Revision History for AT90PWM1 290

30.1 Changes from 4378A to 4378B 290
30.2 Changes from 4378B to 4378C 290

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 Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.