Features

- High performance, low power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC architecture
  - 131 powerful instructions - most single clock cycle execution
  - 32 × 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
  - 8/16Kbytes of in-system programmable program memory flash
    - Endurance: 10,000 write/erase cycles
    - Lock bits protection
    - Optional 2/4Kbytes 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
    - Four bytes page size
  - 256/1024 Bytes Internal SRAM
- On-chip debug support (debugWIRE)
- Peripheral features
  - One 12-bit high speed PSC (Power Stage Controllers with extended PSC2 features)
    - Non overlapping inverted PWM output pins with flexible dead-time
    - Variable PWM duty cycle and frequency
    - Synchronous update of all PWM registers
    - Enhanced resolution mode (16 bits)
    - Additional register for ADC synchronization
    - Input capture
    - Four output pins and output matrix
  - One 12-bit high speed PSC (Power Stage Controller)
    - Auto-stop function for event driven PFC implementation
    - Non overlapping inverted PWM output pins with flexible dead-time
    - Variable PWM duty cycle and frequency
    - Synchronous update of all PWM registers
    - Enhanced resolution mode (16 bits)
    - Input capture
  - One 16-bit simple general purpose timer/counter
  - 10-bit ADC
    - Up to 11 single ended channels and one fully differential ADC channel pair
    - Programmable gain (5×, 10×, 20×, 40× on differential channel)
    - Internal reference voltage
  - One 10-bit DAC
  - Three analog comparators with
    - Resistor-array to adjust comparison voltage
    - DAC to adjust comparison voltage
  - One SPI
  - Three external interrupts
  - Programmable watchdog timer with separate on-chip oscillator
- Special microcontroller features
  - Low power idle, noise reduction, and power down modes

8-bit Atmel Microcontroller with 8/16K Bytes In-System Programmable Flash

AT90PWM81
AT90PWM161

7734Q–AVR–02/12

  - **Power-on reset and programmable brown-out detection**
  - **Flag array in bit-programmable I/O space (three bytes)**
  - **In-system programmable via SPI port**
  - **Internal low power calibrated RC oscillator (8MHz or 1MHz, low jitter)**
  - **On chip PLL for fast PWM (32MHz, 48MHz, 64MHz) and CPU (12MHz, 16MHz); PLL source RC & XTAL**
  - **Dynamic clock switch**
  - **Temperature sensor**
- **Operating voltage: 2.7V - 5.5V**
- **Operating temperature:**
  - **-40°C to +105°C or -40°C to +125°C**
- **Operating speed**
  - **5V: 16MHz core, 64MHz PLL**
  - **3.3V: 12MHz core, 48MHz PLL**

1. Products Configuration

The different product configurations are described per Table 1-1.

Table 1-1. PWM81/PWM161 configurations.

Package	**SO20**	**QFN32**
Pins	20	32
Flash size	8/16K (1)	8/16K (1)
EEPROM size	512	512
RAM size	256/1024 (2)	256/1024 (2)
PSC 12 bits with extended features	1	1
PSC 12 bits	1	1
Timer 8 bits	-	-
Timer 16 bits	1	1
ADC inputs	8	11
Amplifiers for ADC	1	1
Temperature sensor	1	1
Analog Comparators	3	3
DAC	1	1
DAC amplifiers	-	-
UART/DALI	-	-
SPI	1	1

Notes: 1. Flash size is 8Kbytes for AT90PWM81 and 16Kbytes for AT90PWM161.
2. RAM size is 256 bytes for AT90PWM81 and 1024 bytes for AT90PWM161.

2. Pin Configurations

Figure 2-1. 20-pin packages.

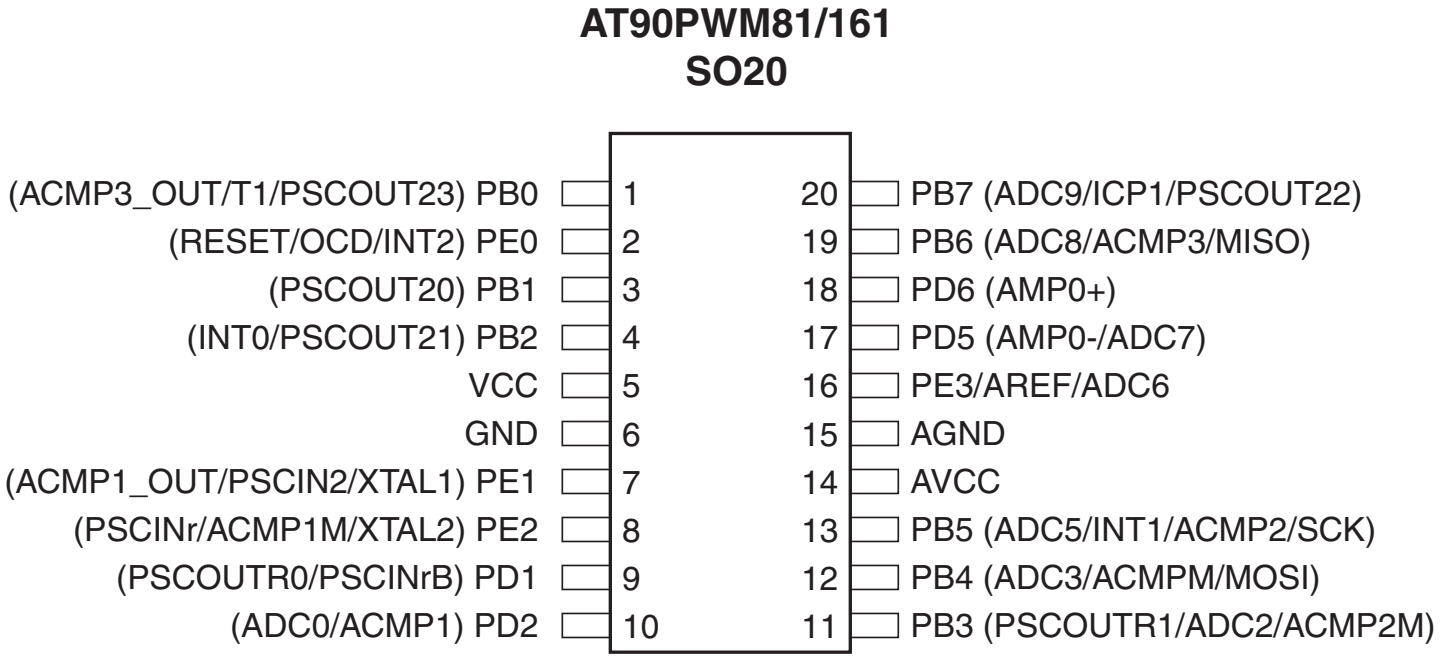


Figure 2-2. 32-pin packages.

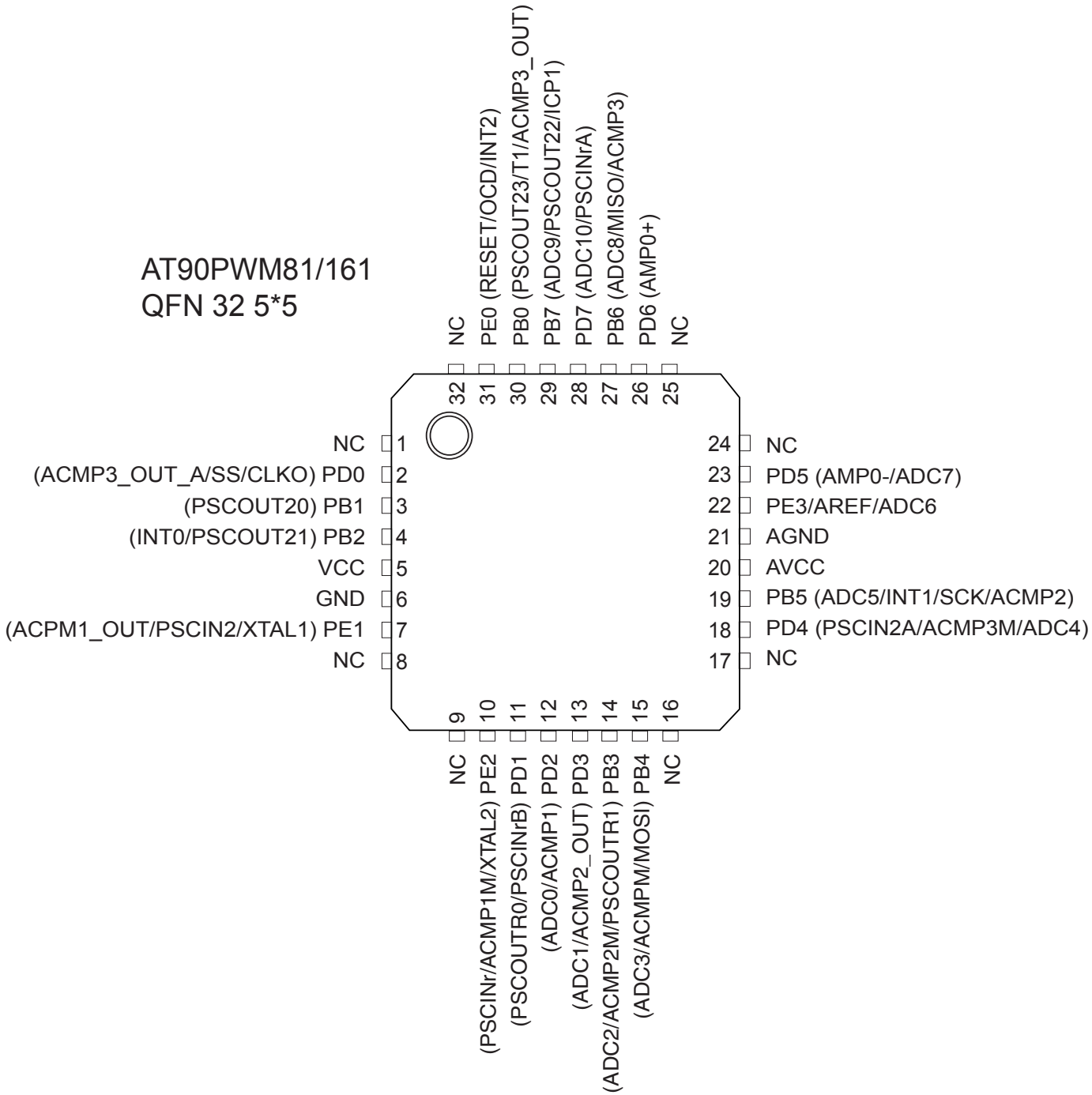


Table 2-1. Functions description.

MNEMONIC	NAME, FUNCTION & ALTERNATE FUNCTION
GND	**Ground:** 0V reference
AGND	**Analog Ground:** 0V reference for analog part
V_{CC}	**Power Supply**
AV_{CC}	**Analog Power Supply:** This is the power supply voltage for analog part For a normal use this pin must be connected.
AREF	**Analog Reference:** Reference for analog converter. This is the reference voltage of the A/D converter. As output, can be used by external analog
CLKO	System Clock Output
RESET# OCD	Reset Input On Chip Debug I/O
XTAL1	XTAL Input
XTAL2	XTAL Output
MISO	SPI Master In Slave Out
MOSI	SPI Master Out Slave In
SCK	SPI Clock
SS	SPI Slave Select
INTn	External interrupt n
Tn	Timer n clock input
PSCOUTxn	PSCx output n
PSCINx	PSCx Digital Input
PSCOUT0n	PSC reduced output n
PSCINr	PSC reduced Digital Input
ACMPn	Analog Comparator n Positive Input
ACMPMn	Analog Comparator n Negative Input
ACMPM	Negative input for analog comparators
ACOMPn_OUT	Analog Comparator n Output
AMPn-	Analog Differential Amplifier n Input Channel
AMPn+	Analog Differential Amplifier n Input Channel
ADCn	Analog Converter Input Channel n

Table 2-2. Pin out description.

Port	SO 20 pins	QFN32 pins	GP	PSC	ADC	Analog
PB0	1	30	T1	PSCOUT23		ACMP3_OUT
PE0	2	31	RESET# OCD, INT2			
PD0	NA	2	CLKO, SS			ACMP3_OUT_A
PB1	3	3		PSCOUT20		
PB2	4	4	INT0	PSCOUT21		
VCC	5	5	Power Supply			
GND	6	6	Ground			
PE1	7	7	XTAL1	PSCIN2		ACMP1_OUT
PE2	8	10	XTAL2	PSCINr		ACMP1M
PD1	9	11		PSCOUTR0, PSCINrB		
PD2	10	12			ADC0	ACMP1
PD3	NA	13			ADC1	ACMP2_OUT
PB3	11	14		PSCOUTR1	ADC2	ACMP2M
PB4	12	15	MOSI		ADC3	ACMPM
PD4	NA	18		PSCIN2A	ADC4	ACMP3M
PB5	13	19	INT1, SCK		ADC5	ACMP2
AVCC	14	20	Analog Supply			
AGND	15	21	Analog Ground			
	16	22	AREF, Analog Ref		ADC6	
PD5	17	23			ADC7	AMP0-
PD6	18	26				AMP0+
PB6	19	27	MISO		ADC8	ACMP3
PD7	NA	28		PSCINrA	ADC10	
PB7	20	29	ICP1	PSCOUT22	ADC9	

2.1 Pin Descriptions

2.1.1 V_{CC}

Digital supply voltage.

2.1.2 GND

Ground.

2.1.3 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 AT90PWM81/161 as listed on Table 9-3 on page 75.

2.1.4 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 AT90PWM81/161 as listed on Table 9-6 on page 78.

2.1.5 Port E (P32..0) $\overline{\text{RESET}}$/XTAL1/XTAL2/AREF

Port E is an 4-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.

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 7-1 on page 51. 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 Table 9-9 on page 80 and Section “Clock Systems and their Distribution”, page 27.

2.1.6 AV_{CC}

AV_{CC} is the supply voltage pin for the A/D converter. 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.

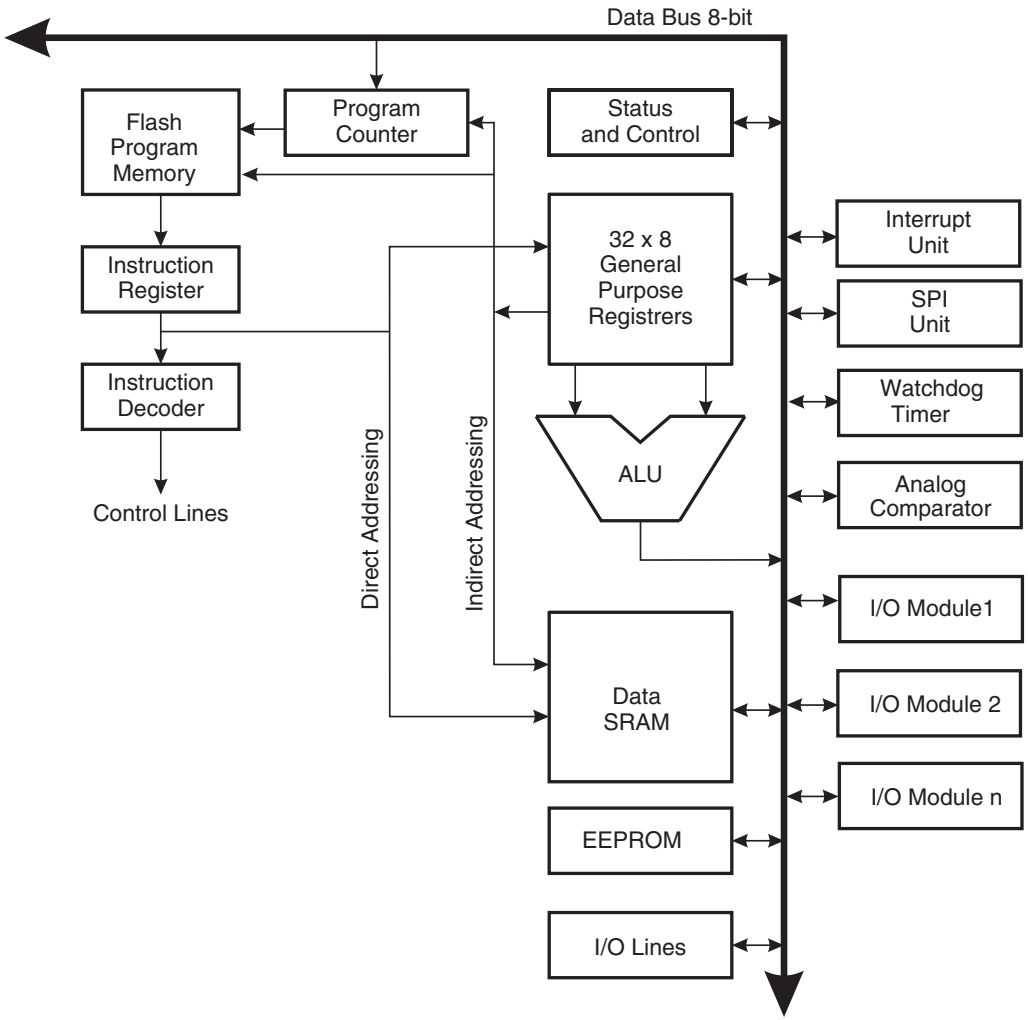
3. AVR CPU Core

3.1 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.

3.2 Architectural Overview

Figure 3-1. 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 × 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-register, Y-register, 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-bit 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 AT90PWM81/161 has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

3.3 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 Summary” on page 301 for a detailed description.

3.4 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 Summary” on page 301. 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 Summary” on page 301.

- **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 Summary” on page 301 for detailed information.

- **Bit 4 – S: Sign Bit, S = N Å 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 Summary” on page 301 for detailed information.

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

The Two’s Complement Overflow Flag V supports two’s complement arithmetic. See the “Instruction Set Summary” on page 301 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 Summary” on page 301 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 Summary” on page 301 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 Summary" on page 301 for detailed information.

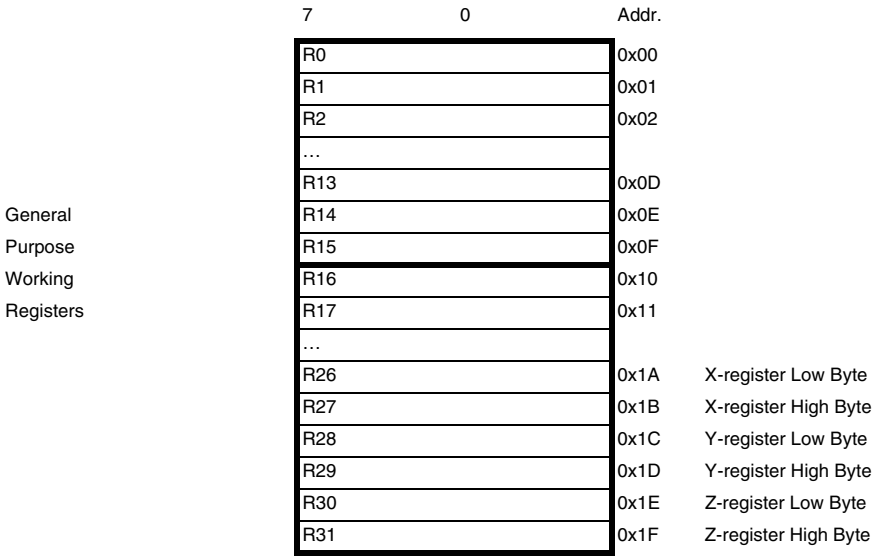
3.5 General Purpose Register File

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 3-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 3-2. AVR CPU General Purpose Working Registers.



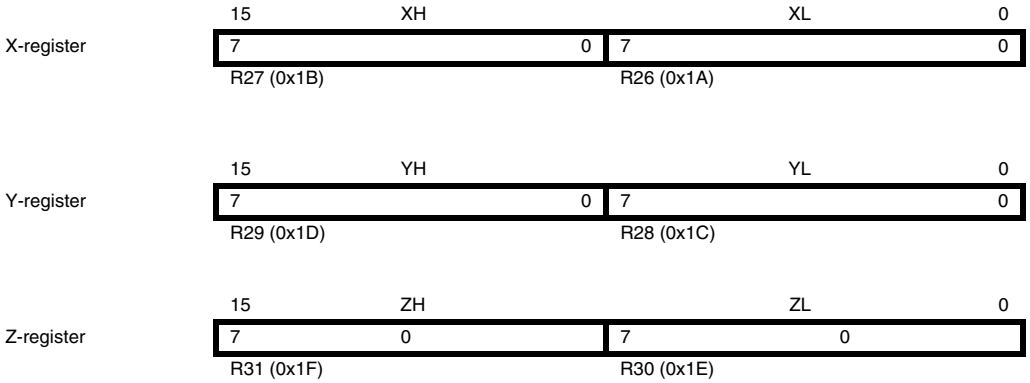
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 3-2, 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.

3.5.1 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 3-3 on page 12.

Figure 3-3. The X-register, Y-register, and Z-register.



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see “Instruction Set Summary” on page 301 for details).

3.6 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	

3.7 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 3-4 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 3-4. The parallel instruction fetches and instruction executions.

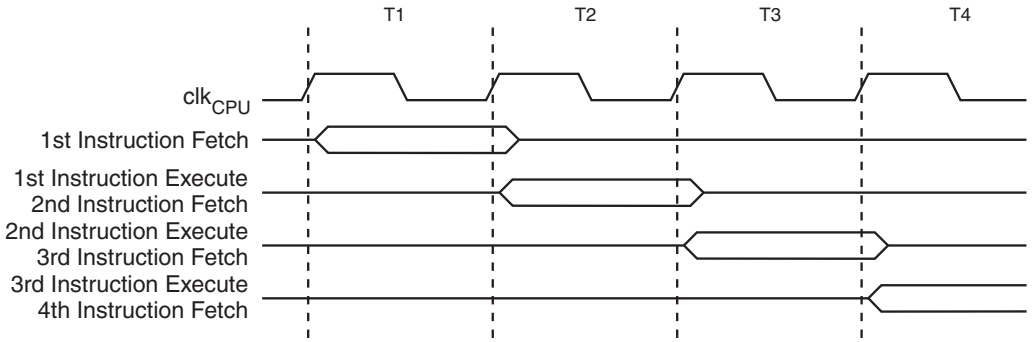
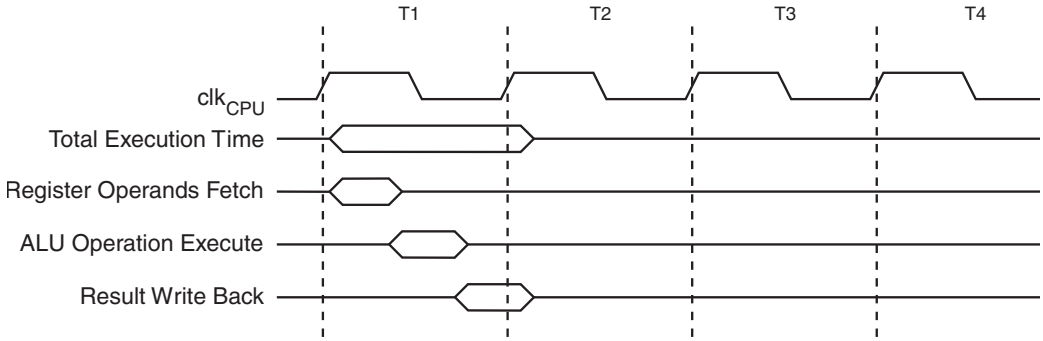


Figure 3-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 3-5. Single cycle ALU operation.



3.8 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 248 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 62. 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 62 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 233.

3.8.1 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) */`

3.8.2 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.

4. Memories

This section describes the different memories in the Atmel AT90PWM81/161. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the AT90PWM81/161 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

4.1 In-System Reprogrammable Flash Program Memory

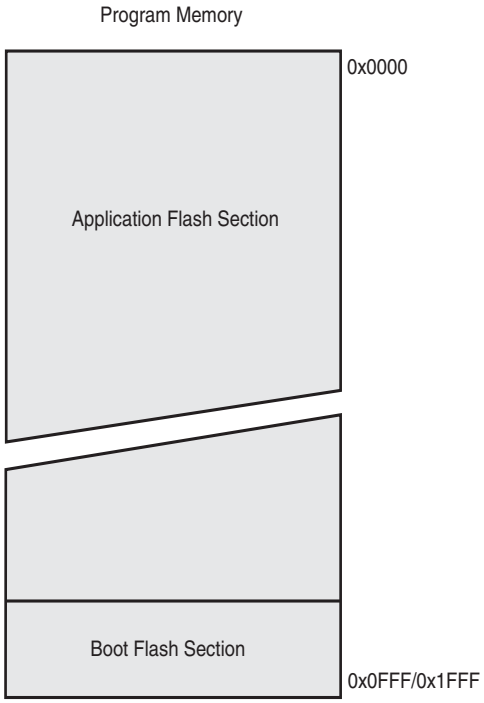
The AT90PWM81/161 contains 8/16Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K × 16bits for the AT90PWM81, and 8K × 16bits for the AT90PWM161. For software security, the Flash Program memory space 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 AT90PWM81 Program Counter (PC) is 12 bits wide, thus addressing the 8Kbytes program memory locations. The AT90PWM161 Program Counter (PC) is 13 bits wide, thus addressing the 16Kbytes 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 233. “Memory Programming” on page 248 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 description of LPM – Load Program Memory in “Instruction Set Summary” on page 301).

Timing diagrams for instruction fetch and execution are presented in “Instruction Execution Timing” on page 12.

Figure 4-1. Program memory map.



4.2 SRAM Data Memory

Figure 4-2 shows how the Atmel AT90PWM81/161 SRAM memory is organized.

The AT90PWM81/161 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 512 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 256 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.

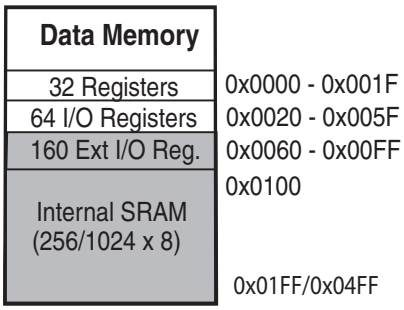
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y-register 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 256/1024 bytes of internal data SRAM in the AT90PWM81/161 are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 11.

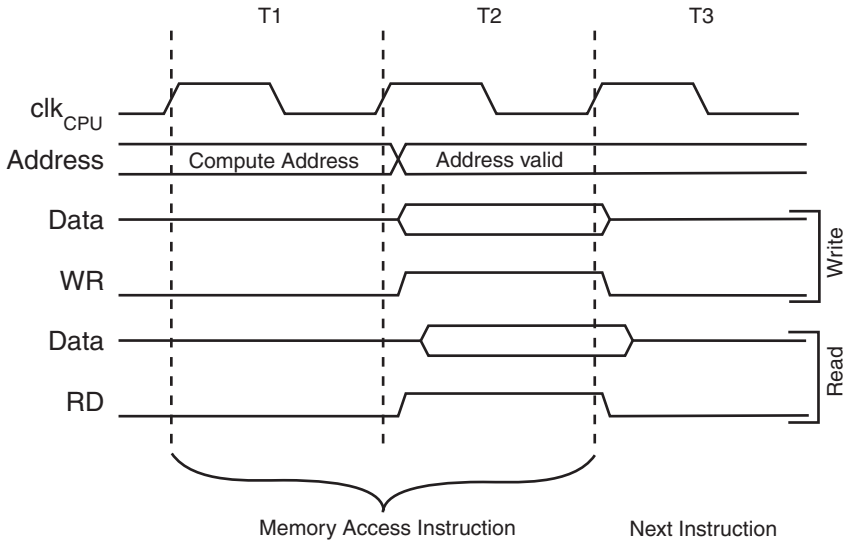
Figure 4-2. Data memory map.



4.2.1 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 4-3 on page 18.

Figure 4-3. On-chip data SRAM access cycles.



4.3 EEPROM Data Memory

The AT90PWM81/161 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 261, and "Parallel Programming Parameters, Pin Mapping, and Commands" on page 252 respectively.

4.3.1 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 4-2 on page 21. 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 see "Preventing EEPROM Corruption" on page 25.

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.

4.3.2 EEARH and EEARL - EEPROM Address Registers

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 AT90PWM81/161 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.

4.3.3 EEDR - EEPROM Data Register

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.

4.3.4 EECR - EEPROM Control Register

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

- **Bits 7 – NVMBSY: Non-volatile memory busy**

The NVMBSY bit is a status bit that indicates that the NVM memory (FLASH, EEPROM, Lock-bits) is busy programming. Once a program operation is started, the bit will be set and it remains set until the program operation is completed.

Bits 6 – EEPAGE: EEPROM page access (multiple bytes access mode)

Writing EEPAGE to one enables the multiple bytes access mode. That means that several bytes can be programmed simultaneously into the EEPROM. When the EEPAGE bit has been written to one, the EEPAGE bit remains set until an EEPROM program operation is completed. Alternatively the bit is cleared when the temporary EEPROM buffer is flushed in software (see EEPMn bits description). Any write to EEPAGE while EEPE is one will be ignored. See Section “Program multiple bytes in one Atomic operation”, page 21 for details on how to load data into the temporary EEPROM page and the usage of the EEPAGE bit.

- **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 4-1. 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 4-1. EEPROM mode bits.

EEPM1	EEPM0	Programming time	Operation
0	0	3.4ms	Erase and write in one operation (atomic operation)
0	1	1.8ms	Erase only
1	0	1.8ms	Write only
1	1	–	Flush temporary EEPROM page buffer

- **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 233 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 4-2 lists the typical programming time for EEPROM access from the CPU.

Table 4-2. EEPROM programming time.

Symbol	Number of calibrated RC oscillator cycles	Typical programming time
EEPROM write (from CPU)	26368	3.3ms

4.3.5 Program multiple bytes in one Atomic operation

It is possible to write multiple bytes into the EEPROM. Before initiating a programming (erase/write), the data to be written has to be loaded into the temporary EEPROM page buffer. Writing EEPAGE to one enables a load operation.

When EEPAGE bit is written to one, the temporary EEPROM page buffer is ready for loading. To load data into the temporary EEPROM page buffer, the address and data must be written into EEARL and EEDR respectively. Note that the data is loaded when EEDR is updated. Therefore, the address must be written before data. This operation is repeated until the temporary EEPROM page buffer is filled up or until all data to be written have been loaded. The number of bytes that is loaded must not exceed the temporary EEPROM page size before performing a program operation. Note that it is not possible to write more than one time to each byte in the temporary EEPROM page buffer before executing a program operation. If the same byte is written multiple times, the content in the temporary EEPROM page will be bit wise AND between the written data (that is, if 0xaa and 0x55 is loaded to the same byte, the result will be 0x00). The temporary EEPROM buffer will be ready for new data after the program operation has completed. Alternatively, the temporary EEPROM buffer is flushed and ready for new data by writing EEPE (within four cycles after EEMPE is written) if the EEPMn bits are 0b11. When the temporary EEPROM buffer is flushed, the EEPAGE bit will be cleared. Loading data into the temporary EEPROM buffer takes three CPU clock cycles. If EEDR is written while EEPAGE is set, the CPU is halted to ensure that the operation takes three cycles.

The order the different bits and registers should be accessed is:

1 Write EEPAGE in EECR (loading of temporary EEPROM buffer is enabled).

2 Write the address bits needed to address bytes within a page into EEARL.

3 Write data to EEDR.

4 Repeat 2 and 3 above until the buffer is filled up or until all data is loaded.

5 Write the remaining address bits into EEARH:EEARL.

a. Select which programming mode that should be executed (EEPMn bits). Write the EEPE bit in EECR (within four cycles after EEMPE has been written) to start a program operation. The temporary EEPROM page buffer will auto-erase after program operation is completed.

OR

b. If an error situation occurred and the loading should be terminated by software: Write EEPM1:0 to 0b11 and trigger the flushing by writing EEPE (within four cycles after EEMPE has been written).

4.4 Fuse Bits

The AT90PWM81/161 has three Fuse bytes. Table 4-3 through Table 4-5 on page 23 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 4-3. Extended low fuse byte.

Extended fuse byte	Bit no.	Description	Default value
PSC2RB	7	PSC2 reset behavior	1
PSC2RBA	6	PSC2 reset behavior for OUT22 & 23	1
PSCRRB	5	PSC reduced reset behavior	1
PSCRV	4	PSCOUT & PSCOUTR reset value	1
PSCINRB	3	PSC & PSCR inputs reset behavior	1
BODLEVEL2 (1)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1 (1)	1	Brown-out detector trigger level	0 (programmed)
BODLEVEL0 (1)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes: 1. See Table 7-2 on page 53 for BODLEVEL fuse decoding.

Table 4-4. 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
BOOTSZ1	2	Select boot size (see Table 20-7 on page 246 for details)	0 (programmed) [4]
BOOTSZ0	1	Select boot size (see Table 20-7 on page 246 for details)	0 (programmed) [4]
BOOTRST	0	Select reset vector	1 (unprogrammed)

Notes:
1. See "Alternate Functions of Port E" on page 80 for description of RSTDISBL fuse.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. See "Watchdog timer configuration." on page 60 for details.
4. The default value of BOOTSZ1..0 results in maximum boot size.

Table 4-5. 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 5-4 on page 30 for details.
2. The default setting of CKSEL3..0 results in internal RC oscillator @ 8MHz. See Table 5-1 on page 28 for details.
3. The CKOUT fuse allows the system clock to be output on PORTD0. See "Clock Output Buffer" on page 34 for details.
4. See "System Clock Prescaler" on page 39 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.

4.4.1 Code examples

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example, 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.

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.

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;
}
```

4.4.2 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.

4.5 I/O Memory

The I/O space definition of the AT90PWM81/161 is shown in “Register Summary” on page 297.

All AT90PWM81/161 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 Summary” on page 301 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 AT90PWM81/161 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.

4.6 General Purpose I/O Registers

The AT90PWM81/161 contains four General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags.

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

4.6.1 GPIOR0 - General Purpose I/O Register 0

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

4.6.2 GPIOR1 - General Purpose I/O Register 1

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

4.6.3 GPIOR2 - General Purpose I/O Register 2

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	

5. System Clock and Clock Options

The Atmel AT90PWM81/161 provides a large number of clock sources. Those can be divided in two categories: internal and external.

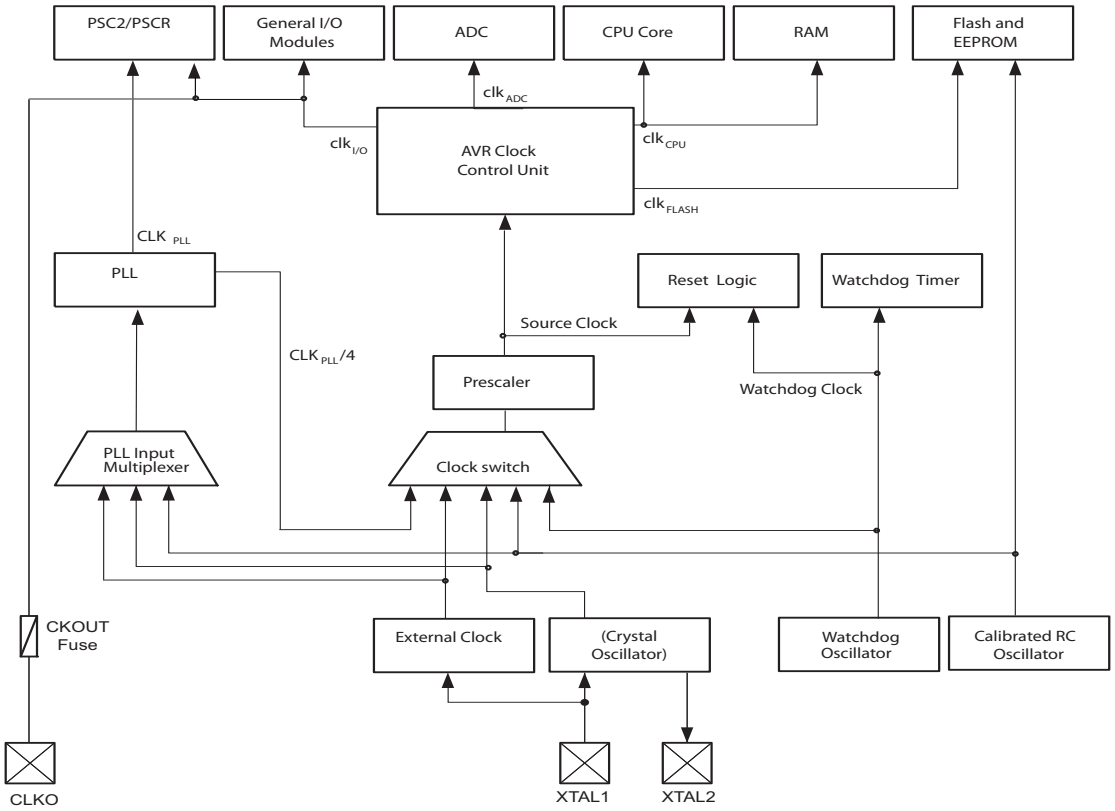
After reset, CKSEL fuses select one clock source. Once the device is running, software clock switching is available on any other clock sources.

Some hardware controls are provided for clock switching management but some specific procedures must be observed. Some settings may lead the user to program the device in an inadequate configuration.

5.1 Clock Systems and their Distribution

Figure 5-1 presents the principal clock systems in the AVR and their distribution. All of the clocks may not be active at a given time. In order to reduce power consumption, the clocks from modules not being used can be halted by using different sleep modes or by using features of the dynamic clock switch ("Power Management and Sleep Modes" on page 45 or "Dynamic Clock Switch" on page 35). The clock systems are detailed below.

Figure 5-1. Clock distribution.



5.1.1 clk_{CPU} - CPU Clock

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.

5.1.2 $clk_{I/O}$ - I/O Clock

The I/O clock is used by the majority of the I/O modules, like Timer/Counter. 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.

5.1.3 clk_{FLASH} - Flash Clock

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

5.1.4 clk_{PLL} - PLL Clock

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

5.1.5 clk_{ADC} - ADC Clock

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.

5.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits (default) or by the CLKSELR register (dynamic clock switch circuit) as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 5-1. Device clocking options select (1) , PLL source and PE1 and PE2 functionality.

Device clocking option	System clock	PLL input (2)	CKSEL3..0 (3) CSEL3..0 (4)	PE1	PE2
External clock	Ext Clk (8)	RC Osc (6)	0000	CLKI	I/O
PLL output divided by 4 : 16MHz driven by internal RC	PLL / 4	RC Osc (6)	0001	I/O	I/O
Calibrated internal RC oscillator 8MHz	RC Osc (6)	RC Osc (6)	0010	I/O	I/O
Internal 128kHz RC oscillator (WD)	WD (7)	N/A	0011	I/O	I/O
PLL output divided by 4 / PLL driven by external crystal/ceramic resonator	PLL / 4	Ext Osc (5)	0100	XTAL1	XTAL2
PLL output divided by 4/ PLL driven by external clock	PLL / 4	Ext Clk (8)	0101	CLKI	I/O
Calibrated internal RC oscillator 1MHz	RC Osc (6)	N/A	0110	I/O	I/O
External crystal/ceramic resonator (3.0MHz - 8.0MHz)	Ext Osc (5)	Ext Osc (5)	0111_b	XTAL1	XTAL2
External crystal/ceramic resonator (0.9MHz - 3.0MHz)	Ext Osc (5)	RC Osc (6)	1000_b	XTAL1	XTAL2
External crystal/ceramic resonator (0.9MHz - 3.0MHz)	Ext Osc (5)	RC Osc (6)	1001_b	XTAL1	XTAL2
External crystal/ceramic resonator (3.0MHz - 8.0MHz)	Ext Osc (5)	RC Osc (6)	1010_b	XTAL1	XTAL2
External crystal/ceramic resonator (3.0MHz - 8.0MHz)	Ext Osc (5)	RC Osc (6)	1011_b	XTAL1	XTAL2
External crystal/ceramic resonator (3.0MHz - 8.0MHz)	Ext Osc (5)	RC Osc (6)	1100_b	XTAL1	XTAL2

Table 5-1. Device clocking options select [1] , PLL source and PE1 and PE2 functionality. (Continued)

Device clocking option	System clock	PLL input [2]	CKSEL3..0 [3] CSEL3..0 [4]	PE1	PE2
External crystal/ceramic resonator (3.0MHz - 8.0MHz)	Ext Osc [5]	RC Osc [6]	1101_b	XTAL1	XTAL2
External crystal/ceramic resonator (8.0MHz - 16.0MHz)	Ext Osc [5]	RC Osc [6]	1110_b	XTAL1	XTAL2
External crystal/ceramic resonator (8.0MHz - 16.0MHz)	Ext Osc [5]	RC Osc [6]	1111_b	XTAL1	XTAL2

Note:
1. For all fuses “1” means unprogrammed while “0” means programmed.
2. PLL must be driven by a nominal 8MHz clock source.
3. Flash fuse bits.
4. CLKSELR register bits.
5. Ext Osc: External oscillator.
6. RC Osc: Internal RC oscillator (1MHz or 8MHz).
7. WD: Internal watch dog RC oscillator 128kHz.
8. 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 when a new clock source is enabled by the dynamic clock switch circuit, 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 commencing 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 5-2.

Table 5-2. Number of watchdog oscillator cycles.

Typical time-out	Number of cycles
4ms	512
64ms	8K (8,192)

5.2.1 Default Clock Source

The device will always starts up from reset using the clock source defined by CKSEL Fuses the start-up time defined by SUT Fuses. This configuration is latched in CLKSELR register at reset. The device will always starts up at Power-on using the clock source defined by CLKSELR register (CSEL3..0 and CSUT1:0).

The device is shipped with CKSEL Fuses = 0010_b, SUT Fuses = 10_b, and CKDIV8 Fuse programmed. The default clock source setting is therefore the Internal RC Oscillator running at 8MHz 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 High-voltage Programmer. This set-up must be taken into account when using ISP tools.

5.2.2 Calibrated Internal RC Oscillator

By default, the Internal RC OScillator provides an approximate 8.0MHz clock or a 1MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user.

The switch between 8MHz and 1MHz is done by the CKRC81 bit in MCUCR register. See "MCUCR - MCU Control Register" on page 42 for more details.The RC oscillator can be accessed by two CKSEL or CSEL configurations. At reset, the CKRC81 bit is initialised with the value compatible with CKSEL value (1 for CKSEL3..0 = 0110, 0 for all other values).

The RC oscillator is active for any CKSEL3..0 or CSEL3..0 configuration where it is used as system clock or PLL source clock. The RC oscillator is diabled in the following CKSEL3..0 or CSEL3..0 cases:

- 0011 (128k oscillator)
- 0100, 0101 (PLL/4 system clock driven by external clock or oscillator)
- 1100, 1101 (External oscillator)

The device is shipped with the CKDIV8 Fuse programmed. See "System Clock Prescaler" on page 39 for more details. This clock may be selected as the system clock by programming the CKSEL Fuses or CSEL field as shown in Table 5-1 on page 28. If selected, it will operate with no external components. During reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. The accuracy of this calibration is shown as Factory calibration in Table 22-1 on page 270.

By changing the OSCCAL register from SW, see "OSCCAL – Oscillator Calibration Register" on page 39, 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 22-1 on page 270.

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 252.

Table 5-3. Internal calibrated RC oscillator operating modes (1)(3).

Frequency range (2) (MHz)	CKSEL3..0
7.6 - 8.4	0010
0.95 - 1.05 (4)	0010

Notes:
1. The device is shipped with this option selected.
2. The frequency ranges are preliminary values. Actual values are TBD.
3. If 8MHz 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.
4. Switch between 8MHz and 1MHz is done by CKRC81 bit in MCUCR register.

When this oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 5-4 on page 30.

Table 5-4. Start-up times for the internal calibrated RC Oscillator clock selection.

Power conditions	Start-up time from power-down	Additional delay from reset (V_{CC} = 5.0V)	SUT1..0
BOD enabled	6CK	14CK (1)	00
Fast rising power	6CK	14CK + 4.1ms	01
Slowly rising power	6CK	14CK + 65ms (2)	10
Reserved			11

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

5.2.2.1 *RC Oscillator calibration at Factory*

The RC oscillator is calibrated at 3V, 25°C for an 8MHz target frequency with an Accuracy ±1%.

The corresponding value OSCAL (@Amb.) is stored in the signature row and automatically loaded in the OSCAL register at reset.

The RC oscillator is monitored at 105°C or 125°C (versus Product version) with an accuracy within ±5% limits.

5.2.3 128KHz Internal Oscillator

The 128kHz internal Oscillator is a low power Oscillator providing a clock of 128kHz. The frequency is nominal at 3V and 25°C. This clock may be select as the system clock by programming CKSEL Fuses or CSEL field as shown in Table 5-1 on page 28.

When this clock source is selected, start-up times are determined by the SUT Fuses or by CSUT field as shown in Table 5-5.

Table 5-5. Start-up times for the 128kHz internal oscillator.

SUT1..0 (1) CSUT1..0 (2)	Start-up time from power-down	Additional delay from reset	Recommended usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4ms	Fast rising power
10	6 CK	14CK + 64ms	Slowly rising power
11	Reserved		

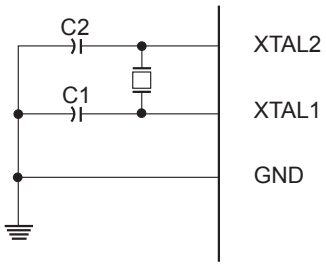
Notes: 1. Flash Fuse bits.
2. CLKSELR register bits.

5.2.4 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 5-2 on page 31. Either a quartz crystal or a ceramic resonator may be used.

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 5-6. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 5-2. Crystal oscillator connections.



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by CKSEL3..1 fuses or by CSEL3..1 field as shown in Table 5-6.

Table 5-6. Crystal oscillator operating modes.

CKSEL3..1 (1) CSEL3..1 (2)	Frequency range [MHz]	Recommended range for capacitors C1 and C2 for use with crystals [pF]
100 (3)	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 - 16.0	12 - 22

Notes:
1. Flash fuse bits.
2. CLKSELR register bits.
3. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses or CSEL0 together with CSUT1..0 field select the start-up times as shown in Table 5-7.

Table 5-7. Start-up times for the crystal oscillator clock selection.

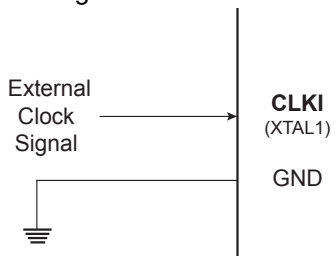
CKSEL0 (1) CSEL0 (2)	SUT1..0 (1) CSUT1..0 (2)	Start-up time from power-down and power-save	Additional delay from reset (V_{CC} = 5.0V)	Recommended usage
0	00	258CK (3)	14CK + 4.1ms	Ceramic resonator, fast rising power
0	01	258CK (3)	14CK + 65ms	Ceramic resonator, slowly rising power
0	10	1K (1024)CK (4)	14CK	Ceramic resonator, BOD enabled
0	11	1K (1024)CK (4)	14CK + 4.1ms	Ceramic resonator, fast rising power
1	00	1K (1024)CK (4)	14CK + 65ms	Ceramic resonator, slowly rising power
1	01	16K (16384)CK	14CK	Crystal oscillator, BOD enabled
1	10	16K (16384)CK	14CK + 4.1ms	Crystal oscillator, fast rising power
1	11	16K (16384)CK	14CK + 65ms	Crystal oscillator, slowly rising power

Notes:
1. Flash fuse bits.
2. CLKSELR register bits.
3. 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.
4. 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.

5.2.5 External Clock

To drive the device from this external clock source, CLKI should be driven as shown in Figure 5-3. To run the device on an external clock, the CKSEL Fuses or CSEL field must be programmed as shown in Table 5-1 on page 28.

Figure 5-3. External clock drive configuration.



When this clock source is selected, start-up times are determined by the SUT Fuses or CSUT field as shown in Table 5-8.

Table 5-8. Start-up times for the external clock selection.

SUT1..0 (1) CSUT1..0 (2)	Start-up time from power-down	Additional delay from reset	Recommended usage
00	6CK	14CK	BOD enabled
01	6CK	14CK + 4ms	Fast rising power
10	6CK	14CK + 64ms	Slowly rising power
11	Reserved		

Notes: 1. Flash Fuse bits.
2. CLKSELR register bits.

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 39 for details.

5.2.6 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 configured by software.

The internal PLL in AT90PWM81/161 generates a clock frequency multiplied from nominally 8MHz input. The source of the 8MHz PLL input clock can be selected from three possible sources (see the Figure 5-4 on page 34):

- Internal RC Oscillator
- Crystal oscillator
- External clock

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.

When selected as clock source by fuse, the PLL multiplication factor is initialized at the value of 6, compatible with a 3V supply.

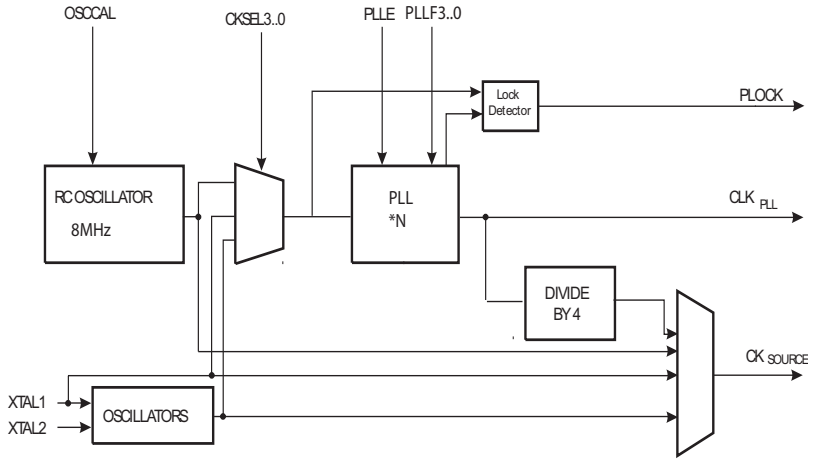
The PLL is locked on the source oscillator which must remains close to 8MHz to assure proper lock of the PLL.

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

Table 5-9. Start-up times when the PLL is selected as system clock.

CKSEL3..0	SUT1..0	Start-up time from power-down	Additional delay from reset (V_{CC} = 5.0V)	Clock source
0100	00	1K CK	14CK	External crystal or resonator
	01	1K CK	14CK + 4ms	
	10	1K CK	14CK + 64ms	
	11	16K CK	14CK	
0101	00	16K CK	14CK	External clock
	01	16K CK	14CK + 4ms	
	10	16K CK	14CK + 4ms	
	11	16K CK	14CK + 64ms	
0001	00	1K CK	14CK	Internal RC oscillator
	01	1K CK	14CK + 4ms	
	10	1K CK	14CK + 64ms	

Figure 5-4. PCK clocking system.



5.2.7 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT Fuse or COUT bit of CLKSELR register has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. Note that the clock will not be output during reset and the normal operation of I/O pin will be overridden when the fuses are programmed. Any clock source can be selected when the clock is output on CLKO. If the System Clock Prescaler is used, it is the divided system clock that is output.

5.3 Dynamic Clock Switch

5.3.1 Features

AT90PWM81/161 provides a powerful dynamic clock switch that allows users to turn on and off clocks of the device on the fly. The built-in de-glitching circuitry allows clocks to be enabled or disabled asynchronously. This enables efficient power management schemes to be implemented easily and quickly. In a safety application, the dynamic clock switch circuit may continuously monitor the external clock fails.

The AT90PWM81/161 provides one register for Clock Fuse substitution (CLKSELR) and one register to control the dynamic clock switch circuit (CLKCSR). The watchdog is used to monitor external clock source if needed. The control of the dynamic clock switch circuit must be supervised by software. The low level control is performed by hardware through the CLKCSR register. The features are:

- **Safe commands**, to avoid unintentional commands, a special write procedure must be followed to change the CLKCSR register bits (see “CLKCSR – Clock Control & Status Register” on page 42):
- **Exclusive action**, the actions are controlled by a decoding (command table). The main commands of the dynamic clock switching are:
  - ‘*Disable Clock Source*’,
  - ‘*Enable Clock Source*’,
  - ‘*Request for Clock Availability*’,
  - ‘*Clock Source Switching*’,
  - ‘*Recover System Clock Source*’.
- **Status**, a status on the availability of the enabled clock and the code recovering of clock source used to drive the system clock are provided.

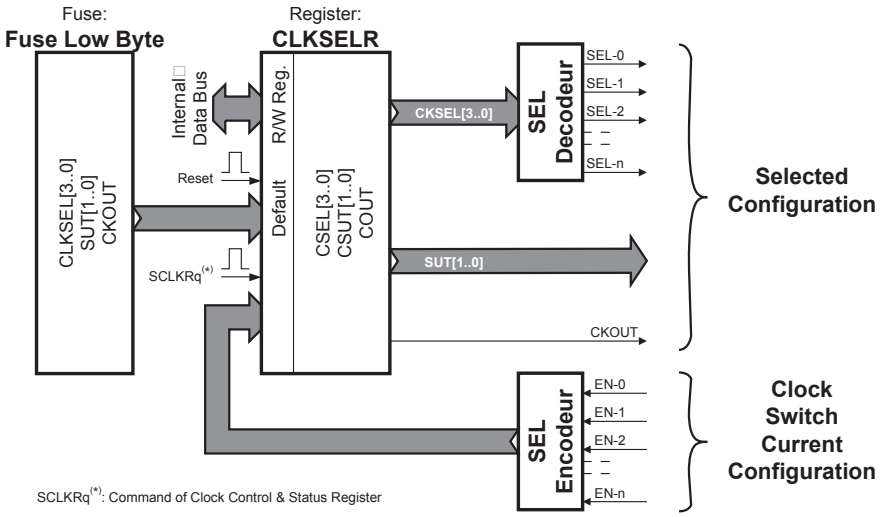
5.3.2 Fuses substitution

During reset, bits of the Low Fuse Byte are latched in the CLKSELR register. The content of this register can operate as well as the Low Fuse Byte. CKSEL3..0, SUT1..0 and CKOUT fuses are substituted as shown in Figure 5-5 and replaced respectively by CSEL3..0, CSUT1:0 and COUT.

5.3.3 Clock Source Selection

The available codes of clock source is given are in Table 5-1 on page 28.

Figure 5-5. Fuses substitution and clock source selection.



When '*Enable/Disable Clock Source*', '*Request for Clock Availability*' or '*Clock Source Switching*' command is entered, the selected configuration provided by the CLKSELR register is latched for each targeted clock source.

'*Recover System Clock Source*' command enables the code recovering of clock source used to drive the system clock. The CKSEL field of CLKSELR register is then updated with this code. There is no information on the SUT used or status on CKOUT.

Because the selected configuration is latched at clock source level, it is possible to enable many clock sources at a given time (ex: the internal RC oscillator for system clock + an oscillator with external crystal). The user's software has the responsibility of this management.

'*Request for Clock Availability'* command returns the working order of the clock source addressed. The status is set in the CLKRDY bit of CLKCSR register.

5.3.4 Enable/Disable Clock Source

'*Enable Clock Source*' command selects and enables the clock source provided by the setting of CLKSELR register (CSEL3..0 and CSUT1:0). CSEL field will select the clock source and CSUT field will select the start-up time (as CKSEL and SUT fuse bits do it). To be sure that a clock source has been enabled, it will be better to perform a '*Request for Clock Availability'* command after the '*Enable Clock Source*' command.

'*Disable Clock Source*' command disables the clock source provided by the setting of CLKSELR register (only CSEL3..0). If the clock source is the one that is used to drive the system clock, the command is not taken into account.

5.3.5 Clock Availability

'Request for Clock Availability' command enables an oscillation-counting of the selected source clock, CSEL3..0. The count is provided by CSUT1..0. The clock is declared ready (CLKRDY = 1) when the count is finished. This flag remains unchanged up to a new count. The CLKRDY flag is reset when the count starts. To perform this checking, the CKSEL and CSUT fields should not change all long the operation is running.

Two usages are possible:

1. Clock stability before switching
   Once the new clock source is selected, the count procedure is running. The user (code) should wait for the setting of the CLKRDY flag in CLKSCR register before to perform a switching.
2. Clock available on request
   AT any time, the user (code) can ask for the availability of a clock source. The user (code) can request it writing the appropriate command in the CLKSCR register. A full status on clock sources then can be done.

5.3.6 Clock Switching

To drive the system clock, the user can switch from the current clock source to the following ones (one of them is the current clock source):

1. Calibrated internal RC oscillator 8.0MHz/1.0MHz,
2. Internal watchdog oscillator 128kHz,
3. External clock,
4. External Crystal/Ceramic Resonator,
5. PLL output divided by four.

The clock switching is performed in a sequence of commands. First, the user (code) must make sure that the new clock source is running. Then the switching command can be entered. At the end, the user (code) can stop the previous clock source. It will be better to run this sequence once the interrupts disabled. The user (code) has the responsibility of the clock switching sequence.

Here is a “light” C-code that describes such a sequence of commands.

C code example

```
void ClockSwiching (unsigned char clk-number, unsigned char sut) {

#define CLOCK-RECOVER   0x05
#define CLOCK-ENABLE    0x02
#define CLOCK-SWITCH    0x04
#define CLOCK-DISABLE   0x01

unsigned char previous-clk, temp;

  // Disable interrupts
    asm ("cli"); temp = SREG;
  // “Recover System Clock Source” command
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK-RECOVER;
    previous-clk = CLKSELR & 0x0F;
  // “Enable Clock Source” command
    CLKSELR = ((sut << 4 ) & 0x30) | (clk-number & 0x0F);
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK-ENABLE;
  // Wait for clock availability
    while ((CLKCSR & (1 << CLKRDY)) == 0);
  // “Clock Source Switching” command
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK-SWITCH;
  // Wait for effective switching
    while (1){
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK-RECOVER;
        if ((CLKSELR & 0x0F) == (clk-number & 0x0F)) break;
    }
  // “Disable Clock Source” command
    CLKSELR = previous-clk;
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK-DISABLE;
  // Re-enable interrupts
    SREG = temp; asm ("sei");
}
```

Warning:

In the AT90PWM81/161, only one among the external clock sources can be enabled at a given time and it is not possible to switch from external clock to external oscillator as both sources share one pin.

Also, it is not possible to switch the synchronization source of the PLL when the sytem clock is PLL/4. See Table 5-1 on page 28 to identify these cases.

As they are two CSEL adresses to access the Calibrated internal RC oscillator 8.0MHz/1.0MHz, the change between the two frequencies is not allowed by the clock switching features. The CKRC81 bit in MCUCR register must be used for this purpose.

5.4 System Clock Prescaler

5.4.1 Features

The AT90PWM81/161 system clock can be divided by setting the Clock Prescaler 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 5-10 on page 40.

5.4.2 Switching Time

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occur in the clock system and 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 another 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.

5.5 Register Description

5.5.1 OSCCAL – Oscillator Calibration Register

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 8.0MHz 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 7.6MHz - 8.4MHz 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 8.8MHz. Otherwise, the EEPROM or Flash write may fail.

The CAL7..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 CAL7..0 by 1 will give a frequency increment of less than 0.5% in the frequency range 7.6MHz - 8.4MHz.

5.5.2 CLKPR – Clock Prescaler Register

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 simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when the 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 6:4 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM81/161 and will always read as zero.

- **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 5-10.

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 in order not to disturb the procedure.

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 eight 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 selected 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 5-10. 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

Table 5-10. Clock prescaler select. (Continued)

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock division factor
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

5.5.3 PLLCSR - PLL Control and Status Register

Bit	7	6	5	4	3	2	1	0	
$29 ($29)	–	–	PLLF3	PLLF2	PLLF1	PLLF0	PLLE	PLOCK	PLLCSR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	1	0	0	0/1	0	

- **Bit 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM81/161 and always read as zero.

- **Bit 5..2–– PLLF: PLL Factor**

The PLLF bits is used to select the multiplication factor of the PLL.

Table 5-11. PLL multiplication factor.

PLLF3..0	N+2	PLL frequency [MHz]
7-F		Reserved
6	8	64
5	7	56
4	6	48
3	5	40
2	4	32
0-1		Reserved

Note: PLLF3 is used for debug purpose (**must be wired**).

- **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. The time to lock is specified in Table 5-9 on page 34.

5.5.4 MCUCR - MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	PUD	RSTDIS	CKRC81	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0/1 (1)	0	0	0	

Notes: 1. Value is initialized with the fuse CKSEL2.
2. Value is initialized with fuses CKSEL3..0 (1 when CKSEL3..0= 0110, 0 in all other cases).

- **Bit 2– CKRC81: Frequency Selection of the calibrated 8/1MHz RC Oscillator**

Thanks to CKRC81 in MCUCR Sfr, the typical frequency of the calibrated RC oscillator is changed.

– When the CKRC81 bit is written to zero, the RC oscillator frequency is 8MHz.
– When the CKRC81 bit is written to one, the RC oscillator frequency is 1MHz.

Note: This bit only can be changed only when the RC oscillator is enabled.

Note: When the RC oscillator is used as the PLL source, CKRC81 must not be written to 1.

Note: If the RC oscillator is disabled, this bit is cleared by hardware.

5.5.5 CLKCSR – Clock Control & Status Register

Bit	7	6	5	4	3	2	1	0	
	CLKCCE	–	–	CLKRDY	CLKC3	CLKC2	CLKC1	CLKC0	CLKCSR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – CLKCCE: Clock Control Change Enable**

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

- **Bits 6:5 – Res: Reserved Bits**

These bits are reserved bits in the AT90PWM81/161 and will always read as zero.

- **Bits 4 – CLKRDY: Clock Ready Flag**

This flag is the output of the '*Clock Availability*' logic.
This flag is reset once the '*Request for Clock Availability*' command is entered.
It is set when '*Clock Availability*' logic confirms that the (selected) clock is running and is stable. The delay from the request and the flag setting is not fixed, it depends on the clock start-up time,

the clock frequency and, of course, if the clock is alive. The user's has itself to do the difference between '***no_clock_signal***' and '***clock_signal_not_yet_available***'.

- **Bits 3:0 – CLKC3:0: Clock Control Bits 3 - 0**

These bits define the command to provide to the '*Clock Switch*' module. The special write procedure must be followed to change the CLKC bits (see "Bit 7 – CLKCCE: Clock Control Change Enable" on page 42).

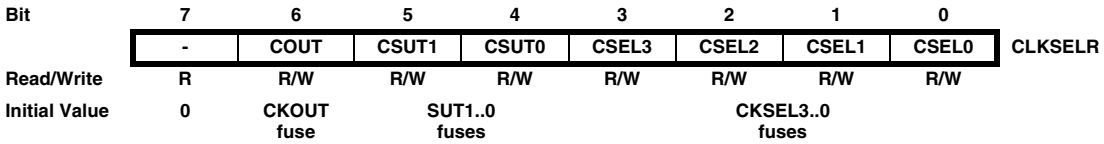
1. Write the Clock Control Change Enable (CLKCCE) bit to one and all other bits in CLKCSR to zero.
2. Within four cycles, write the desired value to CLKCSR register while clearing CLKCCE bit.

Interrupts should be disabled when setting CLKCSR register in order not to disturb the procedure.

Table 5-12. Clock command list.

Clock command	CLKC3..0
No command	0000_b
Disable clock source	0001_b
Enable clock source	0010_b
Request for clock availability	0011_b
Clock source switch	0100_b
Recover system clock source code	0101_b
CKOUT command	0111_b
No command	$1xxx_b$

5.5.6 CLKSELR - Clock Selection Register



- **Bit 7– Res: Reserved Bit**

This bit is reserved bit in the AT90PWM81/161 and will always read as zero.

- **Bit 6 – COUT: Clock Out**

The COUT bit is initialized with CKOUT Fuse bit.
The COUT bit is only used in case of '*CKOUT*' command. Refer to Section 5.2.7 "Clock Output Buffer" on page 34 for using.
In case of '*Recover System Clock Source*' command, COUT it is not affected (no recovering of this setting).

- **Bits 5:4 – CSUT1:0: Clock Start-up Time**

CSUT bits are initialized with the values of SUT Fuse bits.
In case of '*Enable/Disable Clock Source*' command, CSUT field provides the code of the clock start-up time. Refer to subdivisions of Section 5.2 "Clock Sources" on page 28 for code of clock

start-up times.
In case of '*Recover System Clock Source*' command, CSUT field is not affected (no recovering of SUT code).

- **Bits 3:0 – CSEL3:0: Clock Source Select**

CSEL bits are initialized with the values of CKSEL Fuse bits.
In case of '*Enable/Disable Clock Source*', '*Request for Clock Availability*' or '*Clock Source Switch*' command, CSEL field gets back the code of the clock source. Refer to Table 5-1 on page 28 and subdivisions of Section 5.2 "Clock Sources" on page 28 for clock source codes.
In case of '*Recover System Clock Source*' command, CSEL field receives the code of the clock source used to drive the Clock Control Unit as described in Figure 5-1 on page 27.

6. 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.

6.1 Sleep Modes

Figure 5-1 on page 27 presents the different clock systems in the AT90PWM81/161, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 6-1 shows the different sleep modes, their wake up sources.

Table 6-1. 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	Other/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	
Standby (1)						X	X (2)				X	

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

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 or Standby) will be activated by the SLEEP instruction. See Table 6-2 on page 48 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.

6.2 Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, 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 interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by clearing the ACnEN bit in the Analog Comparator Control and Status Register – ACnCON. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

6.3 ADC Noise Reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, 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 INT2:0 can wake up the MCU from ADC Noise Reduction mode.

6.4 Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the External Oscillator is stopped, while the External Interrupts 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 INT2:0 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 83 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.

6.5 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.

6.6 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 predictable behavior 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.

6.7 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.

6.7.1 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 "ADC Noise Canceler" on page 210 for details on ADC operation.

6.7.2 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 NOT automatically disabled, so it should be disabled if not used.

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 194 for details on how to configure the Analog Comparator.

6.7.3 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 significantly to the total current consumption. Refer to "Brown-out Detection" on page 53 for details on how to configure the Brown-out Detector.

6.7.4 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 55 for details on the start-up time.

6.7.5 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 56 for details on how to configure the Watchdog Timer.

6.7.6 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 68 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 "DIDR1 - Digital Input Disable Register 1" on page 222 and "DIDR0 - Digital Input Disable Register 0" on page 202 for details.

6.7.7 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.

6.8 Register description

6.8.1 SMCR - Sleep Mode Control Register

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 6-2.

Table 6-2. 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.

6.8.2 PRR - Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
	PRPSC2	-	PRPSCR	PRTIM1	-	PRSPI	-	PRADC	PRR
Read/Write	R/W	R	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	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 - Reserved**

- **Bit 5 - PRPSCR: Power Reduction PSC reduced**

Writing a logic one to this bit reduces the consumption of the PSCR by stopping the clock to this module. When waking up the PSCR again, the PSCR 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 - Reserved**

- **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 1 - Reserved**

- **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.

7. System Control and Reset

7.1 System Control overview

7.1.1 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 7-1 on page 51 shows the reset logic. Table 7-1 on page 51 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.

7.1.2 Reset Sources

The Atmel AT90PWM81/161 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. The external reset pin can be disabled in 2 ways:
  - By the RSTDISBL fuse. In this case , the SPI programming is disabled.
  - By software using the RSTDIS bit in MCUCR register. In this case , the SPI programming is still active at power up time.
- **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 7-1. Reset logic.

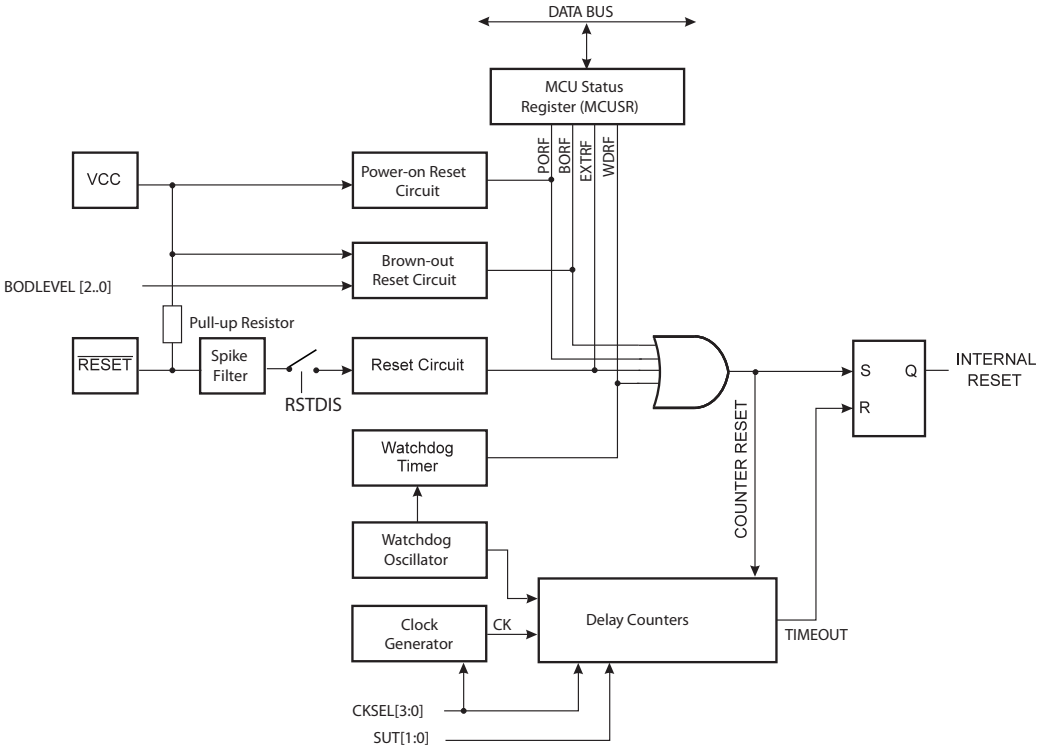


Table 7-1. Reset characteristics (1).

Symbol	Parameter	Condition	Minimum	Typical	Maximum	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		$0.2V_{CC}$		$0.85V_{CC}$	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).

7.1.3 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 7-1. 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 7-2. MCU start-up, $\overline{\text{RESET}}$ tied to V_{CC}.

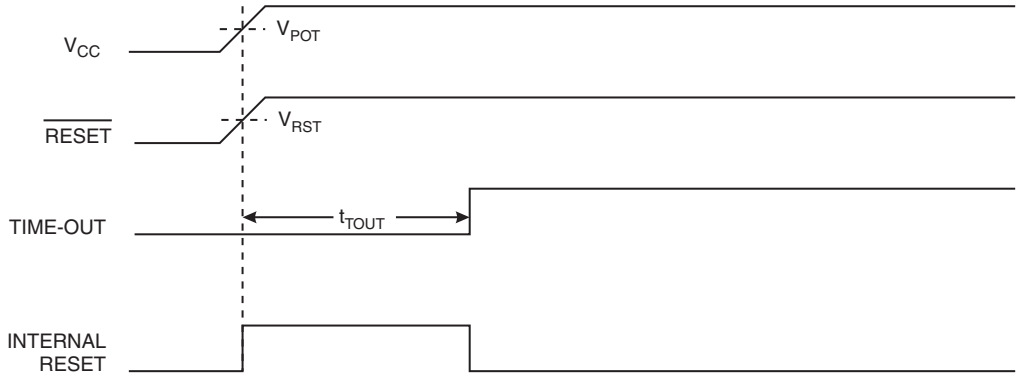
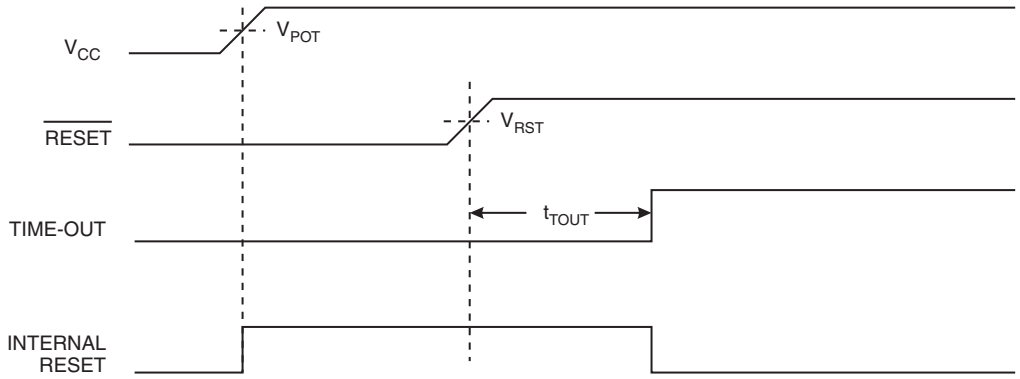


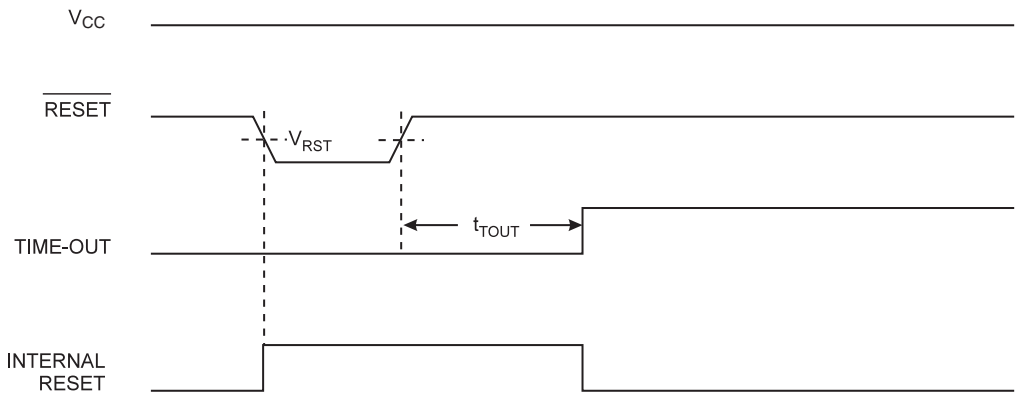
Figure 7-3. MCU start-up, $\overline{\text{RESET}}$ extended externally.



7.1.4 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 7-1 on page 51) 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 7-4. External reset during operation.



7.1.5 Brown-out Detection

AT90PWM81/161 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 7-2. BODLEVEL fuse coding [1][2].

BODLEVEL 2..0 fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	Forbidden, BOD must be enabled			
110		1.8		V
101 (default configuration)		2.7		
100	3.9	4.3	4.6	
011		2.3		
010		2.2		
001		1.9		
000	2.5	2.0	2.9	

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 Voltageand BODLEVEL = 101 for High Operating Voltage.
2. Values are guidelines only.

Table 7-3. Brown-out characteristics [1].

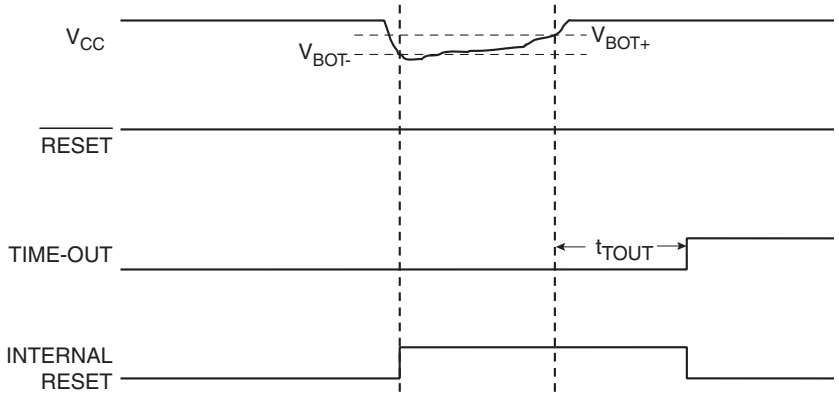
Symbol	Parameter	Minimum	Typical	Maximum	Units
V_{HYST}	Brown-out detector hysteresis		70		mV
t_{BOD}	Minimum pulse width on brown-out reset		2		µs

Notes:
1. Values are guidelines only.

When V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 7-5 on page 54), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 7-5 on page 54), 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 7-3.

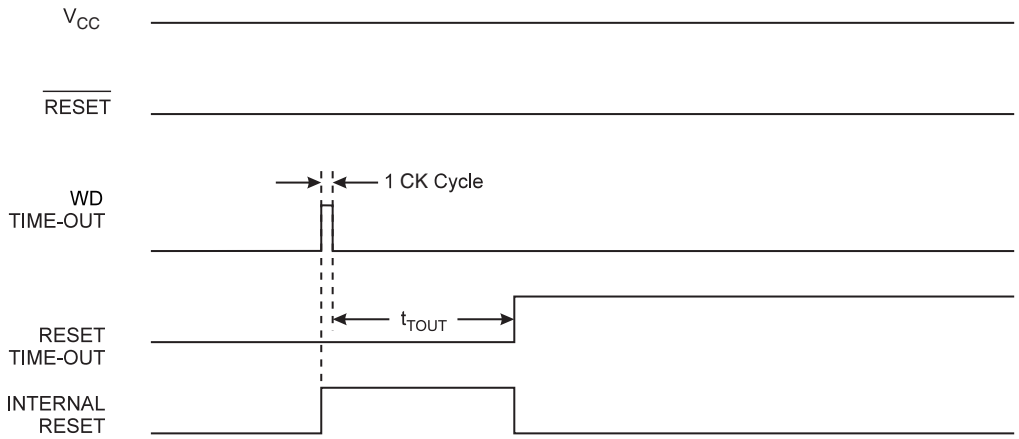
Figure 7-5. Brown-out reset during operation.



7.1.6 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 56 for details on operation of the Watchdog Timer.

Figure 7-6. Watchdog reset during operation.



7.2 System Control registers

7.2.1 MCUSR - MCU Status Register

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.

7.2.2 MCUCR - MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	PUD	RSTDIS	CKRC81	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3– RSTDIS: Reset Pin Disable**

Thanks to RSTDIS in MCUCR Sfr, the reset function can be disabled, leaving this pin for functional purpose.

– When the RSTDIS bit is written to zero, the reset signal is active.

– When the RSTDIS bit is written to one, the reset signal is inactive.

7.3 Internal Voltage Reference

AT90PWM81/161 features an internal bandgap reference. This bandgap reference is used for Brown-out Detection and can be used as analog input for the analog comparators or the ADC.

The internal voltage reference for the DAC and/or the ADC and the comparators is derived from this bandgap voltage, see “On Chip voltage Reference and Temperature sensor overview” on page 189.

The V_{REF} voltage is configured thanks to the REFS1 and REFS0 bits in the ADMUX register; see “ADMUX - ADC Multiplexer Register” on page 217.

7.3.1 Bandgap and Internal Voltage Reference Enable Signals and Start-up Time

The bandgap and the internal voltage reference characteristics is given on Table 7-4 on page 56. To save power, the reference is not always turned on. The bandgap and the internal reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] Fuse).
2. When the internal reference is selected (REFS1 = 1).
3. When the bandgap reference is connected to the Analog Comparator.
4. When the ADC is enabled.

Thus, when the BOD is not enabled, after enabling the ADC, comparator or the internal reference, 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 four conditions above to ensure that the reference is turned off before entering Power-down mode.

7.3.2 Voltage Reference Characteristics

Table 7-4. Internal voltage reference characteristics [1].

Symbol	Parameter	Condition	Minimum	Typical	Maximum	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

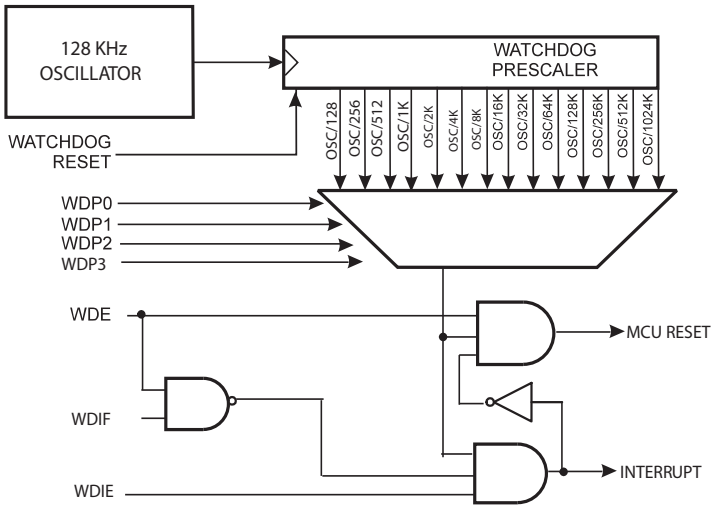
Note: 1. Values are guidelines only.

7.4 Watchdog Timer

AT90PWM81/161 has an Enhanced Watchdog Timer (WDT). The main features are:

- **Clocked from separate On-chip Oscillator**
- **Three operating modes**
  - **Interrupt**
  - **System reset**
  - **Interrupt and system reset**
- **Selectable time-out period from 1ms to 8s**
- **Possible hardware fuse watchdog always on (WDTON) for fail-safe mode**

Figure 7-7. Watchdog timer.



The Watchdog Timer (WDT) is a timer counting cycles of a separate on-chip 128kHz 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 (for example 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.

7.4.1 WDTCSR - Watchdog Timer Control Register

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 7-5. 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 7-6 on page 61.

Table 7-6. Watchdog timer prescaler 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	16ms
0	0	0	1	4K (4096) cycles	32ms
0	0	1	0	8K (8192) cycles	64ms
0	0	1	1	16K (16384) cycles	0.125s
0	1	0	0	32K (32768) cycles	0.25s
0	1	0	1	64K (65536) cycles	0.5s
0	1	1	0	128K (131072) cycles	1.0s
0	1	1	1	256K (262144) cycles	2.0s
1	0	0	0	512K (524288) cycles	4.0s
1	0	0	1	1024K (1048576) cycles	8.0s
1	0	1	0	1K (1024) cycles	8ms
1	0	1	1	512 cycles	4ms
1	1	0	0	256 cycles	2ms
1	1	0	1	128 cycles	1ms
1	1	1	0	Reserved	
1	1	1	1		

8. Interrupts

This section describes the specifics of the interrupt handling as performed in the Atmel AT90PWM81/161. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 13.

8.1 Interrupt Vectors in AT90PWM81/161

Table 8-1. Reset and interrupt vectors.

Vector no.	AT90PWM81 program address	AT90PWM161 program address	Source	Interrupt definition
1	0x0000	0x0000	RESET	External pin, power-on reset, brown-out reset, watchdog reset, and emulation AVR reset
2	0x0001	0x0002	PSC2 CAPT	PSC2 capture event
3	0x0002	0x0004	PSC2 EC	PSC2 end cycle
4	0x0003	0x0006	PSC2 EEC	PSC2 end of enhanced cycle
5	0x0004	0x0008	PSCr CAPT	PSC reduced capture event
6	0x0005	0x000A	PSCr EC	PSC reduced end cycle
7	0x0006	0x000C	PSCr EEC	PSC reduced end of enhanced cycle
8	0x0007	0x000E	ANACOMP 0	Analog comparator 0
9	0x0008	0x0010	ANACOMP 1	Analog comparator 1
10	0x0009	0x0012	ANACOMP 2	Analog comparator 2
11	0x000A	0x0014	INT0	External interrupt request 0
12	0x000B	0x0016	TIMER1 CAPT	Timer/Counter1 capture event
13	0x000C	0x0018	TIMER1 OVF	Timer/Counter1 overflow
14	0x000D	0x001A	ADC	ADC conversion complete
15	0x000E	0x001C	INT1	External interrupt request 1
16	0x000F	0x001E	SPI, STC	SPI serial transfer complete
17	0x0010	0x0020	INT2	External interrupt request 2
18	0x0011	0x0022	WDT	Watchdog time-out interrupt
19	0x0012	0x0024	EE READY	EEPROM ready
20	0x0013	0x0026	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 233.
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 8-2 on page 63 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 8-2. Reset and interrupt vectors placement in AT90PWM81/161 (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 20-7 on page 246. 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 AT90PWM81 is:

(for AT90PWM161, interrupt vector address have an increment equal to 2 in place of 1 for AT90PWM81)

```
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   PSC2_EEC       ; PSC2 End Enhanced Cycle Handler
0x004          rjmp   PSCR_CAPT      ; PSCr Capture event Handler
0x005          rjmp   PSCR_EC        ; PSC0 End Cycle Handler
0x006          rjmp   PSCR_EEC       ; PSCr End Enhanced 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_OVF       ; Timer1 Overflow Handler
0x00D          rjmp   ADC            ; ADC Conversion Complete Handler
0x00E          rjmp   EXT_INT1       ; IRQ1 Handler
0x00F          rjmp   SPI_STC        ; SPI Transfer Complete Handler
0x010          rjmp   EXT_INT2       ; IRQ2 Handler
0x011          rjmp   WDT            ; Watchdog Timer Handler
0x012          rjmp   EE_RDY         ; EEPROM Ready Handler
0x013          rjmp   SPM_RDY        ; Store Program Memory Ready Handler
0x014          rjmp
0x015          rjmp
0x016          rjmp
0x017          rjmp
0x018          rjmp
0x019          rjmp
0x01A          rjmp
0x01B          rjmp
```

```
0x01C          rjmp
0x01F          rjmp
;
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 2Kbytes 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 AT90PWM81/161 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
;
```

For AT90PWM81

```
.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
```

For AT90PWM161

```
.org 0xC01
0xC02          rjmp   PSC2_CAPT      ; PSC2 Capture event Handler
0xC04          rjmp   PSC2_EC        ; PSC2 End Cycle Handler
...            ...    ...            ;
0xC3F          rjmp   SPM_RDY        ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 2Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses in AT90PWM81/161 is:

```
Address  Labels Code            Comments
```

For AT90PWM81

```
.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
```

For AT90PWM161

```
.org 0x001
0x002          rjmp   PSC2_CAPT      ; PSC2 Capture event Handler
0x004          rjmp   PSC2_EC        ; PSC2 End Cycle Handler
...            ...    ...            ;
0x03F          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 2Kbytes 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 AT90PWM81/161 is:

```
Address  Labels Code                 Comments
;
```

For AT90PWM81

```
.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
```

For AT90PWM161

```
.org 0xC00
0xC00          rjmp   RESET          ; Reset handler
0xC02          rjmp   PSC2_CAPT      ; PSC2 Capture event Handler
0xC04          rjmp   PSC2_EC        ; PSC2 End Cycle Handler
...            ...    ...            ;
0xC3F          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
```

8.1.1 Moving Interrupts Between Application and Boot Space

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

8.1.2 MCUCR - MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	PUD	RSTDIS	CKRC81	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R/W	R/W	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 233 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

a. Write the Interrupt Vector Change Enable (IVCE) bit to one.
b. 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 233 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 next page.

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

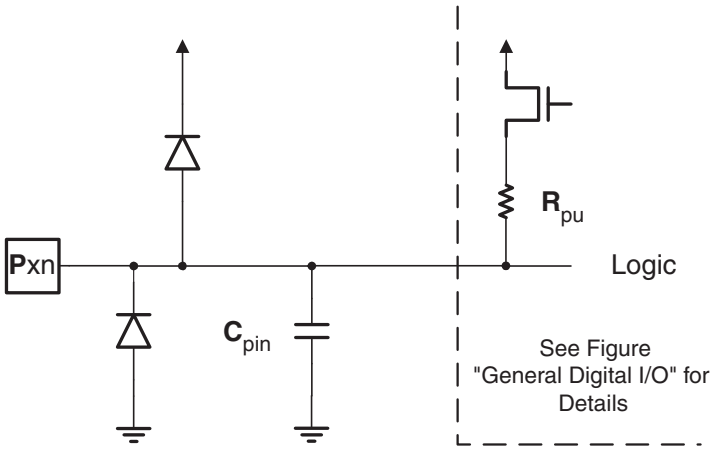
```
void Move_interrupts(void)
{
  /* Enable change of Interrupt Vectors */
  MCUCR = (1<<IVCE);
  /* Move interrupts to Boot Flash section */
  MCUCR = (1<<IVSEL);
}
```

9. I/O-Ports

9.1 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 9-1. Refer to "Electrical Characteristics (1)" on page 265 for a complete list of parameters.

Figure 9-1. 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" on page 81.

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.

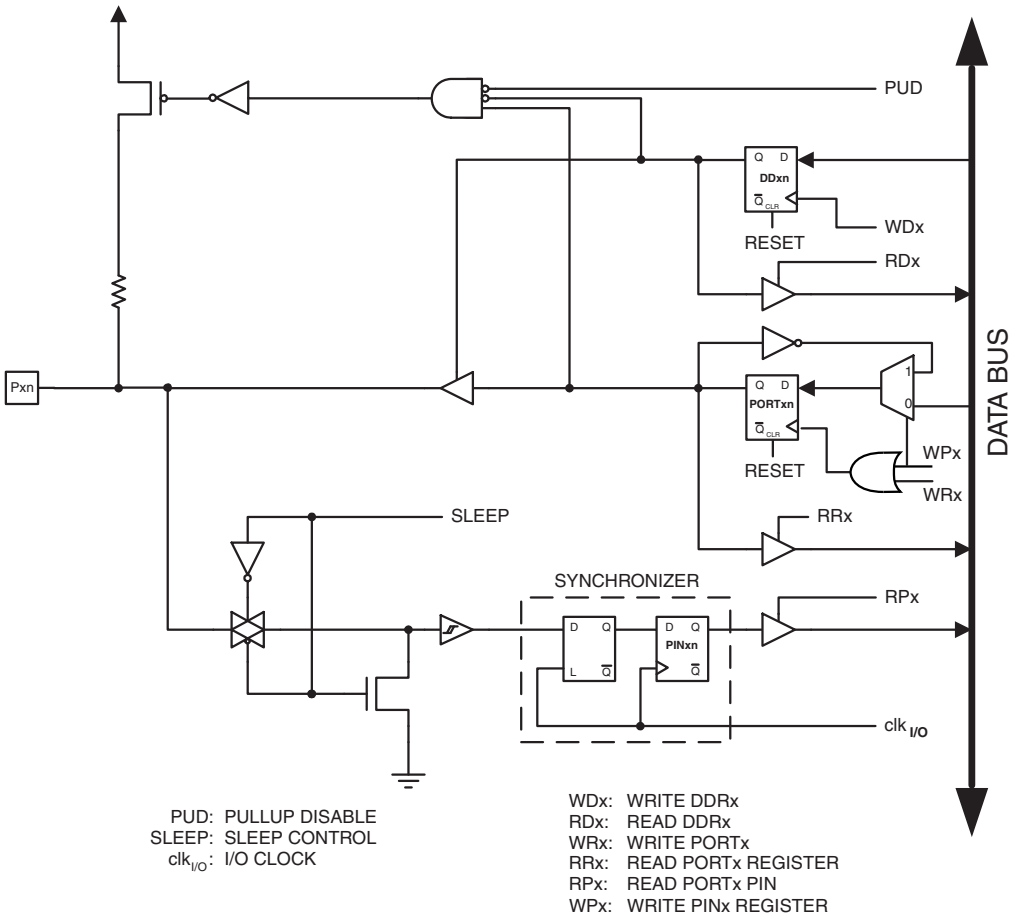
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 69. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 73. 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.

9.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 9-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 9-2. 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.

9.2.1 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 81, 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).

9.2.2 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.

9.2.3 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 9-1 summarizes the control signals for the pin value.

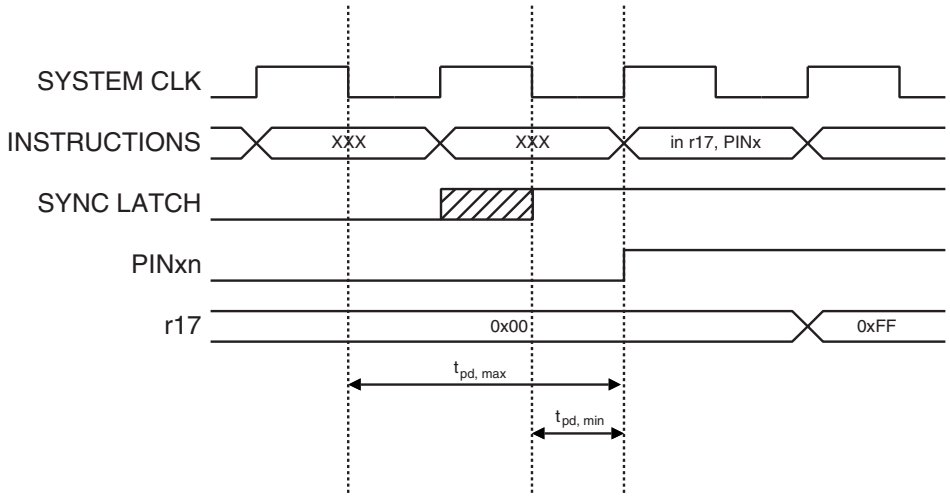
Table 9-1. Port pin configurations.

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)

9.2.4 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 9-2 on page 69, 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 9-3 on page 71 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.

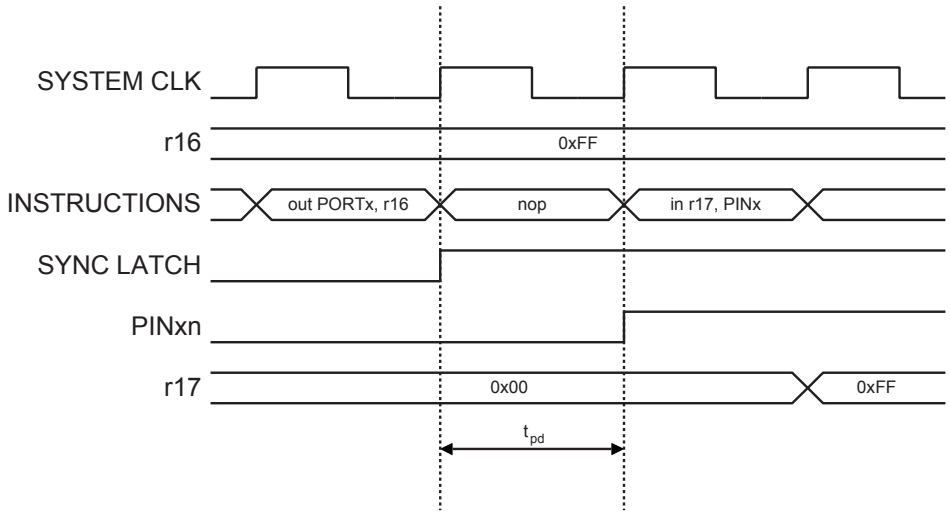
Figure 9-3. Synchronization when reading an externally applied pin value.



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 9-4. 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 9-4. 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.

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.

9.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 9-2 on page 69, 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, and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

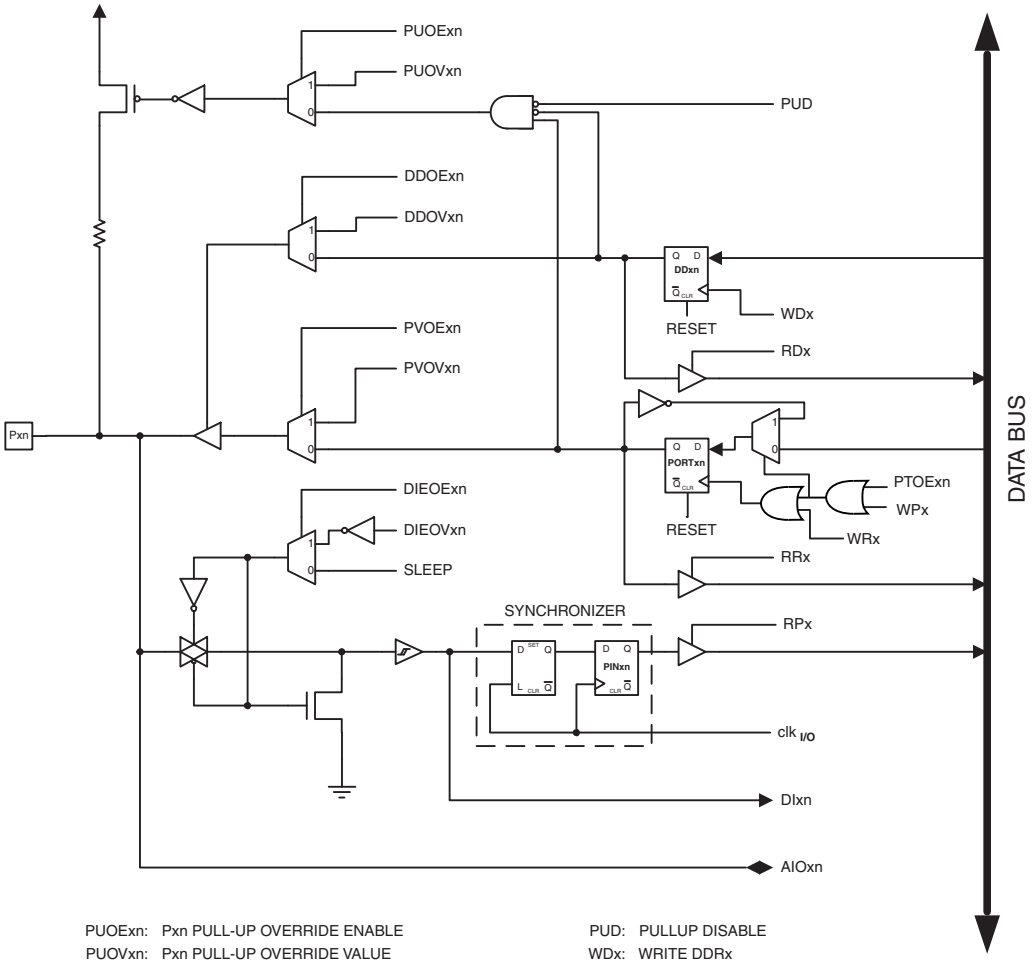
SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 73.

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.

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-5 shows how the port pin control signals from the simplified Figure 9-2 on page 69 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 9-5. 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 9-2 on page 74 summarizes the function of the overriding signals. The pin and port indexes from Figure 9-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 9-2. 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 9-5 on page 73, 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 Table 9-2 for further details.

9.3.1 MCUCR - MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	PUD	RSTDIS	**CKRC81**	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	RW	R/W	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). See "Configuring the Pin" on page 69 for more details about this feature.

9.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 9-3.

Table 9-3. Port B pins alternate functions.

Port pin	Alternate functions
PB7	PSCOUT22 output ICP1 (Timer/Counter1 Input Capture Pin) ADC9 (Analog Input Channel 9)
PB6	MISO (SPI Master In Slave Out) ACMP3 (Analog Comparator 3 Positive Input) ADC8 (Analog Input Channel 8)
PB5	ADC5 (Analog Input Channel 5) ACMP2 (Analog Comparator 2 Positive Input) INT1(External Interrupt 1 Input) SCK (SPI Clock)
PB4	MOSI (SPI Master Out Slave In) ADC3 (Analog Input Channel 3) ACMPM reference for analog comparators
PB3	PSCOUTR1 Output ADC2 (Analog Input Channel 2) ACMP2M (Analog Comparator 2 Negative Input)
PB2	INT0 (External Interrupt 0 Input) PSCOUT21 output
PB1	PSCOUT20 output
PB0	T1 counter source PSCOUT23 output ACMP3_OUT(Analog Comparator3 Output)

The alternate pin configuration is as follows:

- **PSCOUT22/ICP1/ADC9 – Bit 7**

PSCOUT22: Output 2 of PSC 2
ICP1 – Input Capture Pin1: This pin can act as an input capture pin for Timer/Counter1.
ADC9: Analog to Digital Converter, input channel 9.

- **MISO/ACMP3/ADC8– Bit 6**

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 DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 and PUD bits.

ACMP3: Analog Comparator 3 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.

ADC8: Analog to Digital Converter, input channel 8.

- **ADC5/ACMP2/$\overline{\text{INT1}}$/SCK – Bit 5**

ADC5: Analog to Digital Converter, input channel 5.

ACMP2: Analog Comparator 2 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.

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

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 DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit.

- **MOSI/ADC3/ACMPM– Bit 4**

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 DDB4 When the SPI is enabled as a master, the data direction of this pin is controlled by DDB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 and PUD bits.

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.

- **PSCOUTR1/ADC2/ACMP2M– Bit 3**

PSCOUTR1: Output 1 of PSCR.

ADC2: Analog to Digital Converter, input channel 2.

ACMP2M: Analog Comparator 2 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.

- **$\overline{\text{INT0}}$/PSCOUT21 – Bit 2**

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

PSCOUT21: Output 1 of PSC 2.

- **PSCOUT20 – Bit 1**

PSCOUT20: Output 0 of PSC 2.

- **T1/PSCOUT23/ACMP3_OUT – Bit 0**

T1: Timer/Counter1 counter source.

PSCOUT23: Output 3 of PSC 2.

ACMP3_OUT: Analog Comparator3 Output.

Table 9-4 and Table 9-5 relates the alternate functions of Port B to the overriding signals shown in Figure 9-5 on page 73.

Table 9-4. Overriding signals for alternate functions in PB7..PB4.

Signal name	PB7/PSCOUT22/ ICP1/ADC9	PB6/MISO/ ACMP3/ADC8	PB5/ADC5/ ACMP2/$\overline{\text{INT1}}$/SCK	PB4/MOSI/ADC3 /ACMPM
PUOE	0	SPE.MSTR	SPE.$\overline{\text{MSTR}}$	SPE.$\overline{\text{MSTR}}$
PUOV	0	PB6.$\overline{\text{PUD}}$	PB5.$\overline{\text{PUD}}$	PB4.$\overline{\text{PUD}}$
DDOE	PSCen22	SPE.MSTR	SPE.$\overline{\text{MSTR}}$	SPE.$\overline{\text{MSTR}}$
DDOV	1	0	0	0
PVOE	PSCen22	SPE.$\overline{\text{MSTR}}$	SPE.MSTR	SPE.MSTR
PVOV	PSCout22	MISO	SCK	MOSI
DIEOE	ADC9D	ADC8D	ADC5D|In1en	ADC3D
DIEOV	0	0	In1en	0
DI	ICP1	-	INT1	-
AIO	ADC9	ACMP3/ADC8	ADC5/ACMP2	ADC3/ACMPM

Table 9-5. Overriding signals for alternate functions in PB3..PB0.

Signal name	PB3/PSCOUTR1/ ADC2/ACMP2M	PB2/INT0/PSCOUT21/ ADC2/ACMP2M	PB1/PSCOUT20	PB0/T1/PSCOUT23 /ACMP3_OUT
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	PSCen01	PSCen21	PSCen20	PSCen23|AC3EN
DDOV	1	1	1	1
PVOE	PSCen01	PSCen21	PSCen20	PSCen23|AC3EN
PVOV	PSCOUTR1	PCOUT21	PCOUT20	(1)
DIEOE	ADC2D	In0en	-	-
DIEOV	0	In0en	-	-
DI	-	INT0	-	-
AIO	ADC2/ACMP2M	-	-	-

Note: 1. If PSCen23 : PSCOUT23
else
{if AC3EN : ACMP3OUT
else : Ø
}

The alternate pin configuration is as follows:

9.3.3 Alternate Functions of Port D

The Port D pins with alternate functions are shown in Table 9-6.

Table 9-6. Port D pins alternate functions.

Port pin	Alternate function
PD7	ADC10 (Analog Input Channel 10) PCSINrA (PSCR first Alternate Digital Input)
PD6	AMP0+ (Analog Differential Amplifier 0 Input Channel)
PD5	AMP0- (Analog Differential Amplifier 0 Input Channel) ADC7 (Analog Input Channel 7)
PD4	ACMP3M (Analog Comparator 3 Negative Input) ADC4 (Analog Input Channel 4) PCSIN2A (PSC 2 Digital Input)
PD3	ADC1 (Analog Input Channel 1) ACMP2_OUT (Analog Comparator 2 Output)
PD2	ADC0 (Analog Input Channel 0) ACMP1 (Analog Comparator 1 Positive Input)
PD1	PSCOUTR0 Output 0 PCSINrB (PSCR Second Alternate Digital Input)
PD0	ACMP3_OUT_A (Analog Comparator 2 Alternate Output) CLKO (System Clock) SS (SPI Slave Select)

The alternate pin configuration is as follows:

- **ADC10/PSCINrA – Bit 7**

ADC10: Analog to Digital Converter, input channel 10.

PCSINrA: PSCR First Alternate Digital Input.

- **APM0+ – Bit 6**

AMP0+: Analog Differential Amplifier 0 Positive Input Channel.

- **AMP0-/ADC7 – Bit 5**

AMP0-: Analog Differential Amplifier 0 Negative Input Channel.

ADC7: Analog to Digital Converter, input channel 7.

- **ACMP3M/ADC4/PSCIN2A – Bit 4**

ACMP3M: Analog Comparator 3 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.

ADC4: Analog to Digital Converter, input channel 4.

PCSIN2A: PSC 2 Alternate Digital Input.

- **ADC1/ACMP2_OUT, Bit 3**

ADC1: Analog to Digital Converter, input channel 1.

ACMP2_OUT: Analog Comparator 2 Output.

- **ADC0/ACMP1, Bit 2**

ADC0: Analog to Digital Converter, input channel 0.

ACMP1: 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.

- **PSCOUTR0/PSCINrB – Bit 1**

PSCOUTR0: Output 0 of PSCR.

PCSINrB: PSCR Second Alternate Digital Input.

- **ACMP3_OUT_A/$\overline{SS}$/CLKO – Bit 0**

ACMP2_OUT_A: Analog Comparator 2 Alternate Output.

$\overline{SS}$: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDDn. 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 DDDn. When the pin is forced to be an input, the pull-up can still be controlled by the PORTDn bit.

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 PORTDn and DDDn settings. It will also be output during reset.

Table 9-7 and Table 9-8 on page 80 relates the alternate functions of Port D to the overriding signals shown in Figure 9-5 on page 73.

Table 9-7. Overriding signals for alternate functions PD7..PD4.

Signal name	PD7/ADC10/ PSCINrA	PD6/APM0+	PD5/AMP0-/ADC7	PD4/ACMP3M/ ADC4/PSCIN2A
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	ADC10D	AMP0+D	ADC7D	ADC4D
DIEOV	0	0	0	0
DI	PSCiNrA	-	-	PSCIN2A
AIO	ADC10	AMP0+	ADC7/AMP0-	ADC4/ACMP3M

Table 9-8. Overriding signals for alternate functions in PD3..PD0.

Signal name	PD3/ADC1/ ACMP2_OUT	PD2/ADC0/ ACMP1	PD1/PSCOUTR0/ PSCINrB	PD0/ACMP3_OUT/$\overline{SS}$/ CLKO
PUOE	0	0	0	SPE.$\overline{MSTR}$
PUOV	0	0	0	PD0.$\overline{PUD}$
DDOE	ACE2EN	0	PSCen00	ACMP3D|(SPE.$\overline{MSTR}$)
DDOV	1	0	1	AC3EN
PVOE	AC2EN	0	PSCen00	AC3EN
PVOV	ACMP2_OUT	0	PSCOUT00	AVCMP3_OUT
DIEOE	ADC1D	ADC0D	0	0
DIEOV	0	0	0	0
DI	-	-	PSCINrB	$\overline{SS}$
AIO	ADC1	ADC0/ACMP1	-	-

9.3.4 Alternate Functions of Port E

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

Table 9-9. Port E pins alternate functions.

Port pin	Alternate function
PE3	AREF (Analog reference voltage) ADC6 (Analog input channel 6)
PE2	XTAL2: XTAL Output ACMP1M (Analog Comparator 1 Negative Input) PCSINr (PSCR Digital Input)
PE1	XTAL1: XTAL Input PCSIN2 (PSC 2 Digital Input) ACMP1_OUT (Analog Comparator 1 Output)
PE0	$\overline{RESET}$ (Reset Input) OCD (On Chip Debug I/O) INT2 (External Interrupt 2 Input)

The alternate pin configuration is as follows:

- **AREF/ADC6, Bit 3**

AREF: Analog reference voltage. See Table 17-3 on page 218 for the definition of this pin.

ADC6: Analog to Digital Converter, input channel 6.

This pin can only be used as a digital output pin. It cannot be read as a digital input.

- **XTAL2/ACMP1M/PSCINr – 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.

ACMP1M: Analog Comparator 1 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.

PCSINr: PSCR Digital Input.

- **XTAL1/PSCIN2/ACMP1_OUT – 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.

PCSIN2: PSC 2 Digital Input.

ACMP1_OUT: Analog Comparator 1 Output.

- **$\overline{\text{RESET}}$/OCD/INT2 – 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.

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

Table 9-10 relates the alternate functions of Port E to the overriding signals shown in Figure 9-5 on page 73.

Table 9-10. Overriding signals for alternate functions in PE2..PE0.

Signal name	PE3/AREF/ADC6	PE2/XTAL2/ACMP1M/PSCINr	PE1/XTAL1/PSCIN2/ ACMP1_OUT	PE0/$\overline{\text{RESET}}$/OCD/INT2
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	AC1EN	0
DDOV	0	0	1	0
PVOE	0	0	AC1EN	0
PVOV	0	0	ACMP1_OUT	0
DIEOE	0	ACMP1MD	0	In2en
DIEOV	0	0	0	In2en
DI	-	PSCINr	PSCIN2	INT2
AIO	ADC6	ACMP1M	-	-

9.4 Register Description for I/O-Ports

9.4.1 PORTB - Port B Data Register

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	

9.4.2 DDRB - Port B Data Direction Register

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	

9.4.3 PINB - Port B Input Pins Address

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	

9.4.4 PORTD - Port D Data Register

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	

9.4.5 DDRD - Port D Data Direction RegisterS

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	

9.4.6 PIND - Port D Input Pins Address

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	

9.4.7 PORTE - Port E Data Register

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	

9.4.8 DDRE - Port E Data Direction Register

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	

9.4.9 PINE - Port E Input Pins Address

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	

10. External Interrupts

The External Interrupts are triggered by the INT2:0 pins. Observe that, if enabled, the interrupts will trigger even if the INT2:0 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 (INT2:0). 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 INT2:0 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 1µs (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 265. 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 and Clock Options” 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.

10.0.1 EICRA - External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
	-	-	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	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 – ISC21, ISC20 – ISC01, ISC00: External Interrupt 2 - 0 Sense Control Bits**

The External Interrupts 3 - 0 are activated by the external pins INT2:0 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 10-1. Edges on INT3..INT0 are registered asynchronously.The value on the INT2:0 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 10-1. Interrupt sense Ccntrol [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.

10.0.2 EIMSK - External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	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 2..0 – INT2 – INT0: External Interrupt Request 3 - 0 Enable**

When an INT2 – 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.

10.0.3 EIFR - External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	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 2..0 – INTF2 - INTF0: External Interrupt Flags 3 - 0**

When an edge or logic change on the INT2:0 pin triggers an interrupt request, INTF2:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT2:0 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 INT2:0 are configured as level interrupt.

11. Reduced 16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management). The main features are:

- **Clear timer on compare match (auto reload)**
- **One input capture unit**
- **Input capture noise canceler**
- **External event counter**
- **Two independent interrupt sources (TOV1, ICF1)**

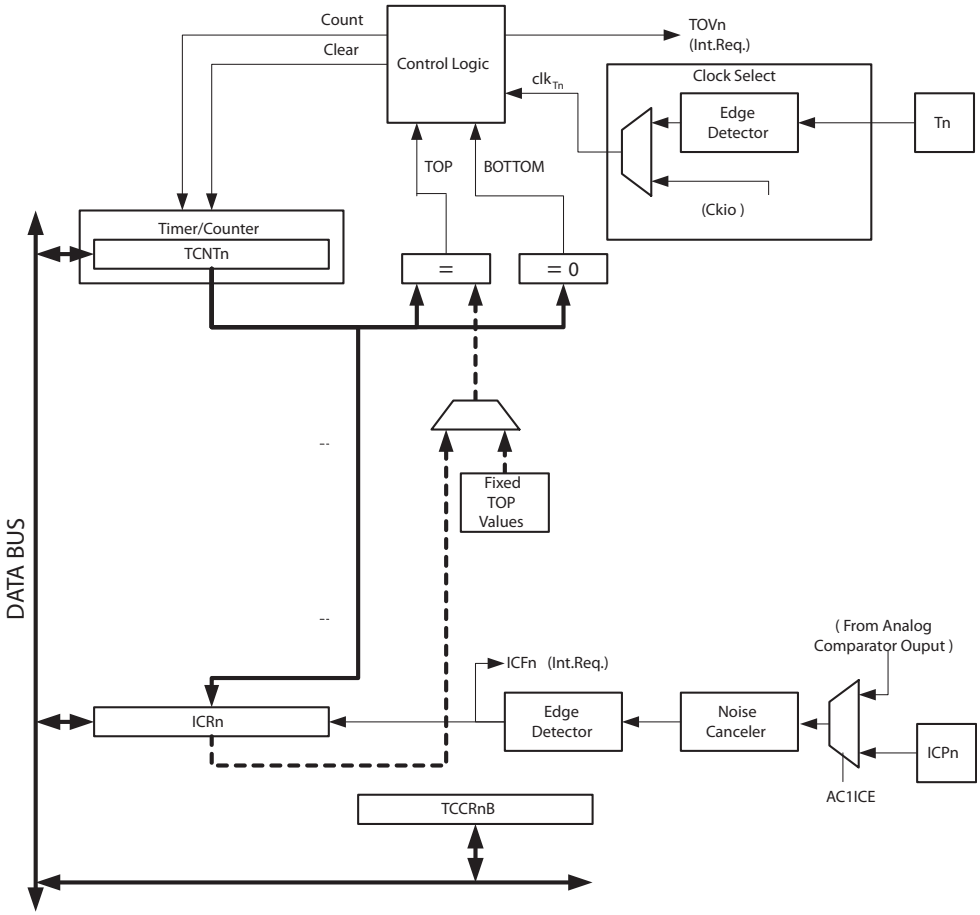
11.1 Overview

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, that is, TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 11-1 on page 86. For the actual placement of I/O pins, refer to Table 2-2 on page 6. 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 97.

The PRTIM1 bit in “Power Reduction Register” on page 46 must be written to zero to enable Timer/Counter1 module.

Figure 11-1. 16-bit timer/counter block diagram [1].



Note: 1. Refer to Table 2-1 on page 5 for Timer/Counter1 pin placement and description.

11.1.1 Registers

The *Timer/Counter* (TCNT1), and *Input Capture Register* (ICR1) 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 87. The *Timer/Counter Control Registers* (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in Figure 11-1) signals are all visible in the *Timer Interrupt Flag Register* (TIFR1). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK1). TIFR1 and TIMSK1 are not shown in Figure 11-1.

The Timer/Counter can be clocked internally, or by an external clock source on the T1 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_{T1}).

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1). 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 the ICR1 Register, or by a set of fixed values.

11.1.2 Definitions

The following definitions are used extensively throughout the section:

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 ICR1 Register. The assignment is dependent of the mode of operation.

11.2 Accessing 16-bit Registers

The TCNT1, and ICR1 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.

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 ICR1 Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly code examples [1]

```
...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in  r16,TCNT1L
in  r17,TCNT1H
...
```

C code examples [1]

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

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 TCNT1 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 TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly code example [1]

```
TIM16_ReadTCNT1:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNT1 into r17:r16
  in  r16,TCNT1L
  in  r17,TCNT1H
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C code example [1]

```
unsigned int TIM16_ReadTCNT1( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Read TCNT1 into i */
  i = TCNT1;
  /* 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 TCNT1 value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly code example [1]

```
TIM16_WriteTCNT1:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNT1 to r17:r16
  out TCNT1H,r17
  out TCNT1L,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C code example [1]

```
void TIM16_WriteTCNT1( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNT1 to i */
  TCNT1 = 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 TCNT1.

11.2.1 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.

11.3 Timer/Counter Clock Sources

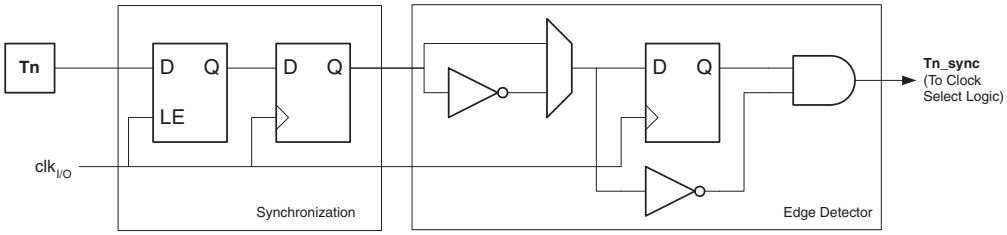
The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter control Register B* (TCCR1B).

11.3.1 External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/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 11-2 shows a functional equivalent block diagram of the T1/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 11-2. T1/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 T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/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.

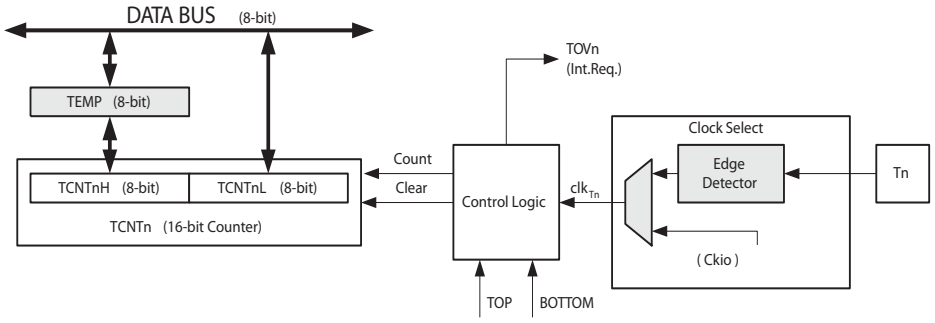
Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, 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.

11.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 11-3 shows a block diagram of the counter and its surroundings.

Figure 11-3. Counter unit block diagram.



Signal description (internal signals):

Count	Increment TCNT1 by 1.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/Counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L 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 TCNT1 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_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} 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* bit (WGM13) located in the *Timer/Counter Control Registers* B (TCCR1B).

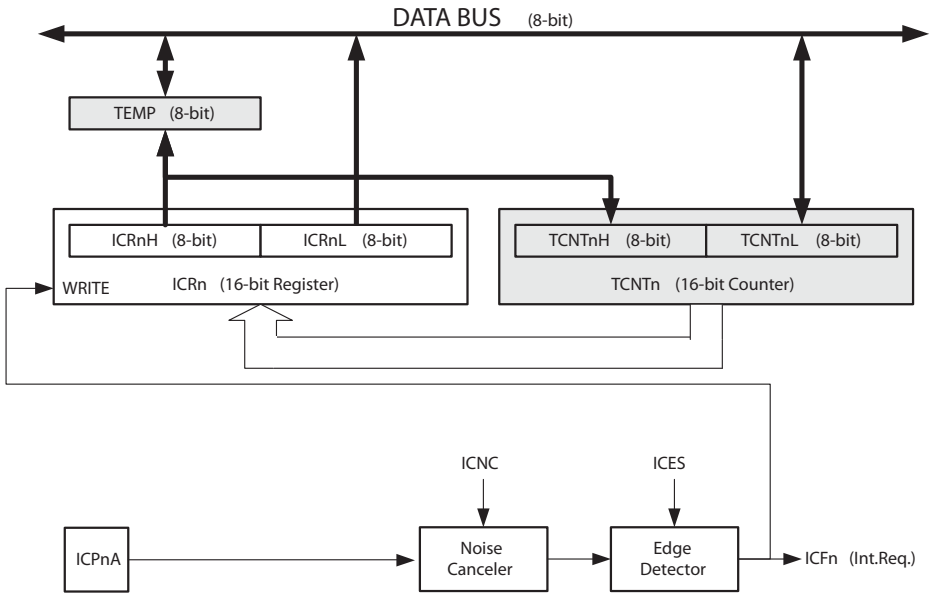
The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM13 bit. TOV1 can be used for generating a CPU interrupt.

11.5 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 ICP1 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 11-4 on page 93. 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 11-4. Input capture unit block diagram.



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP1), 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 (TCNT1) is written to the *Input Capture Register* (ICR1). The *Input Capture Flag* (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (ICIE1 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 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* (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the *Waveform Generation mode* (WGM13) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 87.

11.5.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the *Analog Comparator Input Capture* (AC1ICE) bit in the *Analog Comparator Extended Control Register* (AC1ECON). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator 1 output* (AC1O) inputs are sampled using the same technique as for the T1 pin (see Figure 11-2 on page 91). 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 ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

11.5.2 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* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). 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 ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

11.5.3 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 ICR1 Register before the next event occurs, the ICR1 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 ICR1 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 ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 Flag is not required (if an interrupt handler is used).

11.6 Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the Output Compare pins, is defined by the *Waveform Generation mode* (WGM1).

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 95.

11.6.1 Normal Mode

The simplest mode of operation is the *Normal mode* (WGM13:0 = 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* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 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 TOV1 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.

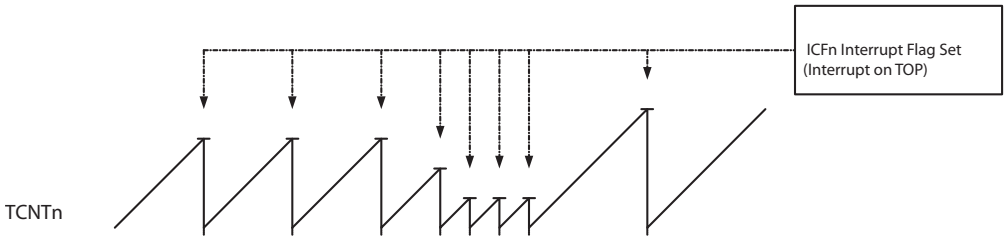
The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt must be used to extend the resolution for the capture unit.

11.6.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGM13 = 1, previous mode 12), the ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches the ICR1 . The ICR1 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 11-5. The counter value (TCNT1) increases until a compare match occurs with ICR1, and then counter (TCNT1) is cleared.

Figure 11-5. CTC mode, timing diagram.



An interrupt can be generated at each time the counter value reaches the TOP value by using the ICF1 Flag . 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 ICR1 is lower than the current value of TCNT1, 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.

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

11.7 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set.

Figure 11-6 on page 96 shows the count sequence close to TOP in various modes.

Figure 11-6. Timer/counter timing diagram, no prescaling.

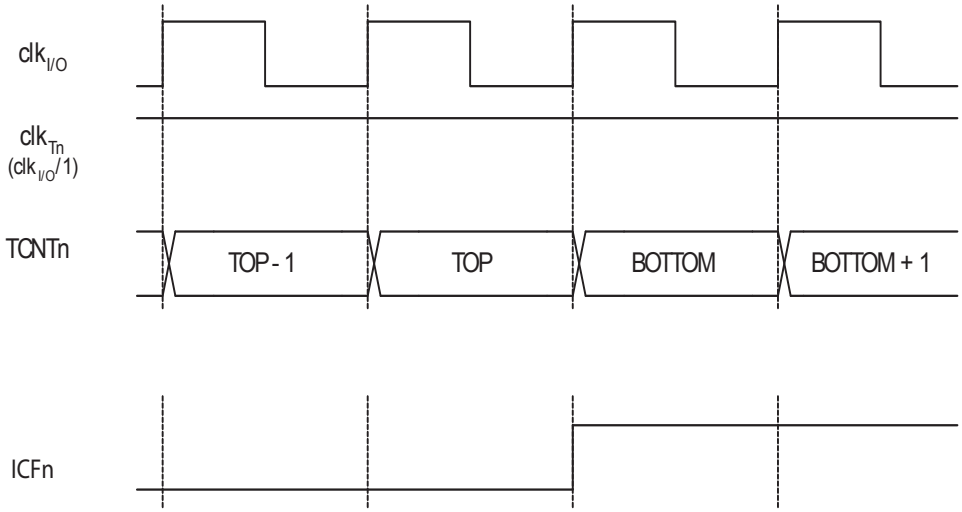
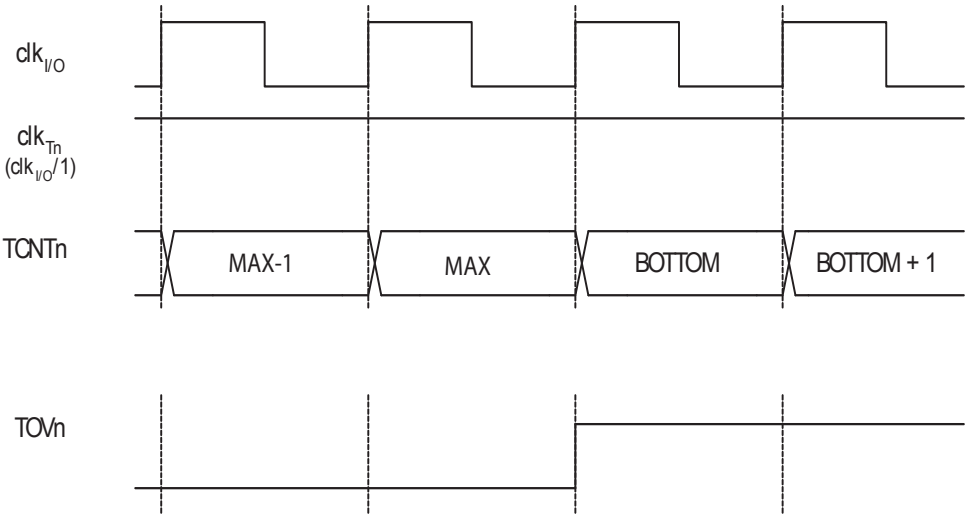


Figure 11-7 shows the count sequence close to MAX in various modes.

Figure 11-7. Timer/counter timing diagram, no prescaling.



11.8 16-bit Timer/Counter Register Description

11.8.1 TCCR1B - Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	-	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: 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 (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The Input Capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

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

This bit selects which edge on the Input Capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

- **Bit 5 – Reserved**

- **Bit 4 – WGM13: Waveform Generation Mode**

See Table 11-1 for the modes definition

Table 11-1. Waveform generation mode bit description.

Mode	WGM13	Timer/counter mode of operation	TOP	TOV1 flag set on
0	0	Normal	0xFFFF	MAX
12	1	CTC	ICR1	MAX

- **Bit 3 – Reserved**

- **Bit 2:0 – CS12:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Table 11-2.

Table 11-2. Clock select bit description.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	Reserved

Table 11-2. Clock select bit description. (Continued)

CS12	CS11	CS10	Description
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

11.8.2 TCNT1H and TCNT1L - Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The two *Timer/Counter* I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 87.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

11.8.3 ICR1H and ICR1L - Input Capture Register 1

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the Analog Comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 87.

11.8.4 TIMSK1 - Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	–	–	ICIE1	–	–	–	–	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 AT90PWM81/161, 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 Table 8-1 on page 62) is executed when the ICF1 Flag, located in TIFR1, is set.

- **Bit 4, 3, 2,1 – Res: Reserved Bits**

These bits are unused bits in the AT90PWM81/161, and will always read as zero.

- **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 Table 8-1 on page 62) is executed when the TOV1 Flag, located in TIFR1, is set.

11.8.5 TIFR1 - Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF1	–	–	–	–	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 AT90PWM81/161, and will always read as zero.

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

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bit 4, 3, 2,1 – Res: Reserved Bits**

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

The setting of this flag is dependent of the WG.

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.

12. Power Stage Controller – (PSCn)

The Power Stage Controller is a high performance waveform controller.

The Atmel AT90PWM81 includes one PSC2 block.

12.1 Features

- **PWM waveform generation function (two complementary programmable outputs)**
- **Dead time control**
- **Standard mode up to 12 bit resolution**
- **Frequency and pulse width resolution enhancement mode (12 + 4 bits)**
- **Frequency up to 64Mhz**
- **Conditional waveform on external events (zero crossing, current sensing ...)**
- **All on chip PSC synchronization**
- **ADC synchronization with digital delay register**
- **Input blanking**
- **Overload protection function**
- **Abnormality protection function, emergency input to force all outputs to low level**
- **Center aligned and edge aligned modes synchronization**
- **Fast emergency stop by hardware**

12.2 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 2. However, when using the register or bit defines in a program, the precise form must be used, that is, PSOC2 for accessing PSC 2 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, that is, PFRC2A for accessing PSC n Fault/Retrigger 2 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 PSCn and four outputs on PSC2.

These outputs can be used in various ways:

- “Two Outputs” 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 ...)

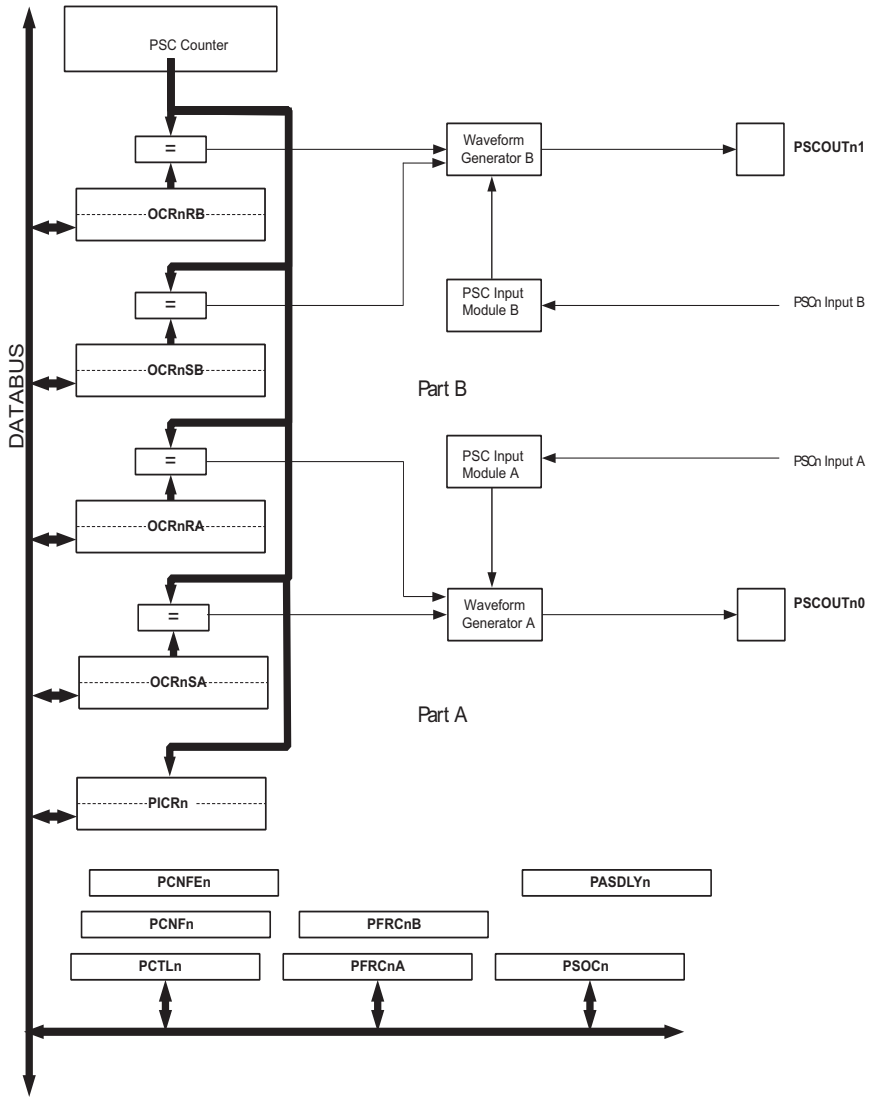
Each PSC has two inputs the purpose of which is to provide means to act directly on the generated waveforms:

- 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.

12.3 PSC Description

Figure 12-1. Power Stage Controller 0 or 1 block diagram.



Note: n = 0, 1.

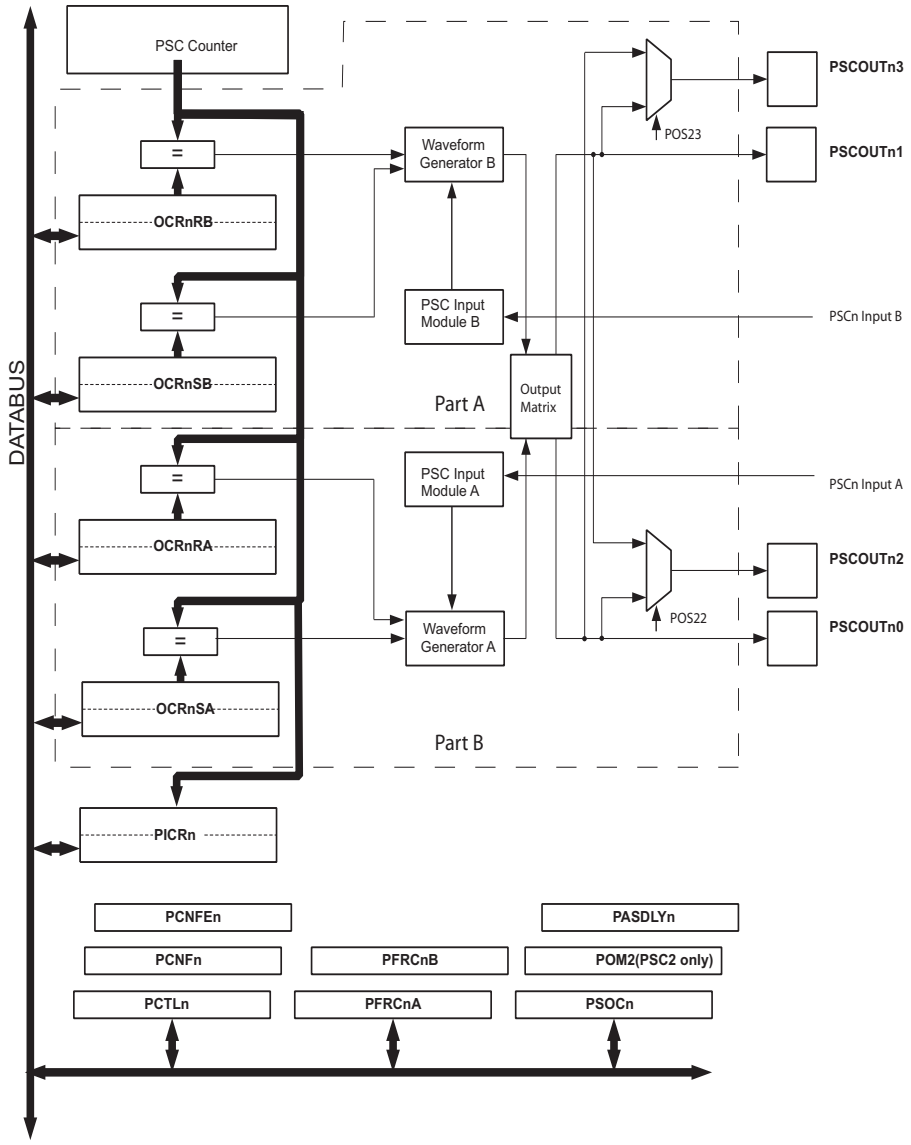
The principle of the PSC is based on the use of a counter (PSC counter). This counter is able to count up and count down from and to values stored in registers according to the selected running mode.

The PSC is seen as two symmetrical 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.

12.3.1 PSC2 Distinctive Feature

Figure 12-2. PSC2 versus PSC1&PSC0 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.

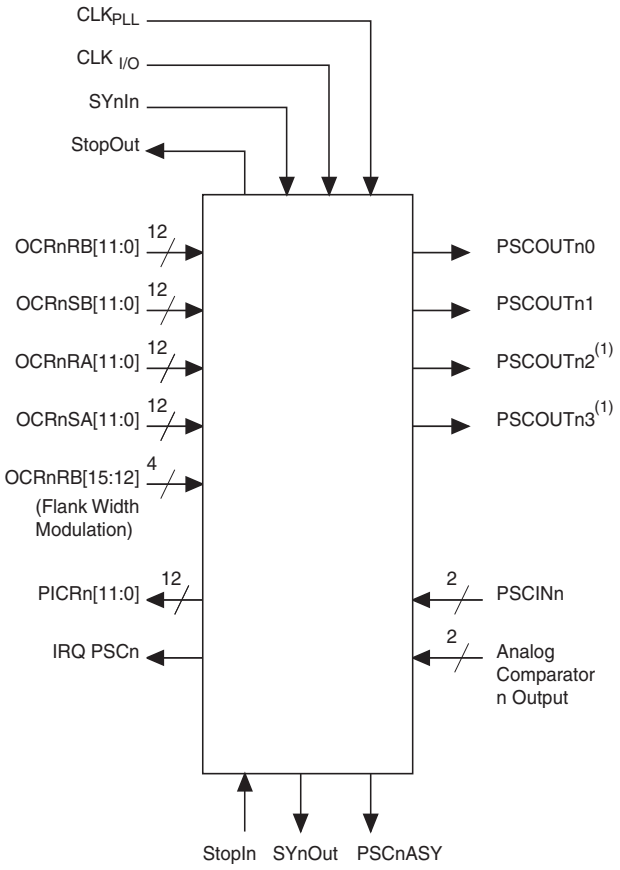
The Output Matrix is a kind of 2 × 2 look up table which gives the possibility to program the output values according to a PSC sequence (see “Output Matrix” on page 129).

12.3.2 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.

12.4 Signal Description

Figure 12-3. PSC external block view.



Note: 1. available only for PSC2.
2. n = 0, 1 or 2.

12.4.1 Input Description

Table 12-1. 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 12-41 on page 132

Table 12-2. Block inputs.

Name	Description	Type width
PSCINn	Input 0 used for Retrigger or Fault functions	Signal
from 1st A C	Input 1 used for Retrigger or Fault functions	Signal
PSCINnA	Input 2 used for Retrigger or Fault functions	Signal
from 2nd A C	Input 3 used for Retrigger or Fault functions	Signal

12.4.2 Output Description

Table 12-3. 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 12-4. 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 sources, overflow, fault, and input capture	Signal
PSCnASY	ADC Synchronization (+ Amplifier Syncho.) (2)	Signal
StopOut	Stop Output (for synchronized mode)	

Note: 1. See Figure 12-41 on page 132
2. See "Analog Synchronization" on page 131

12.5 Functional Description

12.5.1 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 Figure 12-4 on page 105.

The second waveform is relative to PSCOUTn1 output and part B of PSC. The part of this waveform is sub-cycle B in Figure 12-4 on page 105.

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

Figure 12-4. Cycle presentation in 1, 2, and 4 ramp mode.

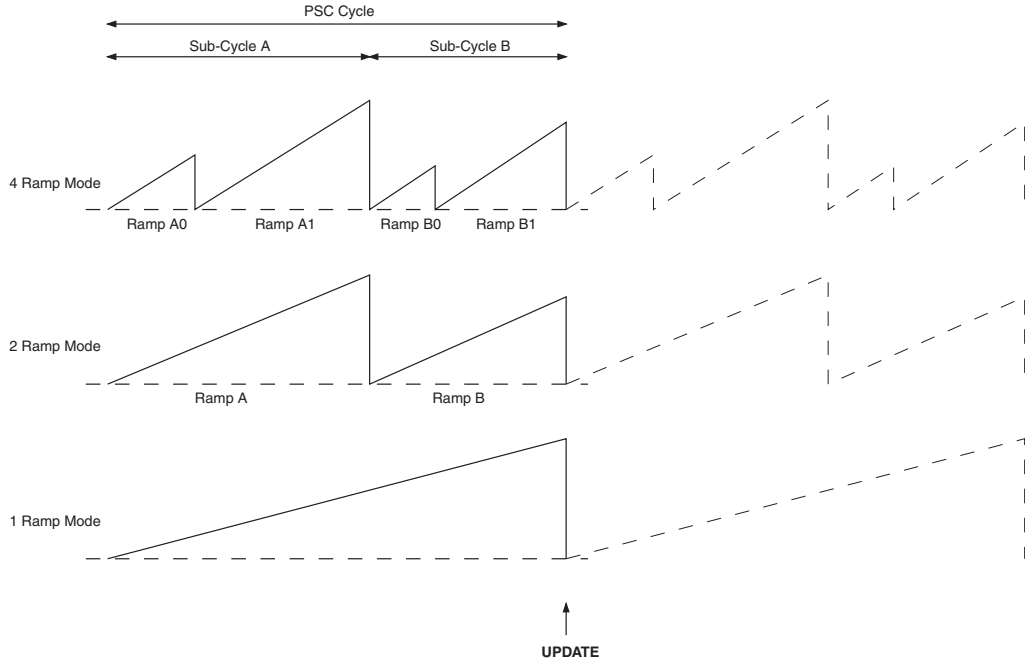
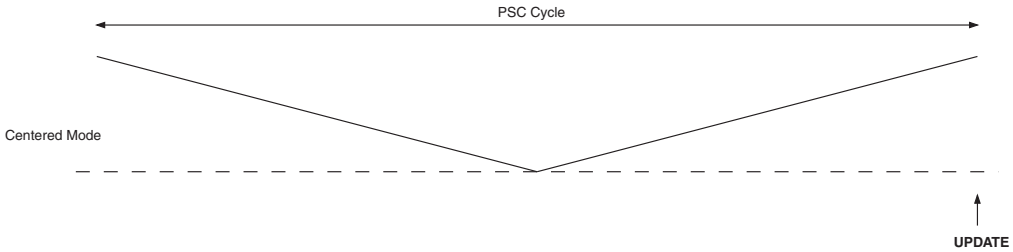


Figure 12-5. Cycle presentation in centered mode.



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.

12.5.2 Running Mode Description

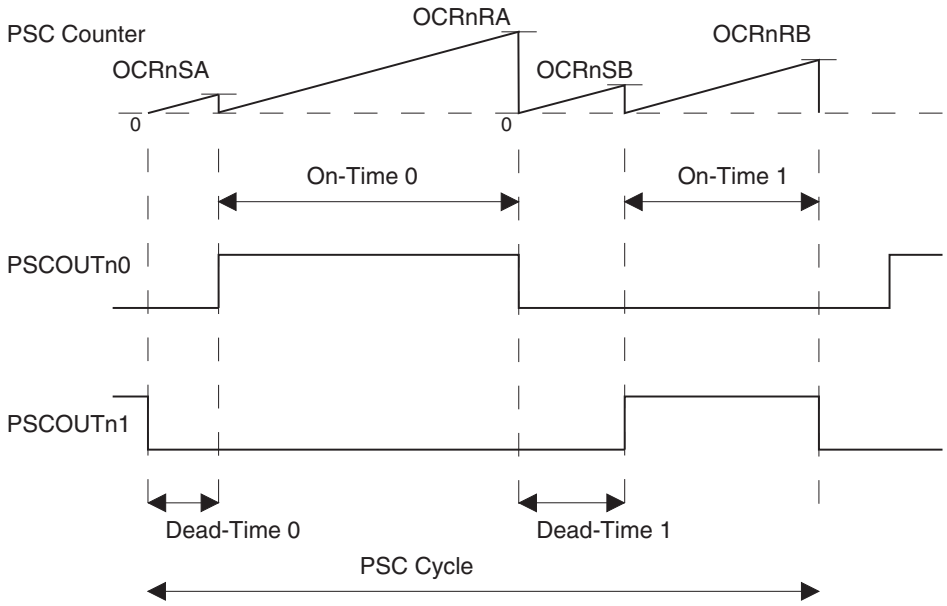
Waveforms and length of output signals are determined by Time Parameters (DT0, OT0, DT1, OT1) and by the running mode. Four modes are possible:

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

12.5.2.1 Four Ramp Mode

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

Figure 12-6. 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.

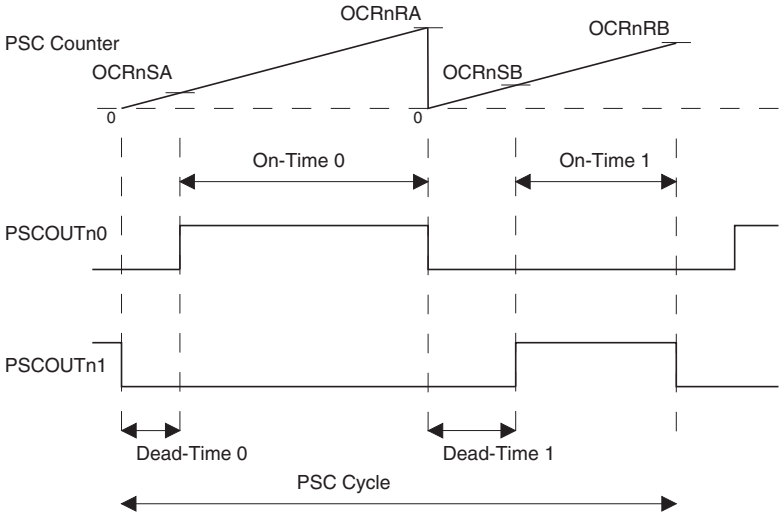
12.5.2.2 Two Ramp Mode

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 12-7. 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

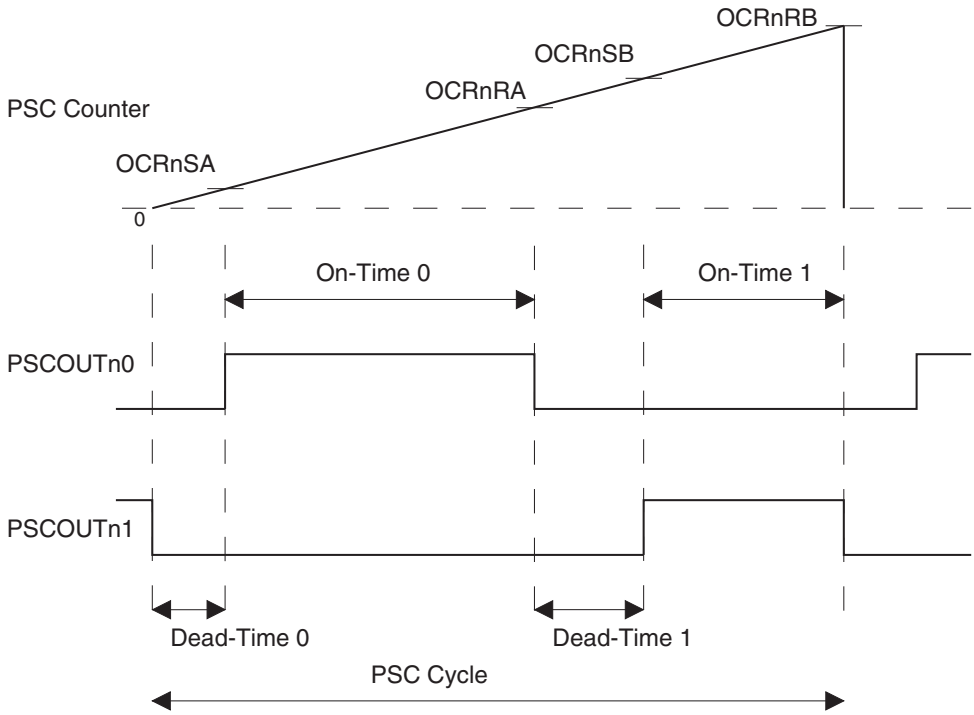
Dead-Time 1 = (OCRnSBH/L + 1) × 1/Fclkpsc

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

12.5.2.3 One Ramp Mode

In One Ramp mode, PSCOUTn0 and PSCOUTn1 outputs can overlap each other.

Figure 12-8. 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

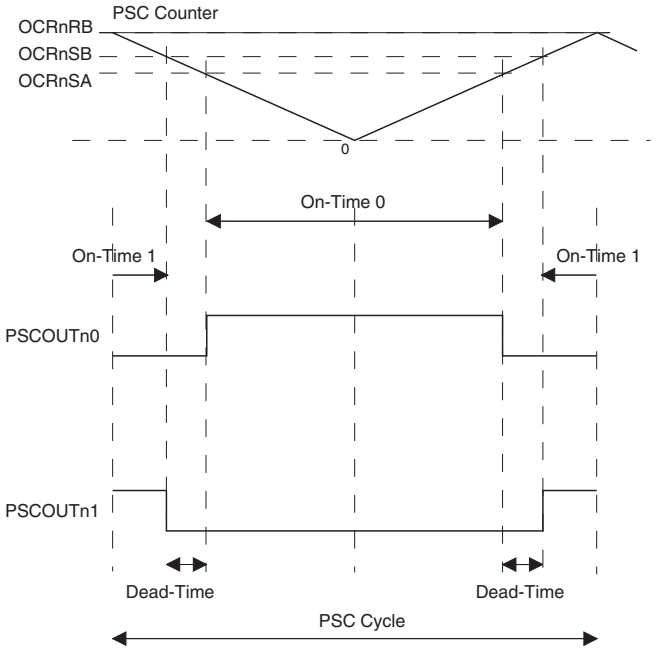
Dead-Time 1 = (OCRnSBH/L - OCRnRAH/L) × 1/Fclkpsc

Note: Minimal value for Dead-Time 0 = 1/Fclkpsc.

12.5.2.4 Center Aligned Mode

In center aligned mode, the center of PSCn00 and PSCn01 signals are centered.

Figure 12-9. 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

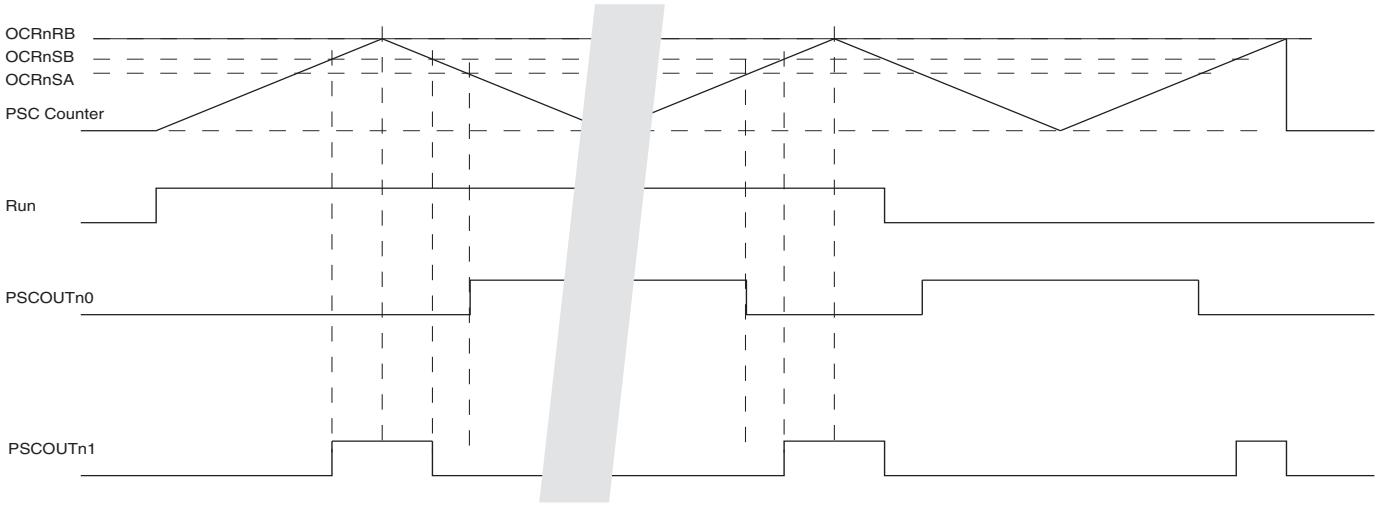
Dead-Time = (OCRnSBH/L - OCRnSAH/L) × 1/Fclkpsc

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 131).

Figure 12-10. Run and stop mechanism in Centered mode.



Note: See “PCTL2 - PSC 2 Control Register” on page 140 (or PCTL1 or PCTL2).

12.5.3 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.

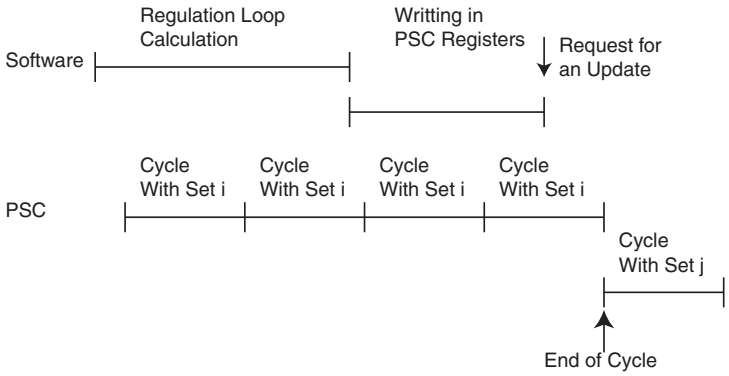
12.6 Update of Values

The update of PSC waveform registers are done in the following way:

- Immediately when the PSC is stopped
- At the PSC end of cycle when the PSC is running
- At the PSC end of cycle following the required condition when LOCK or AUTOLOCK modes are used

To avoid asynchronous and incoherent values in a cycle, if an update of one of several values is necessary, all values can be updated at the same time at the end of the cycle by the PSC. The new set of values is calculated by software and the update is initiated by software.

Figure 12-11. 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.

12.6.1 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 with the following conditions:

- When AUTOLOCK configuration is selected, the update of the PSC internal registers will be done at the end of the PSC cycle following a write in the Output Compare Register RB. 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 135.

When set, AUTOLOCK configuration bit prevails over LOCK configuration bit.

See "PCNF2 - PSC 2 Configuration Register" on page 136.

12.7 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 + Dead Time (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.

It is possible to apply the Flank Width Modulation (FWM) on RB, RB+RA, SB, SB+SA. The selection is done bit the bits PBFMn0 and PBFMn.

According to the ramp mode and the enhanced resolution mode (defined by PBFMn1:0), the frequency difference Df can take three different values:

$$\Delta f = 0$$

$$\Delta f1 = |f1 - f2| = \left|\frac{f_{PSC}}{k} - \frac{f_{PSC}}{k+1}\right| = f_{PSC} \times \frac{1}{k(k+1)}$$

$$\Delta f2 = |f1 - f2| = \left|\frac{f_{PSC}}{k} - \frac{f_{PSC}}{k+2}\right| = f_{PSC} \times \frac{2}{k(k+2)}$$

with k is the number of CLK_{PSC} period in a PSC cycle and is given by the following formula:

$$k = \frac{f_{PSC}}{f_{OP}}$$

with f_{OP} is the output operating frequency.

Example, in normal mode, with maximum operating frequency 160kHz and f_{PLL} = 64Mhz, k equals 400. The resulting resolution is Delta F equals 64MHz / 400 / 401 = 400Hz.

In enhanced mode, the output frequency is the average of the frame formed by the 16 consecutive cycles.

f_{b1} and f_{b2} are two neighboring base frequencies.

$$f_{AVERAGE} = \frac{16-d}{16} \times f_{b1} + \frac{d}{16} \times f_{b2}$$

Then the frequency resolution is divided by 16. In the example above, the resolution equals 25Hz.

$$f_{AVERAGE} = \frac{16-d}{16} \times \frac{f_{PLL}}{k} + \frac{d}{16} \times \frac{f_{PLL}}{k+1}$$

According to the ramp mode and the enhanced resolution mode (defined by PBFMn1:0), the average frequency deviation Df can take three different values:

$$\Delta f(average) = 0$$

$$\Delta f1(average) = f_{PSC} \times \frac{d}{16k(k+1)}$$

$$\Delta f2(average) = f_{PSC} \times \frac{d}{8k(k+2)}$$

These values are applied according to the running mode and the enhanced resolution mode as per Table 12-5;

It must be noted that, in one and two ramps modes, it is possible to apply the FWM only on **pulse width** while keeping a constant frequency.

Table 12-5. Frequency deviation with Flank Width Modulation.

	PBFMn1:0			
	00	01	10	11
Running mode	RB	RB+RA	SB	SB+SA
Four Ramps	Df1	Df2	Df1	Df2
Two Ramps	Df1	Df2	0 (1)	0
One Ramp	Df1	Df1	0	0
Center aligned	Df2	Df2	Df2	Df2

Notes: 1. The modulation is on the pulse width.

12.7.1 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 equivalent evenly distribution.

At the end of the 15th cycle (numbered 14 on Table 12-6 on page 113) an interrupt can be generated. This is the case if the bit PEOEPEn (PSC n End Of Enhanced Cycle Interrupt Enable) is set. This allows:

- To modify the modulation only on a new enhanced cycle start
- To extend the enhanced modulation accuracy by software

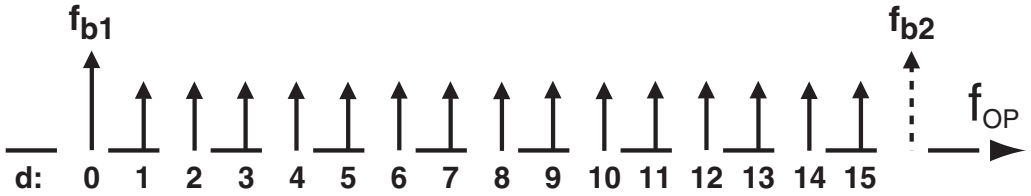
Table 12-6. Distribution of f_{b2} in the modulated frame.

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 12-12. Resulting frequency versus d.



12.7.2 Modes of Operation

12.7.2.1 Normal Mode

The simplest mode of operation is the normal mode. See Figure 12-6 on page 106.

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 adjust the dead time between PSCOUTn0 and PSCOUTn1 active signals.

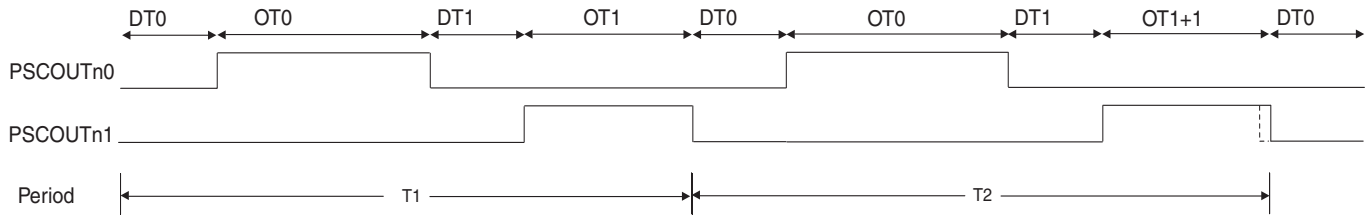
The waveform frequency is defined by the following equation:

$$f_{PSCn} = \frac{1}{PSCnCycle} = \frac{f_{CLK\_PSCn}}{(OT0 + OT1 + DT0 + DT1)}$$

12.7.2.2 Enhanced Mode

The Enhanced Mode uses the previously described method to generate a high resolution frequency. Figure 12-13 gives an example of FWM with PBFMn1:0 = 00.

Figure 12-13. 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. See Table 12-6 on page 113.

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 fractional divider factor.

The FWM can be applied on different locations within the PSC output waveforms as defined per Table 12-15 on page 138.

12.8 PSC Inputs

Part A or B of PSC has its own system to take into account one PSC n internal input. Each part A or B is configured by the PSC n Input A/B Control Register (“PFRCnA - PSC n Input A Control Register” on page 141 and “PFRCnB - PSC n Input B Control Register” on page 141) and the PSC n Extended Configuration Register (see Section “PCNF2 - PSC 2 Configuration Register”, page 136).

The PSC input module A is shown in Table 12-14 on page 115.

According to PSC n Input A Control Register (see “PFRCnA - PSC n Input A Control Register” on page 141), PSC n input A can act as a Retrigger or Fault input.

Each part A or B can be triggered by up to four signals as defined per Table 12-18 on page 139 and Table 12-19 on page 139.

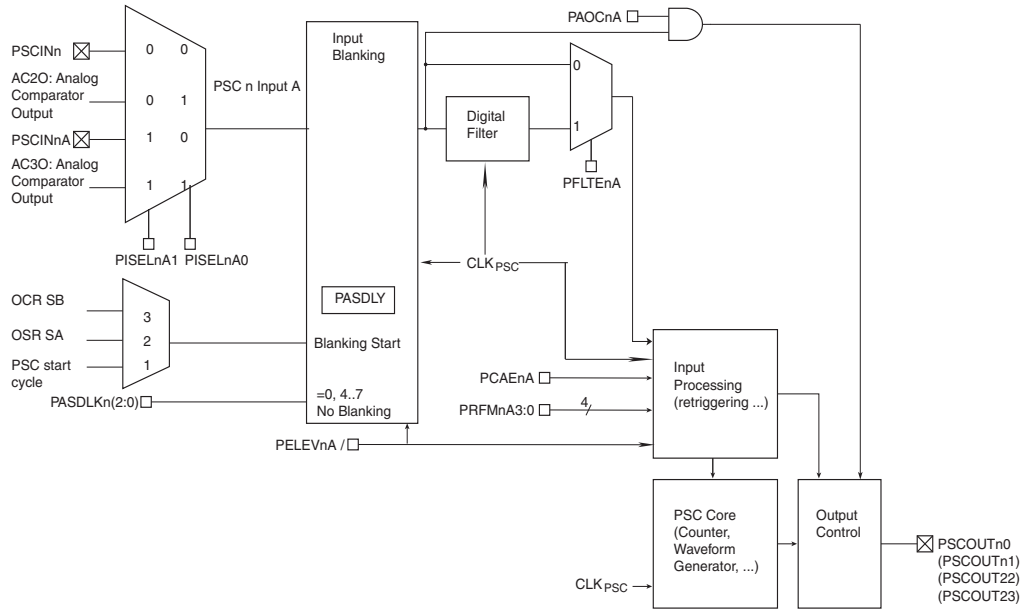
Part A of PSC has also a blanking module allowing to cancel unwanted transitions which may appear on the PSC n input A during a certain period of time.

The blanking start is defined by the bits PASDLKn(2:0) as per Table 12-14 on page 138.

The blanking duration is defined by the register PASDLYn. If the blanking is selected by the corresponding PASDLKn(2:0) bit, all transitions which may appears from the blanking start until a time period are ignored.

Blanking is level sensitive, that is, a pulse started in the blanking window and still at active level after the window will generate a valid retriggering event.

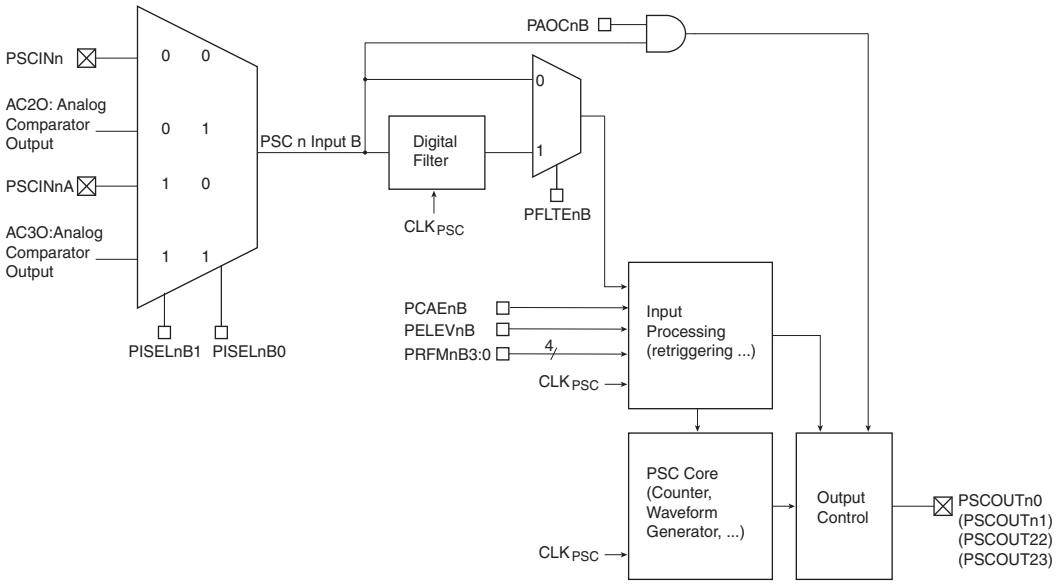
Figure 12-14. PSC input module A.



PSC input module B is shown on Table 12-15 on page 116.

According to PSC n Input B Control Register (see “PFRCnB - PSC n Input B Control Register” on page 141), PSC n input B can act as a Retrigger or Fault input.

Figure 12-15. PSC input module B.



12.8.1 PSC Retrigger Behavior 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.

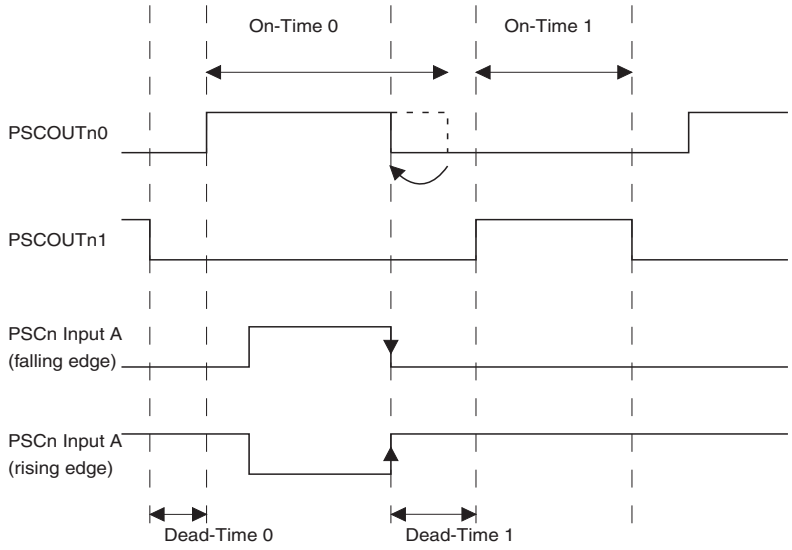
In one ramp mode, Retrigger Inputs A or B reset the current PSC counting to zero.

12.8.2 Retrigger PSCOUTn0 On External Event

PSCOUTn0 output can be reset 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.

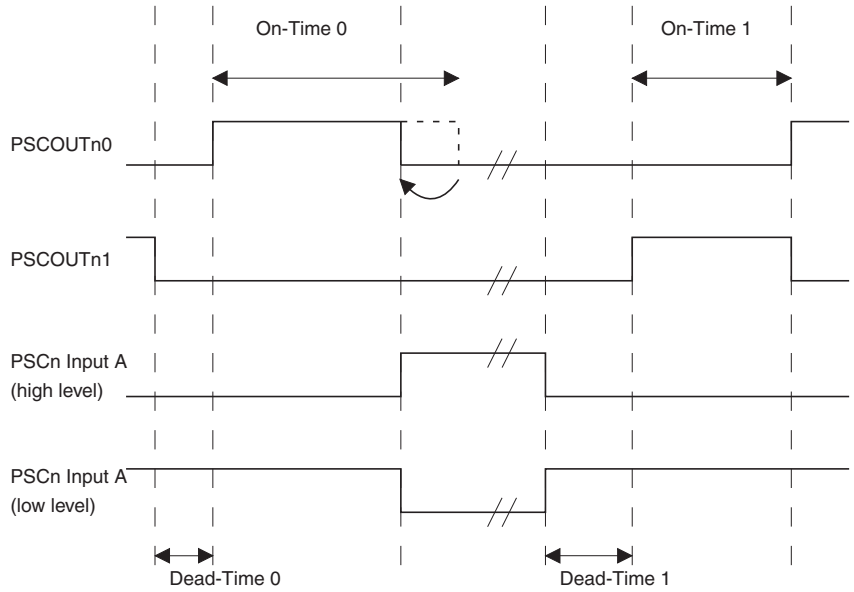
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 12-16. PSCOUTn0 retrograde by PSCn Input A (edge retriggering).



Note: This example is given in “Input Mode 8” in “2 or 4 ramp mode”. See Figure 12-33 on page 126 for details.

Figure 12-17. PSCOUTn0 retriggered by PSCn Input A (level acting).



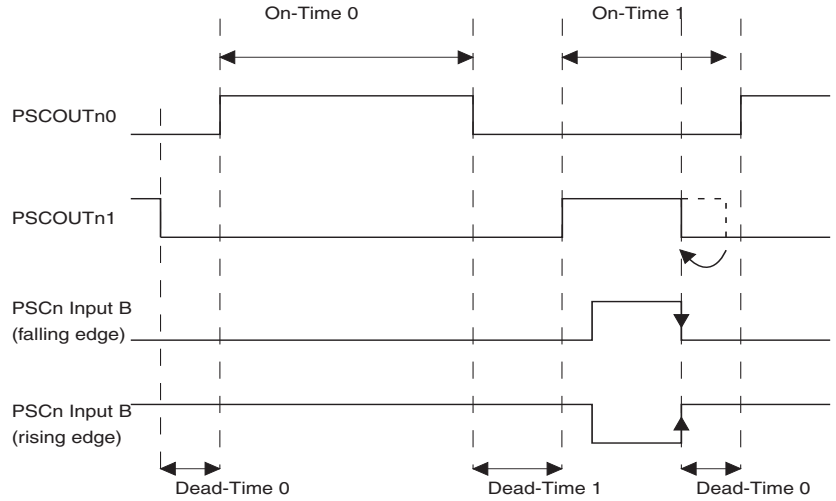
Note: This example is given in “Input Mode 1” in “2 or 4 ramp mode”. See Figure 12-22 on page 121 for details.

12.8.3 Retrigger PSCOUTn1 On External Event

PSCOUTn1 output can be reset 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.

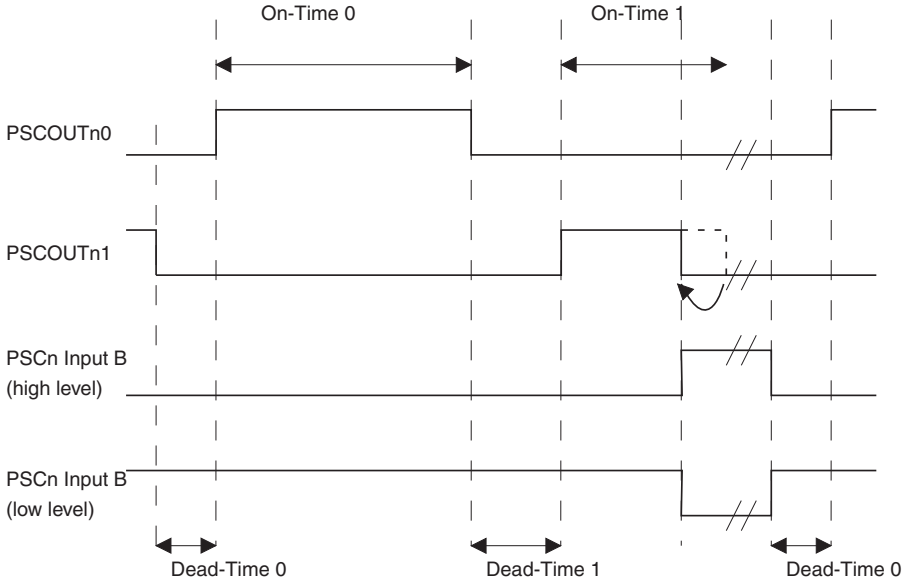
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 12-18. PSCOUTn1 retriggered by PSCn Input B (edge retriggering).



Note: This example is given in “Input Mode 8” in “2 or 4 ramp mode”. See Figure 12-33 on page 126 for details.

Figure 12-19. PSCOUTn1 retriggered by PSCn Input B (level acting).

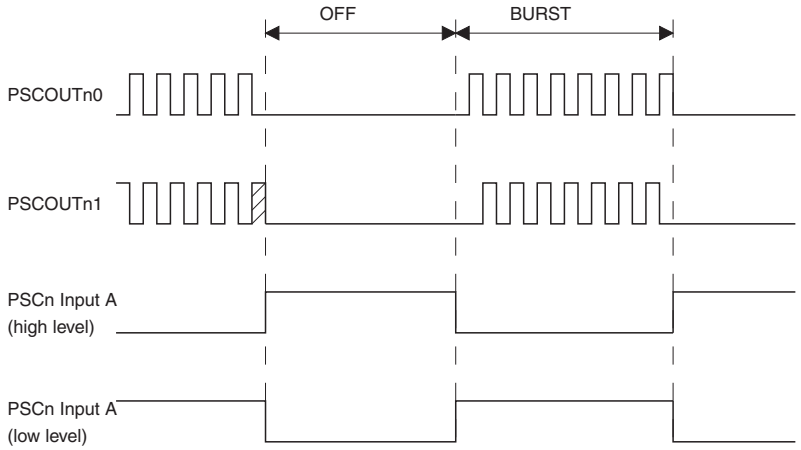


Note: This example is given in “Input Mode 1” in “2 or 4 ramp mode”. See Figure 12-22 on page 121 for details.

12.8.3.1 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 12-26 on page 123 and Figure 12-27 on page 123 for details).

Figure 12-20. Burst generation.



12.8.4 PSC Input Configuration

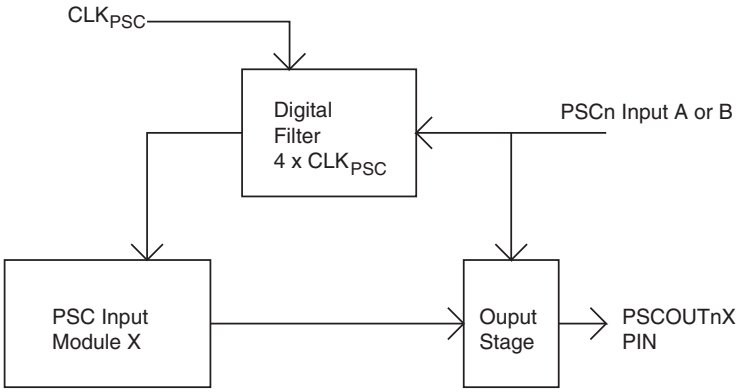
The PSC Input Configuration is done by programming bits in configuration registers.

12.8.4.1 Filter Enable

If the “Filter Enable” bit is set, a digital filter of four 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 deactivate 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.

Figure 12-21. PSC input filtering.



12.8.4.2 Signal Polarity

One can select the active edge (edge modes) or the active level (level modes). See PELEVnx bit description in “PFRCnA - PSC n Input A Control Register” on page 141.

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.

12.8.4.3 *Input Mode Operation*

Thanks to 4 configuration bits (PRFM3:0), it's possible to define the mode of the PSC input. Thanks to four configuration bits (PRFM3:0), it is possible to define all the modes of the PSCR input. These modes are listed in Table 12-7.

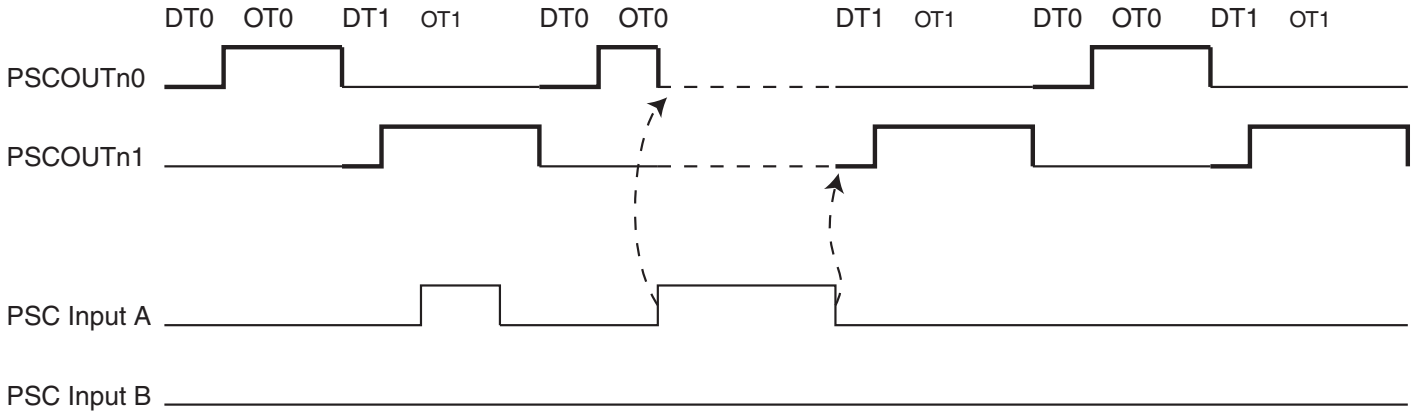
Table 12-7. PSC input mode operation.

	PRFM3:0	Description
0	0000b	PSCn input has no action on PSC output
1	0001b	See "PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait" on page 121.
2	0010b	See "PSC Input Mode 2: Stop signal, Execute Opposite Pulse and Wait" on page 122.
3	0011b	See "PSC Input Mode 3: Stop signal, Execute Opposite Pulse while Fault active" on page 123.
4	0100b	See "PSC Input Mode 4: Deactivate outputs without changing timing" on page 124.
5	0101b	See "PSC Input Mode 5: Stop signal and Insert Dead-Time" on page 124.
6	0110b	See "PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait" on page 125.
7	0111b	See "PSC Input Mode 7: Halt PSC and Wait for Software Action" on page 125.
8	1000b	See "PSC Input Mode 8: Edge Retrigger PSC" on page 126.
9	1001b	See "PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC" on page 127.
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 Deactivate Output" on page 128.
15	1111b	Reserved: Do not use

Note: All following examples are given with rising edge or high level active inputs.

12.9 PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait

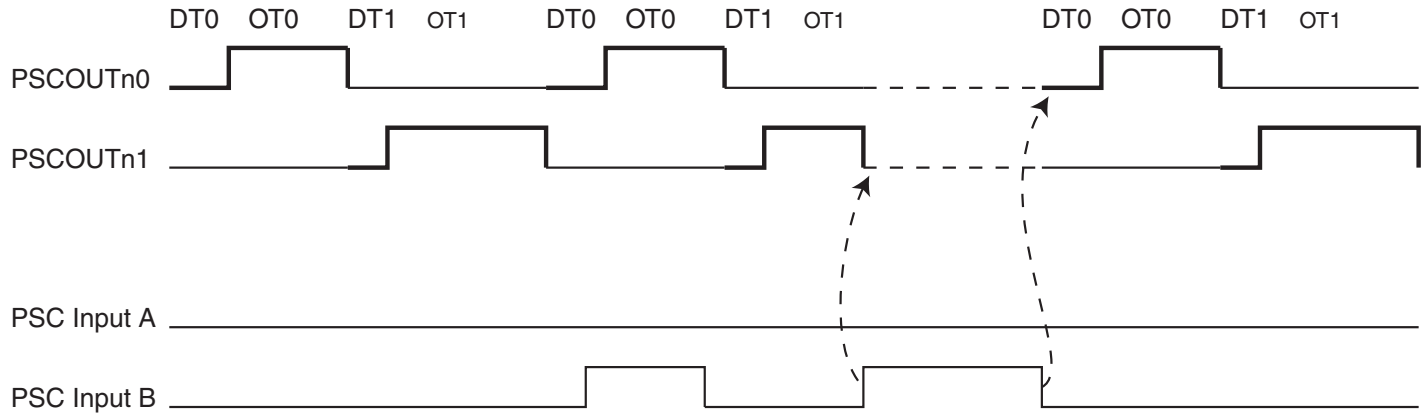
Figure 12-22. PSCn behavior versus PSCn Input A in Fault Mode 1.



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 12-23. PSCn behavior versus PSCn Input B in Fault Mode 1.

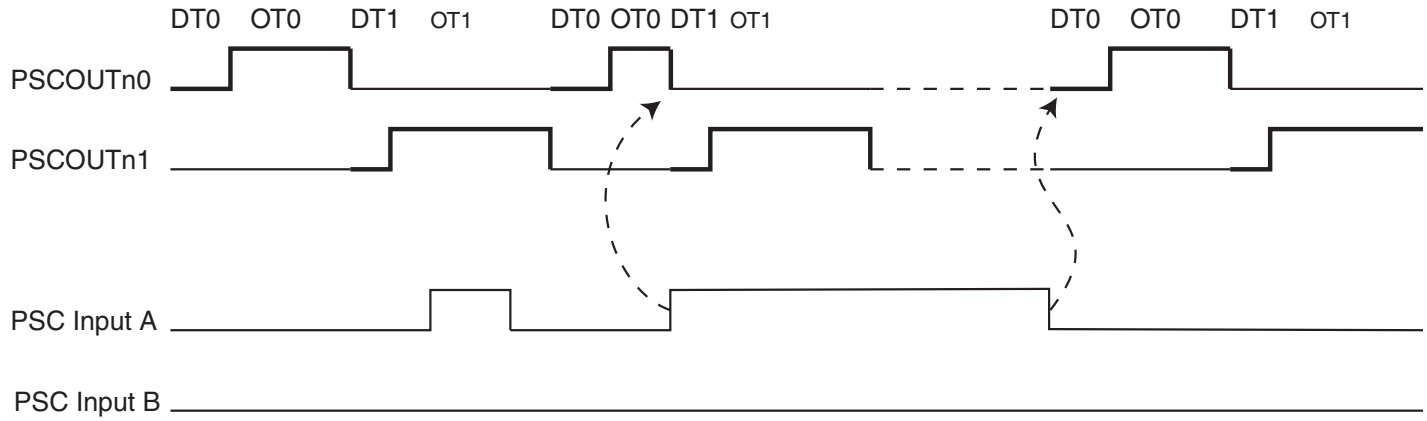


PSC Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSC Input B event occurs, PSC releases PSCOUTn1, waits for PSC Input B inactive state and then jumps and executes DT0 plus OT0.

12.10 PSC Input Mode 2: Stop signal, Execute Opposite Pulse and Wait

Figure 12-24. PSCn behavior versus PSCn Input A in Fault Mode 2.

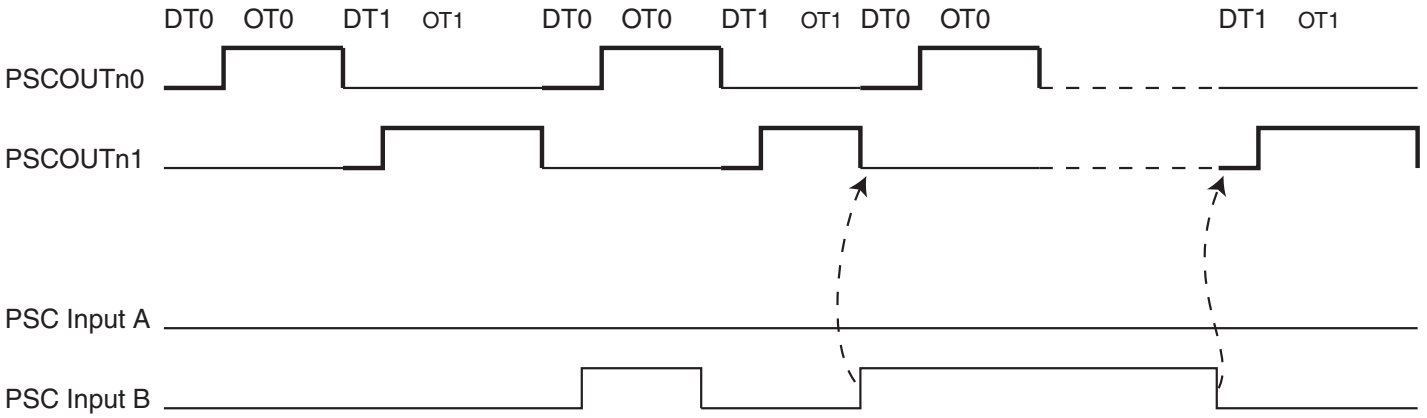


PSC Input A is take into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

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 12-25. PSCn behavior versus PSCn Input B in Fault Mode 2.



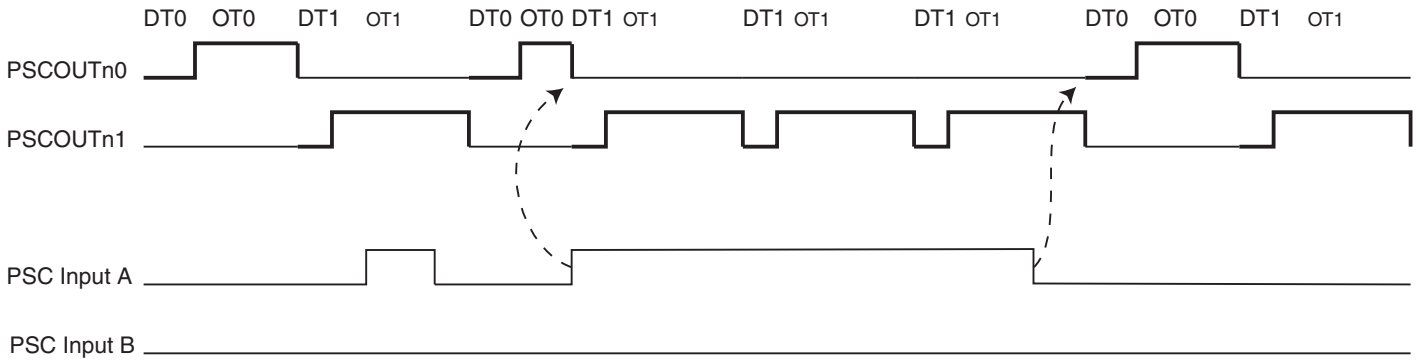
PSC Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

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.

12.11 PSC Input Mode 3: Stop signal, Execute Opposite Pulse while Fault active

Figure 12-26. PSCn behavior versus PSCn Input A in Mode 3.

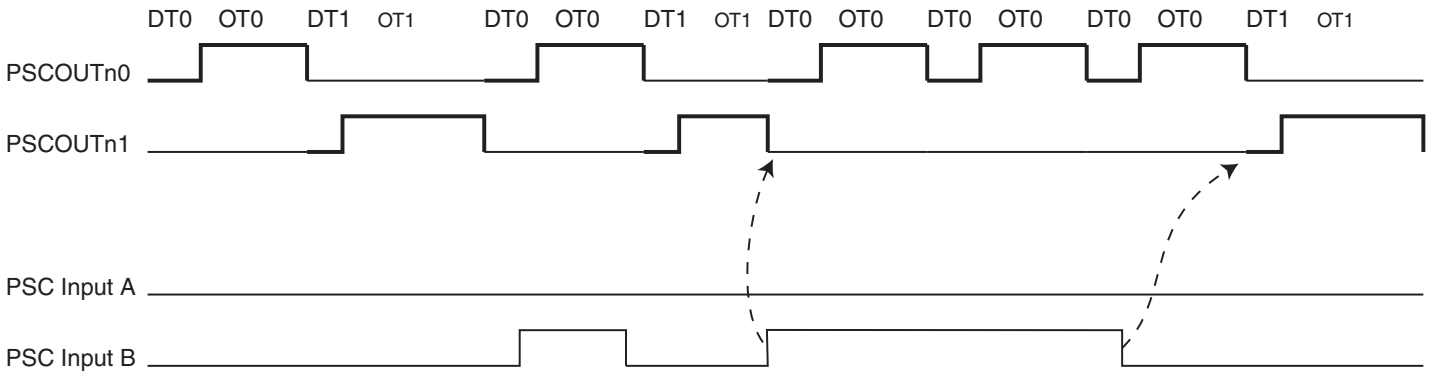


PSC Input A is taken into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

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 12-27. PSCn behavior 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.

12.12 PSC Input Mode 4: Deactivate outputs without changing timing

Figure 12-28. PSC behavior versus PSCn Input A or Input B in Mode 4.

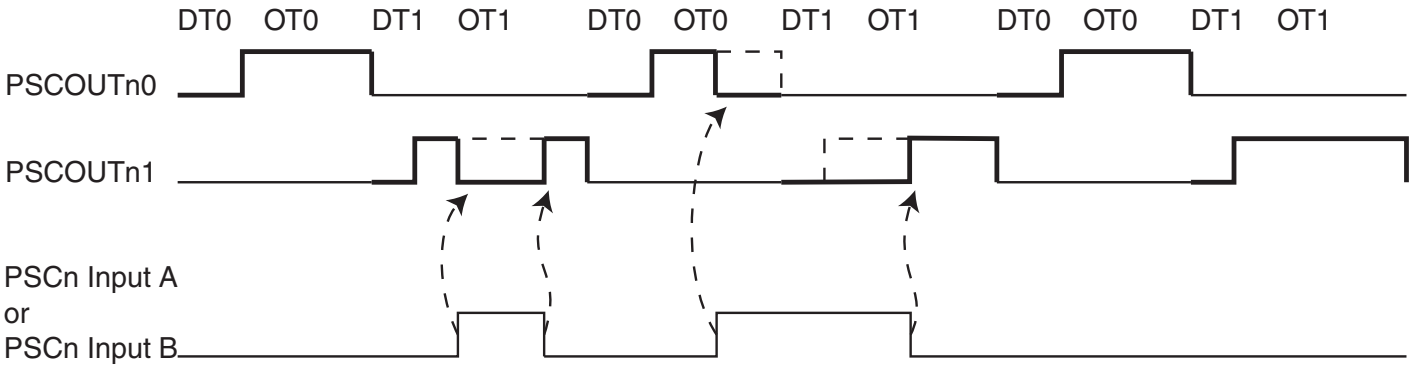
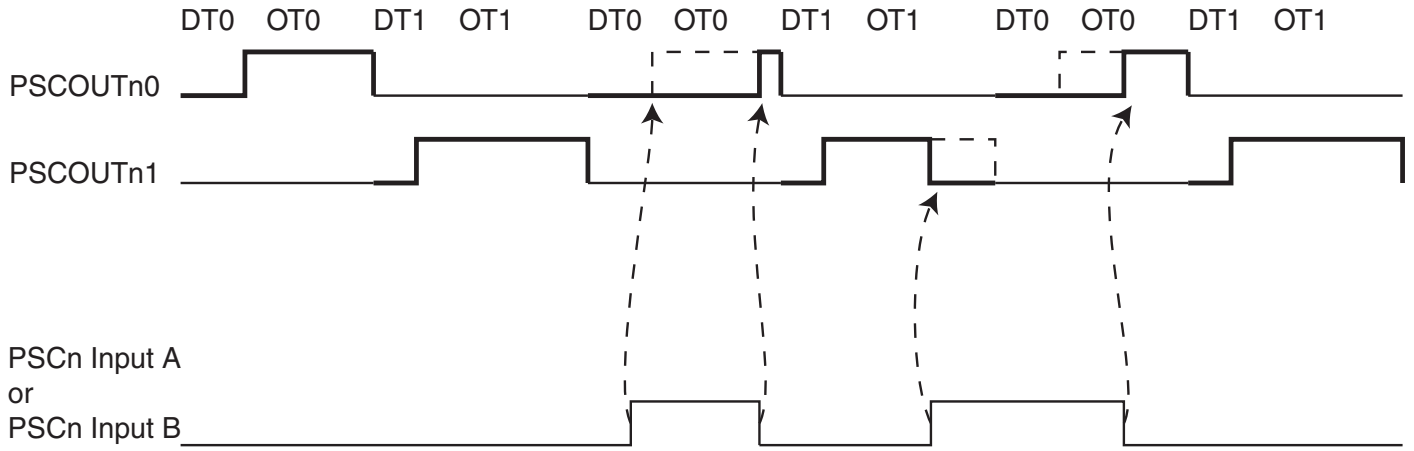


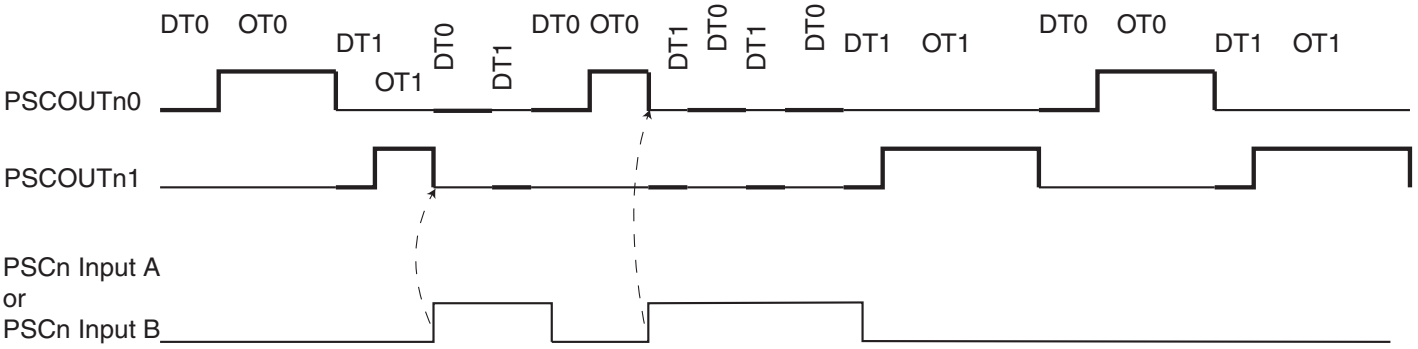
Figure 12-29. PSC behavior versus PSCn Input A or Input B in Fault Mode 4.



PSCn Input A or PSCn Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

12.13 PSC Input Mode 5: Stop signal and Insert Dead-Time

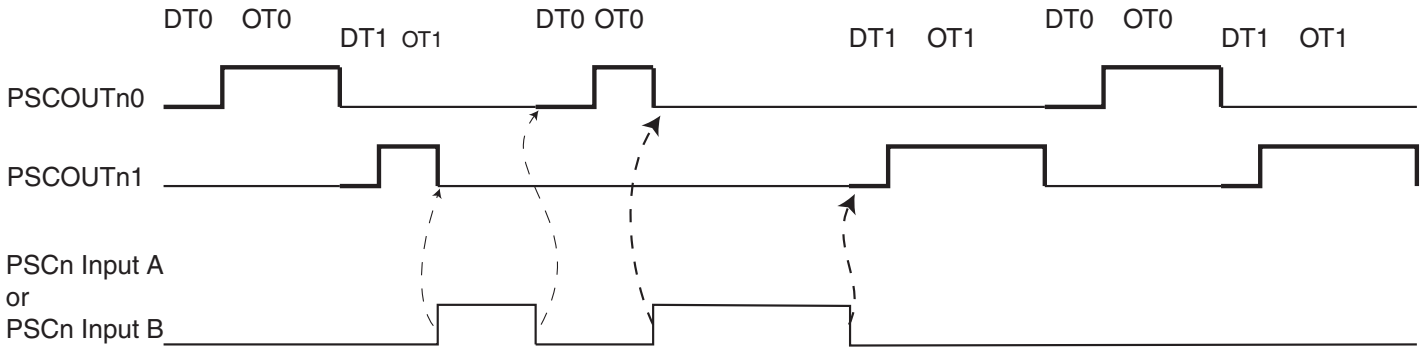
Figure 12-30. PSC behavior versus PSCn Input A in Fault Mode 5.



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.

12.14 PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait

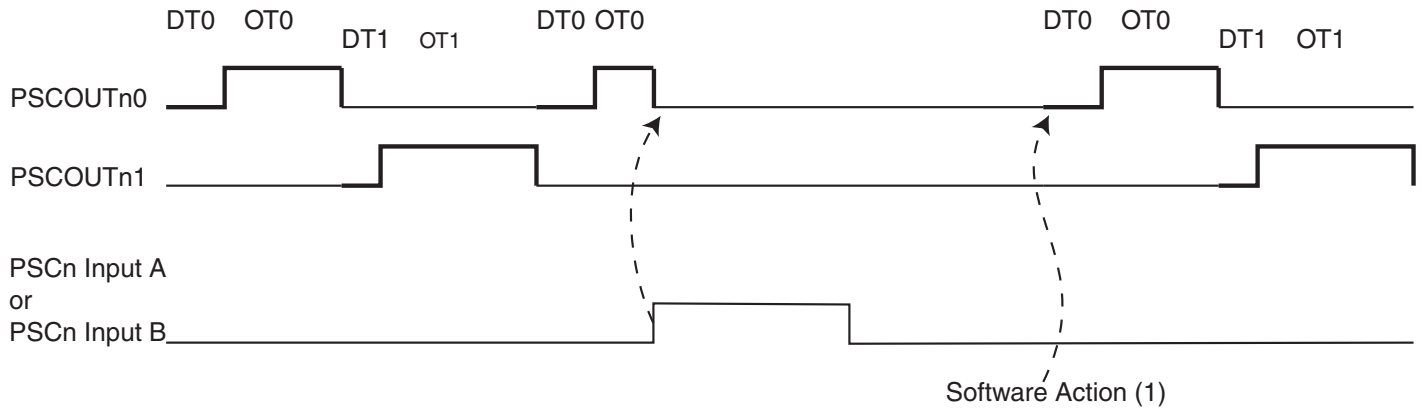
Figure 12-31. PSC behavior versus PSCn Input A in Fault Mode 6.



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.

12.15 PSC Input Mode 7: Halt PSC and Wait for Software Action

Figure 12-32. PSC behavior versus PSCn Input A in Fault Mode 7.

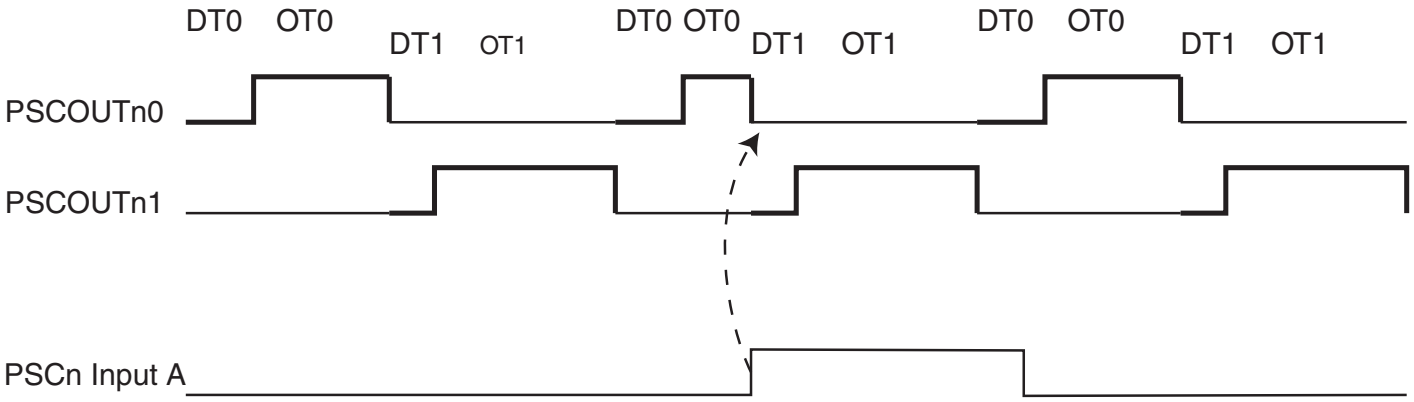


Note: 1. Software action is the setting of the PRUNn bit in PCTLn register.

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.

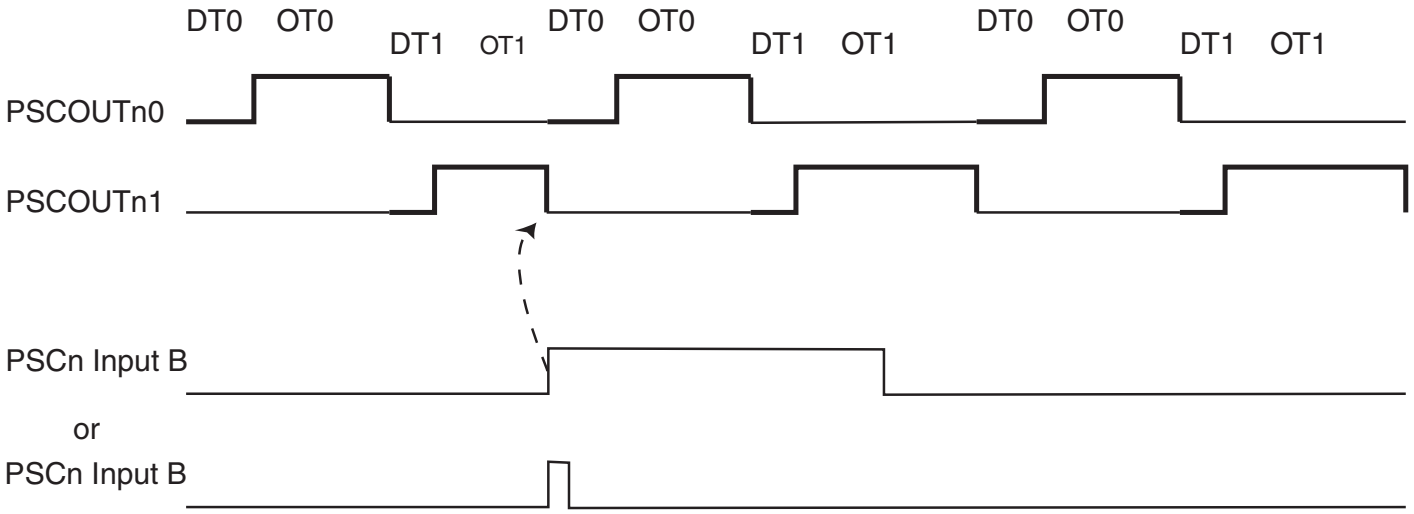
12.16 PSC Input Mode 8: Edge Retrigger PSC

Figure 12-33. PSC behavior versus PSCn Input A in Mode 8.



The output frequency is modulated by the occurrence of significative edge of retriggering input.

Figure 12-34. PSC behavior versus PSCn Input B in Mode 8.



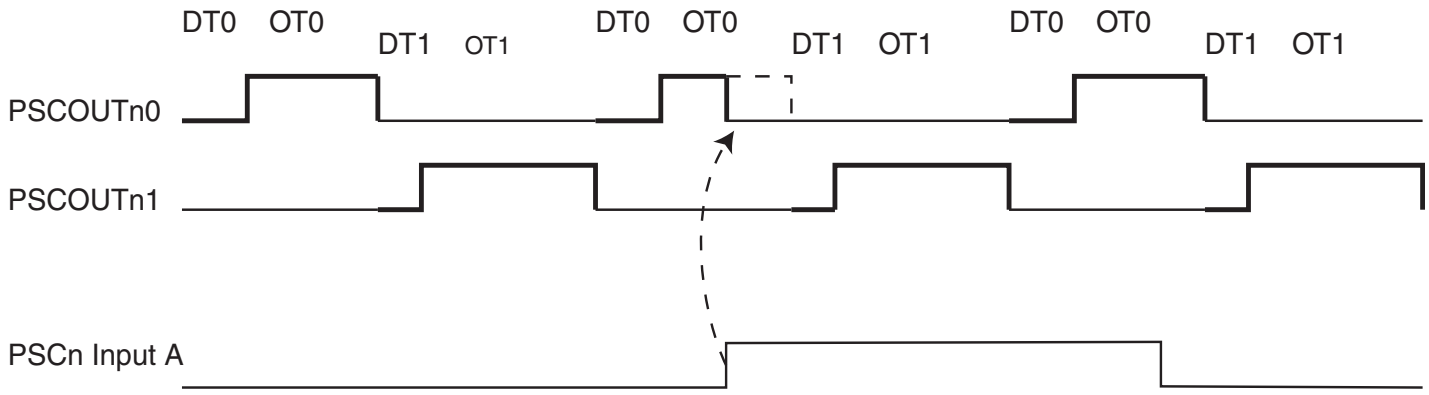
The output frequency is modulated by the occurrence of significative edge of retriggering input.

The retrigger event is taken into account only if it occurs during the corresponding On-Time.

Note: In one ramp mode, the retrigger event on input A resets the whole ramp. So the PSC doesn't jump to the opposite dead-time.

12.17 PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC

Figure 12-35. PSC behavior versus PSCn Input A in Mode 9.

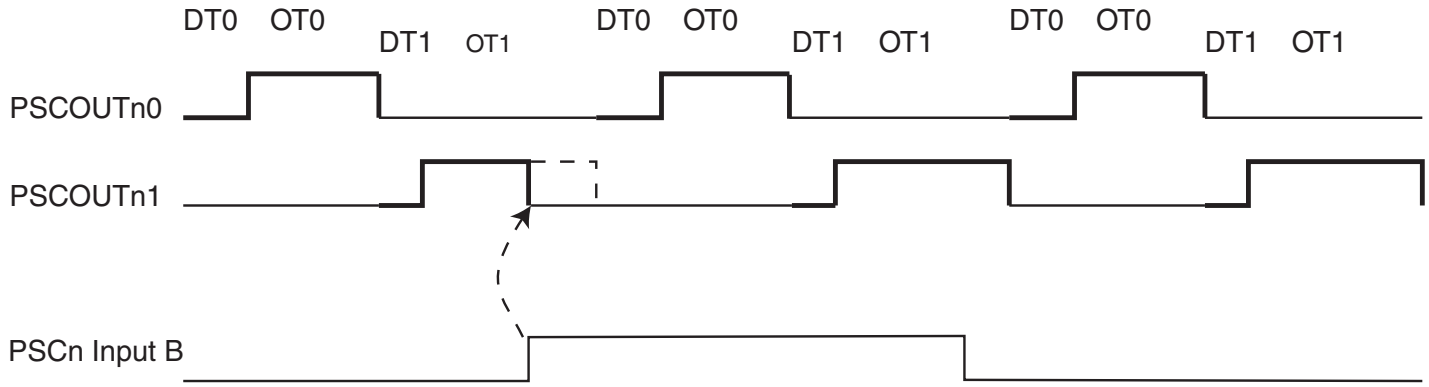


The output frequency is not modified by the occurrence of significative edge of retriggering input.

Only the output is deactivated 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 active. Only the significative edge of Retrigger/Fault input is taken into account.

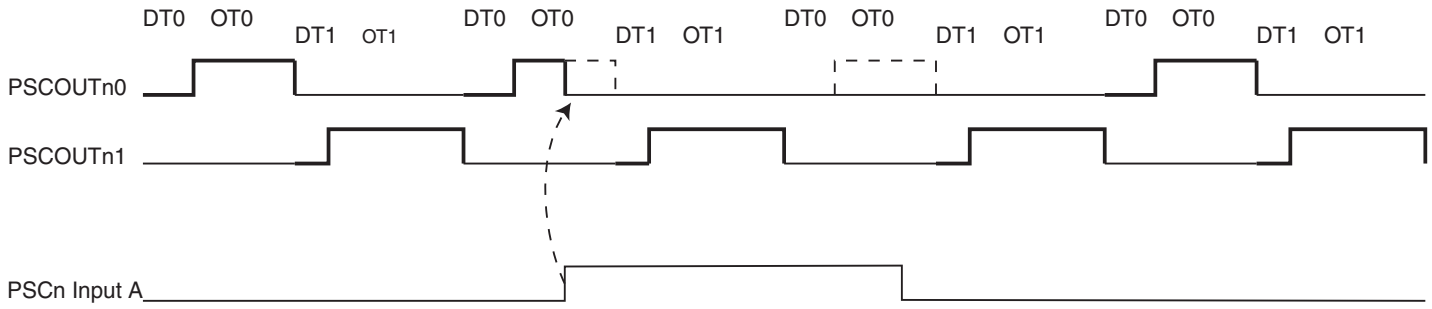
Figure 12-36. PSC behavior versus PSCn Input B in Mode 9.



The retrigger event is taken into account only if it occurs during the corresponding On-Time.

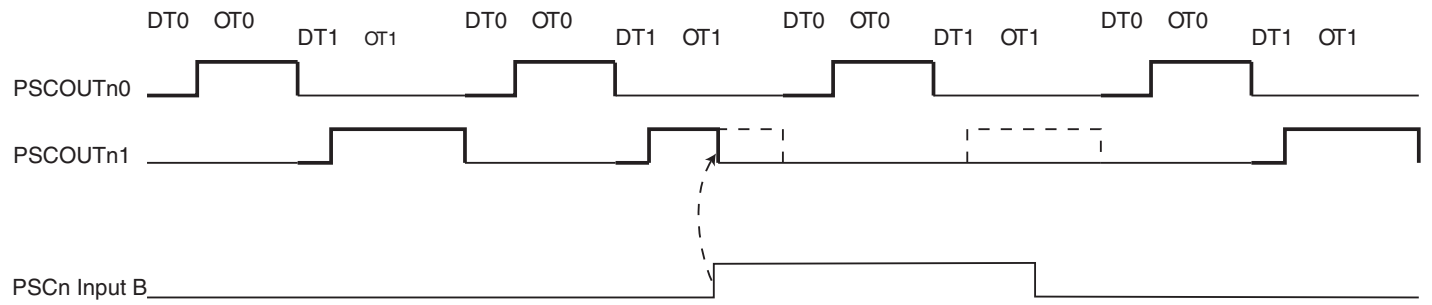
12.18 PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Deactivate Output

Figure 12-37. PSC behavior versus PSCn Input A in Mode 14.



The output frequency is not modified by the occurrence of significative edge of retriggering input.

Figure 12-38. PSC behavior versus PSCn Input B in Mode 14.



The output is deactivated 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 active. The PSC runs at constant frequency.

12.18.1 Available Input Mode according to Running Mode

Some input modes are not consistent with some running modes. Table 12-8 gives the input modes which are valid according to running modes.

Table 12-8. 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

Table 12-8. Available input modes according to running modes. (Continued)

Input mode number:	1 ramp mode	2 ramp mode	4 ramp mode	Centered mode
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			

12.18.2 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 software in PICRnH/L register.

12.18.3 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 PICR1 Register before the next event occurs, the PICR1 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 PICR1 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.

12.19 PSC2 Outputs

12.19.1 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 12-9. 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 12-9. 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.

12.19.2 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.

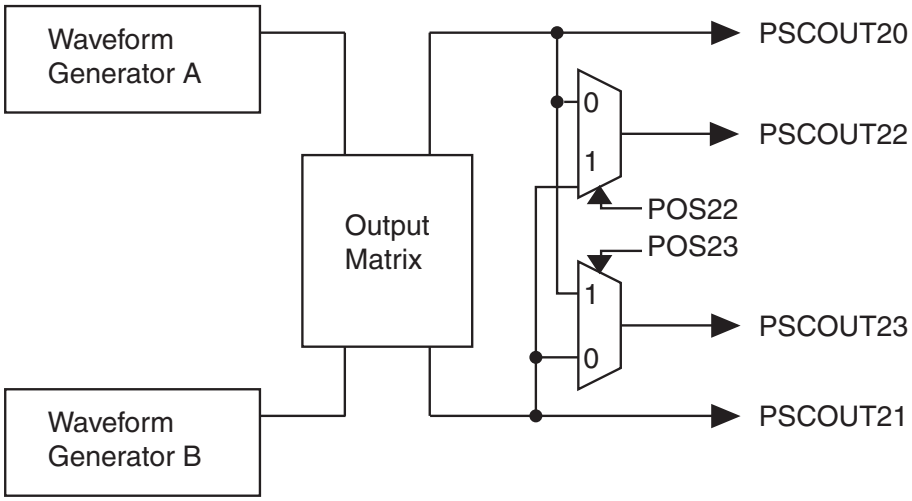
If POS22 bit in PSOC2 register is clear, PSCOUT22 duplicates PSCOUT20.

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 12-39. PSCOUT22 and PSCOUT23 outputs.



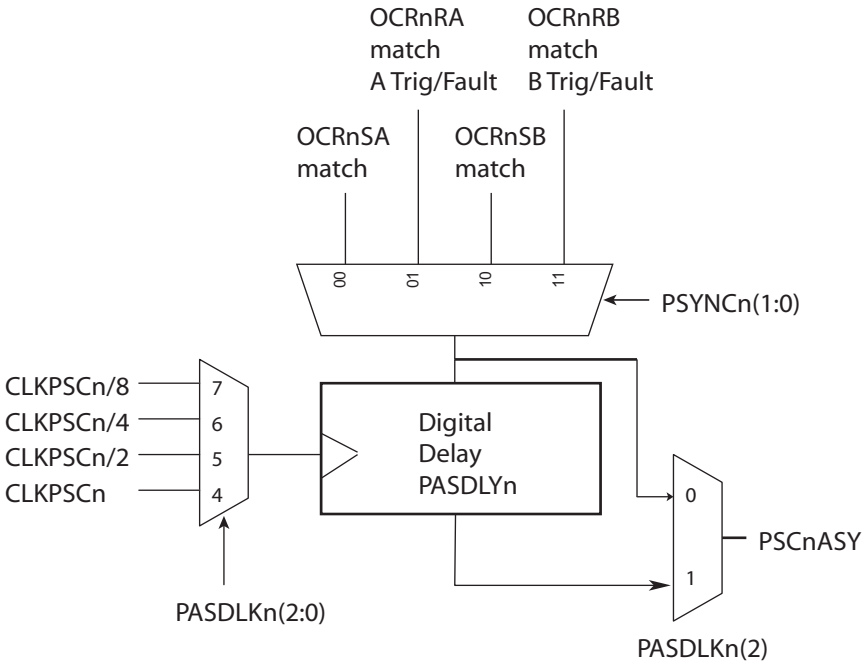
12.20 Analog Synchronization

PSC generates a signal to synchronize the sample and hold or the ADC start; synchronization is mandatory for measurements.

This signal can be selected between all falling or rising edge of PSCn0 or PSCn1 outputs as defined per Table 12-11 on page 134 and Table 12-12 on page 135.

The signal can be shifted by a digital delay defined by the register PASDLY. The shifting clock can be either Clkpsc or Clkpsc/4, as described per Bit 7, 6, 5– PASDLKn(2:0): Analog Synchronization Output Delay or Input Blanking select on page 137.

Figure 12-40. Analog synchronization.



12.21 Interrupt Handling

As each PSC can be dedicated for one function, each PSC has its own interrupt system.

List of interrupt sources:

- Counter reload (end of On Time 1)
- End of Enhanced Cycle
- PSC Input event (active edge or at the beginning of level configured event)
- PSC Mutual Synchronization Error

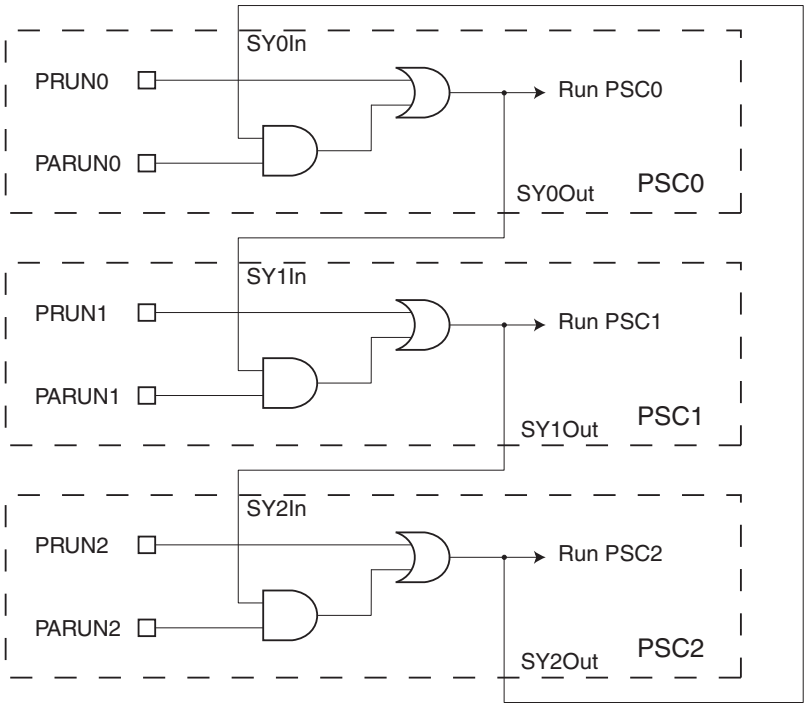
12.22 PSC Synchronization

Note: In AT90PWM81/161, this feature is not relevant and PRUN2, PARUN2 are stuck at zero.

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 12-41. 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 “PCTL2 - PSC 2 Control Register” on page 140.

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).

12.22.1 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” between the three PSC, only the fault event (mode 7) which is able to “stop” the PSC through the PRUN bits is transmitted 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.

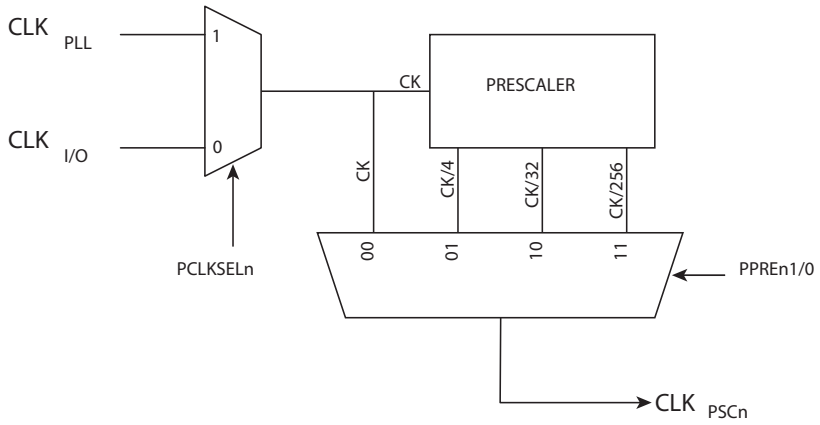
12.23 PSC Clock Sources

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 12-42. 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 12-10. Output Clock versus selection and prescaler.

PCLKSELn	PPREn1	PPREn0	CLKPSCn output
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

12.24 Interrupts

This section describes the specifics of the interrupt handling as performed in AT90PWM81/161.

12.24.1 List of Interrupt Vector

Each PSC provides three interrupt vectors

- **PSCn EC (End of Cycle)**: When enabled and when a match with OCRnRB occurs
- **PSCn EEC (End of Enhanced Cycle)**: When enabled and when a match with OCRnRB occurs at the 15th enhanced cycle
- **PSCn CAPT (Capture Event)**: When enabled and one of the two following events occurs: retrigger, capture of the PSC counter or Synchro Error.

See “PIM2 - PSC2 Interrupt Mask Register” on page 143.

12.25 PSC Register Definition

Registers are explained for PSC0. They are identical for PSC1. For PSC2 only different registers are described.

12.25.1 PSOC2 - PSC 2 Synchro and Output Configuration

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 12-11. 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 12-12. Synchronization source description in centered mode.

PSYNCn1	PSYNCn0	Description
0	0	Send signal on match with OCRnRA (during counting down of PSC) The minimum value of OCRnRA must be 1
0	1	Send signal on match with OCRnRA (during counting up of PSC) The minimum 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.

12.25.2 OCRnSAH and OCRnSAL - Output Compare SA Register

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	

12.25.3 OCRnRAH and OCRnRAL - Output Compare RA Register

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	

12.25.4 OCRnSBH and OCRnSBL - Output Compare SB Register

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	

12.25.5 OCRnRBH and OCRnRBL - Output Compare RB Register

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 16-bit 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.

12.25.6 PCNF2 - PSC 2 Configuration Register

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 12-13. 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 129.

When Output Matrix is used, the PSC n Output Polarity POPn has no action on the outputs.

12.25.7 PCNFE2 - PSC 2 Extended Configuration Register

Bit	7	6	5	4	3	2	1	0	
	PASDLKn2	PASDLKn1	PASDLKn0	PBFMn1	PELEVnA1	PELEVnB1	PISELnA1	PISELnB1	PCNFE2
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 Extended Configuration Register is used to configure the running mode of the PSC.

- **Bit 7, 6, 5– PASDLKn(2:0): Analog Synchronization Output Delay or Input Blanking select**

Defines the modes for Analog signal synchronization delay or Input Blanking.

Table 12-14. Analog signal synchronization or Input Blanking mode selection.

PASDLKn2	PASDLKn1	PASDLKn0	Description
0	0	0	No Analog signal synchronization delay, no Input Blanking
0	0	1	No Analog signal synchronization delay, Input Blanking using PSC clock, started on PSC end of cycle
0	1	0	No Analog signal synchronization delay, Input Blanking using PSC clock, started on OCR SA event
0	1	1	No Analog signal synchronization delay, Input Blanking using PSC clock, started on OCR SB event
1	0	0	Analog signal synchronization delay with PSC clock, no Input Blanking
1	0	1	Analog signal synchronization delay with PSC clock /2, no Input Blanking
1	1	0	Analog signal synchronization delay with PSC clock /4, no Input Blanking
1	1	1	Analog signal synchronization delay with PSC clock /8, no Input Blanking

- **Bit 4- PBFMn1: Balance Flank Width Modulation, bit 1**

Defines the Flank Width Modulation, together with PBFMn0 bit in PCTLn register.

Table 12-15. Flank Width Mode selection.

PBFMn1	PBFMn0	Description
0	0	Flank Width Modulation operates on RB (On-Time 1 only)
0	1	Flank Width Modulation operates on RB + RA (On-Time 0 and On-Time 1)
1	0	Flank Width Modulation operates on SB (Dead-Time 1 only) (1)
1	1	Flank Width Modulation operates on SB +SA (Dead-Time 0 and Dead-Time 1)

Note: 1. In one ramp mode, changing SA or SA+BSB also affect On-Time; see Figure 12-8 on page 108.

- **Bit 3– PELEVnA1: PSC n Input Select for part A**

Together with PELEVnA0, defines active edge or level on PSC part A.

Table 12-16. PSC edge & level input selection.

PELEVnA1	PELEVnA0	Description
0	0	The falling edge or low level of selected input generates the significative event for retrigger or fault function
0	1	The rising edge or high level of selected input generates the significative event for retrigger or fault function
1	0	The toggle of selected input generates the significative event for retrigger or fault function
1	1	Reserved

- **Bit 2– PELEVnB1: PSC n Input Select for part B**

Together with PELEVnB0, defines active edge or level on PSC part B.

Table 12-17. PSC edge & level input selection.

PELEVnB1	PELEVnB0	Description
0	0	The falling edge or low level of selected input generates the significative event for retrigger or fault function
0	1	The rising edge or high level of selected input generates the significative event for retrigger or fault function
1	0	The toggle of selected input generates the significative event for retrigger or fault function
1	1	Reserved

- **Bit 1– PISELnA1: PSC n Input Select for part A**

Together with PISELnA0, defines active signal on PSC part A.

Table 12-18. PSC trigger & fault input selection.

PISELnA1	PISELnA0	Description
0	0	PSCINn
0	1	First analog comparator output
1	0	PSCINnA
1	1	Second analog comparator output

- **Bit 0– PISELnB1: PSC n Input Select for part B**

Together with PISELnB0, defines active signal on PSC part B.

Table 12-19. PSC trigger & fault input selection.

PISELnB1	PISELnB0	Description
0	0	PSCINn
0	1	First analog comparator output
1	0	PSCINnA
1	1	Second analog comparator output

12.25.8 PASDLYn - Analog Synchronization Delay Register

Bit	7	6	5	4	3	2	1	0	
	PASDLYn[7:0]								PASDLYn
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The Analog Synchronization Delay Register store an 8 bit delay used:

- For the input signal blanking. See Section “PSC Inputs”, page 114
- For shifting the PSCOUTnx edges and the PSCnASY signal. See Section “Analog Synchronization”, page 131

See also the bit definition Section “Bit 7, 6, 5– PASDLKn(2:0): Analog Synchronization Output Delay or Input Blanking select”, page 137 and Section “Bit 5:4 – PSYNCn1:0: Synchronization Out for ADC Selection”, page 134.

12.25.9 PCTL2 - PSC 2 Control Register

Bit	7	6	5	4	3	2	1	0	
	PPRE21	PPRE20	PBFM20	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 – PPREn1:0 : PSC n Prescaler Select**

This two bits select the PSC input clock division factor. All generated waveform will be modified by this factor.

Table 12-20. PSC n Prescaler selection.

PPREn1	PPREn0	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 – PBFMn0: Balance Flank Width Modulation bit 0**

Defines the Flank Width Modulation, together with PBFMn1 bit in PCNFEn register. See Table 12-15 on page 138.

- **Bit 4 – PAOCnB: PSC n Asynchronous Output Control B**

When this bit is set, Fault input selected to block B can act directly to PSCOUTn1 and PSCOUT23 outputs. See Section “PSC Clock Sources”, page 133.

- **Bit 3 – PAOCnA: PSC n Asynchronous Output Control A**

When this bit is set, Fault input selected to block A can act directly to PSCOUTn0 and PSCOUT22 outputs. See Section “PSC Clock Sources”, page 133.

- **Bit 2 – PARUNn: PSC n Autorun**

When this bit is set, the PSC n starts with PSCn-1. That means that PSC n starts:

- when PRUNn bit in PCTLn register is set,
- or when PARUNn bit in PCTLn is set and PRUNn-1 bit in PCTLn-1 register is set (or PARUN0 bit and PRUN0)

- **Bit 1 – PCCYCn: PSC n Complete Cycle**

When this bit is set, the PSC n completes the entire waveform cycle before halt operation requested by clearing PRUNn. This bit is not relevant in slave mode (PARUNn = 1).

- **Bit 0 – PRUNn: PSC n Run**

Writing this bit to one starts the PSC n.

When set, this bit prevails over PARUNn bit.

12.25.10 PFRCnA - PSC n Input A Control Register

Bit	7	6	5	4	3	2	1	0	
	PCAEnA	PISELnA0	PELEVnA0	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	

12.25.11 PFRCnB - PSC n Input B Control Register

Bit	7	6	5	4	3	2	1	0	
	PCAEnB	PISELnB0	PELEVnB0	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 PISELnx1:0 bit in the same register).

- **Bit 6 – PISELnx0: PSC n Input Select for Part x**

Together with PISELnx1 in PCNFEn register, defines active signal on PSC module A. See Table 12-18 on page 139 and Table 12-19 on page 139.

- **Bit 5 –PELEVnx0: PSC n Edge Level Selector of Input Part x**

Together with PELEVnx1 n PCNFEn register, defines active edge & level on PSC part x; See Table 12-16 on page 138 and Table 12-17 on page 139.

- **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 Table 12-7 on page 120 for more explanations).

Table 12-21. 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", page 121
0010b	"PSC Input Mode 2: Stop signal, Execute Opposite Pulse and Wait", page 122
0011b	"PSC Input Mode 3: Stop signal, Execute Opposite Pulse while Fault active", page 123

Table 12-21. Level sensitivity and Fault Mode operation. (Continued)

PRFMnx3:0	Description
0100b	"PSC Input Mode 4: Deactivate outputs without changing timing", page 124
0101b	"PSC Input Mode 5: Stop signal and Insert Dead-Time", page 124
0110b	"PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait", page 125
0111b	"PSC Input Mode 7: Halt PSC and Wait for Software Action", page 125
1000b	"PSC Input Mode 8: Edge Retrigger PSC", page 126
1001b	"PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC", page 127
1010b	Reserved (do not use)
1011b	
1100b	
1101b	
1110b	"PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Deactivate Output", page 128
1111b	Reserved (do not use)

12.25.12 PICR2H and PICR2L - PSC 2 Input Capture Register

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.

12.26 PSC2 Specific Register

12.26.1 POM2 - PSC 2 Output Matrix

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.

12.26.2 PIM2 - PSC2 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	-	-	PSEIE2	PEVE2B	PEVE2A	-	PEOEPE2	PEOPE2	PIM2
Read/Write	R	R	R/W	R/W	R/W	R	R/W	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 1– PEOEPEn: PSC n End Of Enhanced Cycle Interrupt Enable**

When this bit is set, an interrupt is generated when PSC reaches the end of the 15th PSC cycle. This allows to update the PSC values in the interrupt routine and to start a new enhanced cycle with the new values at the next PSC cycle end.

- **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.

12.26.3 PIFR2 - PSC2 Interrupt Flag Register

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**

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 frozen external input signal.

- **Bit 6 – POACnA: PSC n Output A Activity**

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 frozen 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 occurred.

Table 12-22. PSC n ramp number description.

PRNn1	PRNn0	Description
0	0	The last event which has generated an interrupt occurred during ramp 1
0	1	The last event which has generated an interrupt occurred during ramp 2
1	0	The last event which has generated an interrupt occurred during ramp 3
1	1	The last event which has generated an interrupt occurred 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.

12.26.4 PSC Output Behavior During Reset

For external component safety reason, the state of PSC outputs during Reset can be programmed by fuses PSCRV, PSCRRB & PSC2RB.

These fuses are located in the Extended Fuse Byte:

Table 12-23. Extended Low Fuse byte.

Extended fuse byte	Bit No	Description	Default value
PSC2RB	7	PSC2 reset behavior	1
PSC2RBA	6	PSC2 reset behavior for OUT22 & 23	1
PSCRRB	5	PSC reduced reset behavior	1
PSCRV	4	PSCOUT & PSCOUTR reset value	1
PSCINRB	3	PSC & PSCR inputs reset behavior	1
BODLEVEL2 (1)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1 (1)	1	Brown-out detector trigger level	0 (programmed)
BODLEVEL0 (1)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes: 1. See Table 7-2 on page 53 for BODLEVEL Fuse decoding.

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 PSCRRB fuse equals 1 (unprogrammed), PSCOUTR0 & PSCOUTR1 keep a standard port behavior. If PSC0RB fuse equals 0 (programmed), PSCOUTR0 & PSCOUTR1 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUTR0 & PSCOUTR1 keep the forced state until PSOC0 register is written.

If PSC2RB fuse equals 1 (unprogrammed), PSCOUT20 & PSCOUT21 keep a standard port behavior. If PSC2RB fuse equals 0 (programmed), PSCOUT20 & PSCOUT21 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT20 & PSCOUT21 keep the forced state until PSOC2 register is written.

If PSC2RBA fuse equals 1 (unprogrammed), PSCOUT22 & PSCOUT23 keep a standard port behavior. If PSC2RBA fuse equals 0 (programmed), PSCOUT22 & PSCOUT23 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT22 & PSCOUT23 keep the forced state until PSOC2 register is written.

12.26.5 PSC Input Behavior During Reset

For power consumption under reset reason, the state of PSC & PSCR inputs during Reset can be programmed by fuse PSCINRB.

If PSCINRB fuse equals 1 (unprogrammed), PSC & PSCR input keep a standard port behavior. If PSCINRB fuse equals 0 (programmed), PSC & PSCR input pull-up are forced while the reset is active. Affected pins are PSCIN2, PSCINr, PSCIN2A, PSCINrA. To prevent any conflict on PD1, this fuse has no effect on PSCINrB.

13. Reduced Power Stage Controller – (PSCR)

The Reduced Power Stage Controller is a high performance waveform controller.

13.1 Features

- **PWM waveform generation function (two complementary programmable outputs)**
- **Dead time control**
- **Standard mode up to 12-bit resolution**
- **Enhanced resolution up to 16 bits**
- **Frequency up to 64Mhz**
- **Conditional waveform on external events (zero crossing, current sensing ...)**
- **ADC synchronization**
- **Overload protection function**
- **Abnormality protection function, emergency input to force all outputs to high impedance or in inactive state (fuse configurable)**
- **Fast emergency stop by hardware**

13.2 Overview

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

- A lower case “r” (or “n” replaces the PSC number, in this case 0. However, when using the register or bit defines in a program, the precise form must be used, that is, PSOC0 for accessing PSCR 0 Synchro and Output Configuration register and so on
- A lower case “x” replaces the PSCR part , in this case A or B. However, when using the register or bit defines in a program, the precise form must be used, that is, PFRC0A for accessing PSCR 0 Fault/Retrigger 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.

These outputs can be used in various ways:

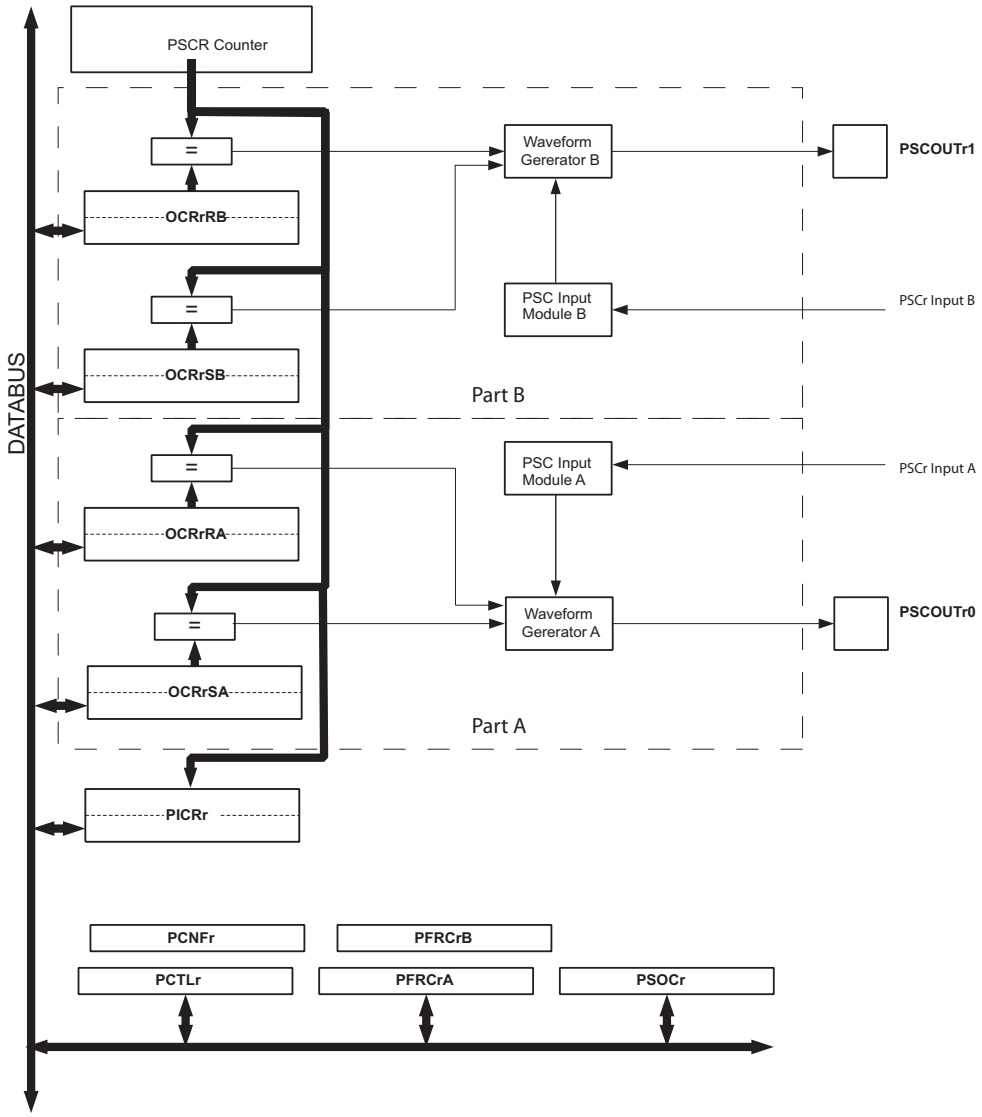
- “Two Outputs” to drive a half bridge (for example, Lighting applications)
- “One Output” to drive single power transistor (for example, DC/DC converter, PFC applications)

The PSCR 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

13.3 PSCR Description

Figure 13-1. Power Stage Controller block diagram.



The principle of the PSCR is based on the use of a counter (PSCR 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 PSCR is seen as two symmetrical entities. One part named part A which generates the output PSCOUTr0 and the second one named part B which generates the PSCOUTr1 output.

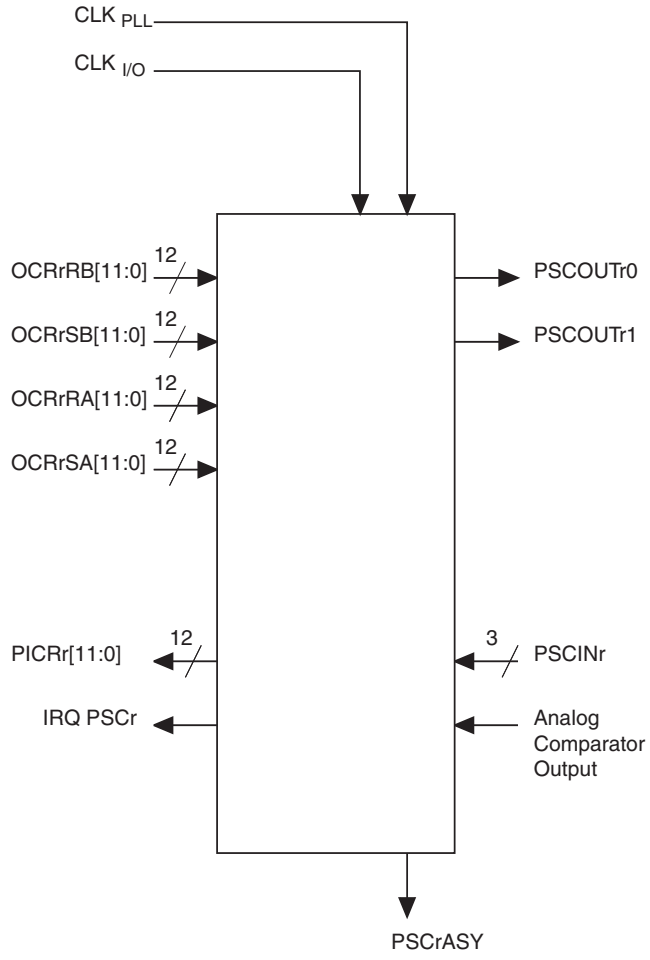
Each part A or B has its own PSCR Input Module to manage selected input.

13.3.1 Output Polarity

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

13.4 Signal Description

Figure 13-2. PSCR external block view.



13.4.1 Input Description

Table 13-1. Internal inputs.

Name	Description	Type width
OCRrRB[11:0]	Compare value which reset signal on Part B (PSCOUTr1)	Register 12 bits
OCRrSB[11:0]	Compare value which set signal on Part B (PSCOUTr1)	Register 12 bits
OCRrRA[11:0]	Compare value which reset signal on Part A (PSCOUTr0)	Register 12 bits
OCRrSA[11:0]	Compare value which set signal on Part A (PSCOUTr0)	Register 12 bits
CLK I/O	Clock input from I/O clock	Signal
CLK PLL	Clock input from PLL	Signal

Table 13-2. Block inputs.

Name	Description	Type width
PSCINr	Input 0 used for retrigger or fault functions	Signal
From analog comparator	Input 1 used for retrigger or fault functions	Signal
PSCINrA	Input 2 used for retrigger or fault functions	Signal
PSCINrB	Input 3 used for retrigger or fault functions	Signal

13.4.2 Output Description

Table 13-3. Block outputs.

Name	Description	Type width
PSCOUTr0	PSCR Output 0 (from part A of PSC)	Signal
PSCOUTr1	PSCR Output 1 (from part B of PSC)	Signal

Table 13-4. Internal Outputs.

Name	Description	Type width
PICRr[11:0]	PSCR Input Capture Register Counter value at retriggering event	Register 12 bits
IRQPSCr	PSCR Interrupt Request: three sources, overflow, fault, and input capture	Signal
PSCrASY	ADC synchronization (+ amplifier synchro.) (2)	Signal

2. See “Analog Synchronization” on page 169.

13.5 Functional Description

13.5.1 Waveform Cycles

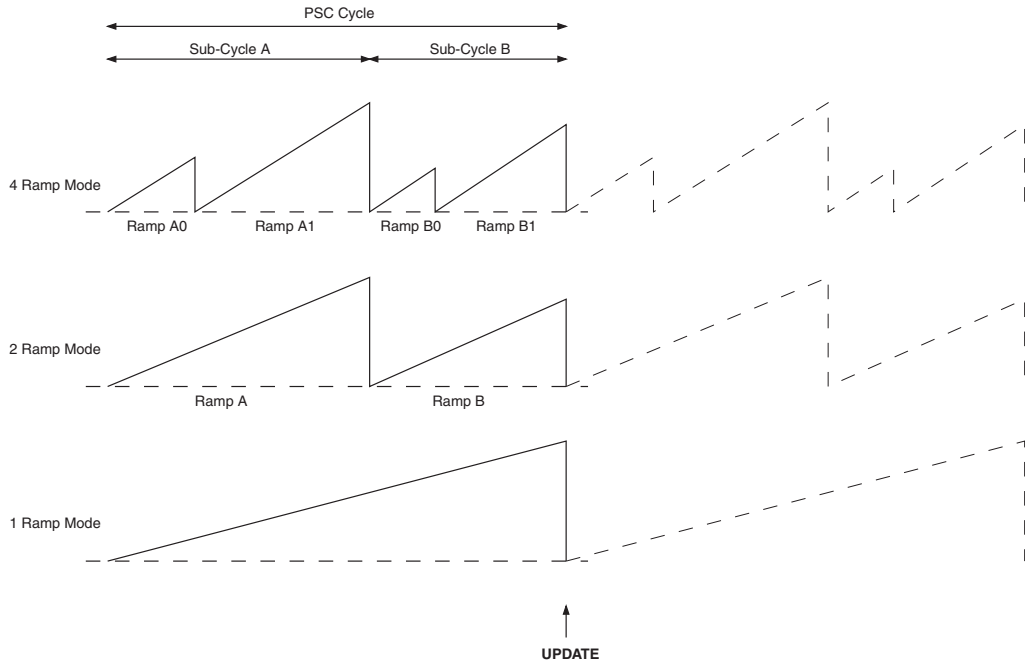
The waveform generated by PSCR can be described as a sequence of two waveforms.

The first waveform is relative to PSCOUTr0 output and part A of PSC. The part of this waveform is sub-cycle A in Figure 13-3.

The second waveform is relative to PSCOUTr1 output and part B of PSC. The part of this waveform is sub-cycle B in Figure 13-3.

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

Figure 13-3. Cycle presentation in 1, 2, and 4 Ramp mode.



Ramps illustrate the output of the PSCR 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.

13.5.2 Running Mode Description

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

- Four Ramp mode
- Two Ramp mode
- One Ramp mode

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 adjust the dead time between PSCOUTn0 and PSCOUTn1 active signals.

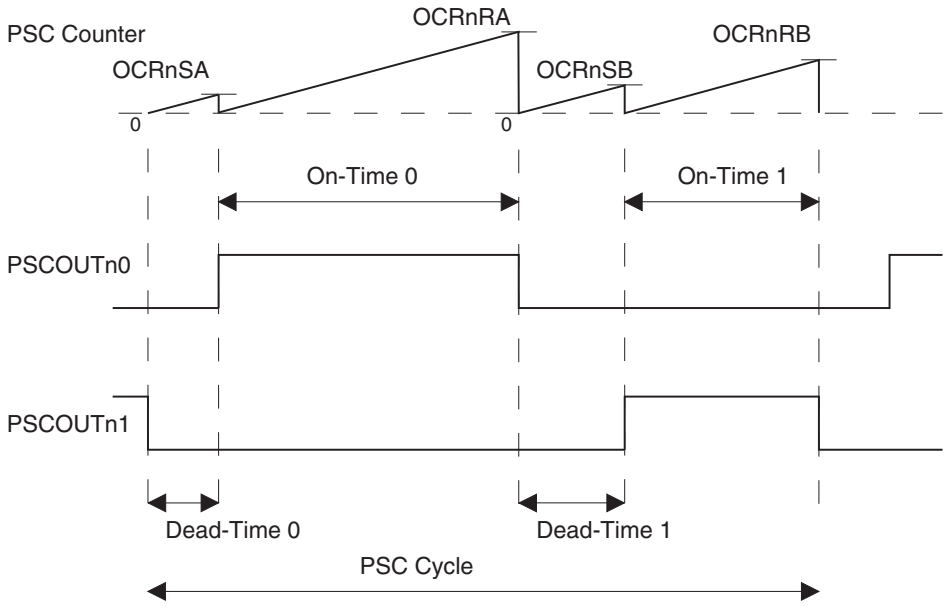
The waveform frequency is defined by the following equation:

$$f_{PSCn} = \frac{1}{PSCnCycle} = \frac{f_{CLK\_PSCn}}{(OT0 + OT1 + DT0 + DT1)}$$

13.5.2.1 *Four Ramp Mode*

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

Figure 13-4. PSCr0 & PSCr1 basic waveforms in Four Ramp mode.



The input clock of PSCR is given by CLKPSC.

PSCOUTr0 and PSCOUTr1 signals are defined by On-Time 0, Dead-Time 0, On-Time 1 and Dead-Time 1 values with:

On-Time 0 = OCRrRAH/L × 1/Fclkpsc

On-Time 1 = OCRrRBH/L × 1/Fclkpsc

Dead-Time 0 = (OCRrSAH/L + 2) × 1/Fclkpsc

Dead-Time 1 = (OCRrSBH/L + 2) × 1/Fclkpsc

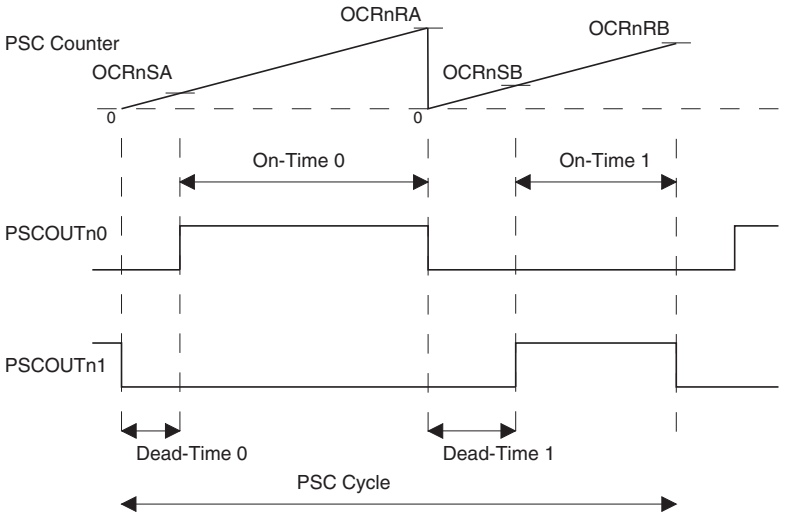
Note: Minimal value for Dead-Time 0 and Dead-Time 1 = 2 × 1/Fclkpsc.

13.5.2.2 *Two Ramp Mode*

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

- One moment for PSCr0 description with OT0 which gives the time of the whole moment
- One moment for PSCr1 description with OT1 which gives the time of the whole moment

Figure 13-5. PSCr0 & PSCr1 basic waveforms in Two Ramp mode.



PSCOUTr0 and PSCOUTr1 signals are defined by On-Time 0, Dead-Time 0, On-Time 1 and Dead-Time 1 values with:

On-Time 0 = (OCRrRAH/L - OCRrSAH/L) × 1/Fclkpsc

On-Time 1 = (OCRrRBH/L - OCRrSBH/L) × 1/Fclkpsc

Dead-Time 0 = (OCRrSAH/L + 1) × 1/Fclkpsc

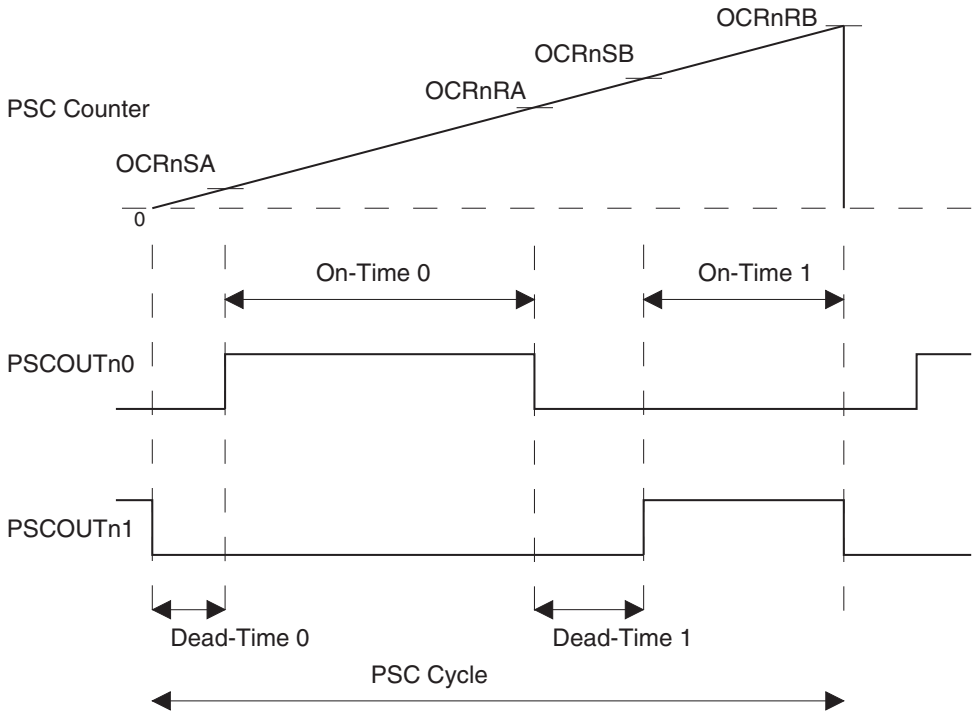
Dead-Time 1 = (OCRrSBH/L + 1) × 1/Fclkpsc

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

13.5.2.3 *One Ramp Mode*

In One Ramp mode, PSCOUTr0 and PSCOUTr1 outputs can overlap each other.

Figure 13-6. PSCr0 & PSCr1 basic waveforms in One Ramp mode.



On-Time 0 = (OCRrRAH/L - OCRrSAH/L) × 1/Fclkpsc

On-Time 1 = (OCRrRBH/L - OCRrSBH/L) × 1/Fclkpsc

Dead-Time 0 = (OCRrSAH/L + 1) × 1/Fclkpsc

Dead-Time 1 = (OCRrSBH/L - OCRrRAH/L) × 1/Fclkpsc

Note: Minimal value for Dead-Time 0 = 1/Fclkpsc.

13.5.3 Fifty Percent Waveform Configuration

When PSCOUTr0 and PSCOUTr1 have the same characteristics, it's possible to configure the PSCR in a Fifty Percent mode. When the PSCR is in this configuration, it duplicates the OCRrSBH/L and OCRrRBH/L registers in OCRrSAH/L and OCRrRAH/L registers. So it is not necessary to program OCRrSAH/L and OCRrRAH/L registers.

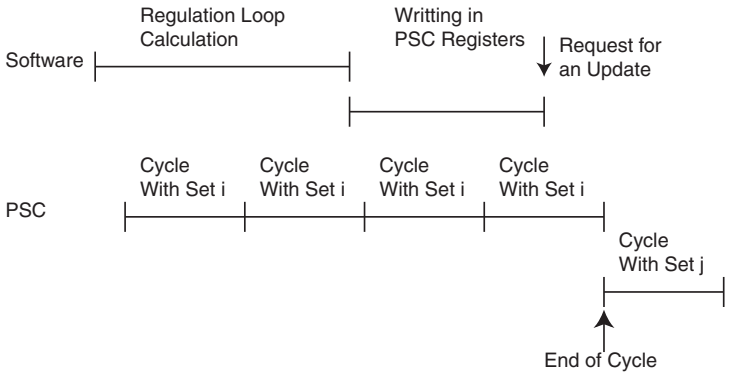
13.6 Update of Values

The update of PSCR waveform registers are done in the following way:

- Immediately when the PSC is stopped
- At the PSC end of cycle when the PSC is running
- At the PSC end of cycle following the required condition when LOCK or AUTOLOCK modes are used

To avoid asynchronous 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 software and the update is initiated by software.

Figure 13-7. Update at the end of complete PSCR cycle.



The software can stop the cycle before the end to update the values and restart a new PSCR cycle.

13.6.1 Value Update Synchronization

New timing values or PSCR output configuration can be written during the PSCR cycle. Thanks to LOCK and AUTOLOCK configuration bits, the new whole set of values can be taken into account with the following conditions:

- When AUTOLOCK configuration is selected, the update of the PSCR internal registers will be done at the end of the PSCR cycle following a write in the Output Compare Register RB. The AUTOLOCK configuration bit is taken into account at the end of the first PSCR cycle
- When LOCK configuration bit is set, there is no update. The update of the PSCR internal registers will be done at the end of the PSCR cycle if the LOCK bit is released to zero

The registers which update is synchronized thanks to LOCK and AUTOLOCK are OCRrSAH/L, OCRrRAH/L, OCRrSBH/L, OCRrRBH/L and PSOCr. PISELrA1 and PISELrB1 bits of PSOCr are immediatly updated in order to behave as PISELrA0 and PISELrB0.

See these register's description starting on page 172.

When set, AUTOLOCK configuration bit prevails over LOCK configuration bit.

See "PCNF0 - PSCR Configuration Register" on page 173.

13.7 Enhanced resolution

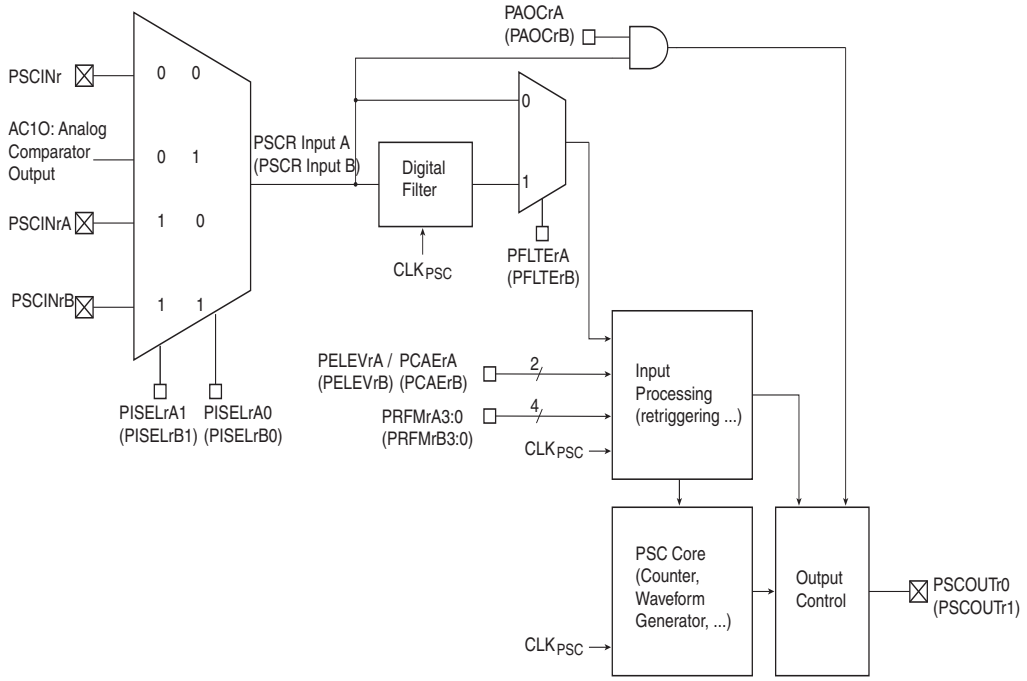
The PSCR includes the same resolution enhancement as in PSC. Please see Section "Enhanced Resolution", page 111 for the description of this feature.

13.8 PSCR Inputs

Each part A or B of PSCR has its own system to take into account one PSCR input. According to PSCR Input A/B Control Register (see description "PFRC0A - PSCR Input A Control Register" on page 175), PSCrIN0/1 input can act has a Retrigger or Fault input.

This system A or B is also configured by this PSCR Input A/B Control Register (PFRCrA/B).

Figure 13-8. PSCR input module.



13.8.1 PSCR Retrigger Behavior versus PSCR running modes

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.

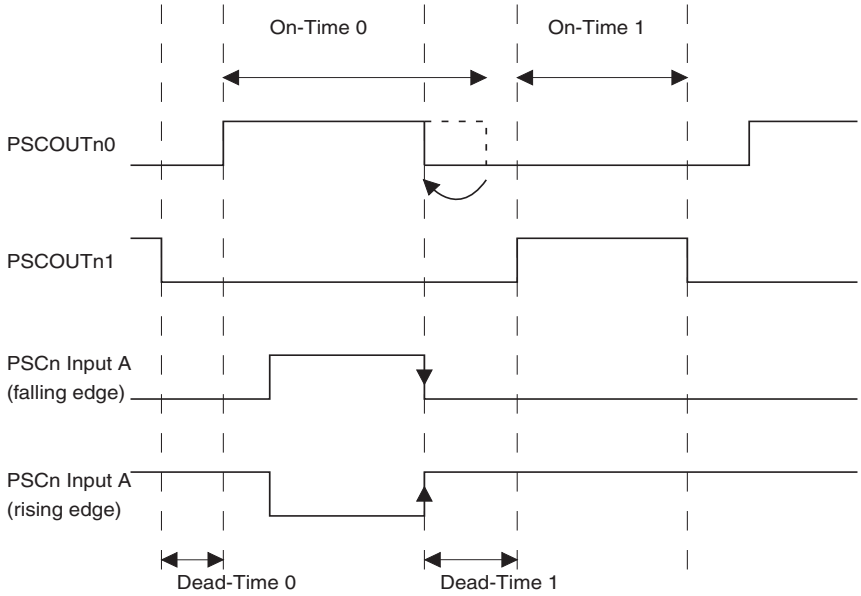
In one ramp mode, Retrigger Inputs A or B reset the current PSCR counting to zero.

13.8.2 Retrigger PSCOUTr0 On External Event

PSCOUTr0 output can be reset before end of On-Time 0 on the change on PSCr Input A. PSCr Input A can be configured to do not act or to act on level or edge modes. The polarity of PSCr Input A is configurable thanks to a sense control block. PSCr Input A can be the Output of the analog comparator or the PSCINr input.

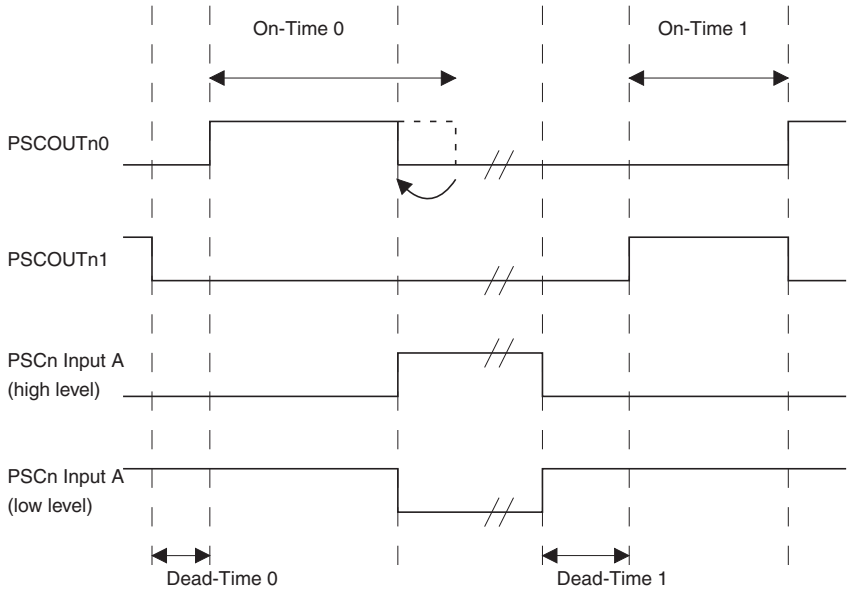
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 13-9. PSCOUTr0 retriggered by PSCr Input A (edge retriggering).



Note: This example is given in “Input Mode 8” in “2 or 4 ramp mode”. See Figure 13-26 on page 166 for details.

Figure 13-10. PSCOUTr0 retriggered by PSCr Input A (level acting).



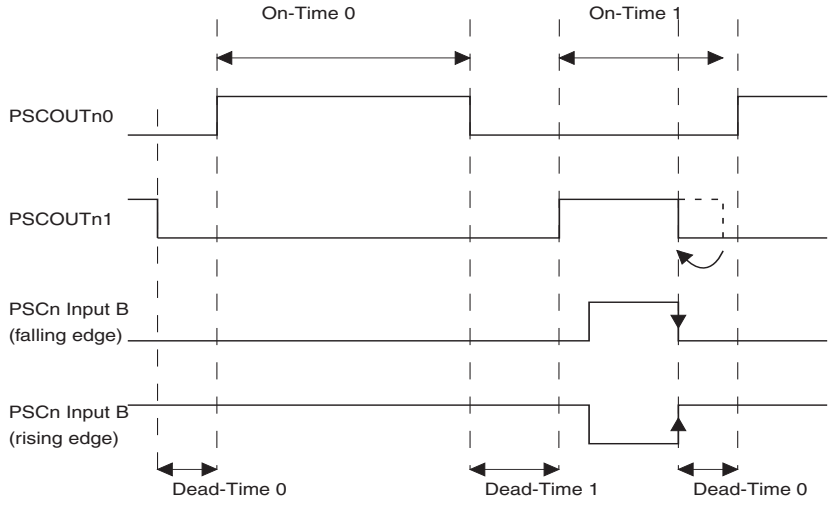
Note: This example is given in “Input Mode 1” in “2 or 4 ramp mode”. See Figure 13-15 on page 161 for details.

13.8.3 Retrigger PSCOUTr1 On External Event

PSCOUTr1 output can be reset before end of On-Time 1 on the change on PSCr Input B. The polarity of PSCr Input B is configurable thanks to a sense control block. PSCr Input B can be configured to do not act or to act on level or edge modes. PSCr Input B can be the Output of the analog comparator or the PSCINr input.

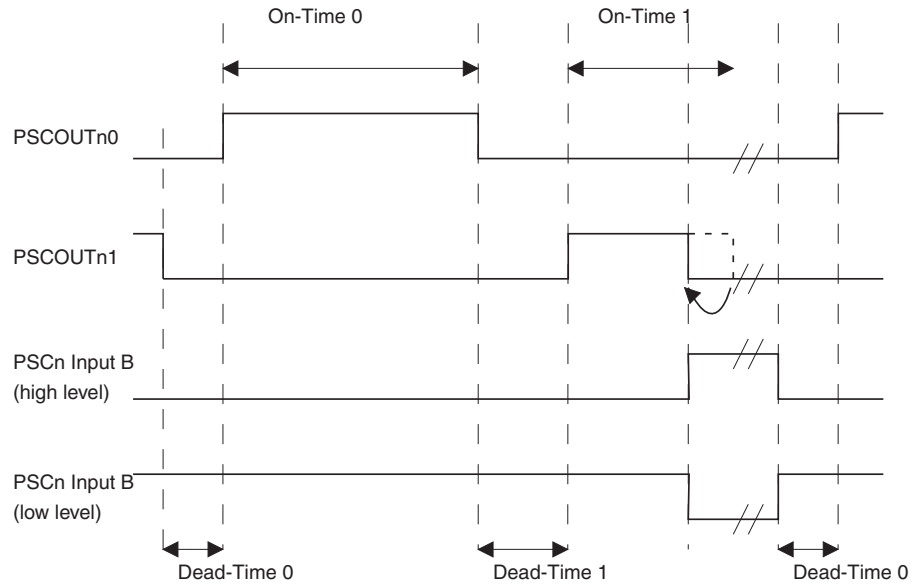
As the period of the cycle decreases, the instantaneous frequency of the two outputs increases.

Figure 13-11. PSCOUTr1 retriggered by PSCr Input B (edge retriggering).



Note: This example is given in “Input Mode 8” in “2 or 4 ramp mode”. See Figure 13-26 on page 166 for details.

Figure 13-12. PSCOUTr1 retriggered by PSCr Input B (level acting).

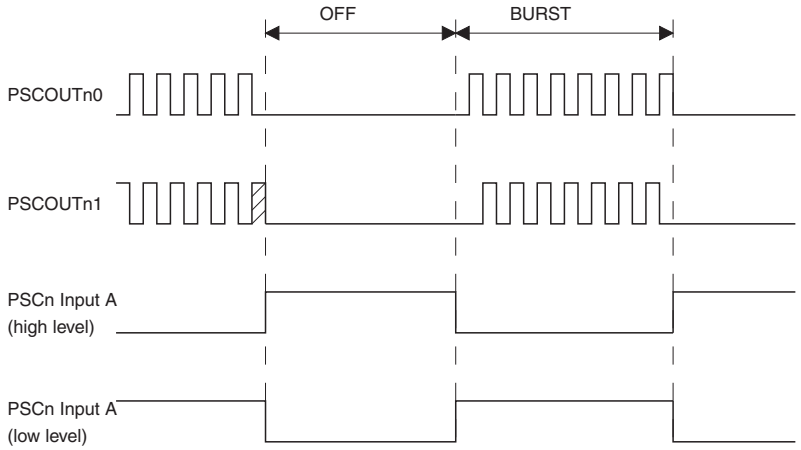


Note: This example is given in “Input Mode 1” in “2 or 4 ramp mode”. See Figure 13-15 on page 161 for details.

13.8.3.1 Burst Generation

Note: On level mode, it’s possible to use PSCR to generate burst by using Input Mode 3 or Mode 4 (see Figure 13-19 on page 163 and Figure 13-20 on page 163 for details.)

Figure 13-13. Burst generation.



13.8.4 PSCR Input Configuration

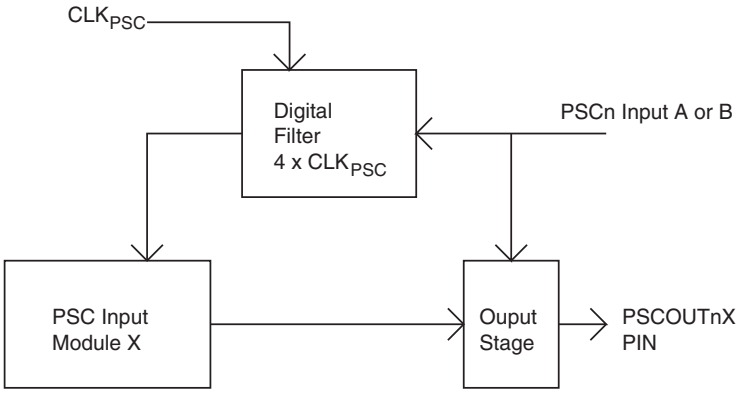
The PSCR Input Configuration is done by programming bits in configuration registers.

13.8.4.1 Filter Enable

If the “Filter Enable” bit is set, a digital filter of four cycles is inserted before evaluation of the signal. The disable of this function is mainly needed for prescaled PSCR 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 PSCR clock to deactivate the outputs (emergency protection of external component). Likewise when used as fault input, PSCr Input A or Input B have to go through PSCR to act on PSCOUTr0/1/2/3 output. This way needs that CLK_{PSCR} is running. So thanks to PSCR Asynchronous Output Control bit (PAOCrA/B), PSCrIN0/1 input can deactivate directly the PSCR output. Notice that in this case, input is still taken into account as usually by Input Module System as soon as CLK_{PSCR} is running.

Figure 13-14. PSCR input filtering.



13.8.4.2 Signal Polarity

One can select the active edge (edge modes) or the active level (level modes). See PELEV0x bit description in Section “PFRC0A - PSCR Input A Control Register”, page 175.

If PELEV0x bit set, the significant edge of PSCr 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, PSCr Input A is taken into account only during Dead-Time0 and On-Time0 period (respectively Dead-Time1 and On-Time1 for PSCr Input B).

- In 1-ramp-mode PSCR Input A or PSCR Input B act on the whole ramp.

13.8.4.3 *Input Mode Operation*

Thanks to four configuration bits (PRFM3:0), it is possible to define all the modes of the PSCR input. These modes are listed in Table 13-5.

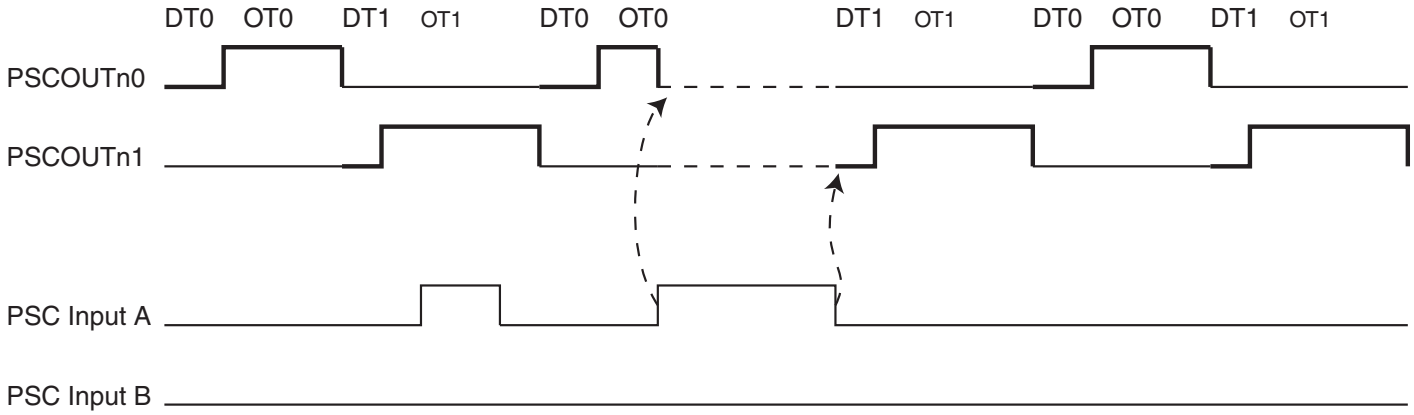
Table 13-5. PSCR input mode operation.

	PRFM3:0	Description
0	0000b	PSCr Input has no action on PSCR output
1	0001b	See “PSCR Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait” on page 161.
2	0010b	See “PSCR Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait” on page 162.
3	0011b	See “PSCR Input Mode 3: Stop signal, Execute Opposite while Fault active” on page 163.
4	0100b	See “PSCR Input Mode 4: Deactivate outputs without changing timing” on page 164.
5	0101b	See “PSCR Input Mode 5: Stop signal and Insert Dead-Time” on page 164.
6	0110b	See “PSCR Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait” on page 165.
7	0111b	See “PSCR Input Mode 7: Halt PSCR and Wait for Software Action” on page 165.
8	1000b	See “PSCR Input Mode 8: Edge Retrigger PSC” on page 166.
9	1001b	See “PSCR Input Mode 9: Fixed Frequency Edge Retrigger PSC” on page 167.
10	1010b	Reserved: Do not use
11	1011b	
12	1100b	
13	1101b	
14	1110b	See “PSCR Input Mode 14: Fixed Frequency Edge Retrigger PSCR and Deactivate Output” on page 168.
15	1111b	Reserved: Do not use

Note: All the following examples are given with rising edge or high level active inputs.

13.9 PSCR Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait

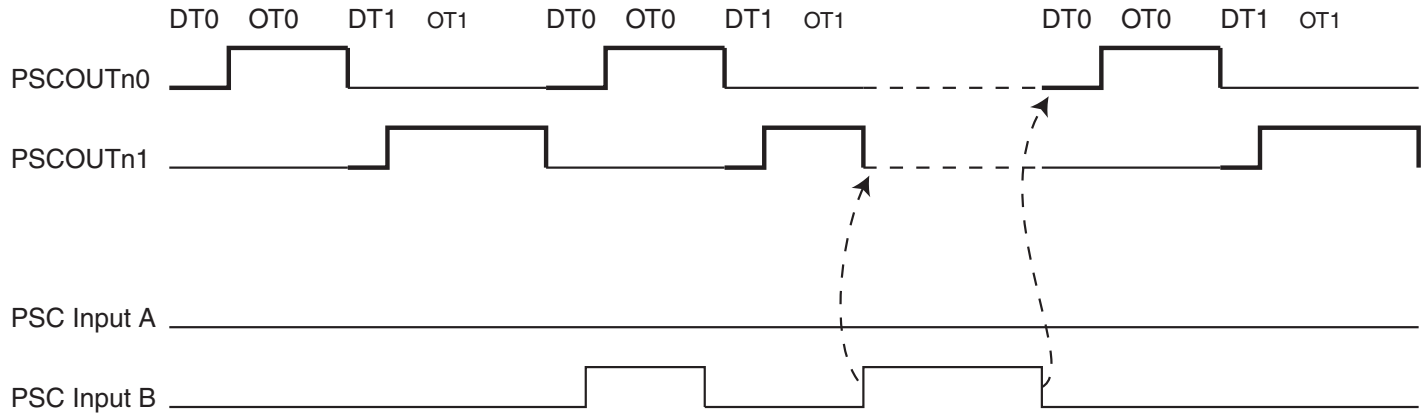
Figure 13-15. PSCr behavior versus PSCr Input A in Fault Mode 1.



PSCR Input A is taken into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSCR Input A event occurs, PSCR releases PSCOUTr0, waits for PSCR Input A inactive state and then jumps and executes DT1 plus OT1.

Figure 13-16. PSCr behavior versus PSCr Input B in Fault Mode 1.

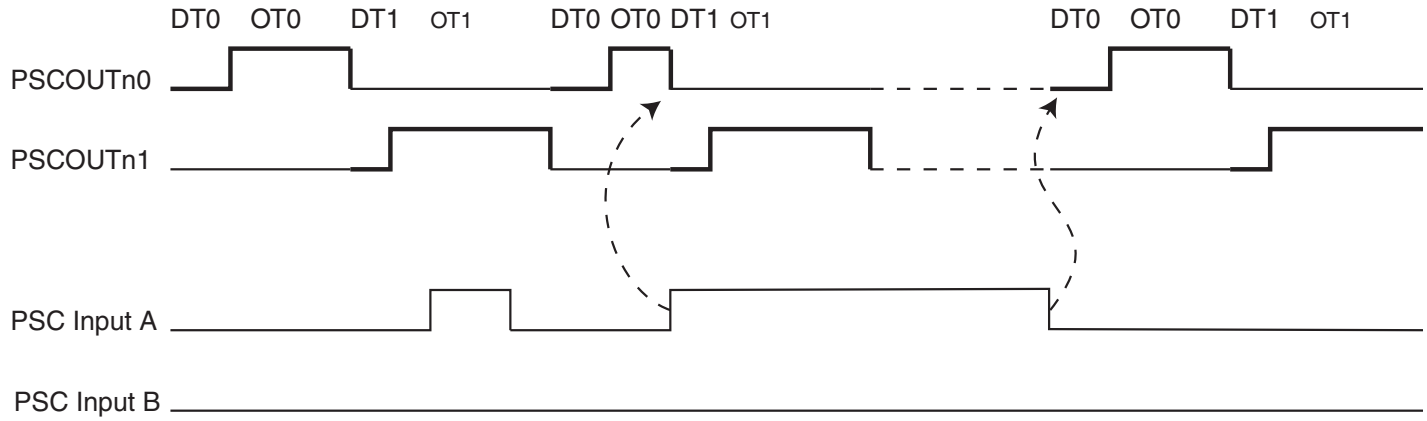


PSCR Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSCR Input B event occurs, PSCR releases PSCOUTr1, waits for PSCR Input B inactive state and then jumps and executes DT0 plus OT0.

13.10 PSCR Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait

Figure 13-17. PSCr behavior versus PSCr Input A in Fault Mode 2.

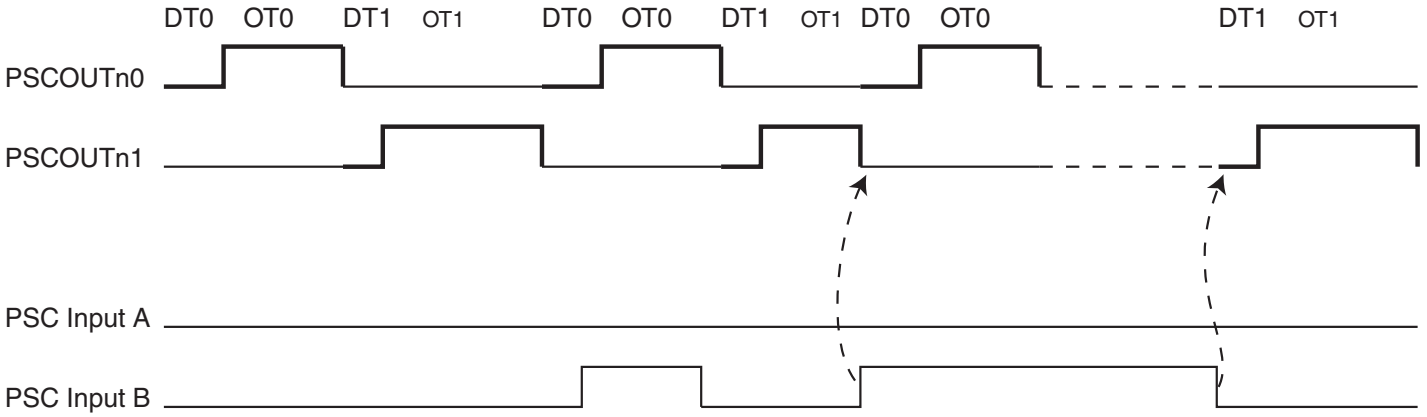


PSCR Input A is take into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSCr Input A event occurs, PSCR releases PSCOUTr0, jumps and executes DT1 plus OT1 and then waits for PSCR Input A inactive state.

Even if PSCR Input A is released during DT1 or OT1, DT1 plus OT1 sub-cycle is always completely executed.

Figure 13-18. PSCr behavior versus PSCr Input B in Fault Mode 2.



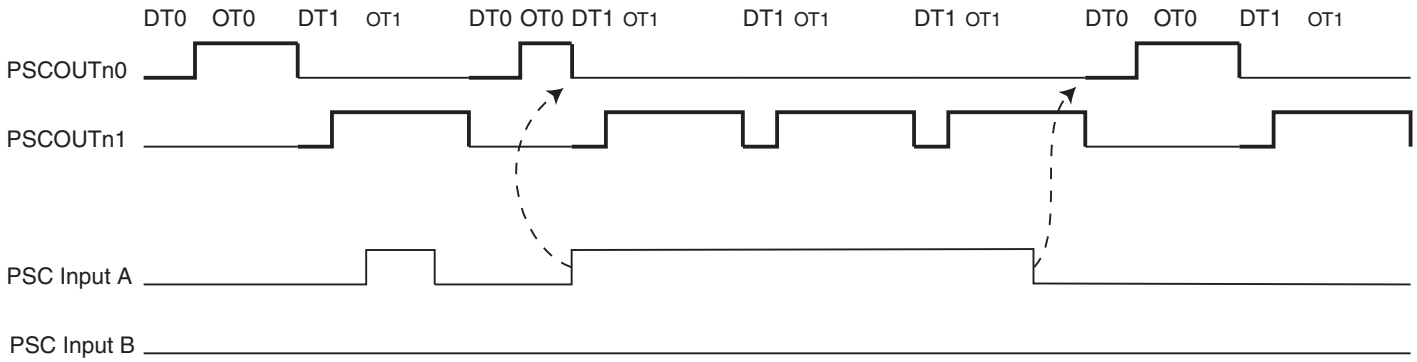
PSCR Input B is take into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSCR Input B event occurs, PSCR releases PSCOUTr1, jumps and executes DT0 plus OT0 and then waits for PSCR Input B inactive state.

Even if PSCR Input B is released during DT0 or OT0, DT0 plus OT0 sub-cycle is always completely executed.

13.11 PSCR Input Mode 3: Stop signal, Execute Opposite while Fault active

Figure 13-19. PSCr behavior versus PSCr Input A in Mode 3.

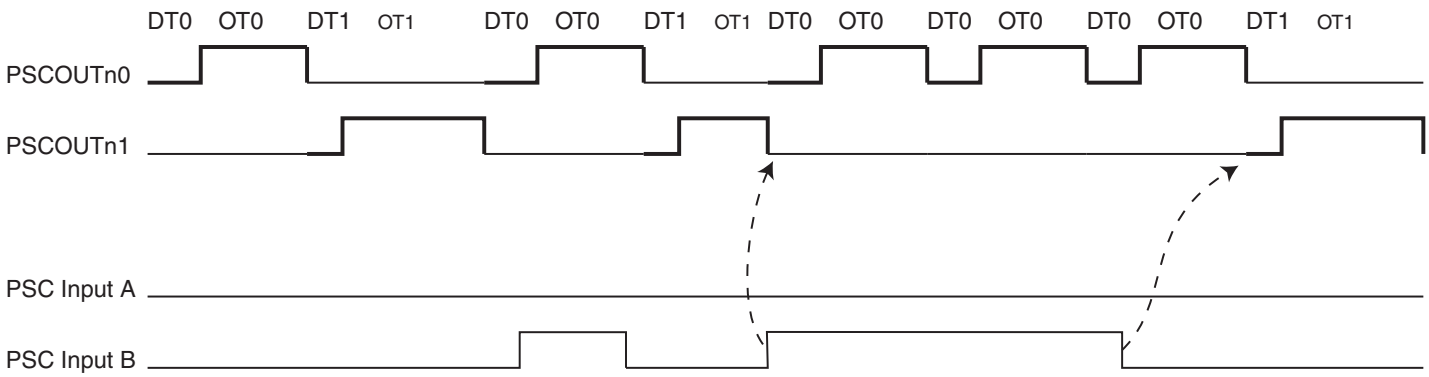


PSCR Input A is taken into account during DT0 and OT0 only. It has no effect during DT1 and OT1.

When PSCR Input A event occurs, PSCR releases PSCOUTr0, jumps and executes DT1 plus OT1 plus DT0 while PSCR Input A is in active state.

Even if PSCR Input A is released during DT1 or OT1, DT1 plus OT1 sub-cycle is always completely executed.

Figure 13-20. PSCr behavior versus PSCr Input B in Mode 3.



PSCR Input B is taken into account during DT1 and OT1 only. It has no effect during DT0 and OT0.

When PSCR Input B event occurs, PSCR releases PSCOUTR1, jumps and executes DT0 plus OT0 plus DT1 while PSCR Input B is in active state.

Even if PSCR Input B is released during DT0 or OT0, DT0 plus OT0 sub-cycle is always completely executed.

13.12 PSCR Input Mode 4: Deactivate outputs without changing timing

Figure 13-21. PSCR behavior versus PSCr Input A or Input B in Mode 4.

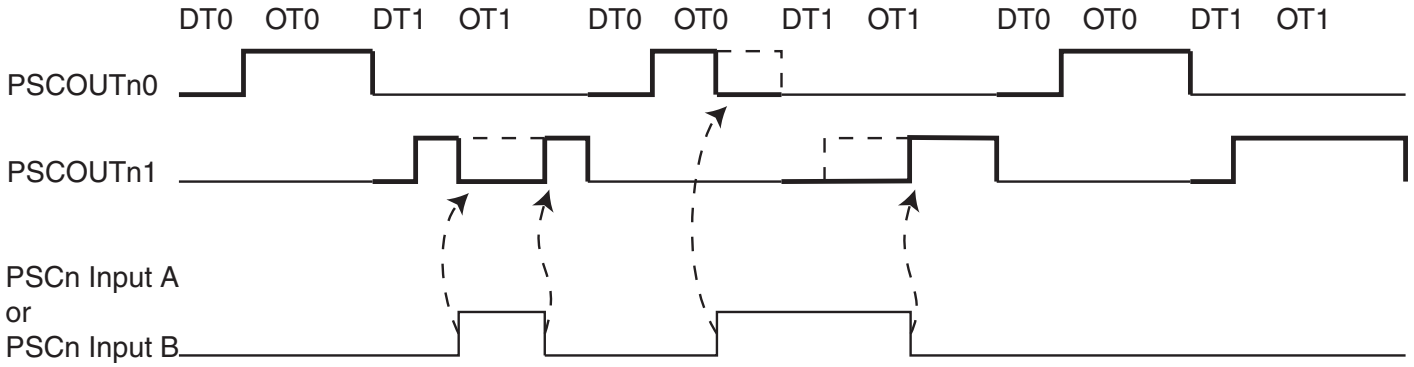
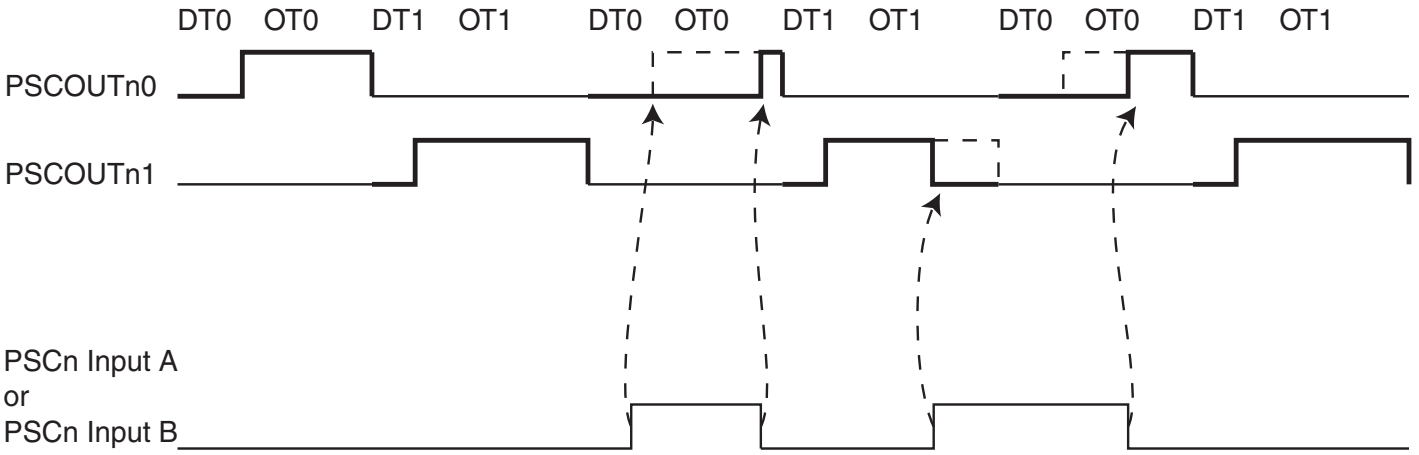


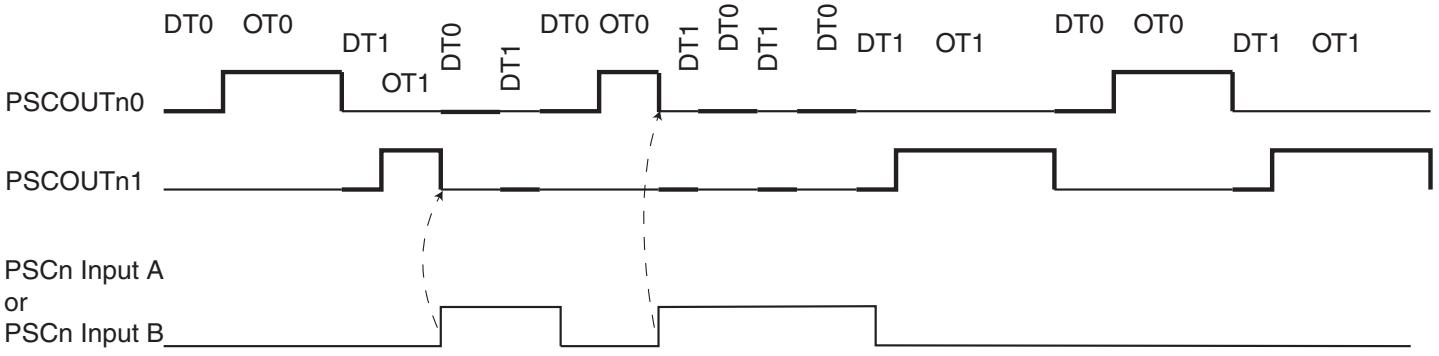
Figure 13-22. PSCR behavior versus PSCr Input A or Input B in Fault Mode 4.



PSCr Input A or PSCr Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

13.13 PSCR Input Mode 5: Stop signal and Insert Dead-Time

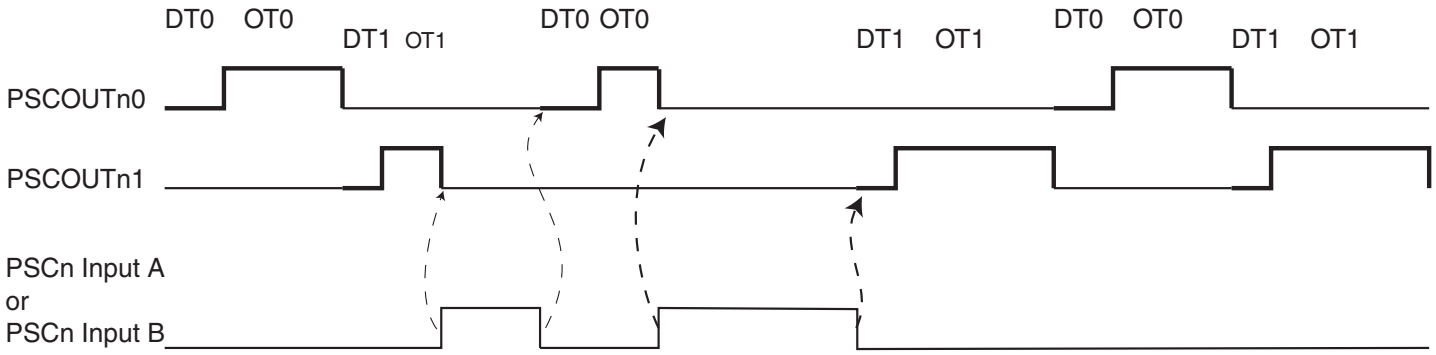
Figure 13-23. PSCR behavior versus PSCr Input A in Fault Mode 5.



Used in Fault mode 5, PSCr Input A or PSCr Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

13.14 PSCR Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait

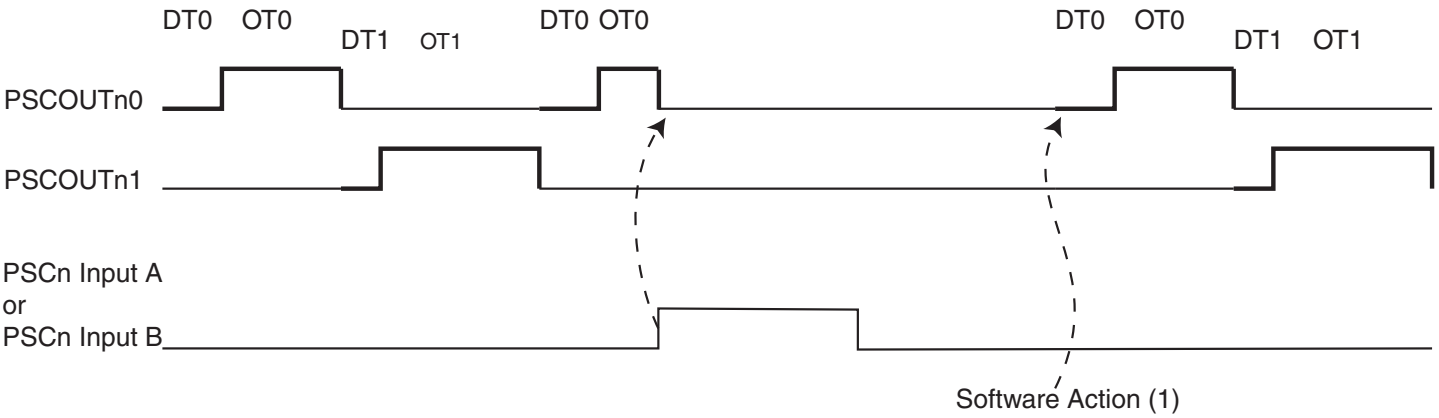
Figure 13-24. PSCR behavior versus PSCr Input A in Fault Mode 6.



Used in Fault mode 6, PSCr Input A or PSCr Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

13.15 PSCR Input Mode 7: Halt PSCR and Wait for Software Action

Figure 13-25. PSCR behavior versus PSCr Input A in Fault Mode 7.

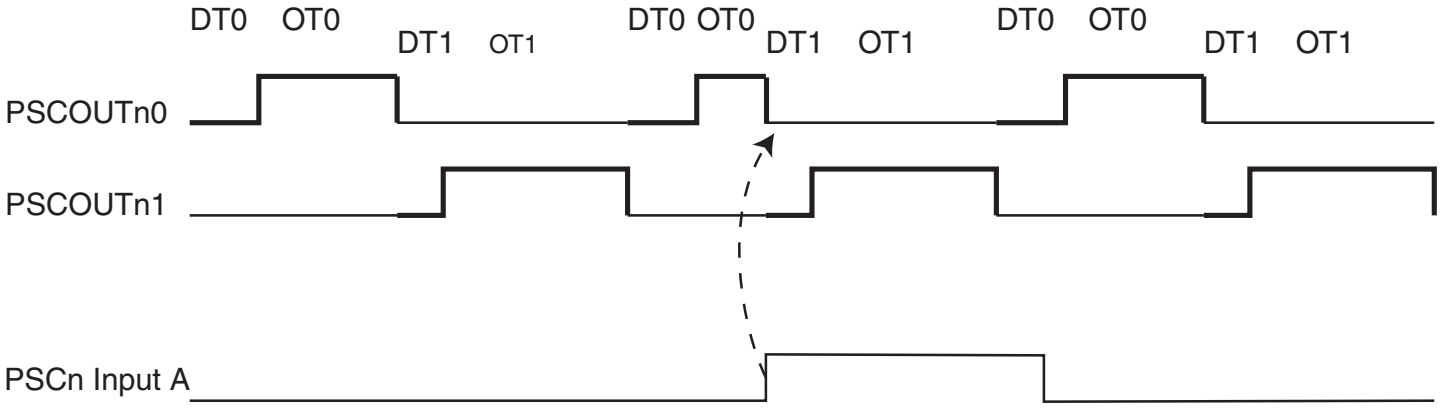


Note: 1. Software action is the setting of the PRUNn bit in PCTLr register.

Used in Fault mode 7, PSCr Input A or PSCr Input B act indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

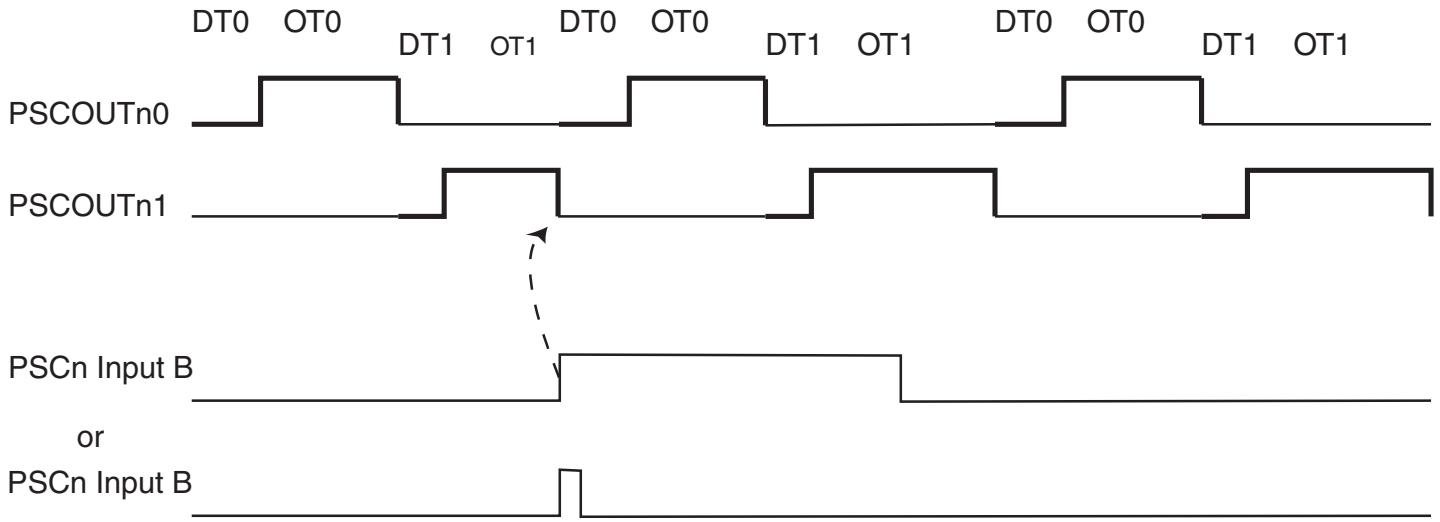
13.16 PSCR Input Mode 8: Edge Retrigger PSC

Figure 13-26. PSCR behavior versus PSCr Input A in Mode 8.



The output frequency is modulated by the occurrence of significative edge of retriggering input.

Figure 13-27. PSCR behavior versus PSCr Input B in Mode 8.



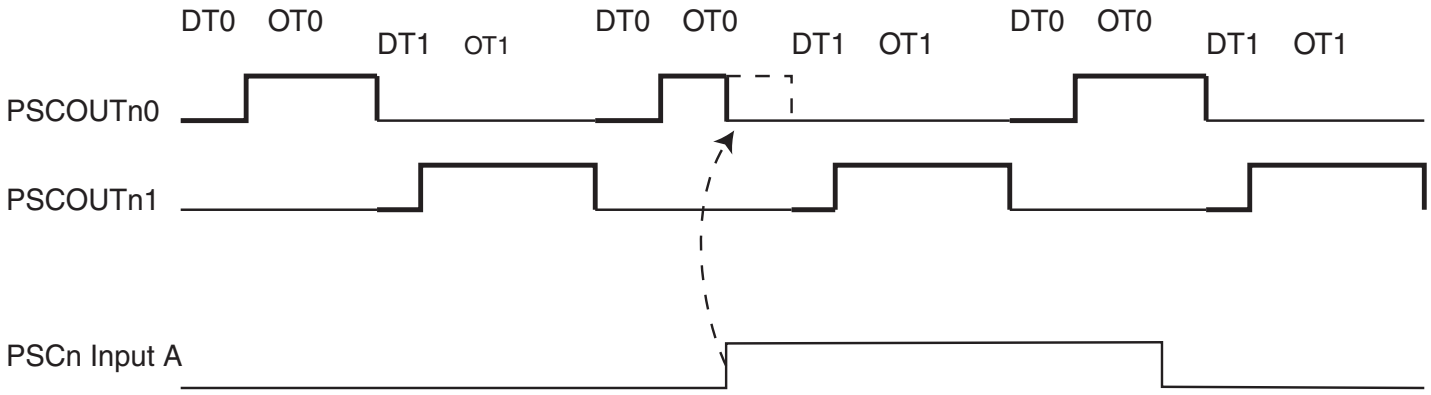
The output frequency is modulated by the occurrence of significative edge of retriggering input.

The retrigger event is taken into account only if it occurs during the corresponding On-Time.

Note: In one ramp mode, the retrigger event on input A resets the whole ramp. So the PSCR doesn't jump to the opposite dead-time.

13.17 PSCR Input Mode 9: Fixed Frequency Edge Retrigger PSC

Figure 13-28. PSCR behavior versus PSCr Input A in Mode 9.

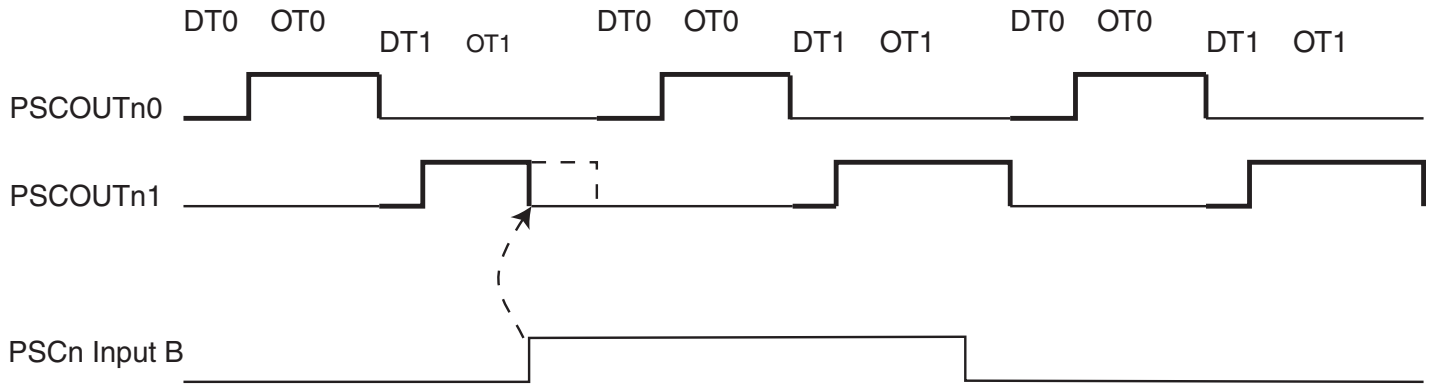


The output frequency is not modified by the occurrence of significative edge of retriggering input.

Only the output is deactivated when significative edge on retriggering input occurs.

Note: In this mode the output of the PSCR becomes active during the next ramp even if the Retrigger/Fault input is active. Only the significative edge of Retrigger/Fault input is taken into account.

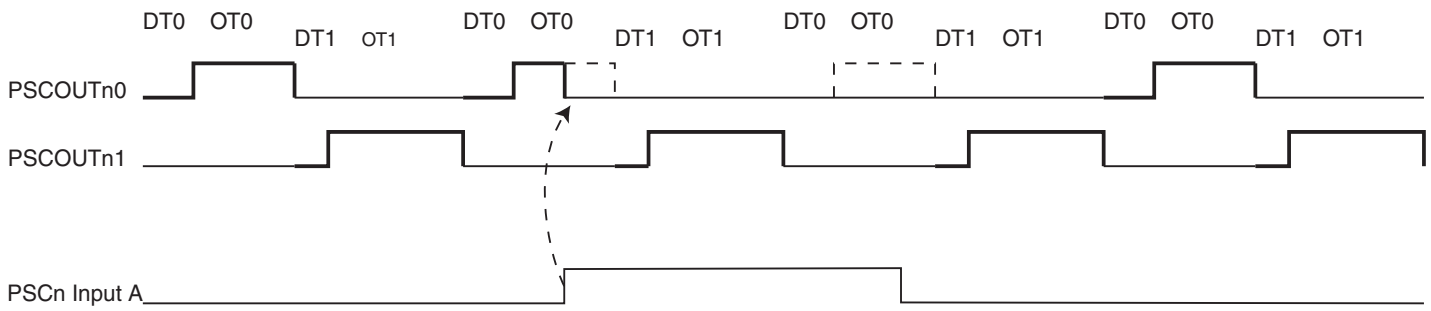
Figure 13-29. PSCR behavior versus PSCr Input B in Mode 9.



The retrigger event is taken into account only if it occurs during the corresponding On-Time.

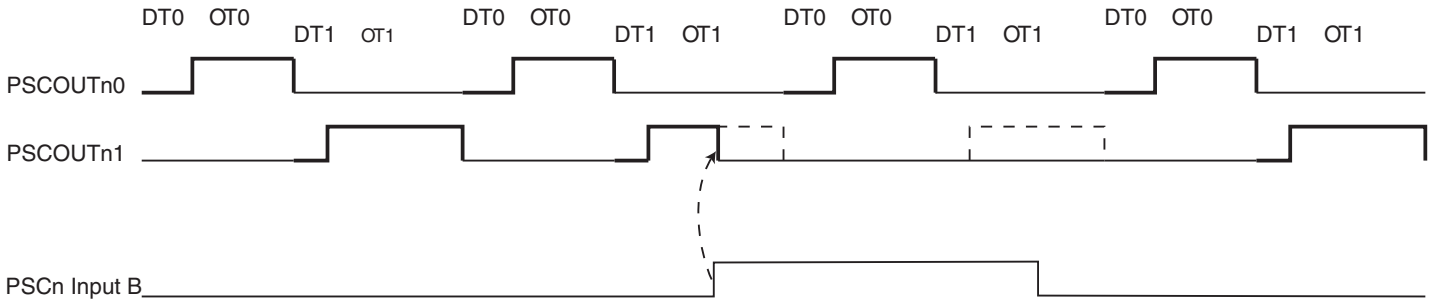
13.18 PSCR Input Mode 14: Fixed Frequency Edge Retrigger PSCR and Deactivate Output

Figure 13-30. PSCR behavior versus PSCr Input A in Mode 14.



The output frequency is not modified by the occurrence of significative edge of retriggering input.

Figure 13-31. PSCR behavior versus PSCr Input B in Mode 14.



The output is deactivated while retriggering input is active.

The output of the PSCR 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 active. The PSCR runs at constant frequency.

13.18.1 Available Input Mode according to Running Mode

Some Input Modes are not consistent with some Running Modes. So Table 13-6 gives the input modes which are valid according to running modes.

Table 13-6. Available input modes according to running modes.

Input mode number:	1 Ramp mode	2 Ramp mode	4 Ramp mode
1	Valid	Valid	Valid
2	Do not use	Valid	Valid
3	Do not use	Valid	Valid
4	Valid	Valid	Valid
5	Do not use	Valid	Valid
6	Do not use	Valid	Valid
7	Valid	Valid	Valid
8	Valid	Valid	Valid
9	Valid	Valid	Valid

Table 13-6. Available input modes according to running modes. (Continued)

Input mode number:	1 Ramp mode	2 Ramp mode	4 Ramp mode
10	Do not use		
11			
12			
13			
14	Valid	Valid	Valid
15	Do not use		

13.18.2 Event Capture

The PSCR can capture the value of time (PSCR counter) when a retrigger event or fault event occurs on PSCR inputs. This value can be read by software in PICRrH/L register.

13.18.3 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 PICR1 Register before the next event occurs, the PICR1 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 PICR1 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.

13.19 Analog Synchronization

PSCR generates a signal to synchronize the sample and hold; synchronization is mandatory for measurements.

This signal can be selected between all falling or rising edge of PSCr0 or PSCr1 outputs.

13.20 Interrupt Handling

List of interrupt sources:

- Counter reload (end of On Time 1)
- PSCR Input event (active edge or at the beginning of level configured event)

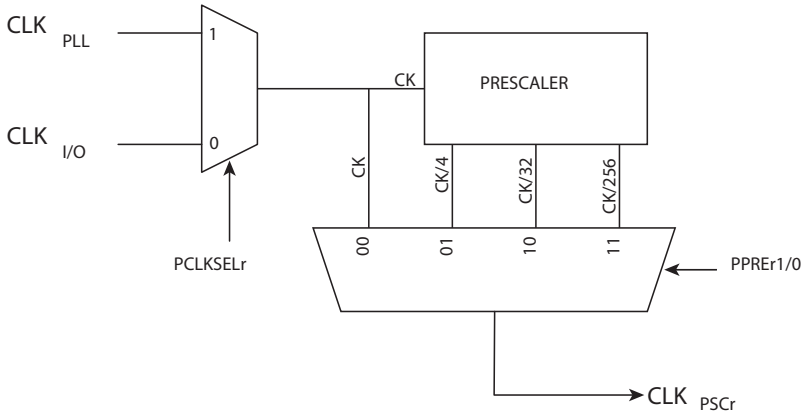
13.21 PSC Clock Sources

PSCR must be able to generate high frequency with enhanced resolution.

The PSCR has two clock inputs:

- CLK PLL from the PLL
- CLK I/O

Figure 13-32. Clock selection.



PCLKSELr bit in PSCR Configuration register (PCNFr) is used to select the clock source.

PPREr1/0 bits in PSCR Control Register (PCTLr) are used to select the divide factor of the clock.

Table 13-7. Output clock versus selection and prescaler.

PCLKSELr	PPREr1	PPREr0	CLKPSCr output
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

13.22 Interrupts

This section describes the specifics of the interrupt handling as performed in AT90PWM81/161.

13.22.1 List of Interrupt Vector

The PSCR provides 3 interrupt vectors:

- **PSC0EC (End of Cycle)**: When enabled and when a match with OCRrRB occurs
- **PSC0EEC (End of Enhanced Cycle)**: When enabled and when a match with OCRrRB occurs at the 15th enhanced cycle
- **PSC0CAPT (Capture Event)**: When enabled and one of the two following events occurs : retrigger, capture of the PSCR counter or Synchro Error

See PSC0 Interrupt Mask Register page 177 and PSC0 Interrupt Flag Register page 178.

13.23 PSCR Register Definition

13.23.1 PSOC0 - PSCR Synchro and Output Configuration

Bit	7	6	5	4	3	2	1	0	
	PISEL0A1	PISEL0B1	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	

- **Bit 7– PISEL0A1: PSC Input Select for part A**

Together with PISEL0A0, defines active signal on PSCR part A.

Table 13-8. PSC trigger & fault input selection.

PISEL0A1	PISEL0A0	Description
0	0	PSCINr
0	1	Analog comparator output
1	0	PSCINrA
1	1	PSCINrB

- **Bit 6– PISEL0B1: PSCR Input Select for part B**

Together with PISEL0B0, defines active signal on PSCR part B.

Table 13-9. PSC trigger & fault input selection.

PISEL0B1	PISEL0B0	Description
0	0	PSCINr
0	1	Analog comparator output
1	0	PSCINrA
1	1	PSCINrB

- **Bit 5:4 – PSYNC01:0: 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 13-10. Synchronization source description in one/two/four Ramp modes.

PSYNC01	PSYNC00	Description
0	0	Send signal on leading edge of PSCOUT00 (match with OCR0SA)
0	1	Send signal on trailing edge of PSCOUT00 (match with OCR0RA or fault/retrigger on part A)
1	0	Send signal on leading edge of PSCOUT01 (match with OCR0SB)
1	1	Send signal on trailing edge of PSCOUT01 (match with OCR0RB or fault/retrigger on part B)

- **Bit 3 – Reserved**

- **Bit 2 – POEN0B: PSCR OUT Part B Output Enable**

When this bit is clear, I/O pin affected to PSCOUT01 acts as a standard port.

When this bit is set, I/O pin affected to PSCOUT01 is connected to the PSCR waveform generator B output and is set and clear according to the PSCR operation.

- **Bit 1 – Reserved**

- **Bit 0 – POEN0A: PSCR OUT Part A Output Enable**

When this bit is clear, I/O pin affected to PSCOUT00 acts as a standard port.

When this bit is set, I/O pin affected to PSCOUT00 is connected to the PSCR waveform generator A output and is set and clear according to the PSCR operation.

13.23.2 OCR0SAH and OCR0SAL - Output Compare SA Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCR0SA[11:8]				OCR0SAH
	OCR0SA[7:0]								OCR0SAL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

13.23.3 OCR0RAH and OCR0RAL - Output Compare RA Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCR0RA[11:8]				OCR0RAH
	OCR0RA[7:0]								OCR0RAL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

13.23.4 OCR0SBH and OCR0SBL - Output Compare SB Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	OCR0SB[11:8]				OCR0SBH
	OCR0SB[7:0]								OCR0SBL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

13.23.5 OCR0RBH and OCR0RBL - Output Compare RB Register

Bit	7	6	5	4	3	2	1	0	
	OCR0RB[15:12]				OCR0RB[11:8]				OCR0RBH
	OCR0RB[7:0]								OCR0RBL
Read/Write	W	W	W	W	W	W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSCR 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 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.

13.23.6 PCNF0 - PSCR Configuration Register

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	

- **Bit 7 - PFIFTY0: PSCR Fifty**

Writing this bit to one, set the PSCR in a fifty percent mode where only OCR0RBH/L and OCR0SBH/L are used. They are duplicated in OCR0RAH/L and OCR0SAH/L during the update of OCR0RBH/L. This feature is useful to perform fifty percent waveforms.

- **Bit 6 - PALOCK0: PSCR Autolock**

When this bit is set, the Output Compare Registers RA, SA, SB, the Output Matrix POM2 and the PSCR Output Configuration PSOC0 can be written without disturbing the PSCR cycles. The update of the PSCR internal registers will be done at the end of the PSCR cycle if the Output Compare Register RB has been the last written.

When set, this bit prevails over LOCK (bit 5).

- **Bit 5 – PLOCK0: PSCR Lock**

When this bit is set, the Output Compare Registers RA, RB, SA, SB, the Output Matrix POM2 and the PSCR Output Configuration PSOC0 can be written without disturbing the PSCR cycles. The update of the PSCR internal registers will be done if the LOCK bit is released to zero.

- **Bit 4:3 – PMODE01: 0: PSCR Mode**

Select the mode of PSC.

Table 13-11. PSCR mode selection.

PMODE01	PMODE00	Description
0	0	One Ramp mode
0	1	Two Ramp mode
1	0	Four Ramp mode
1	1	Reserved

- **Bit 2 – POP0: PSCR Output Polarity**

If this bit is cleared, the PSCR outputs are active Low.

If this bit is set, the PSCR outputs are active High.

- **Bit 1 – PCLKSEL0: PSCR 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 – Reserved**

13.23.7 PCTL0 - PSCR Control Register

Bit	7	6	5	4	3	2	1	0	
	PPRE01	PPRE00	PBFM01	PAOC0B	PAOC0A	PBFM00	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 : PSCR Prescaler Select**

This two bits select the PSCR input clock division factor. All generated waveform will be modified by this factor.

Table 13-12. PSCR prescaler selection.

PPRE01	PPRE00	Description
0	0	No divider on PSCR input clock
0	1	Divide the PSCR input clock by 4
1	0	Divide the PSCR input clock by 32
1	1	Divide the PSCR clock by 256

- **Bit 5- PBFM01: Balance Flank Width Modulation, bit 1**

Defines the Flank Width Modulation, together with PBFM00 bit.

Table 13-13. Flank Width mode selection.

PBFM01	PBFM00	Description
0	0	Flank Width Modulation operates on RB (On-Time 1 only)
0	1	Flank Width Modulation operates on RB + RA (On-Time 0 and On-Time 1)
1	0	Flank Width Modulation operates on SB (Dead-Time 1 only) (1)
1	1	Flank Width Modulation operates on SB +SA (Dead-Time 0 and Dead-Time 1)

Note: 1. In one ramp mode, changing SA or SA+SB also affect On-Time; see Figure 13-6 on page 154.

- **Bit 4 – PAOC0B: PSCR Asynchronous Output Control B**

When this bit is set, Fault input selected to block B can act directly to PSCOUT01 output. See Section “PSCR Input Configuration”, page 159.

- **Bit 3 – PAOC0A: PSCR Asynchronous Output Control A**

When this bit is set, Fault input selected to block A can act directly to PSCOUT00 output. See Section “PSCR Input Configuration”, page 159.

- **Bit 2- PBFM00: Balance Flank Width Modulation, bit 0**

Defines the Flank Width Modulation, together with PBFM01 bit.

- **Bit 1 – PCCYC0: PSCR Complete Cycle**

When this bit is set, the PSCR 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: PSCR Run**

Writing this bit to one starts the PSCR.

When set, this bit prevails over PARUN0 bit.

13.23.8 PFRC0A - PSCR Input A Control Register

Bit	7	6	5	4	3	2	1	0	
	PCAE0A	PISEL0A0	PELEV0A	PFLTE0A	PRFM0A3	PRFM0A2	PRFM0A1	PRFM0A0	PFRC0A
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	

13.23.9 PFRC0B - PSCR Input B Control Register

Bit	7	6	5	4	3	2	1	0	
	PCAE0B	PISEL0B0	PELEV0B	PFLTE0B	PRFM0B3	PRFM0B2	PRFM0B1	PRFM0B0	PFRC0B
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 – PCAE0x: PSCR 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 PISEL0x0 bit in the same register).

- **Bit 6 – PISEL0x0: PSCR Input Select for Part x**

Together with PISEL0x1 in PSOC0 register, defines active signal on PSC module A. See Table 13-8 on page 171 and Table 13-9 on page 171.

- **Bit 5 –PELEV0x: PSCR 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 – PFLTE0x: PSCR 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 – PRFM0x3:0: PSCR Fault Mode**

These four bits define the mode of operation of the Fault or Retrigger functions. (See Table 13-5 on page 160 for more explanations).

Table 13-14. Level sensitivity and Fault mode operation.

PRFM0x3:0	Description
0000b	No action, PSCR Input is ignored
0001b	“PSCR Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait”, page 161
0010b	“PSCR Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait”, page 162
0011b	“PSCR Input Mode 3: Stop signal, Execute Opposite while Fault active”, page 163
0100b	“PSCR Input Mode 4: Deactivate outputs without changing timing”, page 164
0101b	“PSCR Input Mode 5: Stop signal and Insert Dead-Time”, page 164
0110b	“PSCR Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait”, page 165
0111b	“PSCR Input Mode 7: Halt PSCR and Wait for Software Action”, page 165
1000b	“PSCR Input Mode 8: Edge Retrigger PSC”, page 166
1001b	“PSCR Input Mode 9: Fixed Frequency Edge Retrigger PSC”, page 167

Table 13-14. Level sensitivity and Fault mode operation. (Continued)

PRFM0x3:0	Description
1010b	Reserved (do not use)
1011b	
1100b	
1101b	
1110b	"PSCR Input Mode 14: Fixed Frequency Edge Retrigger PSCR and Deactivate Output", page 168
1111b	Reserved (do not use)

13.23.10 PICR0H and PICR0L - PSCR Input Capture Register

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	

- **Bit 7 – PCST0: PSCR Capture Software Trig bit**

Set this bit to trigger off a capture of the PSCR counter. When reading, if this bit is set it means that the capture operation was triggered by PCST0 setting otherwise it means that the capture operation was triggered by a PSCR input.

The Input Capture is updated with the PSCR counter value each time an event occurs on the enabled PSCR input pin (or optionally on the Analog Comparator output) if the capture function is enabled (bit PCAE0x in PFRC0x 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.

13.23.11 PIM0 - PSCR Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	PEVE0B	PEVE0A	–	PEOEPE0	PEOPE0	PIM0
Read/Write	R	R	R	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7- 5 – Reserved**

- **Bit 4 – PEVE0B: PSCR 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 – PEVE0A: PSCR 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 2 – Reserved**

- **Bit 1– PEOEPE0: PSCR End Of Enhanced Cycle Interrupt Enable**

When this bit is set, an interrupt is generated when PSC reduced reaches the end of the 15th PSC cycle. This allows to update the PSCR values in the interrupt routine and to start a new enhanced cycle with the new values at the next PSCR cycle end.

- **Bit 0 – PEOPE0: PSCR End Of Cycle Interrupt Enable**

When this bit is set, an interrupt is generated when PSCR reaches the end of the whole cycle.

13.23.12 PIFR0 - PSCR Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	POAC0B	POAC0A	-	PEV0B	PEV0A	PRN01	PRN00	PEOP0	PIFR0
Read/Write	R	R	R	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – POAC0B: PSCR Output B Activity**

This bit is set by hardware each time the output PSCOUT01 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 PSCR output doesn't change due to a frozen external input signal.

- **Bit 6 – POAC0A: PSCR Output A Activity**

This bit is set by hardware each time the output PSCOUT00 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 PSCR output doesn't change due to a freezen external input signal.

- **Bit 5 – Reserved**

- **Bit 4 – PEV0B: PSCR 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 (PEVE0B bit = 0).

- **Bit 3 – PEV0A: PSCR 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 (PEVE0A bit = 0).

- **Bit 2:1 – PRN01:0 : PSCR Ramp Number**

Memorization of the ramp number when the last PEV0A or PEV0B occurred.

Table 13-15. PSCR ramp number description.

PRN01	PRN00	Description
0	0	The last event which has generated an interrupt occurred during ramp 1
0	1	The last event which has generated an interrupt occurred during ramp 2
1	0	The last event which has generated an interrupt occurred during ramp 3
1	1	The last event which has generated an interrupt occurred during ramp 4

- **Bit 0 – PEOP0: End Of PSCR Interrupt**

This bit is set by hardware when PSCR achieves its whole cycle.

Must be cleared by software by writing a one to its location.

14. Serial Peripheral Interface – SPI:

14.1 Features

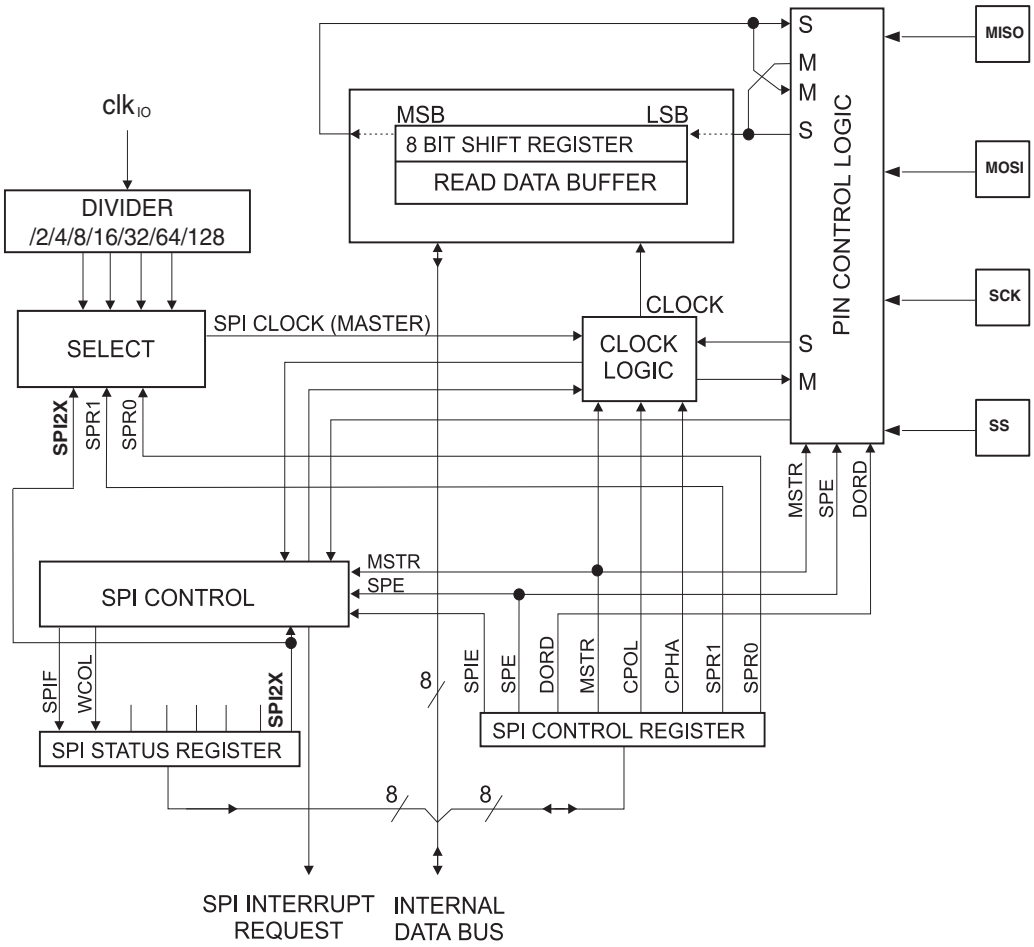
- **Full-duplex, three-wire synchronous data transfer**
- **Master or Slave operation**
- **LSB first or MSB first data transfer**
- **Seven programmable bit rates**
- **End of transmission interrupt flag**
- **Write collision flag protection**
- **Wake-up from idle mode**
- **Double speed (CK/2) Master SPI mode**

14.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT90PWM81/161 and peripheral devices or between several AVR devices.

The AT90PWM81/161 SPI includes the following features.

Figure 14-1. SPI block diagram [(1)].



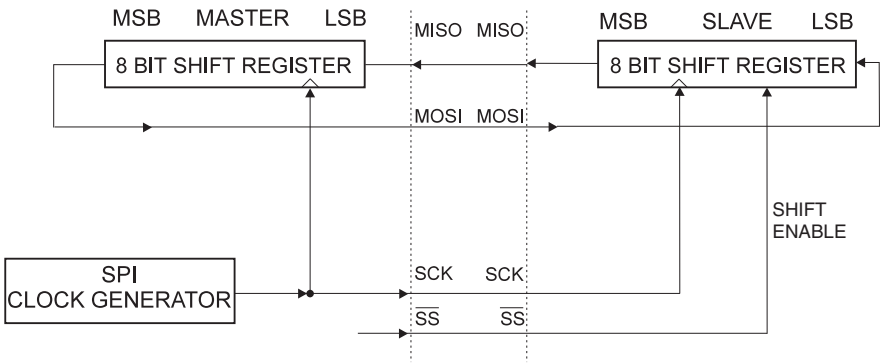
Note: 1. Refer to Figure 2-1 on page 3, and Table 9-3 on page 75 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 14-2. 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 14-2. 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 14-1. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 73.

Table 14-1. SPI pin overrides [1].

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

Note: 1. See “Alternate Functions of Port B” on page 75 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. For example, if MOSI is placed on pin PB2, replace DD_MOSI with DDB2 and DDR_SPI with DDRB.

Assembly code example [1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi  r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out  DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi  r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out  SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out  SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis SPSR,SPIF
  rjmp Wait_Transmit
  ret
```

C code example [1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

Note: 1. 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.

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.

14.3 $\overline{SS}$ Pin Functionality

14.3.1 Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which

means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

14.3.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

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

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

14.4 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 14-3 on page 186 and Figure 14-4 on page 186. 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 14-3 on page 187 and Table 14-4 on page 187, as done below:

Table 14-2. CPOL functionality.

	Leading edge	Trailing edge	SPI mode
CPOL=0, CPHA=0	Sample (rising)	Setup (falling)	0
CPOL=0, CPHA=1	Setup (rising)	Sample (falling)	1
CPOL=1, CPHA=0	Sample (falling)	Setup (rising)	2
CPOL=1, CPHA=1	Setup (falling)	Sample (rising)	3

Figure 14-3. SPI transfer format with CPHA = 0.

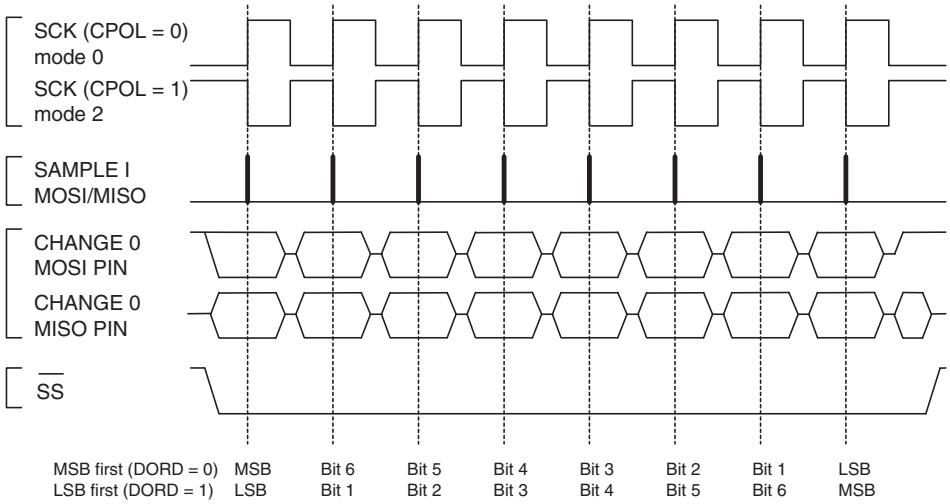
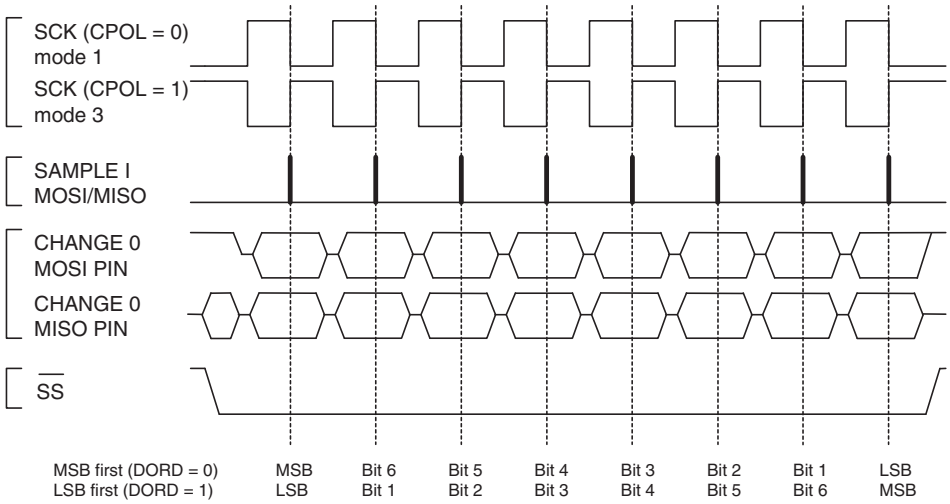


Figure 14-4. SPI transfer format with CPHA = 1.



14.5 $\overline{\text{SPI}}$ registers

14.5.1 SPCR - SPI Control Register

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 14-3 on page 186 and Figure 14-4 on page 186 for an example. The CPOL functionality is summarized below:

Table 14-3. 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 14-3 on page 186 and Figure 14-4 on page 186 for an example. The CPOL functionality is summarized below:

Table 14-4. 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 Table 14-5:

Table 14-5. 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	$f_{clkio}/64$

14.5.2 SPSR - SPI Status Register

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 AT90PWM81/161 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 14-5 on page 187). 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 AT90PWM81/161 is also used for program memory and EEPROM downloading or uploading. See "Serial Programming Algorithm" on page 261 for serial programming and verification.

14.5.3 SPDR - SPI Data Register

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.

15. Voltage Reference and Temperature Sensor

15.1 Features

- **Accurate voltage reference of 2.56V**
- **Internal temperature sensor**
- **Possibility for runtime compensation of temperature drift in both voltage reference and on-chip oscillators**
- **Low power consumption**

15.2 On Chip voltage Reference and Temperature sensor overview

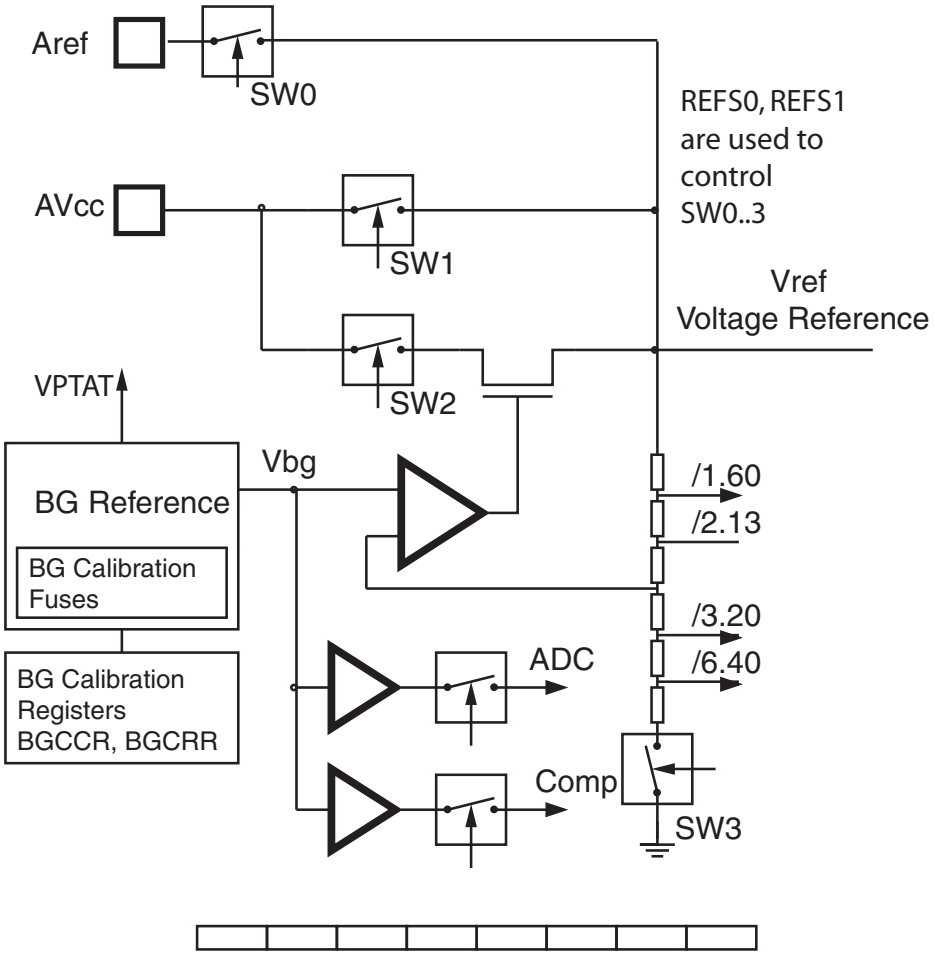
A low power band-gap reference provides AT90PWM81/161 with an accurate On-chip Bandgap voltage of 1.100V (Vbg).

Then when SW1 is off and SW2/SW3 is on, the bandgap voltage is multiplied and generates the internal reference V_{REF} of 2.56V. This reference voltage is used as reference for the ADC, the DAC and can use a buffer with external decoupling capacitor (when SW0 is on) to enable excellent noise performance with minimum power consumption as shown on Figure 15-1 on page 190.

The selection of the Voltage Reference for all the analog components (ADC, DAC, Comparators) is done using the REFS1:0 bits in ADMUX register; see “ADMUX - ADC Multiplexer Register” on page 217.

For conditions using the Bandgap and the internal voltage reference, see “Bandgap and Internal Voltage Reference Enable Signals and Start-up Time” on page 55.

Figure 15-1. Reference circuitry.



AT90PWM81/161 has an On-chip temperature sensor for monitoring the die temperature. A voltage Proportional-To-Absolute-Temperature, VPTAT, is generated in the voltage reference circuit and after buffering, is connected to the ADC multiplexer. This temperature sensor can be used for runtime compensation of temperature drift in both the voltage reference and the On-chip Oscillator. To get the absolute temperature in degrees Kelvin, the measured Vtemp voltage must be scaled with the Vtemp factory calibration value stored in the signature row. See Section “Temperature Measurement”, page 192 for details.

Vbg and Vtemp can be measured with the integrated ADC by selecting the proper ADC channel with ADMUX (see “ADMUX - ADC Multiplexer Register” on page 217).

15.3 Register Description

15.3.1 BGCCR – Bandgap Calibration Current Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	BGCC3	BGCC2	BGCC1	BGCC0	BGCCR
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	1	0	0	0	

- **Bit 7:4 – Res: Reserved Bit**

This bit is reserved for future use.

- **Bit 3:0 – BGCC3:0: BG Calibration of PTAT Current**

These bits are used for trimming of the nominal value of the bandgap reference voltage. These bits are binary coded, so the lowest value for Vbg is reached when BGCC3:0 is 0000 and the maximum value when BGCC3:0 is 1111. The step size is approximately 5mV.

Updating the BGCC bits will affect the BOD detection level. The BOD will react quickly to the new detection level.

15.3.2 BGCRR – Bandgap Calibration Resistor Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	BGCR3	BGCR2	BGCR1	BGCR0	BGCRR
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	1	0	0	0	

- **Bit 7:4 – Res: Reserved Bit**

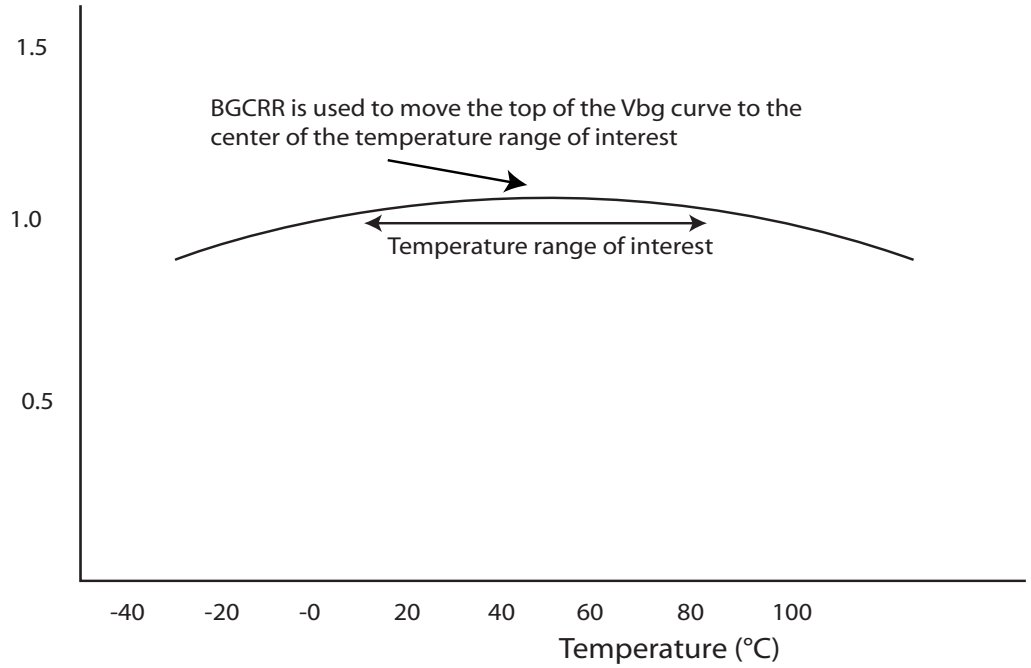
This bit is reserved for future use.

- **Bit 3:0 – BGCR3:0: BG Calibration of Resistor ladder**

These bits are used for temperature gradient adjustment of the bandgap reference. Figure 15-2 on page 192 illustrates Vbg as a function of temperature. Vbg has a positive temperature coefficient at low temperatures and negative temperature coefficient at high temperatures. Depending on the process variations, the top of the Vbg curve may be located at higher or lower temperatures.

To minimize the temperature drift in the temperature range of interest, BGCRR is used to adjust the top of the curve towards the centre of the temperature range of interest. The BGCRR bits are thermometer coded, resulting in 5 possible settings: 0000, 0001, 0011, 0111, 1111. The value 0000 shifts the top of the Vbg curve to the highest possible temperature, and the value 1111 shifts the top of the Vbg curve to the lowest possible temperature.

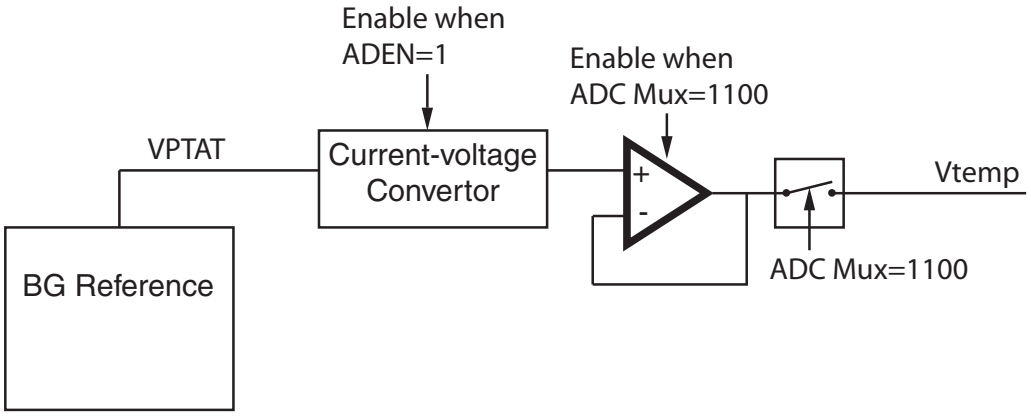
Figure 15-2. Illustration of Vbg as a function of temperature.



15.4 Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC12 channel, as shown on Figure 15-3.

Figure 15-3. Temperature sensor circuitry.



Selecting the ADC12 channel by writing the MUX3..0 bits in ADMUX register to “1100” enables the temperature sensor (see “ADMUX - ADC Multiplexer Register” on page 217). The recommended ADC voltage reference source is the internal 2.56V voltage reference for temperature sensor measurement. When the temperature sensor is enabled, the ADC converter can be used in single conversion mode to measure the voltage over the temperature sensor. The amplifier allows to charge the ADC sample capacitor at full CKadc clock speed. The measured voltage has a linear relationship to temperature as described in Table 15-1 on page 193. When the voltage reference equals 2.56V, the conversion result has approximately a 1LSB/°C (or 2.5mV/°C)

correlation to temperature and the typical accuracy of the temperature measurement is ±10°C after offset calibration.

Table 15-1. Temperature vs. sensor output voltage (typical case).

Temperature	-40°C	25°C	105°C	125°C
Voltage (mV)	600	762		1012
ADC	240	305		405

The values described in Table 15-1 are typical values. However, due to the process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software.

When using temperature sensor, the temperature (in Kelvin) is calculated as follows:

$T = A \times T_{ptat} + B$, where

A - Gain correction multiplier (constant '1', or unsigned fixed point number)

B - Offset correction term (2. complement signed byte)

T_{ptat} - ADC result when measuring temperature sensor voltage, V_{REF} with 2.56V internal reference

T - Temperature in Kelvin (°K = °C + 273)

Example:

If A = 0x80 (=1.00) and B = 8, and ADC result is 0x15E (=350), this gives a measured temperature of:

T = 1.00 × 350 + 8 = 358K (+85°C)

15.4.1 Manufacturing Calibration

One can also use the calibration values available in the signature row. See “Reading the Signature Row from Software” on page 243.

The calibration values are determined from values measured during test at room temperature which is approximatively +25°C. Calibration measures are done at V_{CC} = 3V and with ADC in internal V_{REF} (1.1V) mode.

The temperature in Celsius degrees can be calculated utilizing the formula:

T = ((([(ADCH << 8) | ADCL] -(273 + 25-TSOFFSET)) × TSGAIN)/128) + 25

Where:

a. ADCH & ADCL are the ADC data registers.

b. TSGAIN is the temperature sensor gain (unsigned fixed point 8-bit temperature sensor gain factor in 1/128th units stored as previously in the signature row at address 0x0007). See “Reading the Signature Row from Software” on page 243.

c. TSOFFSET is the temperature sensor offset correction term (signed twos complement 7-bit temperature sensor offset reading stored as previously in the signature row at address 0x0005).

16. Analog Comparator

The Analog Comparator compares the input values on the positive pin ACMPx and negative pin ACMPM or ACMPMx.

16.1 Features

- **Three analog comparators**
- **High speed analog comparators**
- **±25mV or ±10mV or 0 hysteresis**
- **Four reference levels**
- **Generation of configurable interrupts**

16.2 Overview

The AT90PWM81/161 features three fast analog comparators.

Each comparator has a dedicated input on the positive input, and the negative input of each comparator can be configured as:

- A steady value among the four internal reference levels defined by the V_{REF} selected thanks to the REFS1:0 bits in ADMUX register
- A value generated from the internal DAC
- An external analog input ACMPMx

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.

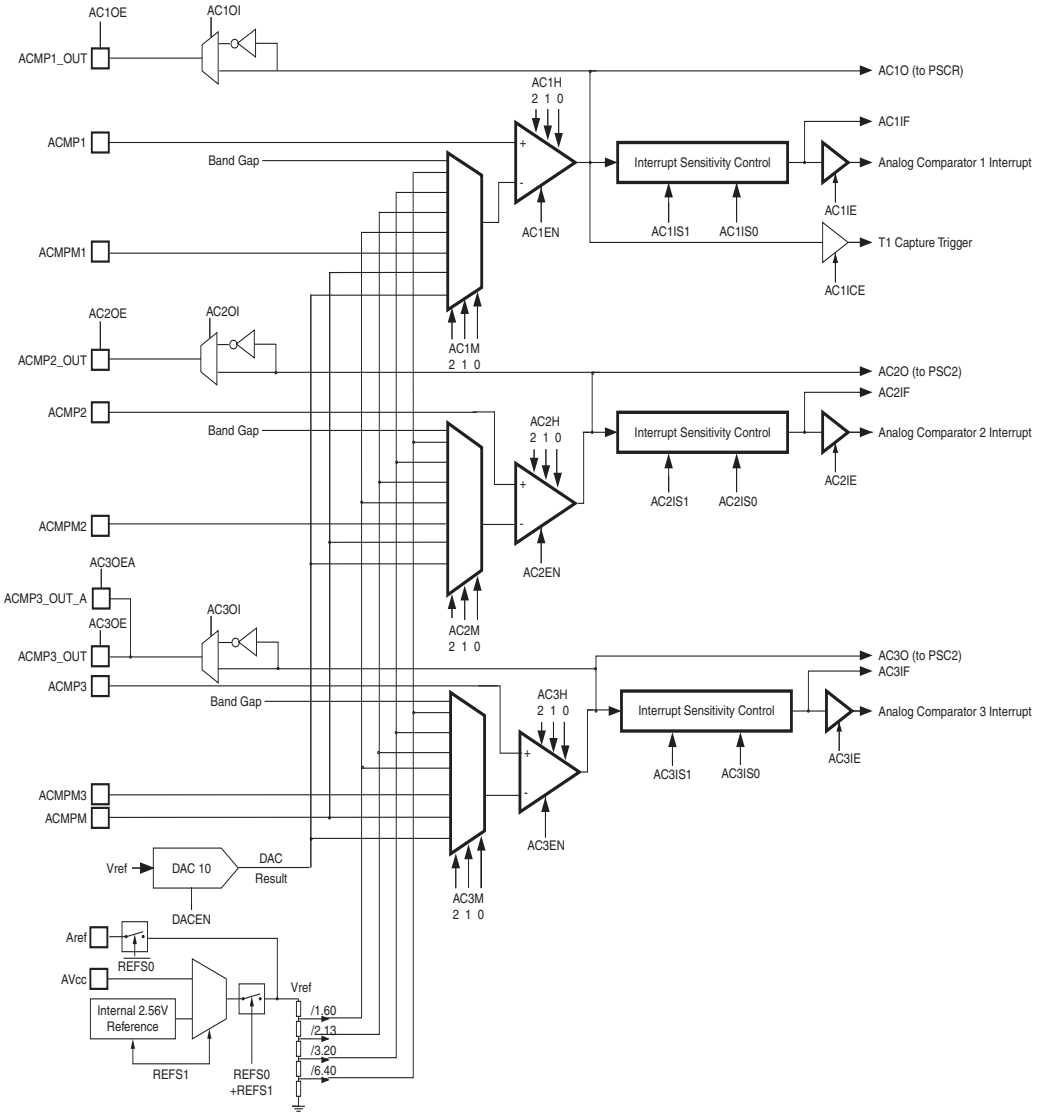
Each comparator can trigger a separate interrupt, exclusive to the Analog Comparator. In addition, the user can select Interrupt triggering on comparator output rise, fall or toggle.

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 comparators and their surrounding logic is shown in Figure 16-1 on page 195.

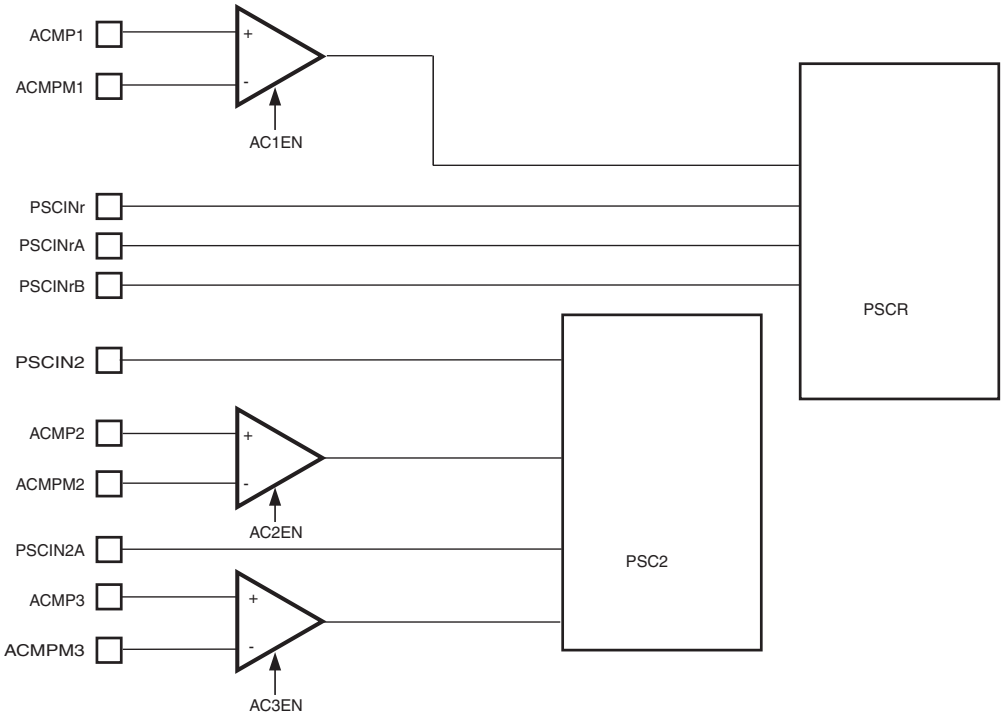
Figure 16-1. Analog comparator block diagram.



Notes: 1. Refer to Figure 2-1 on page 3 and Figure 2-2 on page 4 for Analog Comparator pin placement.

2. The voltage on V_{REF} is defined in Table 17-3, “ADC voltage reference selection.,” on page 218.

Figure 16-2. Comparator PSC links.



16.3 Shared pins between Analog Comparator and ADC

Several Analog comparators input pins can also be used as ADC inputs, so it is possible to measure the comparison voltages. However, when a comparator input is selected as the ADC input, a spike occurs during the sampling phase of the ADC. This may lead to an unwanted transition on the comparator output. So it is a safe software practice to devalidate the comparator output before measuring the voltage on one of the inputs.

16.4 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 three analog comparators.

16.4.1 AC1CON - Analog Comparator 1 Control Register

Bit	7	6	5	4	3	2	1	0	
	AC1EN	AC1IE	AC1IS1	AC1IS0	-	AC1M2	AC1M1	AC1M0	AC1CON
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– AC1EN: Analog Comparator 1 Enable Bit**

Set this bit to enable the analog comparator 1.
Clear this bit to disable the analog comparator 1.

- **Bit 6– AC1IE: Analog Comparator 1 Interrupt Enable bit**

Set this bit to enable the analog comparator 1 interrupt.
Clear this bit to disable the analog comparator 1 interrupt.

- **Bit 5, 4– AC1IS1, AC1IS0: Analog Comparator 1 Interrupt Select bit**

These two bits determine the sensitivity of the interrupt trigger.
The different setting are shown in Table 16-1.

Table 16-1. Interrupt sensitivity selection.

AC1IS1	AC1IS0	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 3– Reserved**

- **Bit 2, 1, 0– AC1M2, AC1M1, AC1M0: Analog Comparator 1 Multiplexer register**

These three bits determine the input of the negative input of the analog comparator.
The different settings are shown in Table 16-2.

Table 16-2. Analog Comparator 1 negative input selection.

AC1M2	AC1M1	AC1M0	Description
0	0	0	“V_{REF}”/6.40
0	0	1	“V_{REF}”/3.20
0	1	0	“V_{REF}”/2.13
0	1	1	“V_{REF}”/1.60
1	0	0	Band gap voltage
1	0	1	DAC result
1	1	0	Analog comparator negative input (ACMPM1 pin)
1	1	1	Analog comparator negative input (ACMPM pin)

16.4.2 AC2CON - Analog Comparator 2 Control Register

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	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 two bits determine the sensitivity of the interrupt trigger.
The different setting are shown in Table 16-3.

Table 16-3. 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 3– Reserved**

- **Bit 2, 1, 0– AC2M2, AC2M1, AC2M0: Analog Comparator 2 Multiplexer register**

These three bits determine the input of the negative input of the analog comparator.
The different setting are shown in Table 16-4.

Table 16-4. Analog Comparator 2 negative input selection.

AC2M2	AC2M1	AC2M0	Description
0	0	0	"V_{REF}"/6.40
0	0	1	"V_{REF}"/3.20
0	1	0	"V_{REF}"/2.13
0	1	1	"V_{REF}"/1.60
1	0	0	Band gap voltage
1	0	1	DAC result
1	1	0	Analog comparator negative input (ACMPM2 pin)
1	1	1	Analog comparator negative input (ACMPM pin)

16.4.3 AC3CON - Analog Comparator 3 Control Register

Bit	7	6	5	4	3	2	1	0	
	AC3EN	AC3IE	AC3IS1	AC3IS0	AC3OEA	AC3M2	AC3M1	AC3M0	AC3CON
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– AC3EN: Analog Comparator 3 Enable Bit**

Set this bit to enable the analog comparator 3.
Clear this bit to disable the analog comparator 3.

- **Bit 6– AC3IE: Analog Comparator 3 Interrupt Enable bit**

Set this bit to enable the analog comparator 3 interrupt.
Clear this bit to disable the analog comparator 3 interrupt.

- **Bit 5, 4– AC3IS1, AC3IS0: Analog Comparator 3 Interrupt Select bit**

These 2 bits determine the sensitivity of the interrupt trigger.
The different setting are shown in Table 16-5.

Table 16-5. Interrupt sensitivity selection.

AC3IS1	AC3IS0	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 3– AC3OEA: Analog Comparator 3 Alternate Output Enable**

Set this bit to enable the analog comparator 3 alternate output pin.
Clear this bit to disable the analog comparator 3 alternate output pin.

- **Bit 2, 1, 0– AC3M2, AC3M1, AC3M0: Analog Comparator 3 Multiplexer register**

These 3 bits determine the input of the negative input of the analog comparator.
The different setting are shown in Table 16-6.

Table 16-6. Analog Comparator 2 negative input selection.

AC3M2	AC3M1	AC3M0	Description
0	0	0	"V_{REF}"/6.40
0	0	1	"V_{REF}"/3.20
0	1	0	"V_{REF}"/2.13
0	1	1	"V_{REF}"/1.60
1	0	0	Band gap voltage
1	0	1	DAC result
1	1	0	Analog comparator negative input (ACMPM3 pin)
1	1	1	Analog comparator negative input (ACMPM pin)

16.4.4 ACnECON - Analog Comparator n Extended Control Register

Bit	7	6	5	4	3	2	1	0	
			ACnOI	ACnOE	AC1ICE	ACnH2	ACnH1	ACnH0	ACnECON
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..6– Reserved**

- **Bit 5– AC1OI: Analog Comparator n Output Invert**

Set this bit to invert the analog comparator n output.
Clear this bit to keep the analog comparator n output.

- **Bit 4– AC1OE: Analog Comparator n Output Enable**

Set this bit to enable the analog comparator n output pin.
Clear this bit to disable the analog comparator n output pin.

- **Bit 3 – AC1ICE: Analog Comparator 1 Interrupt Capture Enable bit**

Set this bit to enable the input capture of the Timer/Counter1 on the analog comparator event. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

In case ICES1 bit (“TCCR1B - Timer/Counter1 Control Register B” on page 97) is set high, the rising edge of AC3O is the capture/trigger event of the Timer/Counter1, in case ICES1 is set to zero, it is the falling edge which is taken into account.

Clear this bit to disable this function. In this case, no connection between the Analog Comparator and the input capture function exists.

- **Bit 2, 1, 0– ACnH2, ACnH1, ACnH0: Analog Comparator n Hysteresis select**

These 3 bits determine the hysteresis value of the analog comparator.
The different setting are shown in Table 16-7.

Table 16-7. Analog cComparator n hysteresis selection.

AC1M2	AC1M1	AC1M0	Description
0	0	0	No hysteresis
0	0	1	Hysteresis + 10mV
0	1	0	Hysteresis - 10mV
0	1	1	Hysteresis ±10mV
1	0	0	Reserved
1	0	1	Hysteresis + 25mV
1	1	0	Hysteresis - 25mV
1	1	1	Hysteresis ±25mV

16.4.5 ACSR - Analog Comparator Status Register

Bit	7	6	5	4	3	2	1	0	
	AC3IF	AC2IF	AC1IF	-	AC3O	AC2O	AC1O	-	ACSR
Read/Write	R/W	R/W	R/W	R/W	-	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– AC3IF: Analog Comparator 3 Interrupt Flag Bit**

This bit is set by hardware when comparator 3 output event triggers off the interrupt mode defined by AC3IS1 and AC3IS0 bits in AC3CON register.
This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC3IE in AC3CON 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 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– AC1IF: Analog Comparator 1 Interrupt Flag Bit**

This bit is set by hardware when comparator 1 output event triggers off the interrupt mode defined by AC1IS1 and AC1IS0 bits in AC1CON register.
This bit is cleared by hardware when the corresponding interrupt vector is executed in case the AC1IE in AC1CON 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 4– Reserved**

- **Bit 3– AC3O: Analog Comparator 3 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 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 1– AC1O: Analog Comparator 1 Output Bit**

AC1O bit is directly the output of the Analog comparator 1.
Set when the output of the comparator is high.
Cleared when the output comparator is low.

- **Bit 0– Reserved**

16.4.6 DIDR0 - Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
	ADC8D ACMP3D	ADC7D AMP0-D	ADC5D ACMP2D	ADC4D ACMP3MD	ADC3D ACMPMD	ADC2D ACMP2MD	ADC1D	ADC0D ACMP1D	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 – ACMPMxD and ACMPxD: ACMPxMD, ACMPxD & APM0+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.

16.4.7 DIDR1 - Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ACMP1MD	AMP0+D	ADC10D	ADC9D	DIDR1
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 3, 0: ACMPxMD, ACMPxD & APM0+ 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.

17. Analog to Digital Converter - ADC

17.1 Features

- **10-bit resolution**
- **0.5LSB integral non-linearity**
- **±2LSB absolute accuracy**
- **8µs - 250µs conversion time**
- **Up to 120kSPS at maximum resolution**
- **11 multiplexed single ended input channels**
- **One 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 2.56V 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**
- **Temperature sensor**

The AT90PWM81/161 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).

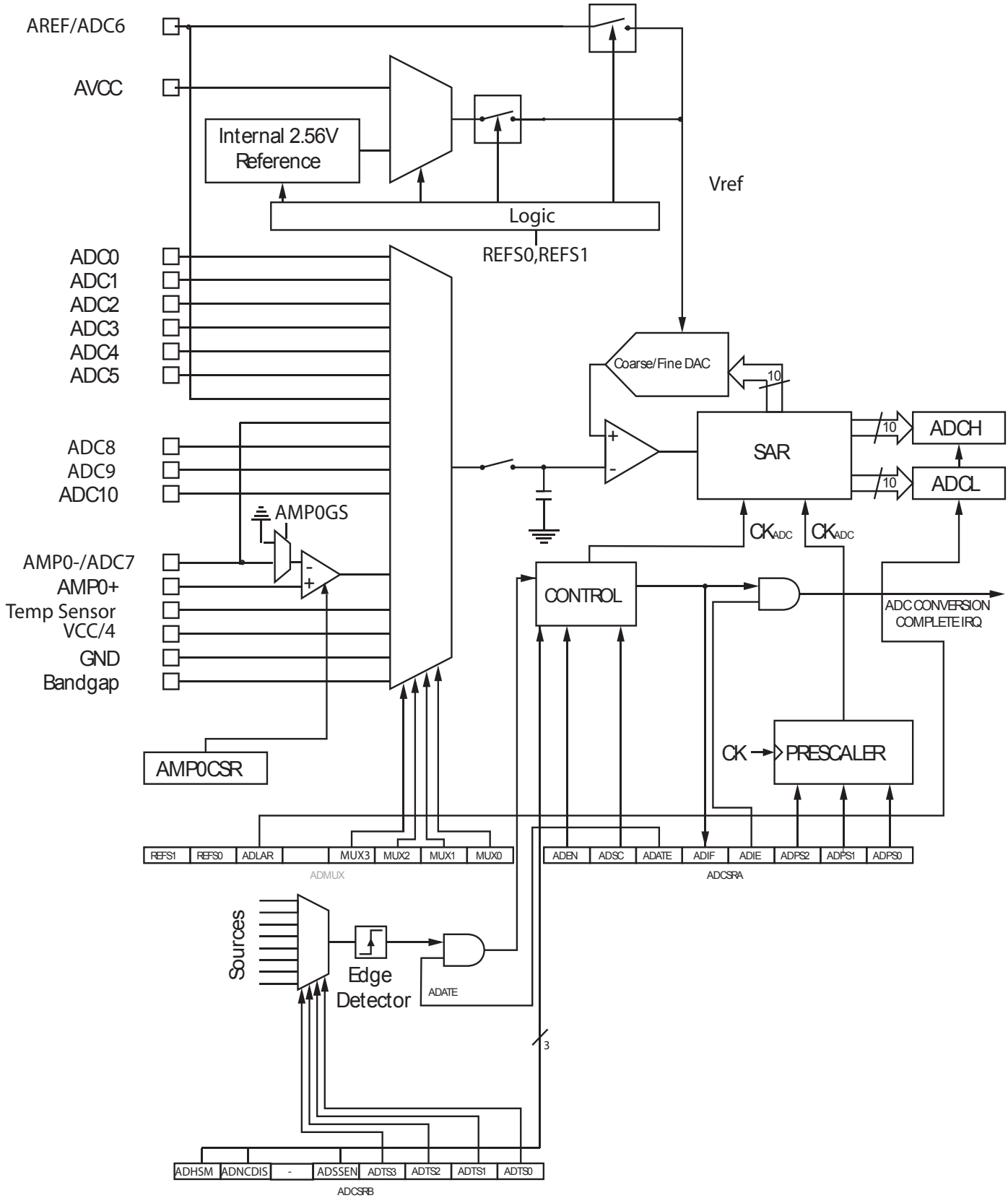
The device also supports 2 differential voltage input combinations which are equipped with a programmable gain stage, providing amplification steps of 14dB (5×), 20dB (10×), 26dB (20×), or 32dB (40×) on the differential input voltage before the A/D conversion. On the amplified channels, 8-bit resolution 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 17-1 on page 204.

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 210 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 17-1. Analog to digital converter block schematic.



17.2 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 1LSB. 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.

17.3 Starting a Conversion

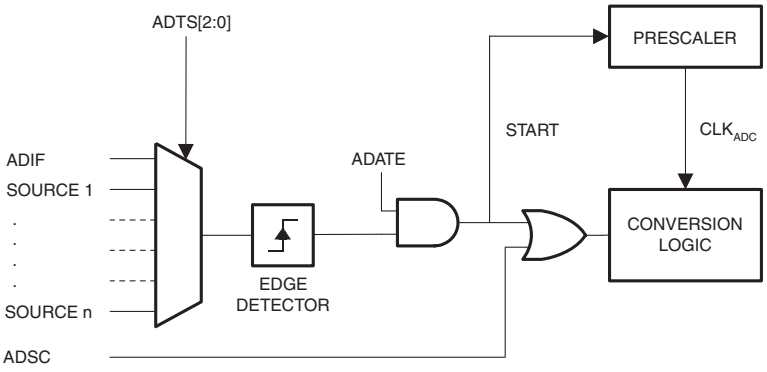
A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the 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.

Triggering from the PSC's synchronization signal is different as there is no flag. In this case, a new conversion is started at each triggering signal. However, a single shot mode can be acti-

vated by setting the bit ADSSEN in ADCSRB register. In this case the synchronization signal is blocked until the ADCH registed is read.

Figure 17-2. ADC auto trigger logic.

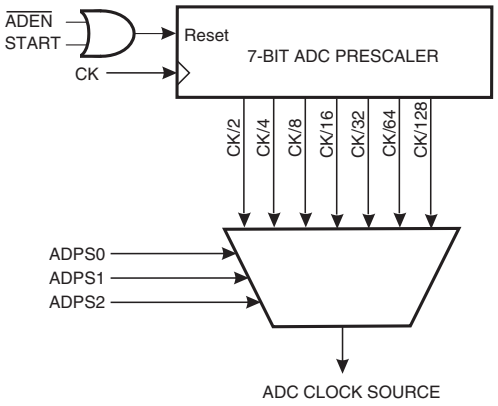


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the 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.

17.4 Prescaling and Conversion Timing

Figure 17-3. ADC prescaler.



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

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA.

The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See “Changing Channel or Reference Selection” on page 209 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 17-1 on page 209.

Figure 17-4. ADC timing diagram, first conversion (single conversion mode).

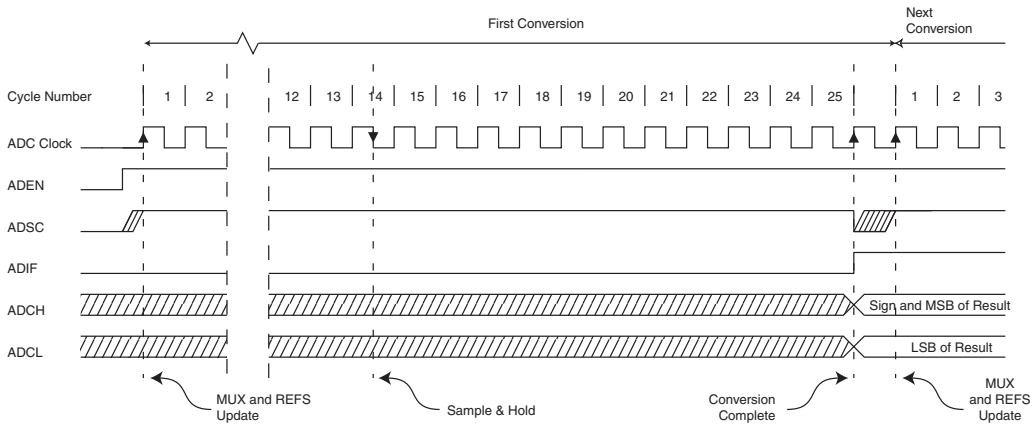


Figure 17-5. ADC timing diagram, single conversion.

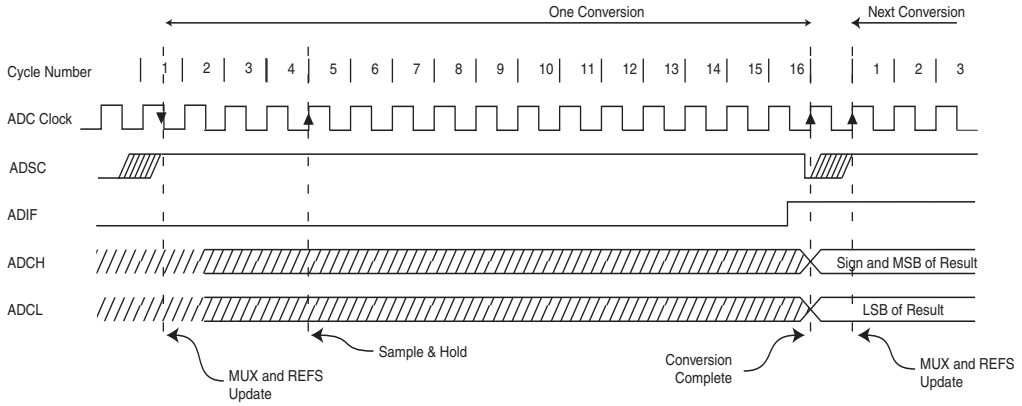


Figure 17-6. ADC timing diagram, auto triggered conversion.

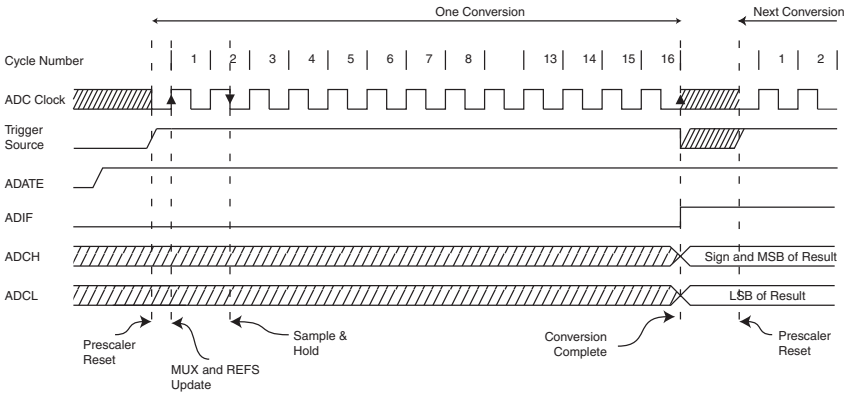


Figure 17-7. ADC timing diagram, free running conversion.

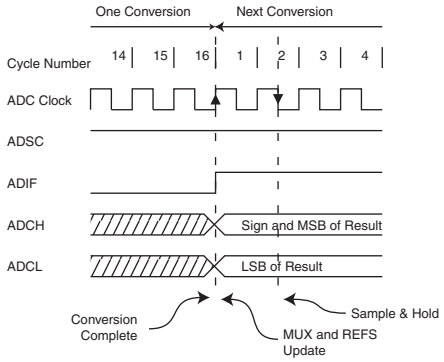


Table 17-1. ADC conversion time.

Condition	First conversion	Normal conversion, single ended	Auto triggered conversion
Sample & hold (cycles from start of conversion)	13.5	3.5	4
Conversion time (cycles)	25	15.5	16

17.5 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:

a. When ADATE or ADEN is cleared.
b. During conversion, minimum one ADC clock cycle after the trigger event.
c. 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.

17.5.1 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

17.5.2 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. If the external AREF pin is connected to the ADC, 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.

The user may switch between $AV_{CC,}$ AREF pin 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.

17.6 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 the ADNCDIS bit is reset
b. Make sure the ADATE bit is reset
c. Make sure that the ADC is enabled and is not busy converting (ADSC reset). Single Conversion mode must be selected and the ADC conversion complete interrupt must be enabled
d. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted
e. 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

Another possible procedure is possible for Auto trigger conversions:

a. Make sure the ADNCDIS bit is set
b. Make sure the ADATE bit is set
c. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion on the next triggering event
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.

17.6.1 Analog Input Circuitry

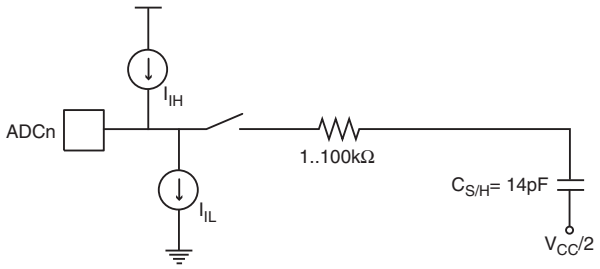
The analog input circuitry for single ended channels is illustrated in Figure 17-8 An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive 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 5kW 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, witch 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 kW 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 17-8. Analog input circuitry.



17.6.2 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:

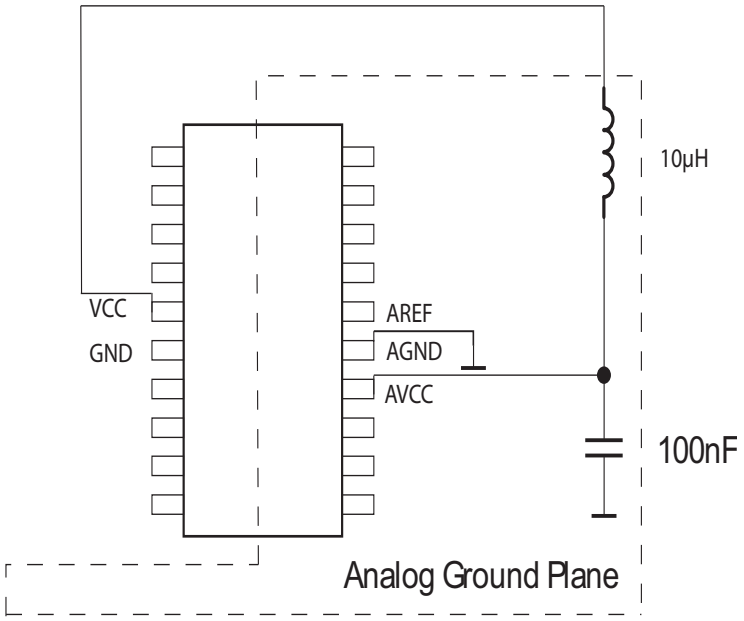
a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. 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 17-9.

c. Use the ADC noise canceler function to reduce induced noise from the CPU.

d. 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 17-9. ADC power connections.



17.6.3 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 “AMP0CSR - Amplifier 0 Control and Status register” on page 225). 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.

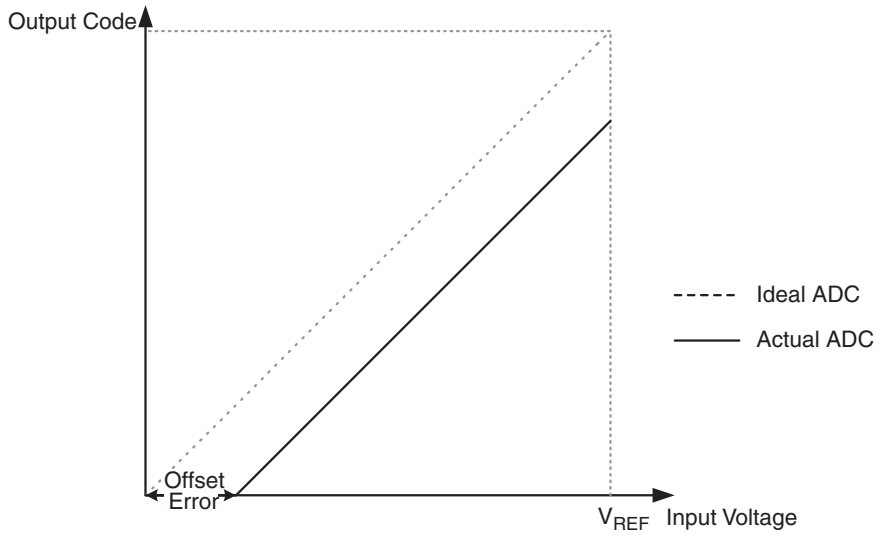
17.6.4 ADC Accuracy Definitions

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:

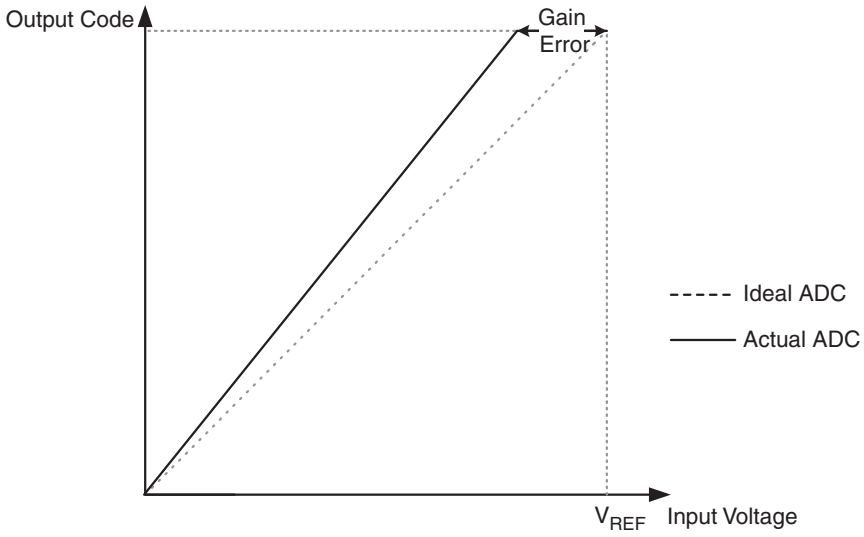
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5LSB). Ideal value: 0LSB

Figure 17-10. Offset error.



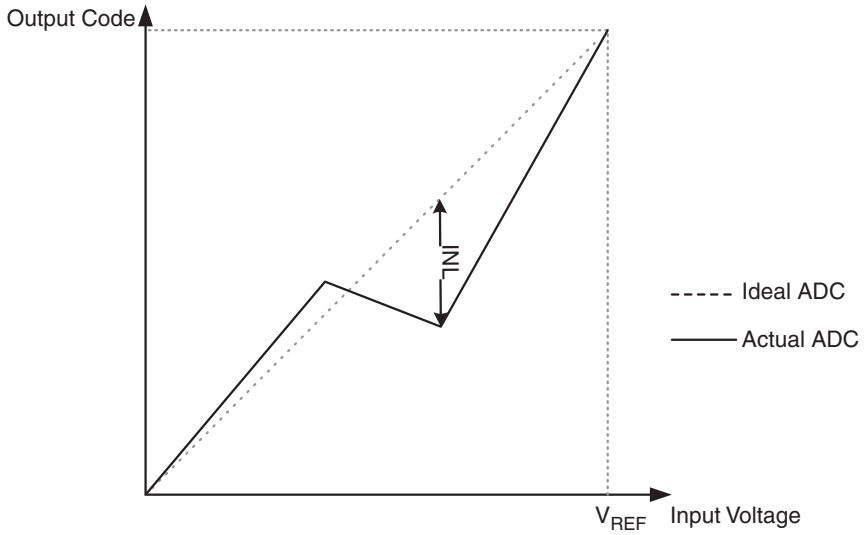
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5LSB below maximum). Ideal value: 0LSB

Figure 17-11. Gain error.



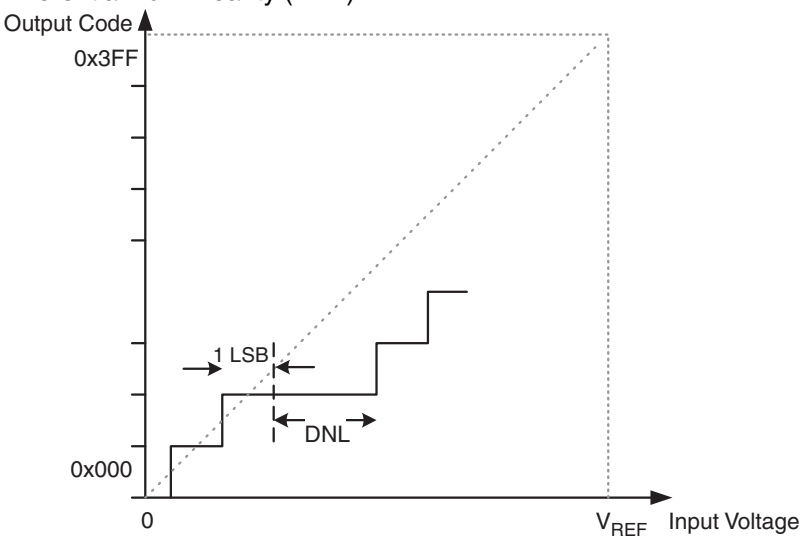
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0LSB

Figure 17-12. 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 (1LSB). Ideal value: 0LSB

Figure 17-13. 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 (1LSB wide) will code to the same value. Always ±0.5LSB
- 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: ±0.5LSB

17.7 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 17-3 on page 218 and Table 17-4 on page 218). 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 17-14 on page 216 shows the decoding of the differential input range.

Table 17-2 on page 217 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a reference voltage of V_{REF}.

Figure 17-14. Differential measurement range.

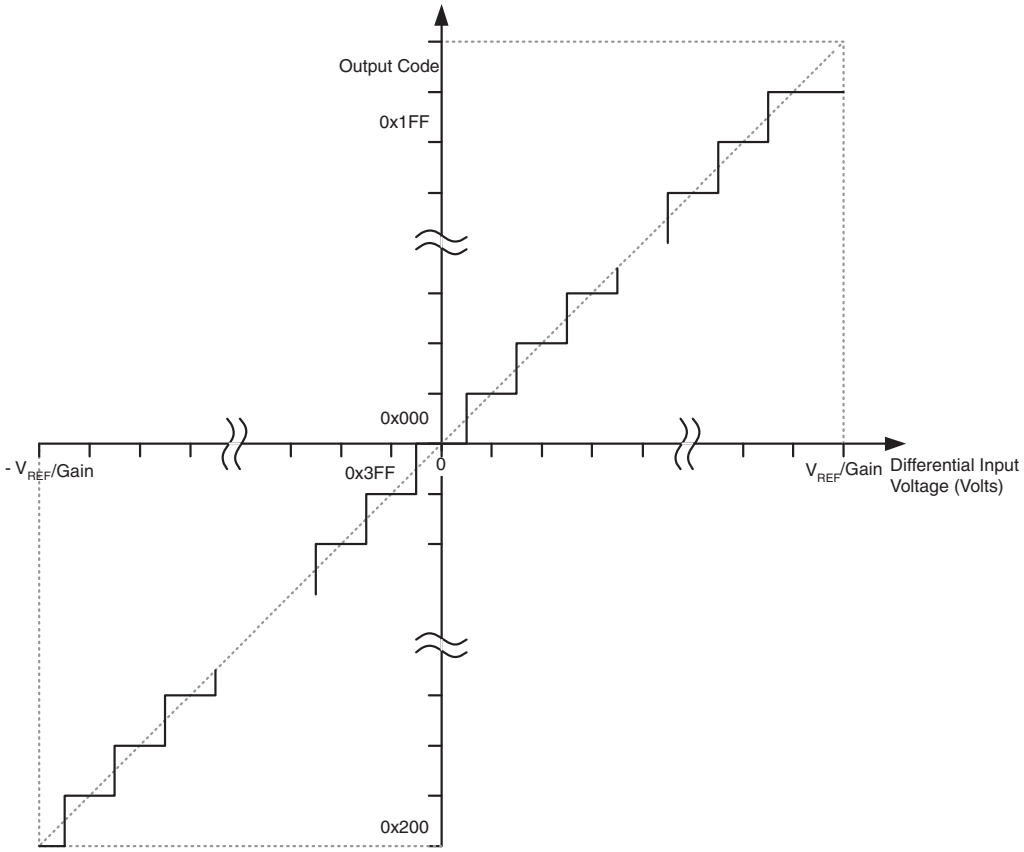


Table 17-2. Correlation between input voltage and output codes.

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, 10× gain, 2.56V reference, left adjusted result)
– Voltage on ADC3 is 300mV, voltage on ADC2 is 500mV
– 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, 1× gain, 2.56V reference, left adjusted result)
– Voltage on ADC3 is 300mV, voltage on ADC2 is 500mV
– 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

17.8 ADC Register Description

The ADC of the AT90PWM81/161 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 V_{REF} 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.

17.8.1 ADMUX - ADC Multiplexer Register

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	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – REFS1, 0: ADC V_{REF} Selection Bits**

These 2 bits determine the voltage reference for the ADC and for the other analog devices. The different setting are shown in Table 17-3.

Table 17-3. ADC voltage reference selection.

REFS1	REFS0	Description	
		Voltage reference	**PE3/AREF pin**
0	0	External V_{REF}	External voltage reference
0	1	AV_{CC}	
1	0	Internal 2.56V reference voltage	External capacitor for decoupling of the internal reference voltage
1	1	Internal 2.56V reference voltage	PE3 pin free as port

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 V_{REF} 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 “ADCH and ADCL - ADC Result Data Registers”, page 221.

- **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 17-4.

Table 17-4. 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	ADC8
1	0	0	1	ADC9
1	0	1	0	ADC10
1	0	1	1	AMP0

Table 17-4. ADC input channel selection. (Continued)

MUX3	MUX2	MUX1	MUX0	Description
1	1	0	0	Temp sensor (Vtemp)
1	1	0	1	$V_{CC}/4$
1	1	1	0	Bandgap (Vbg)
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).

17.8.2 ADCSRA - ADC Control and Status Register A

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.

- **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 17-6 on page 221.

- **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 17-5 on page 220.

Table 17-5. 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

17.8.3 ADCSRB - ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	ADHSM	ADNCDIS	-	ADSSEN	ADTS3	ADTS2	ADTS1	ADTS0	ADCSRB
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 – 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 6 – ADNCDIS: ADC Noise Canceller Disable**

Set this bit to disable automatic ADC start when entering Idle or ADC Noise reduction Modes.
Clear it to enable automatic ADC start when entering Idle or ADC reduction Modes.
The ADNCDIS must be set before entering Idle or ADC Noise reduction Modes if the ADC is running or Auto triggered to prevent false ADC restart.

- **Bit 5 – Reserved**

- **Bit 4 – ADSSEN: ADC Single Shot Enable on PSC's synchronization signals**

Set this bit to enable single shot mode when auto trigger on PSCRASY & PSC2ASY. In this case a single conversion will be performed and PSCRASY & PSC2ASY will be blocked until ADCH reading.
Clear it to enable continuous conversion on PSCRASY & PSC2ASY auto triggering.

- **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 17-6 on page 221, 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, if ADSSEN is reset, a conversion will start each time the trig event appears and the previous conversion is completed.

Table 17-6. ADC auto trigger source selection.

ADTS3	ADTS2	ADTS1	ADTS0	Description
0	0	0	0	Free running mode
0	0	0	1	Analog comparator 1
0	0	1	0	External Interrupt Request 0
0	0	1	1	Timer/Counter1 overflow
0	1	0	0	Timer/Counter1 capture event
0	1	0	1	PSCRASY event
0	1	1	0	PSC2ASY event
0	1	1	1	Analog comparator 2
1	0	0	0	Analog comparator 3
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

17.8.4 ADCH and ADCL - ADC Result Data Registers

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.

17.8.4.1 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	

17.8.4.2 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	

17.8.5 DIDR0 - Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
	ADC8D ACMP3D	ADC7D AMP0-D	ADC5D ACMP2D	ADC4D ACMP3MD	ADC3D ACMPMD	ADC2D ACMP2MD	ADC1D	ADC0D ACMP1D	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: AMP0-D 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.

17.8.6 DIDR1 - Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	ACMP1MD	AMP0+D	ADC10D	ADC9D	DIDR1
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 2:0 – AMP0+D 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.

17.9 Amplifier

The AT90PWM81/161 features one differential amplified channel with programmable 5, 10, 20, and 40 gain stage. Despite the result is given by the 10bit ADC, the amplifier has been sized to give a 8bits resolution.

The negative input on the amplifier can be internally switched to the analog ground. However, amplifier characteristics are specified with differential inputs.

Because the amplifier is a switching capacitor amplifier, it needs to be clocked by a synchronization signal called in this document the amplifier synchronization clock. The amplifier samples the input value at the falling edge of the synchronization signal. This allow to measure analog signals with same period as the synchronization. The maximum clock for the amplifier is 250kHz.

To ensure an accurate result in case of large voltage change, the amplifier input needs to have a quite stable sampled input value during at least four Amplifier synchronization clock periods.

Amplified conversions can be synchronized to PSC events (see “Synchronization source description in one/two/four ramp modes.” on page 134 and “Synchronization source description in centered mode.” on page 135) or to the internal clock CK_{ADC} equal to eighth the ADC clock frequency. In case the synchronization is done by the ADC clock divided by eight, this synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2}. A conversion initiated by the user (that is, all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism.

The normal way to use the amplifier is to select a synchronization clock via the AMPxTS1:0 bits in the AMPxCSR register. Then the amplifier can be switched on, and the amplification is done on each synchronization event. The amplification is done independently of the ADC.

In order to start an amplified Analog to Digital Conversion on the amplified channel, the ADMUX must be configured as specified on Table 17-4 on page 218.

The ADC starting is done by setting the ADSC (ADC Start conversion) bit in the ADCSRB register.

Until the conversion is not achieved, it is not possible to start a conversion on another channel.

On AT90PWM81/161, conversion takes advantage of the amplifier characteristics to ensure minimum conversion time.

As soon as a conversion is requested thanks to the ADSC bit, the Analog to Digital Conversion is started. In order to have a better understanding of the functioning of the amplifier synchronization, a timing diagram example is shown Figure 17-15 on page 224.

In case the amplifier output is modified during the sample phase of the ADC, the on-going conversion is aborted and restarted as soon as the output of the amplifier is stable as shown Figure 17-16 on page 224.

The only precaution to take is to be sure that the trig signal (PSC) frequency is lower than ADCclk/4.

It is also possible to auto trigger conversion on the amplified channel. In this case, the conversion is started at the next amplifier clock event following the last auto trigger event selected thanks to the ADTS bits in the ADCSRB register. In auto trigger conversion, the free running mode is not possible unless the ADSC bit in ADCSRA is set by soft after each conversion.

Figure 17-15. Amplifier synchronization timing diagram with change on analog input signal.

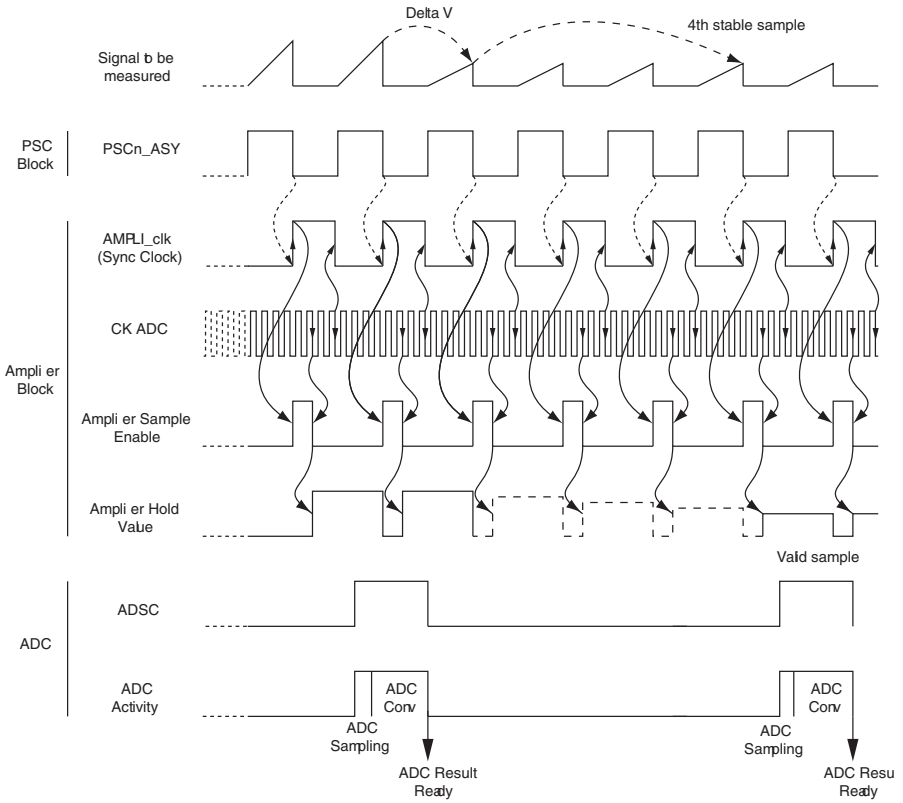
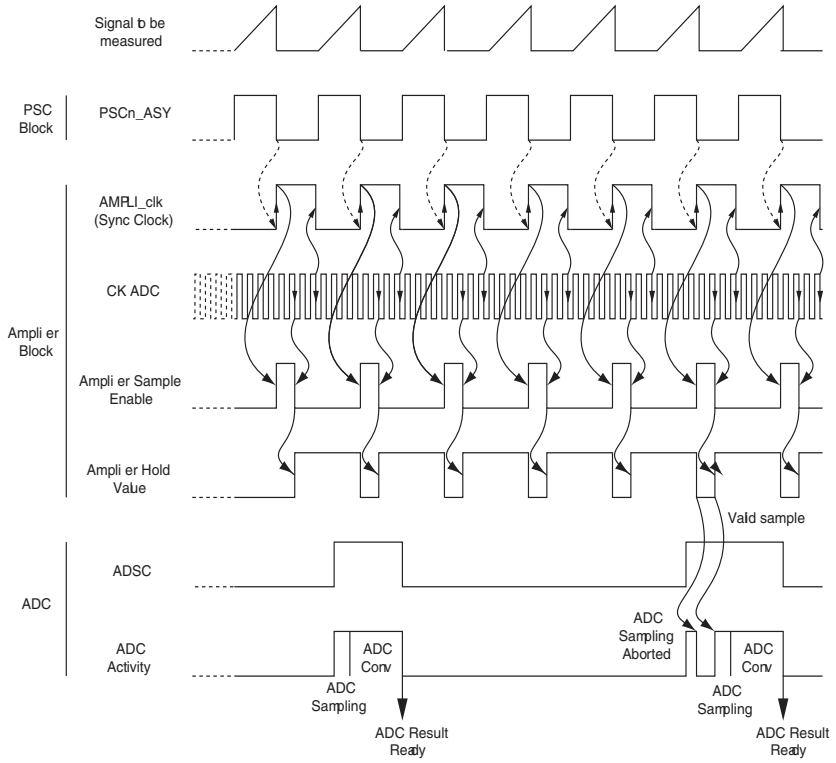
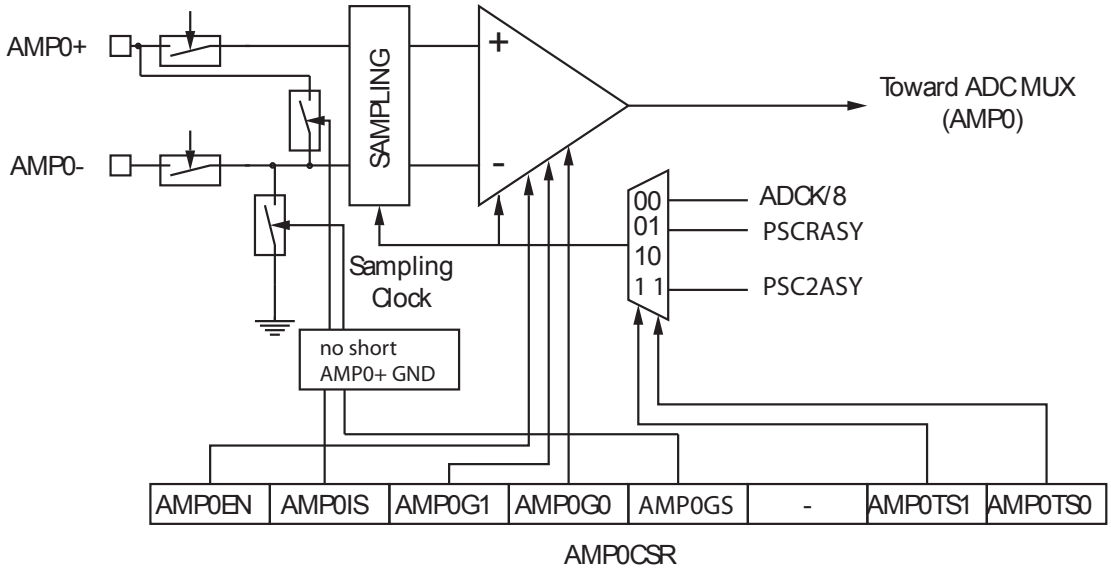


Figure 17-16. Amplifier synchronization timing diagram: behavior when ADSC is set when the amplifier output is changing.



The block diagram of the two amplifiers is shown on Figure 17-17.

Figure 17-17. Amplifiers block diagram.



If APMP0GS bit is set, the AMP0- input is open and PD5/AMP0- pin is free for another use. At the same time the negative input of the Amplifier is internally grounded.

17.10 Amplifier Control Registers

The configuration of the amplifier is controlled via the register AMP0CSR. 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 least significant bits.

17.10.1 AMP0CSR - Amplifier 0 Control and Status register

Bit	7	6	5	4	3	2	1	0	
	AMP0EN	AMP0IS	AMP0G1	AMP0G0	AMP0GS	-	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.

- **Bit 6– AMP0IS: Amplifier 0 Input Shunt**

Set this bit to short-circuit the Amplifier 0 input. If AMP0GS is set, the ground switch is released during shunt of inputs.
Clear this bit to normally use the Amplifier 0.

- **Bit 5, 4– AMP0G1, 0: Amplifier 0 Gain Selection Bits**

These two bits determine the gain of the amplifier 0.
The different setting are shown in Table 17-7.

Table 17-7. 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 four Amplifier synchronization clock periods.

- **Bit 3– AMP0GS: Amplifier 0 Ground Select of AMP0**

This bit select negative input of the amplifier:

Set this bit to ground the Amplifier 0 negative input.
Clear this bit to normally use the Amplifier 0 differential input.

- **Bit 1, 0– AMP0TS1, AMP0TS0: Amplifier 0 Trigger Source Selection Bits**

In accordance with the Table 17-8, these two 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 17-8. AMP0 auto trigger source selection.

AMP0TS1	AMP0TS0	Description
0	0	Auto synchronization on ADC Clock/8
0	1	Trig on PSCRASY
1	0	
1	1	Trig on PSC2ASY

18. Digital to Analog Converter - DAC

18.1 Features

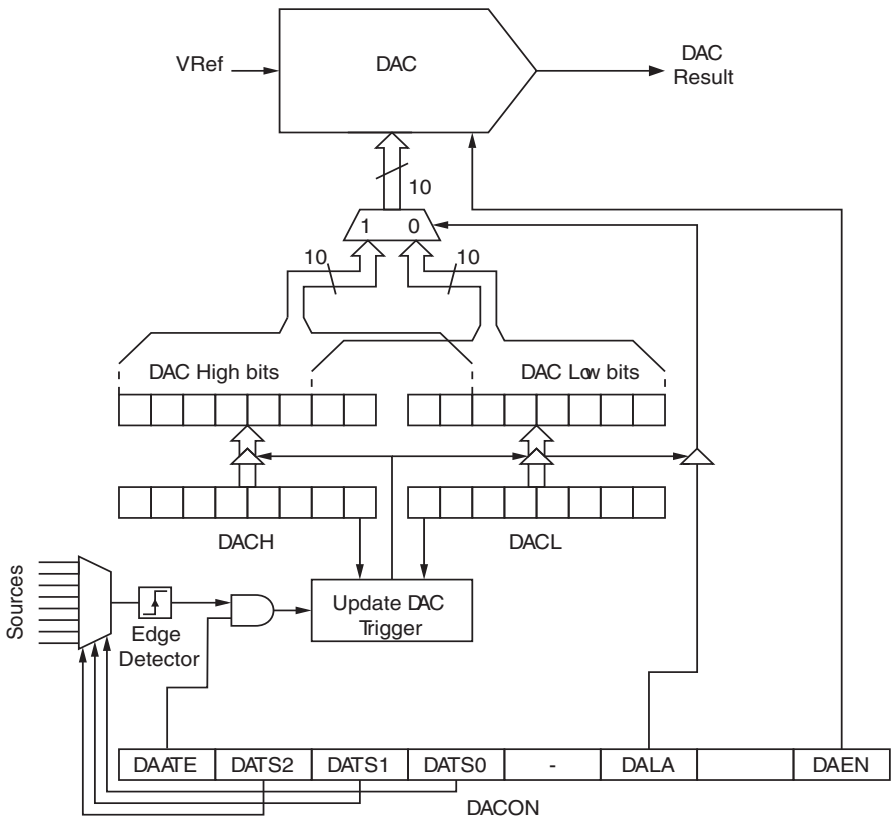
- **10 bits resolution**
- **8 bits linearity**
- **±0.5LSB accuracy between 100mV and AV_{CC} - 100mV**
- **V_{OUT} = DAC × V_{REF}/1023**
- **The DAC could be connected to the negative inputs of the analog comparators**

The AT90PWM81/161 features a 10-bit Digital to Analog Converter. This DAC can be used for the analog comparators.

The DAC 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 210 on how to connect this pin.

The reference voltage is the same as the one used for the ADC. See "ADMUX - ADC Multiplexer Register" on page 217. These nominally 2.56V V_{REF} or AV_{CC} are provided On-chip. The voltage reference may be externally decoupled at the A_{REF} pin by a capacitor for better noise performance.

Figure 18-1. Digital to analog converter block schematic.



18.2 Operation

The Digital to Analog Converter generates an analog signal proportional to the value of the DAC registers value.

In order to have an accurate sampling frequency control, there is the possibility to update the DAC input values through different trigger events.

18.3 Starting a Conversion

The DAC is configured thanks to the DACON register. As soon as the DAEN bit in DACON register is set, the DAC converts the value present on the DACH and DACL registers in accordance with the register DACON setting.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the DAC Auto Trigger Enable bit, DAATE in DACON. The trigger source is selected by setting the DAC Trigger Select bits, DATS in DACON (see description of the DATS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the DAC converts the value present on the DACH and DACL registers in accordance with the register DACON setting. 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.

18.3.1 DAC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the DAC. V_{REF} can be selected as either AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} is connected to the DAC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. When the external AREF pin is connected to the DAC, 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 impedance voltmeter. Note that V_{REF} is a high impedance source, and only a capacitive load should be connected in a system.

The user may switch between $AV_{CC,}$ AV_{CC} and 2.56V as reference selection. The first DAC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

18.4 DAC Register Description

The DAC is controlled via three dedicated registers:

- The DACON register which is used for DAC configuration
- DACH and DACL which are used to set the value to be converted

18.4.1 DACON - Digital to Analog Conversion Control Register

Bit	7	6	5	4	3	2	1	0	
	DAATE	DATS2	DATS1	DATS0	-	DALA	-	DAEN	DACON
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 – DAATE: DAC Auto Trigger Enable bit (not useful, may be left for compatibility)**

Set this bit to update the DAC input value on the positive edge of the trigger signal selected with the DACTS2-0 bit in DACON register.
Clear it to automatically update the DAC input when a value is written on DACH register.

- **Bit 6:4 – DATS2, DATS1, DATS0: DAC Trigger Selection bits (not useful, may be left for compatibility)**

These bits are only necessary in case the DAC works in auto trigger mode. It means if DAATE bit is set.

In accordance with the Table 18-1, these 3 bits select the interrupt event which will generate the update of the DAC input values. The update will be generated by the rising edge of the selected interrupt flag whether the interrupt is enabled or not.

Table 18-1. DAC auto trigger source selection.

DATS2	DATS1	DATS0	Description
0	0	0	Analog comparator 0
0	0	1	Analog comparator 1
0	1	0	External Interrupt Request 0
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

- **Bit 2 – DALA: Digital to Analog Left Adjust**

Set this bit to left adjust the DAC input data.
Clear it to right adjust the DAC input data.
The DALA bit affects the configuration of the DAC data registers. Changing this bit affects the DAC output on the next DACH writing.

- **Bit 1 – Reserved**

- **Bit 0 – DAEN: Digital to Analog Enable bit**

Set this bit to enable the DAC.
Clear it to disable the DAC.

18.4.2 DACH and DACL - Digital to Analog Converter input Register

DACH and DACL registers contain the value to be converted into analog voltage.

Writing the DACL register forbid the update of the input value until DACH has not been written too. So the normal way to write a 10-bit value in the DAC register is firstly to write DACL the DACH.

In order to work easily with only 8 bits, there is the possibility to left adjust the input value. Like this it is sufficient to write DACH to update the DAC value.

18.4.2.1 DALA = 0

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	DAC9	DAC8	DACH
	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0	DACL
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	

18.4.2.2 DALA = 1

Bit	7	6	5	4	3	2	1	0	
	DAC9	DAC8	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DACH
	DAC1	DAC0	-	-	-	-	-	-	DACL
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	

To work with the 10-bit DAC, two registers have to be updated. In order to avoid intermediate value, the DAC input values which are really converted into analog signal are buffering into unreachable registers. In normal mode, the update of the shadow register is done when the register DACH is written.

In case DAATE bit is set, the DAC input values will be updated on the trigger event selected through DATS bits.

In order to avoid wrong DAC input values, the update can only be done after having written respectively DACL and DACH registers. It is possible to work on 8-bit configuration by only writing the DACH value. In this case, update is done each trigger event.

In case DAATE bit is cleared, the DAC is in an automatic update mode. Writing the DACH register automatically update the DAC input values with the DACH and DACL register values.

It means that whatever is the configuration of the DAATE bit, changing the DACL register has no effect on the DAC output until the DACH register has also been updated. So, to work with 10 bits, DACL must be written first before DACH. To work with 8-bit configuration, writing DACH allows the update of the DAC.

19. debugWIRE On-chip Debug System

19.1 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**

19.2 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.

19.3 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 19-1. The debugWIRE setup.

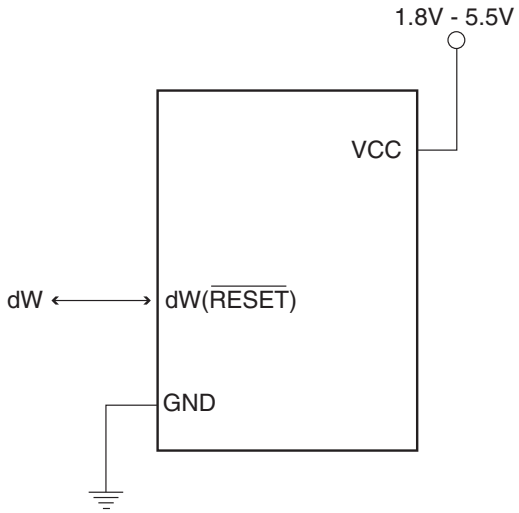


Figure 19-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

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 10kW. 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

19.4 Software Break Points

The 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.

19.5 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, that is, 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.

19.6 debugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

19.6.1 DWDR - debugWire Data Register

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.

20. Boot Loader Support – Read-While-Write Self-Programming

In AT90PWM81/161, 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 configured 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.

20.1 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 21-12 on page 254) used during programming. The page organization does not affect normal operation.

20.2 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 20-2 on page 236). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 20-7 on page 246 and Figure 20-2 on page 236. These two sections can have different level of protection since they have different sets of Lock bits.

20.2.1 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 20-2 on page 237. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

20.2.2 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 20-3 on page 237.

20.3 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 configured 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 20-9 on page 247 and Figure 20-2 on page 236. 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.

20.3.1 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 (that is, 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 “SPMCSR - Store Program Memory Control and Status Register” on page 238 for details on how to clear RWWSB.

20.3.2 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 20-1. 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 20-1. Read-while-write vs. no read-while-write.

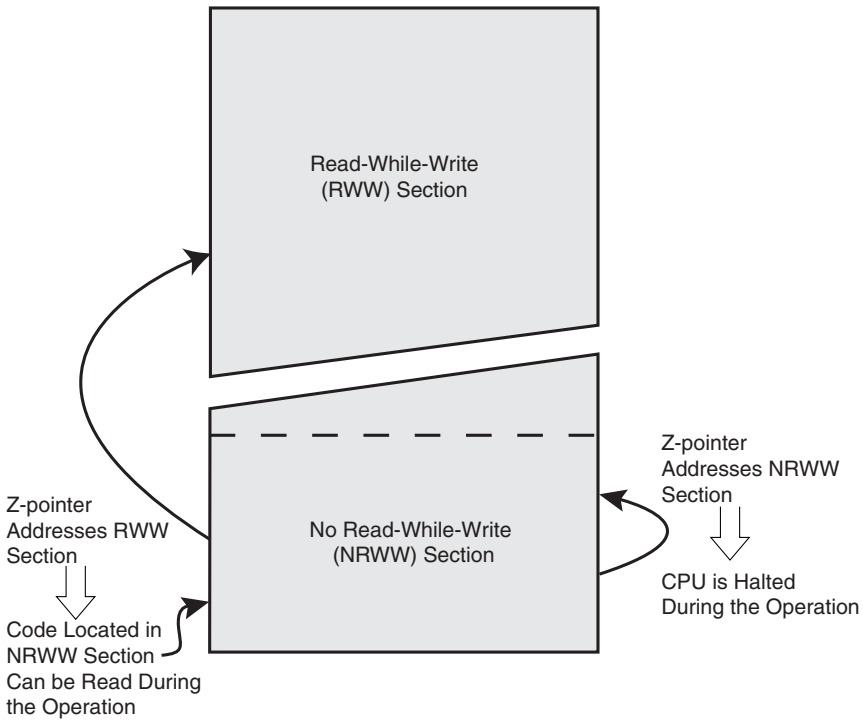
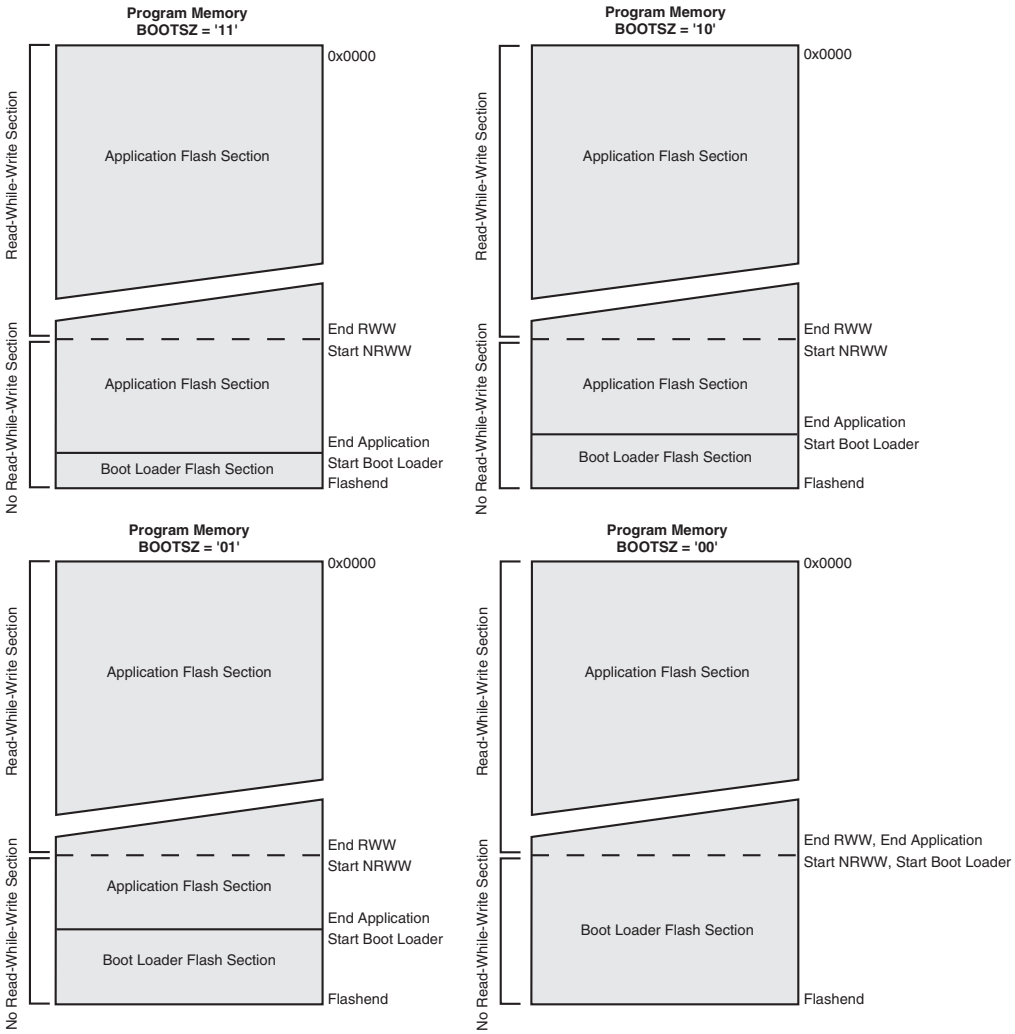


Figure 20-2. Memory sections.



Note: 1. The parameters in the figure above are given in Table 20-7 on page 246.

20.4 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 20-2 on page 237 and Table 20-3 on page 237 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 20-2. 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 20-3. 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.

20.5 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 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 20-4. Boot reset fuse [(1)].

BOOTRST	Reset address
1	Reset vector = application reset (address 0x0000)
0	Reset vector = boot loader reset (see Table 20-7 on page 246)

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

20.5.1 SPMCSR - Store Program Memory Control and Status Register

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	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R/W	R	R/W	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 – SIGRD: Signature Row Read**

If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. See “Reading the Signature Row from Software” on page 243 for details.

An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect. This operation is reserved for future use and should not be used.

- **Bit 4 – RWWSRE: Read-While-Write Section Read Enable**

When programming (Page Erase or Page Write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a Page Erase or a Page Write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits 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 242 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.

20.6 Addressing the Flash During Self-Programming

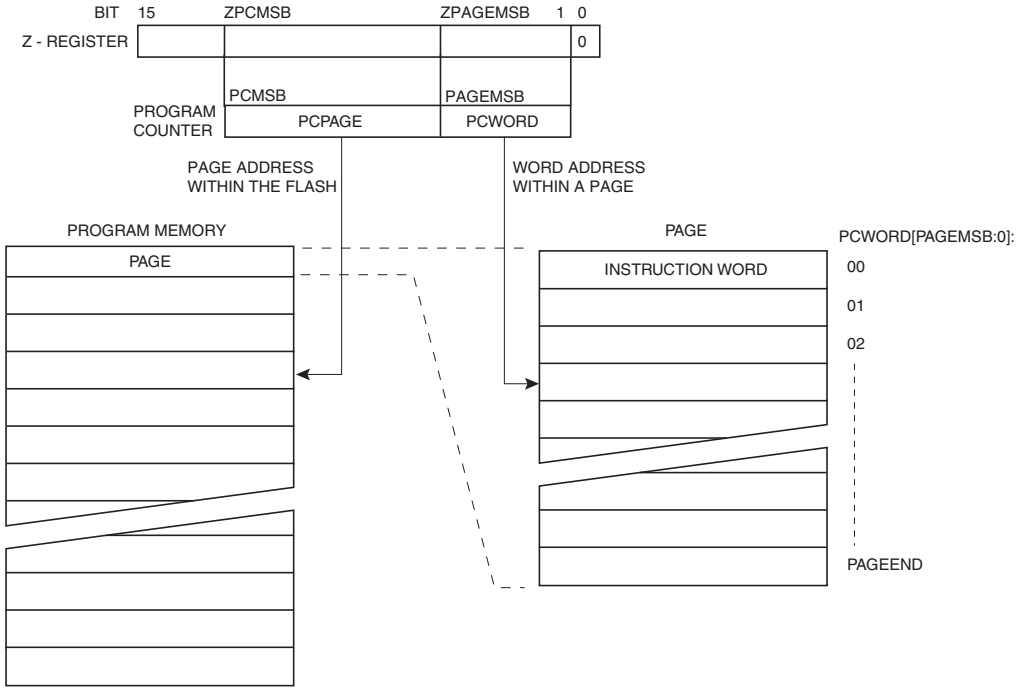
The Z-pointer is used to address the SPM commands.

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 21-12 on page 254), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 20-3 on page 240. 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 20-3. Addressing the flash during SPM [1].



Note: 1. The different variables used in Figure 20-3 are listed in Table 20-10 on page 247.

20.7 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 245 for an assembly code example.

20.7.1 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

20.7.2 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.

20.7.3 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

20.7.4 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 Section “Moving Interrupts Between Application and Boot Space”, page 65.

20.7.5 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.

20.7.6 Prevent Reading the RWW Section During Self-Programming

During Self-Programming (either Page Erase or Page Write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the self programming operation. The RWWSB in the SPMCSR will be set as long as the RWW section is busy. During Self-Programming the Interrupt Vector table should be moved to the BLS as described in Section “Moving Interrupts Between Application and Boot Space”, page 65, 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 245 for an example.

20.7.7 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 20-2 on page 237 and Table 20-3 on page 237 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.

20.7.8 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.

20.7.9 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 21-4 on page 249 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 21-5 on page 250 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 21-4 on page 249 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.

20.7.10 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 20-5 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the Signature Row Lock bits or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the "AVR Instruction Set" description.

Table 20-5. Signature row addressing.

Signature Byte	Address	AT90PWM81 data	AT90PWM161 data
Device ID 0, manufacturer ID	0x00	1EH	1EH
OSCAL 8M, RC-OSC calibration	0x01	XXH	XXH
Device ID 1, flash size	0x02	93H	94H
Reserved	0x03	XXH	XXH
Device ID 2, device	0x04	88H	8BH
Temperature sensor offset : TSOFFSET	0x05	XXH	XXH
Reserved	0x06	XXH	XXH
Temperature sensor gain : TSGAIN (1)	0x07	XXH	XXH
Lot number at sort, byte 2, ASCII	0x0E	XXH	XXH
Lot number at sort, Byte 1, ASCII (most left lot#)	0x0F	XXH	XXH
Lot number at sort, byte 2, ASCII	0x10	XXH	XXH
Lot number at sort, Byte 1, ASCII	0x11	XXH	XXH
Lot number at sort, byte 2, ASCII	0x12	XXH	XXH
Lot number at sort, Byte 1, ASCII	0x13	XXH	XXH
Final test Amb V_{REF}: LOW BYTE (2)	0x3C	XXH	XXH

Table 20-5. Signature row addressing. (Continued)

Signature Byte	Address	AT90PWM81 data	AT90PWM161 data
Final test amb V_{REF}: HIGH BYTE (3)	0x3D	XXH	XXH
Final test hot V_{REF}: LOW BYTE (only a Read) (4)	0x3E	XXH	XXH
Final test hot V_{REF}: HIGH BYTE (only a Read) (5)	0x3F	XXH	XXH

Note:
1. TSGAIN typical value is 0x80=128
2. See Note 3
3. Final Test Amb V_{REF} HIGH BYTE and LOW BYTE:
   Typical values are for V_{REF} = 2.56V:
   HIGH BYTE = 0x0A
   LOW BYTE = 0x00
   This means:
   Final Test Amb V_{REF} = 0x0A00 = 2560 = V_{REF} × 1000
4. See Note 3 which details the value format
5. See Note 3 which details the value format

20.7.11 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.

20.7.12 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 20-6 shows the typical programming time for Flash accesses from the CPU.

Table 20-6. SPM programming time.

Symbol	Min. programming time	Max. programming time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7ms	4.5ms

20.7.13 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
```

20.7.14 Boot Loader Parameters

In Table 20-7 through Table 20-10 on page 247, the parameters used in the description of the self programming are given.

For AT90PWM81.

Table 20-7. 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 20-2 on page 236.

For AT90PWM161.

Table 20-8. 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	256 words	4	0x000 - 0x1EFF	0x1F00 - 0x1FFF	0x1EFF	0x1F00
1	0	512 words	8	0x000 - 0x1DFF	0x1E00 - 0x1FFF	0x1DFF	0x1E00
0	1	1024 words	16	0x000 - 0x1BFF	0x1C00 - 0x1FFF	0x1BFF	0x1C00
0	0	2048 words	32	0x000 - 0x17FF	0x1800 - 0x1FFF	0x17FF	0x1800

The different BOOTSZ Fuse configurations are shown in Figure 20-2 on page 236.

Table 20-9. Read-while-write limit.

Section	Pages	Address
Read-while-write section (RWW)	96	0x000 - 0x17FF
No read-while-write section (NRWW)	32	0x1800 - 0x1FFF

For details about these two section, see “NRWW – No Read-While-Write Section” on page 234 and “RWW – Read-While-Write Section” on page 234.

Table 20-10. Explanation of different variables used in Figure 20-3 on page 240 and the mapping to the Z-pointer.

Variable	AT90PWM81	AT90PWM81 corresponding Z-value (1)	AT90PWM161	AT90PWM161 corresponding Z-value (1)	Description
PCMSB	11		12		Most significant bit in the Program Counter. (The Program Counter is 12 bits PC[11:0])
PAGEMSB	4		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		Z12	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1
ZPAGEMSB		Z5		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	PC[12:6]	Z12:Z6	Program counter page address: Page select, for page erase and page write
PCWORD	PC[4:0]	Z5:Z1	PC[5:0]	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 239 for details about the use of Z-pointer during Self-Programming.

21. Memory Programming

21.1 Program And Data Memory Lock Bits

The AT90PWM81/161 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 21-2. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 21-1. Lock bit byte (1)

Lock bit byte	Bit no.	Description	Default value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

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

Table 21-2. Lock bit protection modes (1)(2).

Memory lock bits			Protection type
LB mode	LB2	LB1	
1	1	1	No memory lock features enabled
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode (1)
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Boot Lock bits and Fuse bits are locked in both Serial and Parallel Programming mode (1)

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

Table 21-3. 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.

21.2 Fuse Bits

The AT90PWM81/161 has three Fuse bytes. Table 21-4 to Table 21-6 on page 251 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 21-4. Extended Low Fuse byte.

Extended fuse byte	Bit No	Description	Default value
PSC2RB	7	PSC2 reset behavior	1
PSC2RBA	6	PSC2 reset behavior for OUT22 & 23	1
PSCRRB	5	PSC reduced reset behavior	1
PSCRV	4	PSCOUT & PSCOUTR reset value	1
PSCINRB	3	PSC & PSCR inputs reset behavior	1
BODLEVEL2 [1]	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1 [1]	1	Brown-out detector trigger level	0 (programmed)
BODLEVEL0 [1]	0	Brown-out detector trigger level	1 (unprogrammed)

Notes: 1. See Table 7-2 on page 53 for BODLEVEL Fuse decoding.

21.2.1 PSC Output Behavior During Reset

For external component safety reason, the state of PSC outputs during Reset can be programmed by fuses PSCRV, PSCRRB & PSC2RB.

These fuses are located in the Extended Fuse Byte (see Table 21-4).

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 PSCRRB fuse equals 1 (unprogrammed), PSCOUTR0 & PSCOUTR1 keep a standard port behavior. If PSC0RB fuse equals 0 (programmed), PSCOUTR0 & PSCOUTR1 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUTR0 & PSCOUTR1 keep the forced state until PSOC0 register is written.

If PSC2RB fuse equals 1 (unprogrammed), PSCOUT20 & PSCOUT21 keep a standard port behavior. If PSC2RB fuse equals 0 (programmed), PSCOUT20 & PSCOUT21 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT20 & PSCOUT21 keep the forced state until PSOC2 register is written.

If PSC2RBA fuse equals 1 (unprogrammed), PSCOUT22 & PSCOUT23 keep a standard port behavior. If PSC2RBA fuse equals 0 (programmed), PSCOUT22 & PSCOUT23 are forced at reset to low level or high level according to PSCRV fuse bit. In this second case, PSCOUT22 & PSCOUT23 keep the forced state until PSOC2 register is written.

21.2.2 PSC Input Behavior During Reset

For power consumption under reset reason, the state of PSC & PSCR inputs during Reset can be programmed by fuse PSCINRB.

If PSCINRB fuse equals 1 (unprogrammed), PSC & PSCR input keep a standard port behavior. If PSCINRB fuse equals 0 (programmed), PSC & PSCR input pull-up are forced while the reset is active. Affected pins are PSCIN2, PSCINr, PSCIN2A, PSCINrA. To prevent any conflict on PD1, this fuse has no effect on PSCINrB.

Table 21-5. 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
BOOTSZ1	2	Select boot size (see Table 20-7 on page 246 for details)	0 (programmed) (4)
BOOTSZ0	1	Select boot size (see Table 20-7 on page 246 for details)	0 (programmed) (4)
BOOTRST	0	Select reset vector	1 (unprogrammed)

Notes:
1. See “Alternate Functions of Port E” on page 80 for description of RSTDISBL Fuse.
2. The SPIEN Fuse is not accessible in serial programming mode.
3. See “Watchdog timer configuration.” on page 60 for details.
4. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 21-8 on page 252 for details.

Table 21-6. 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 5-4 on page 30 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 8MHz. See Table 5-1 on page 28 for details.
3. The CKOUT Fuse allows the system clock to be output on PORTD0. See "Clock Output Buffer" on page 34 for details.
4. See "System Clock Prescaler" on page 39 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.

21.2.3 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.

21.3 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space, the signature row.

21.3.1 Signature Bytes

Table 21-7. Device ID bytes for AT90PWM81/161 devices.

Device	Device ID bytes		
	0x004	0x002	0x000
AT90PWM81	88	93	1E
AT90PWM161	8B	94	1E

For the AT90PWM81/161 the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x002: 0x93 (indicates 8KB Flash memory).
3. 0x004: 0x88 (indicates the AT90PWM81 device when 0x002 is 0x93).
   0x004: 0x8B (indicates the AT90PWM161 device when 0x002 is 0x94).

21.4 Calibration Byte

The AT90PWM81/161 has a byte calibration value for the internal RC Oscillator. This byte resides in the byte of address 0x003 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.

21.5 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 AT90PWM81/161. Pulses are assumed to be at least 250ns unless otherwise stated.

21.5.1 Signal Names

In this section, some pins of the AT90PWM81/161 are referenced by signal names describing their functionality during parallel programming, see Figure 21-1 and Table 21-8. Pins not described in Table 21-8 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 21-10 on page 253.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in Table 21-11 on page 253.

Figure 21-1. Parallel programming.

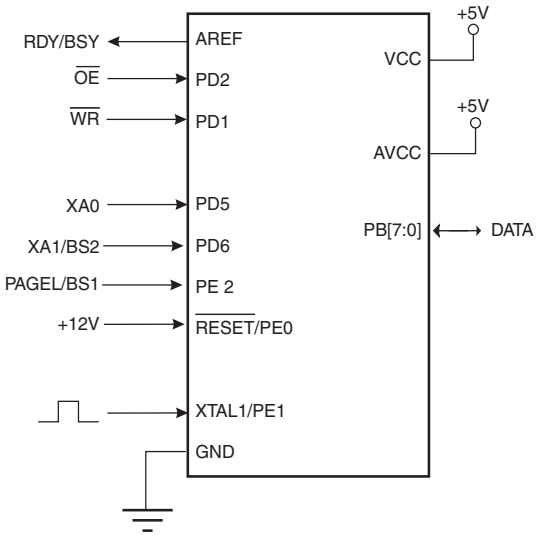


Table 21-8. Pin name mapping.

Signal name in programming mode	Pin name	I/O	Function
RDY/$\overline{BSY}$	AREF	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output enable (active low)
$\overline{WR}$	PD1	I	Write pulse (active low)
XA0	PD5	I	XTAL Action Bit 0

Table 21-8. Pin name mapping. (Continued)

Signal name in programming mode	Pin name	I/O	Function
XA1/BS2	PD6	I	XTAL Action Bit 1 Byte Select 2 (“0” selects Low byte, “1” selects 2’nd High byte)
PAGEL/BS1	PE2	I	Program memory and EEPROM Data Page Load Byte Select 1 (“0” selects Low byte, “1” selects High byte)
DATA	PB[7:0]	I/O	Bi-directional Data bus (Output when $\overline{OE}$ is low)

Table 21-9. Pin Values Used to Enter Programming Mode.

Pin	Symbol	Value
XA1/BS2	Prog_enable[3]	0
XA0	Prog_enable[2]	0
$\overline{OE}$	Prog_enable[1]	0
$\overline{WR}$	Prog_enable[0]	0

Table 21-10. 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 21-11. 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
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 21-12. No. of words in a page and no. of pages in the flash.

Device	Flash size	Page size	PCWORD	No. of pages	PCPAGE	PCMSB
AT90PWM81	4K words (8Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11
AT90PWM161	8K words (16Kbytes)	64 words	PC[5:0]	128	PC[12:6]	12

Table 21-13. No. of words in a page and no. of pages in the EEPROM.

Device	EEPROM size	Page size	PCWORD	No. of pages	PCPAGE	EEAMSB
AT90PWM81/161	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

21.6 Serial Programming Pin Mapping

Table 21-14. Pin mapping serial programming.

Symbol	Pins	I/O	Description
MOSI		I	Serial Data in
MISO		O	Serial Data out
SCK		I	Serial Clock

21.7 Parallel Programming

21.7.1 Enter Programming Mode

The following algorithm puts the device in Parallel (High-voltage) > Programming mode:

1. Set Prog_enable pins listed in Table 21-9 on page 253 to “0000”, RESET pin to “0” and V_{CC} to 0V.
2. Apply 4.5V - 5.5V between V_{CC} and GND. Ensure that V_{CC} reaches at least 1.8V within the next 20µs.
3. Wait 20µs - 60µs, and apply 11.5V - 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 V_{CC} is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set Prog_enable pins listed in Table 21-9 on page 253 to “0000”, RESET pin to “0” and V_{CC} to 0V.
2. Apply 4.5V - 5.5V between V_{CC} and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9V - 1.1V, apply 11.5V - 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 V_{CC} actually reaches 4.5V -5.5V before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing RESET pin to 0V.

21.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading

21.7.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM[1] memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: 1. The 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.

21.7.4 Programming the Flash

The Flash is organized in pages, see Table 21-12 on page 254. 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 21-3 on page 257 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 21-2 on page 257. 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

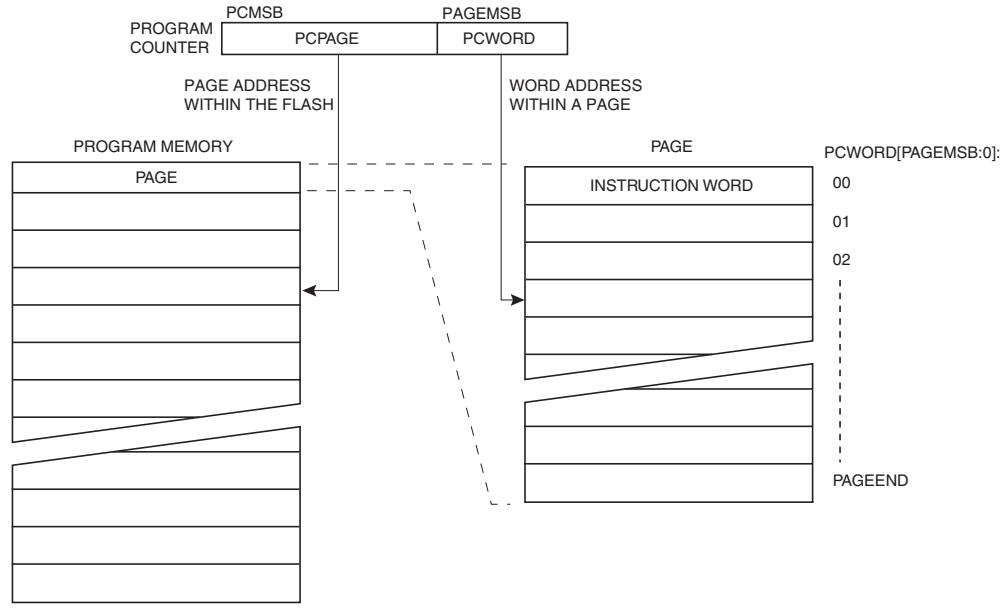
1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (see Figure 21-3 on page 257 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed

J. End Page Programming

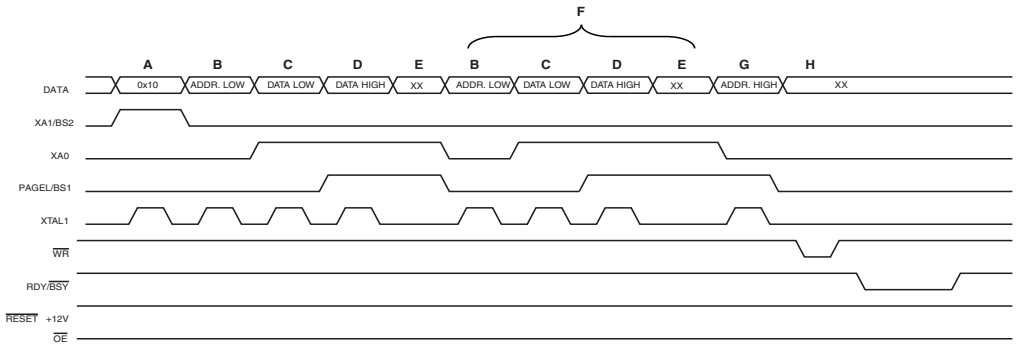
1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 21-2. Addressing the flash, which is organized in pages [1].



Note: 1. PCPAGE and PCWORD are listed in Table 21-12 on page 254.

Figure 21-3. Programming the flash waveforms [1].



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

21.7.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 21-13 on page 254. 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 255 for details on Command, Address and Data loading):

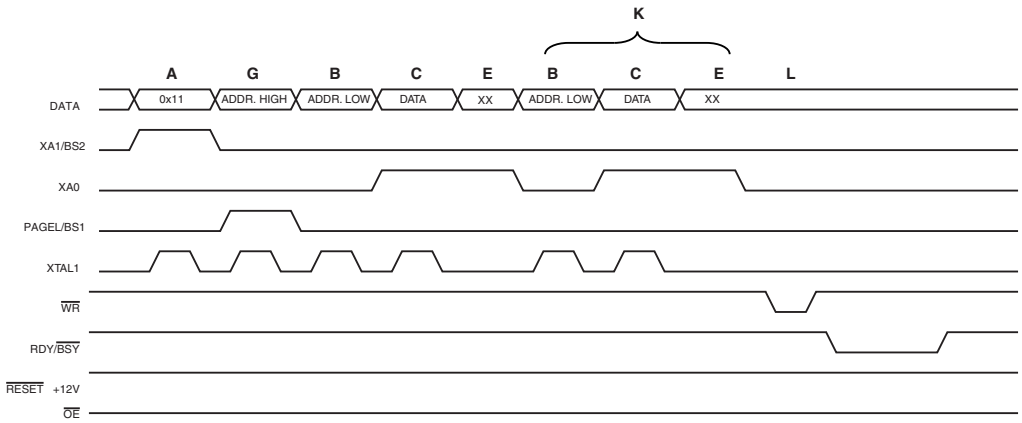
1. A: Load Command "0001 0001".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

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 21-4 for signal waveforms).

Figure 21-4. Programming the EEPROM waveforms.



21.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to “Programming the Flash” on page 255 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”.

21.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 255 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”.

21.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 255 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.

21.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 255 for details on Command and Data loading):

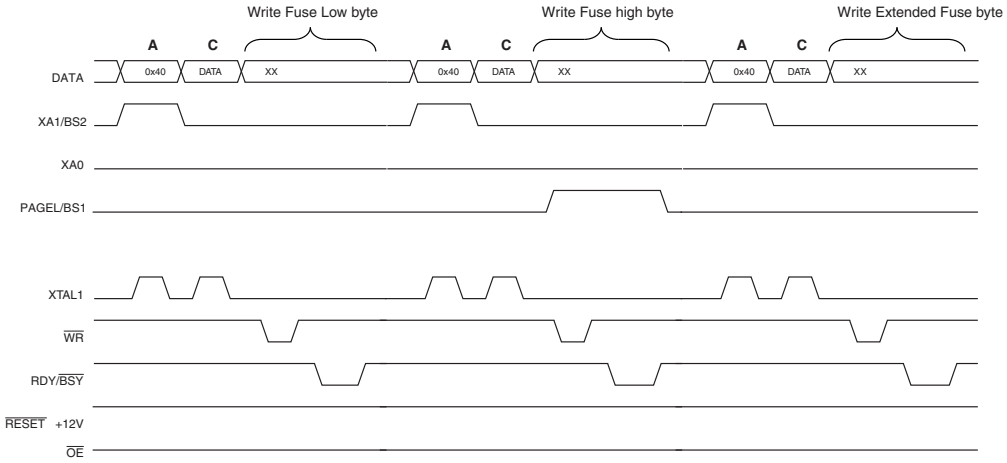
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

21.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 255 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 21-5. Programming the FUSES waveforms.



21.7.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 255 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.

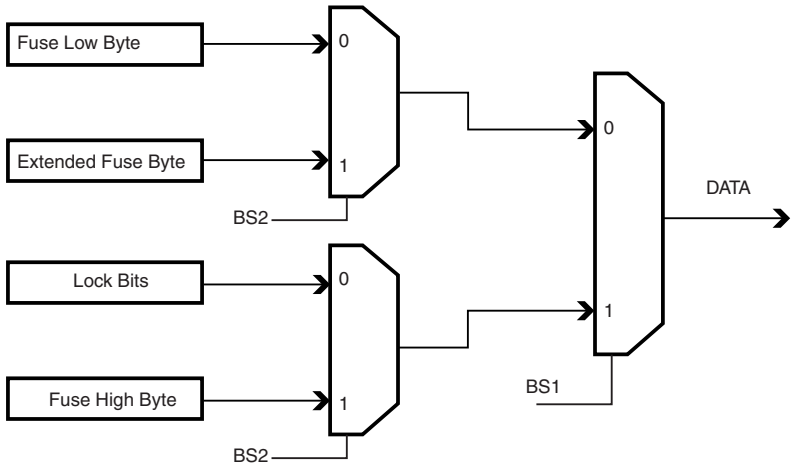
The Lock bits can only be cleared by executing Chip Erase.

21.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 255 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 21-6. Mapping between BS1, BS2 and the Fuse and Lock Bits during read.



21.7.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 255 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".

21.7.14 Reading the Calibration Byte

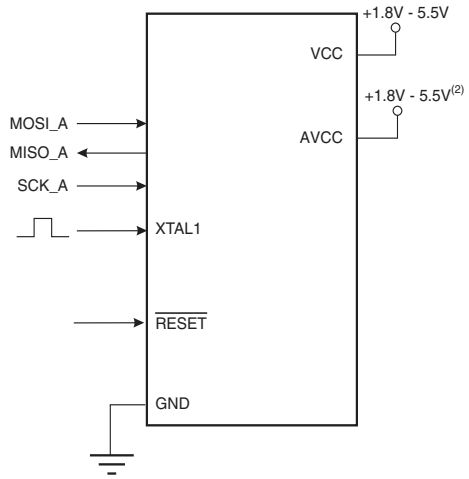
The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 255 for details on Command and Address loading):

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".

21.8 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 21-14 on page 254, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 21-7. 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 < AV_{CC} < V_{CC} + 0.3V, however, AV_{CC} should always be within 1.8V - 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} < 12MHz, 3 CPU clock cycles for f_{ck} >= 12MHz

High: > 2 CPU clock cycles for f_{ck} < 12MHz, 3 CPU clock cycles for f_{ck} >= 12MHz

21.8.1 Serial Programming Algorithm

When writing serial data to the AT90PWM81/161, data is clocked on the rising edge of SCK.

When reading data from the AT90PWM81/161, data is clocked on the falling edge of SCK. See Figure 21-8 on page 263 for timing details.

To program and verify the AT90PWM81/161 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 21-16 on page 263):

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 20ms 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 21-15 on page 263.) 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 21-15 on page 263.) 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.

21.8.2 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 21-15 on page 263 for t_{WD_FLASH} value.

21.8.3 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 21-15 on page 263 for t_{WD_EEPROM} value.

Table 21-15. Minimum wait delay before writing the next flash or EEPROM location.

Symbol	Minimum wait delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	3.6ms
t_{WD_ERASE}	9.0ms

Figure 21-8. Serial programming waveforms.

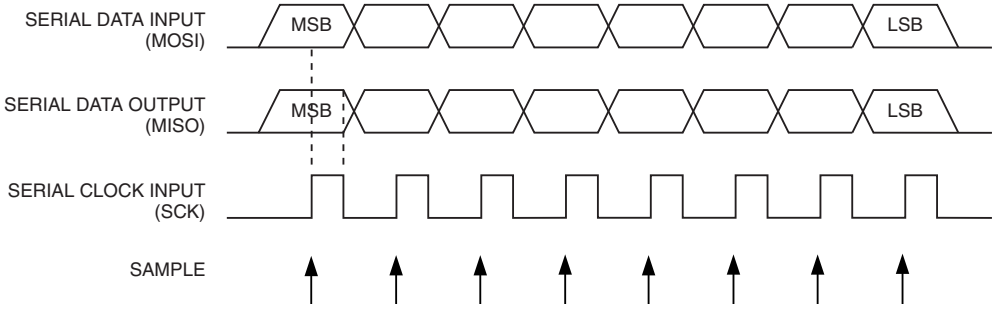


Table 21-16. Serial programming instruction set.

Instruction	Instruction format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after $\overline{\text{RESET}}$ 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**
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 21-1 on page 248 for details

Table 21-16. Serial programming instruction set. (Continued)

Instruction	Instruction format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = “0” to program Lock bits. See Table 21-1 on page 248 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 21-6 on page 251 for details
Write Fuse High bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 21-5 on page 250 for details
Write Extended Fuse Bits	1010 1100	1010 0100	xxxx xxxx	**xxxx xxii**	Set bits = “0” to program, “1” to unprogram. See Table 21-4 on page 249 for details
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 21-6 on page 251 for details
Read Fuse High bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse High bits. “0” = programmed, “1” = unprogrammed. See Table 21-5 on page 250 for details
Read Extended Fuse Bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read Extended Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 21-4 on page 249 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.

21.8.4 SPI Serial Programming Characteristics

For characteristics of the SPI module see “SPI Serial Programming Characteristics” on page 264.

22. Electrical Characteristics [1]

22.1 Absolute Maximum Ratings*

Operating temperature..... -40°C to +105°C Or operating temperature ..... -40°C to +125°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..... 40.0mA DC current V_{CC} and GND pins 200.0mA	*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.

22.2 DC Characteristics

T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted).

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
V_{IL}	Input low voltage	Port B & D and XTAL1, XTAL2 pins as I/O	-0.5		$0.2V_{CC}$ (1)	V
V_{IH}	Input high voltage	Port B D and XTAL1, XTAL2 pins as I/O	$0.6V_{CC}$ (2)		V_{CC}+0.5	
V_{IL1}	Input low voltage	XTAL1 pin , External Clock Selected	-0.5		$0.1V_{CC}$ (1)	
V_{IH1}	Input high voltage	XTAL1 pin , External Clock Selected	$0.7V_{CC}$ (2)		V_{CC}+0.5	
V_{IL2}	Input low voltage	$\overline{\text{RESET}}$ pin	-0.5		$0.2V_{CC}$ (1)	
V_{IH2}	Input high voltage	$\overline{\text{RESET}}$ pin	$0.9V_{CC}$ (2)		V_{CC}+0.5	
V_{IL3}	Input low voltage	$\overline{\text{RESET}}$ pin as I/O	-0.5		$0.2V_{CC}$ (1)	
V_{IH3}	Input high voltage	$\overline{\text{RESET}}$ pin as I/O	$0.8V_{CC}$ (2)		V_{CC}+0.5	
V_{OL}	Output low voltage (3) (Port B & D and XTAL1, XTAL2 pins as I/O)	I_{OL} = 10mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V			0.6 0.5	
V_{OH}	Output high voltage (4) (Port B & D and XTAL1, XTAL2 pins as I/O)	I_{OH} = -10mA, V_{CC} = 5V I_{OH} = -5mA, V_{CC} = 3V	4.3 2.5			
V_{OL3}	Output low voltage (3) ($\overline{\text{RESET}}$ pin as I/O)	I_{OL} = 2.1mA, V_{CC} = 5V I_{OL} = 0.8mA, V_{CC} = 3V			0.7 0.5	
V_{OH3}	Output high voltage (4) ($\overline{\text{RESET}}$ pin as I/O)	I_{OH} = -0.6mA, V_{CC} = 5V I_{OH} = -0.4mA, V_{CC} = 3V	3.8 2.2			
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	
R_{RST}	Reset pull-up resistor		30		200	kW
R_{pu}	I/O pin pull-up resistor		20		50	

T_A = -40°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted). (Continued)

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
I_{CC}	Power supply current	Active 8MHz, V_{CC} = 3V, RC osc., PRR = 0xFF		3.5	5	mA
		Active 16MHz, V_{CC} = 5V, Ext Clock, PRR = 0xFF		10.5	15	
		Idle 8MHz, V_{CC} = 3V, RC Osc		1.5	2	
		Idle 16MHz, V_{CC} = 5V, Ext Clock		4.5	7	
	Power-down mode (5)	WDT enabled,V_{CC} = 3V 25°C		7		µA
		WDT enabled, V_{CC} = 3V 105°C			30	
		WDT enabled, V_{CC} = 5V 25°C		10		µA
		WDT enabled, V_{CC} = 5V 105°C			50	
		WDT disabled, V_{CC} = 3V 25°C		0.5		µA
		WDT disabled, V_{CC} = 3V 105°C			25	
		WDT disabled, V_{CC} = 5V 25°C		1		
		WDT disabled, V_{CC} = 5V 105°C			40	µA
V_{REF}	Internal voltage reference (7)	@25°C	2.46	2.56	2.66	V
	Analog comparator input common mode range		0.1		V_{CC} - 0.1	
V_{ACIO}	Analog comparator input offset voltage	Input offset voltage $0.1<V_{IN}<V_{CC}$ - 0.1V		±1.5	±10	mV
		With ±10mV hysteresis $0.1<V_{IN}<V_{CC}$ - 0.1V		±10	±20	
		With ± 25mV hysteresis $0.1<V_{IN}<V_{CC}$ - 0.1V		±25	±60	
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		50 (6)		ns

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

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
V_{IL}	Input low voltage	Port B & D and XTAL1, XTAL2 pins as I/O	-0.5		$0.2V_{CC}$ (1)	V
V_{IH}	Input high voltage	Port B D and XTAL1, XTAL2 pins as I/O	$0.6V_{CC}$ (2)		V_{CC}+0.5	
V_{IL1}	Input low voltage	XTAL1 pin, External Clock selected	-0.5		$0.1V_{CC}$ (1)	
V_{IH1}	Input high voltage	XTAL1 pin, External Clock selected	$0.7V_{CC}$ (2)		V_{CC}+0.5	
V_{IL2}	Input low voltage	$\overline{\text{RESET}}$ pin	-0.5		$0.2V_{CC}$ (1)	
V_{IH2}	Input high voltage	$\overline{\text{RESET}}$ pin	$0.9V_{CC}$ (2)		V_{CC}+0.5	
V_{IL3}	Input low voltage	$\overline{\text{RESET}}$ pin as I/O	-0.5		$0.2V_{CC}$ (1)	
V_{IH3}	Input high voltage	$\overline{\text{RESET}}$ pin as I/O	$0.8V_{CC}$ (2)		V_{CC}+0.5	
V_{OL}	Output low voltage (3) (Port B & D and XTAL1, XTAL2 pins as I/O)	I_{OL} = 10mA, V_{CC} = 5V I_{OL} = 5mA, V_{CC} = 3V			0.6 0.5	
V_{OH}	Output high voltage (4) (Port B & D and XTAL1, XTAL2 pins as I/O)	I_{OH} = -10mA, V_{CC} = 5V I_{OH} = -5mA, V_{CC} = 3V	4.3 2.5			
V_{OL3}	Output low voltage (3) ($\overline{\text{RESET}}$ pin as I/O)	I_{OL} = 2.1mA, V_{CC} = 5V I_{OL} = 0.8mA, V_{CC} = 3V			0.7 0.5	
V_{OH3}	Output high voltage (4) ($\overline{\text{RESET}}$ pin as I/O)	I_{OH} = -0.6mA, V_{CC} = 5V I_{OH} = -0.4mA, V_{CC} = 3V	3.8 2.2			
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	
R_{RST}	Reset pull-up resistor		30		200	kW
R_{PU}	I/O pin pull-up resistor		20		50	

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
I_{CC}	Power supply current	Active 8MHz, V_{CC} = 3V, RC osc, PRR = 0xFF		3.5	5	mA
		Active 16MHz, V_{CC} = 5V, Ext Clock, PRR = 0xFF		10.5	15	
		Idle 8MHz, V_{CC} = 3V, RC Osc		1.5	2	
		Idle 16MHz, V_{CC} = 5V, Ext Clock		4.5	7	
	Power-down mode (5)	WDT enabled, V_{CC} = 3V 25°C		7		µA
		WDT enabled, V_{CC} = 3V 125°C			70	
		WDT enabled, V_{CC} = 5V 25°C		10		
		WDT enabled, V_{CC} = 5V 125°C			110	
		WDT disabled, V_{CC} = 3V 25°C		0.5		
		WDT disabled, V_{CC} = 3V 125°C			35	
		WDT disabled, V_{CC} = 5V 25°C		1		
		WDT disabled, V_{CC} = 5V 125°C			55	
V_{REF}	Internal voltage reference (7)	@25°C	2.46	2.56	2.66	V
	Analog comparator input common mode range		0.1		V_{CC} - 0.1	
V_{ACIO}	Analog comparator input offset voltage	Input offset voltage $0.1<V_{IN}<V_{CC}$ - 0.1V		±1.5	±10	mV
		With ±10mV Hysteresis $0.1<V_{IN}<V_{CC}$ - 0.1V		±10	±20	
		With ±25mV Hysteresis $0.1<V_{IN}<V_{CC}$ - 0.1V		±25	±60	
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		50 (6)		ns

Note:
1. “Maximum” means the highest value where the pin is guaranteed to be read as low.
2. “Minimum” 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 (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   SO20 and TQFN Package:
   1] The sum of all I_{OL}, for all ports, should not exceed 400mA.
   2] The sum of all I_{OL}, for ports B6 - B7, D0 - D3, E0 should not exceed 100mA.
   3] The sum of all I_{OL}, for ports B0 - B1, D4, E1 - E2 should not exceed 100mA.

If I_{OL} exceeds the test condition, V_{OL} 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 (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   SO20 and TQFN Package:
   1] The sum of all I_{OH}, for all ports, should not exceed 400mA.
   2] The sum of all I_{OH}, for ports B6 - B7, D0 - D3, E0 should not exceed 150mA.
   3] The sum of all I_{OH}, for ports B0 - B1, D4, E1 - E2 should not exceed 150mA.
   If I_{OH} exceeds the test condition, V_{OH} 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. Propagation delay of the internal comparator with 100mV overdrive condition.
7. accuracy : ±8% from -40°C to +125°C.

22.3 Clock Drive Characteristics

22.3.1 Calibrated Internal RC Oscillator Accuracy

Table 22-1. Calibration accuracy of internal RC oscillator.

	Frequency	V_{CC}	Temperature	Calibration accuracy
Factory calibration	8.0MHz	3V	25°C	±1%
Factory calibration	8.0MHz	2.7V - 5.5V	-40°C + or 125°C	±6%
User calibration	7.6MHz - 8.4MHz	2.7V - 5.5V	-40°C +105 or 125°C	±5%

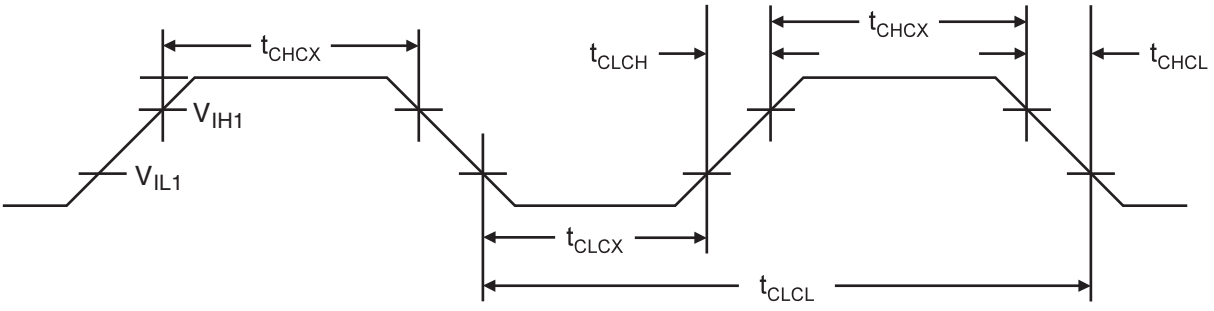
22.3.2 Watchdog Oscillator Accuracy

Table 22-2. Accuracy of watchdog oscillator.

Frequency	Accuracy
128kHz	±40%

22.3.3 External Clock Drive Waveforms

Figure 22-1. External clock drive waveforms.



22.3.4 External Clock Drive

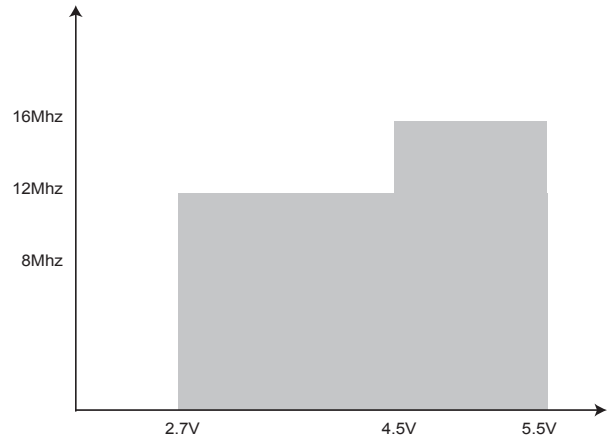
Table 22-3. External clock drive.

Symbol	Parameter	V_{CC} = 2.7V - 5.5V		V_{CC} = 4.5V - 5.5V		Units
		Minimum	Maximum	Minimum	Maximum	
$1/t_{CLCL}$	Oscillator frequency	0	12	0	16	MHz
t_{CLCL}	Clock period	83		62		ns
t_{CHCX}	High time	30		20		
t_{CLCX}	Low time	30		20		
t_{CLCH}	Rise time		1.6		0.5	ms
t_{CHCL}	Fall time		1.6		0.5	
Dt_{CLCL}	Change in period from one clock cycle to the next		2		2	%

22.4 Maximum Speed vs. V_{CC}

Maximum frequency is depending on $V_{CC.}$ As shown in Figure 22-2 on page 271 , the Maximum Frequency equals 12MHz 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 22-2. Maximum frequency vs. V_{CC}, AT90PWM81/161.



22.5 PLL Characteristics

Table 22-4. PLL characteristics - V_{CC} = 2.7V to 5.5V (unless otherwise noted).

Symbol	Parameter	Minimum	Typical	Maximum	Units
PLL_{IF}	Input frequency (1)		8		MHz
PLL_F	PLL factor	4		8 (2)	
PLL_{LT}	Lock-in time			64	µS

Notes: 1. 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.
2. When V_{CC} is below 4.5V, maximum PLL_F is 6.

22.6 SPI Timing Characteristics

See Figure 22-3 on page 273 and Figure 22-4 on page 273 for details.

Table 22-5. SPI timing parameters.

	Description	Mode	Minimum	Typical	Maximum	Units
1	SCK period	Master		See Table 14-5 on page 187		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}			ns
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}			

Notes: 1. In SPI programming mode the minimum SCK high/low period is:
- 2 t_{CLCL} for f_{CK} <12MHz
- 3 t_{CLCL} for f_{CK} >12MHz

Figure 22-3. SPI interface timing requirements (Master mode).

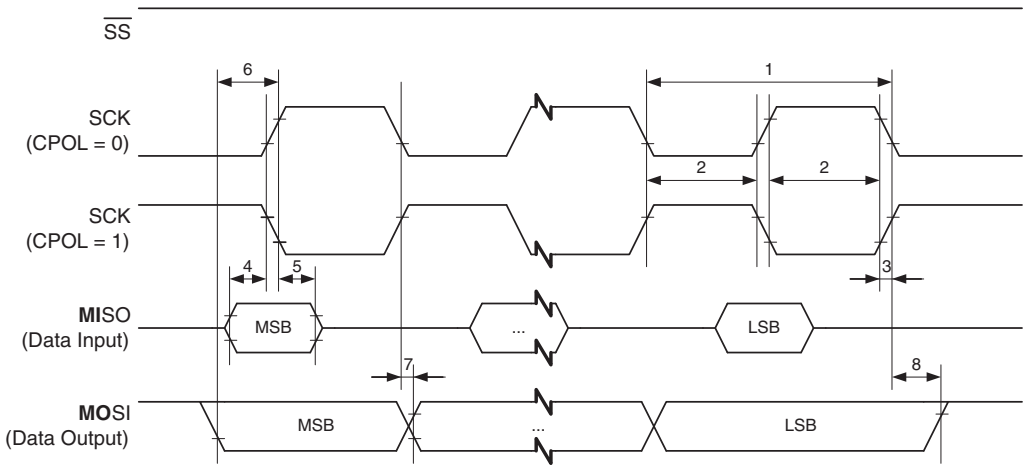
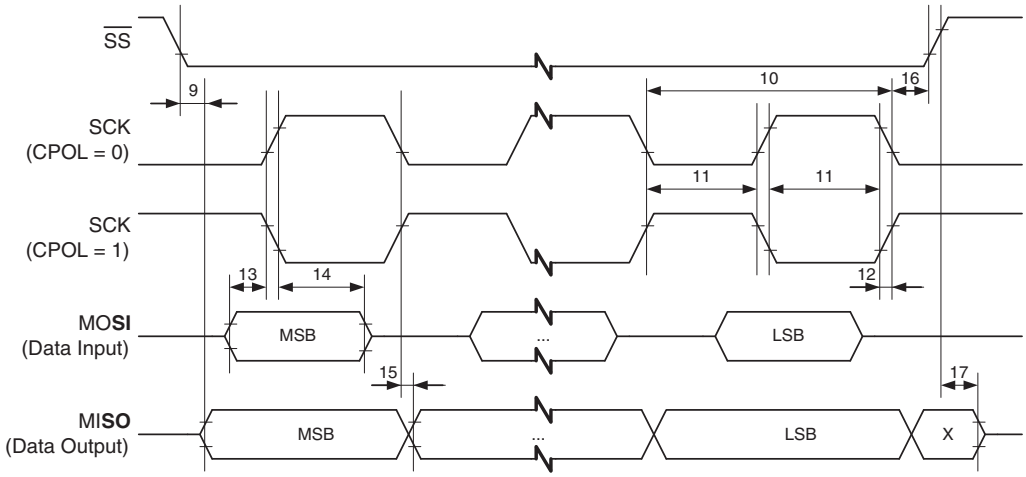


Figure 22-4. SPI interface timing requirements (Slave mode).



22.7 ADC Characteristics

Table 22-6. ADC characteristics - T_A = -45°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted).

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
	Resolution	Single Ended Conversion		10		Bits
		Differential conversion, Gain = 5× or 10×		8		
		Differential conversion, Gain = 20× or 40×		8		
	Absolute accuracy	Single Ended Conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 1MHz		2	4	LSB
		Single Ended Conversion, V_{CC} = 2.7V, V_{REF} =2.56V ADC clock = 2MHz		2.2	4	
		Differential conversion, Gain = 5× or 10× V_{CC} = 5V, V_{REF} = 4V ADC clock = 1MHz		1.2	2.0	
		Differential conversion, Gain = 20× or 40× V_{CC} = 5V, V_{REF} = 4V ADC clock = 2MHz		1.5	3.0	
	Integral non-linearity	Single Ended Conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 1MHz		0.6	1	LSB
		Single Ended Conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 2MHz		0.8	1.5	
		Single Ended Conversion, V_{CC} = 2.7V, V_{REF} = 2.56V ADC clock = 2MHz		1.0	2.5	
		Differential conversion, Gain=5× or 10× V_{CC} = 5V, V_{REF} = 4V ADC clock = 1MHz		0.5	1.0	
		Differential conversion, Gain=20× or 40× V_{CC} = 5V, V_{REF} = 4V ADC clock = 2MHz		0.8	2.0	

Table 22-6. ADC characteristics - T_A = -45°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted). (Continued)

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
	Differential non-linearity	Single Ended conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 1MHz		0.2	0.5	LSB
		Single Ended conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 2MHz		0.6	1	
		Single Ended conversion, V_{CC} = 2.7V, V_{REF} =2.56V ADC clock = 2MHz		1.0	2.5	
		Differential conversion, Gain=5× or 10× V_{CC} = 5V, V_{REF} = 4V ADC clock = 1MHz		0.3	0.8	
		Differential conversion, Gain=20× or 40× V_{CC} = 5V, V_{REF} = 4V ADC clock = 2MHz		0.5	1.0	
	Gain error	Single Ended conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 1MHz	0.0		-6.0	LSB
		Single Ended conversion, V_{CC} = 2.7V, V_{REF} = 2.56V ADC clock = 2MHz	0.0		-6.0	
		Differential conversion, V_{CC} = 5V, V_{REF} = 4V ADC clock = 1MHz	-2.0		+2.0	
	Offset error	Single Ended conversion, V_{CC} = 4V, V_{REF} = 4V ADC clock = 1MHz	-1.0		2.0	LSB
		Single Ended conversion, V_{CC} = 2.7V, V_{REF} = 2.56V ADC clock = 2MHz	1.0		4.0	
		Differential conversion, V_{CC} = 5V, V_{REF} = 4V ADC clock = 1MHz	-1.0		+1.0	
	Conversion time	Single conversion	8		260	µs
	Clock frequency		50		2000	kHz
AV_{CC}	Analog supply voltage		V_{CC} - 0.3		V_{CC} + 0.3	V
V_{REF}	Reference voltage		2.56		AV_{CC} - 0.6	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		

Table 22-6. ADC characteristics - T_A = -45°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted). (Continued)

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
R_{REF}	Reference input resistance			30		kW
R_{AIN}	Analog input resistance			23		
C_{AIN}	Analog input capacitor			10		pF
I_{HSM}	Increased current consumption	High speed mode Single Ended conversion			380	µA

22.8 DAC Characteristics

Table 22-7. DAC characteristics - T_A = -45°C to +105°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted).

Symbol	Parameter	Condition	Minimum	Typical	Maximum	Units
	Resolution	DAC		10		
	Absolute accuracy	V_{CC} = 4V, V_{REF} = 4V		2.5	5	LSB
	Integral non-linearity	V_{CC} = 4V, V_{REF} = 4V		0.8	1.5	
	Differential non-linearity	V_{CC} = 4V, V_{REF} = 4V		0.2	0.5	
	Gain error	V_{CC} = 4V, V_{REF} = 4V	-5.0		0.0	
	Offset error	V_{CC} = 4V, V_{REF} = 4V	0.0		2.0	
V_{REF}	Reference voltage		2.56		AV_{CC}	V

22.9 Parallel Programming Characteristics

Figure 22-5. Parallel programming timing, including some general timing requirements.

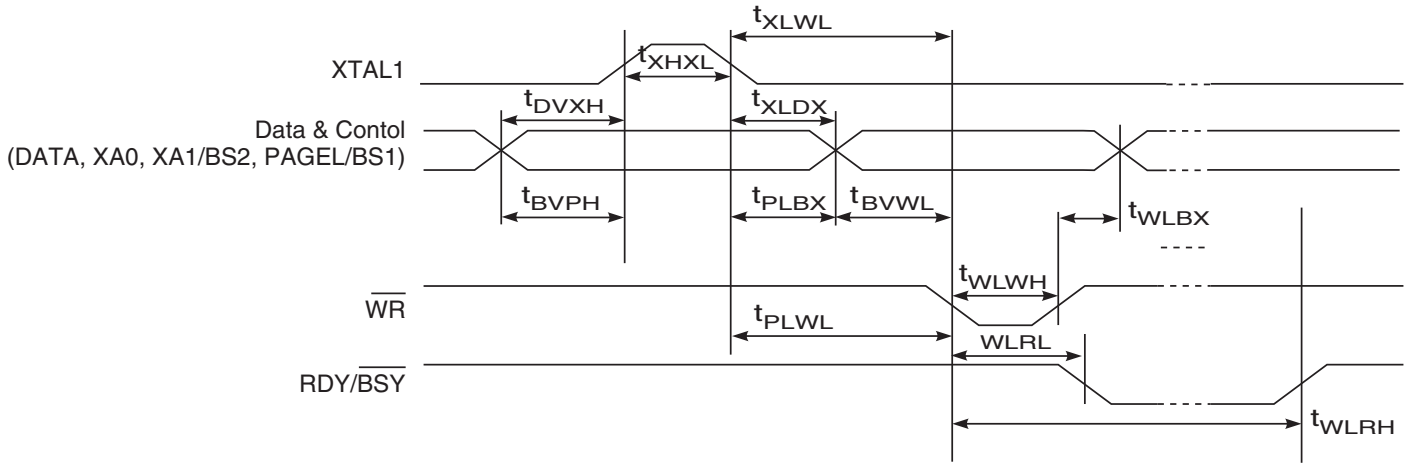
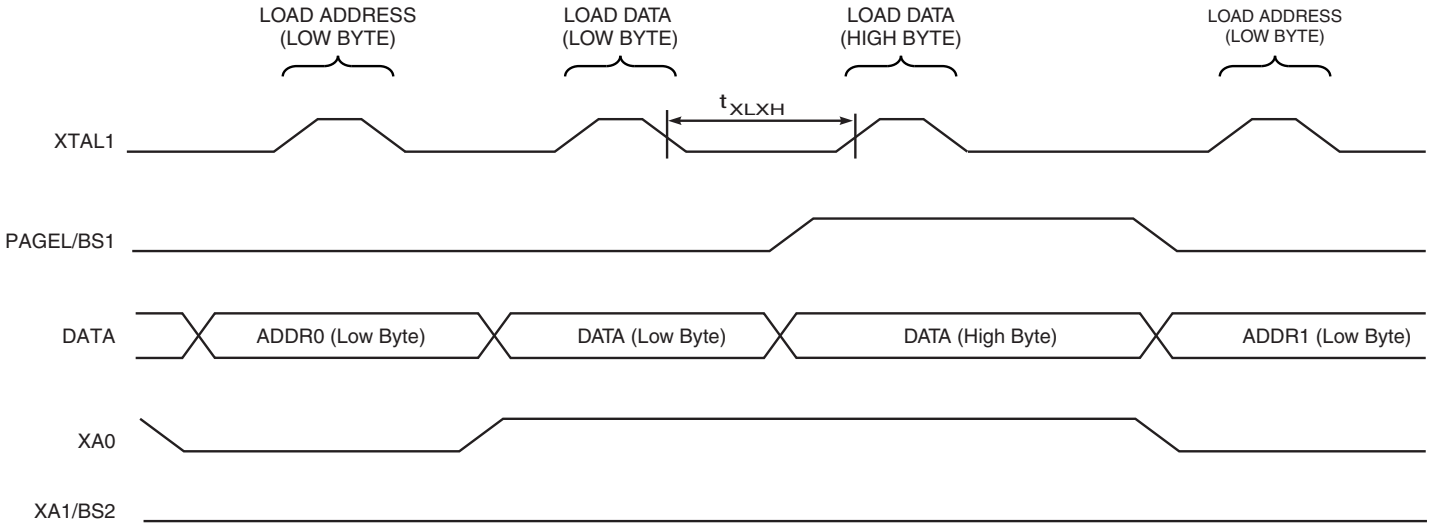
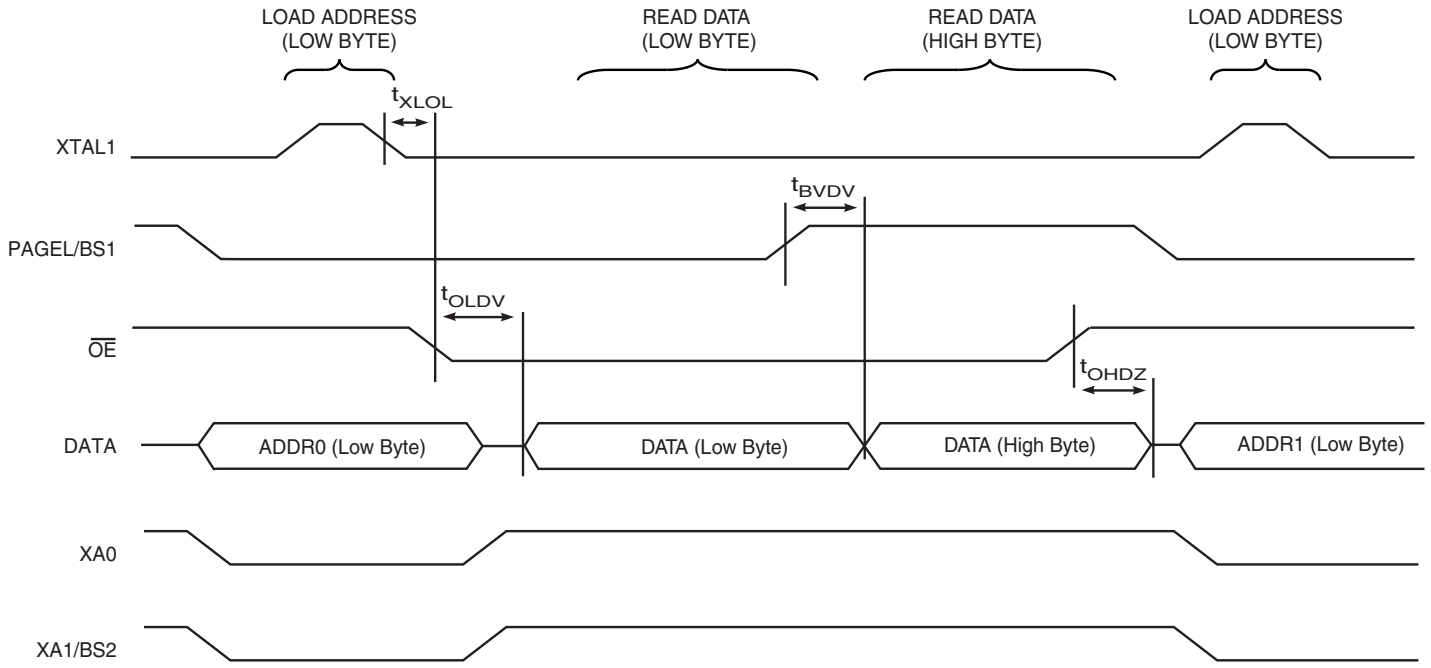


Figure 22-6. Parallel programming timing, loading sequence with timing requirements [1].



Note: 1. The timing requirements shown in Figure 22-5 on page 276 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 22-7. Parallel programming timing, reading sequence (within the same page) with timing requirements [1].



Note: 1. The timing requirements shown in Figure 22-5 on page 276 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 22-8. Parallel programming characteristics, V_{CC} = 5V ±10%.

Symbol	Parameter	Minimum	Typical	Maximum	Units
V_{PP}	Programming enable voltage	11.5		12.5	V
I_{PP}	Programming enable current			250	mA
t_{DVXH}	Data and control valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			
t_{XHXL}	XTAL1 Pulse Width High	150			
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			
t_{XLPH}	XTAL1 Low to PAGEL high	0			
t_{PLXH}	PAGEL low to XTAL1 high	150			
t_{BVPH}	BS1 Valid before PAGEL High	67			
t_{PHPL}	PAGEL Pulse Width High	150			
t_{PLBX}	BS1 Hold after PAGEL Low	67			
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	ms
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	
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	

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.

23. AT90PWM81/161 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.

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 $C_L \times V_{CC} \times f$, 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.

23.1 Active Supply Current

Figure 23-1. Active supply current vs. frequency (0.1MHz - 1.0MHz).

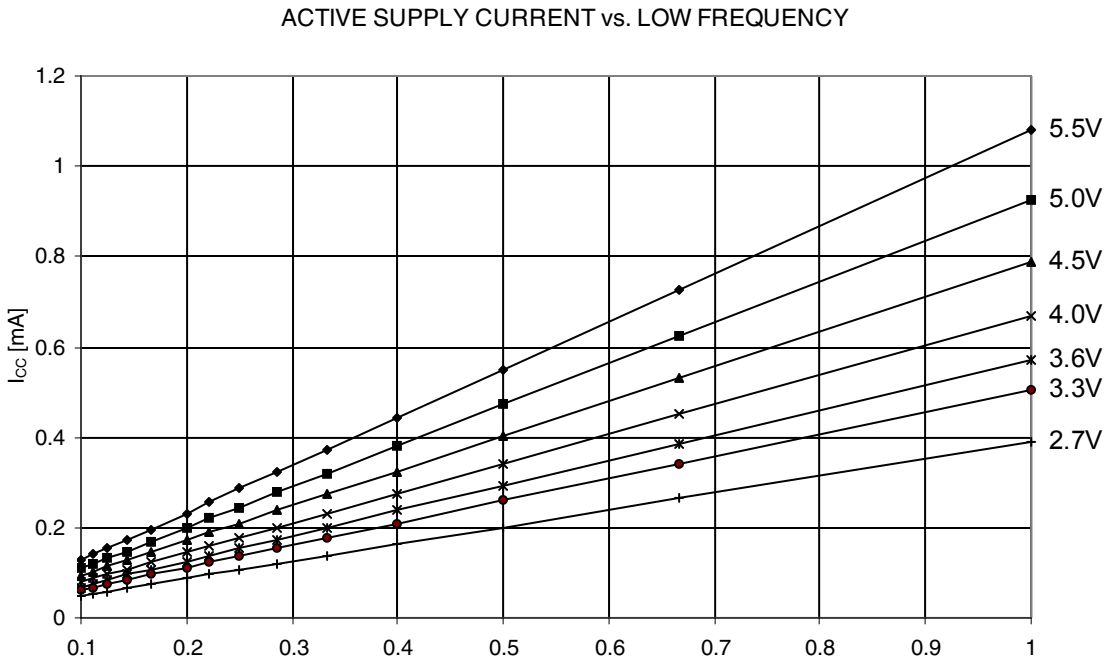


Figure 23-2. Active supply current vs. frequency (1MHz - 16MHz).

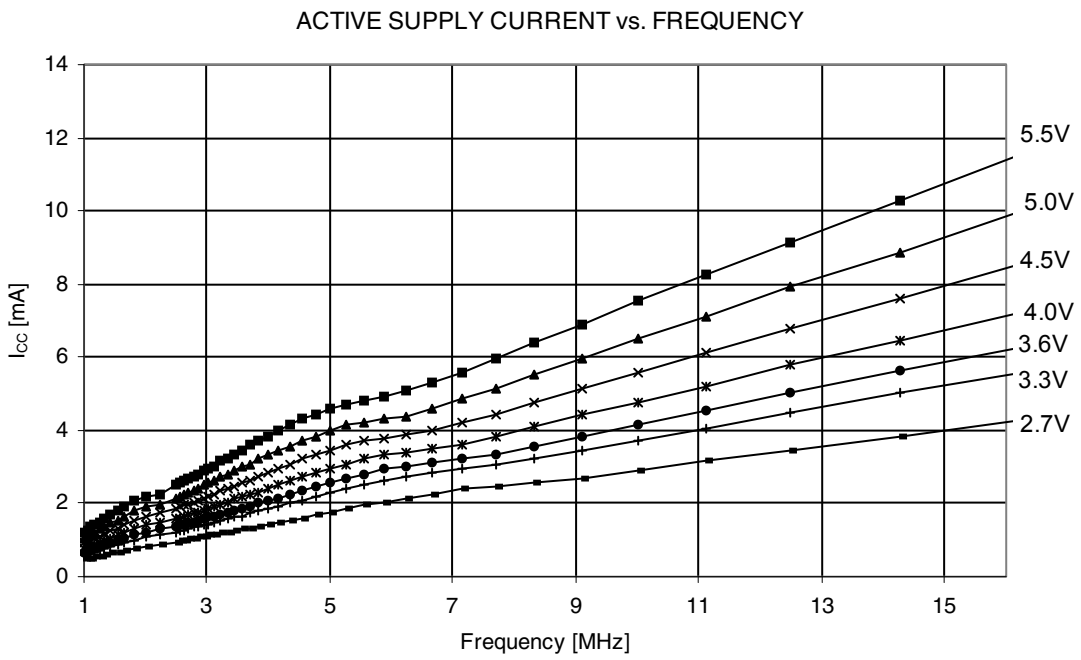


Figure 23-3. Active supply current vs. V_{CC} (internal RC oscillator, 8MHz).

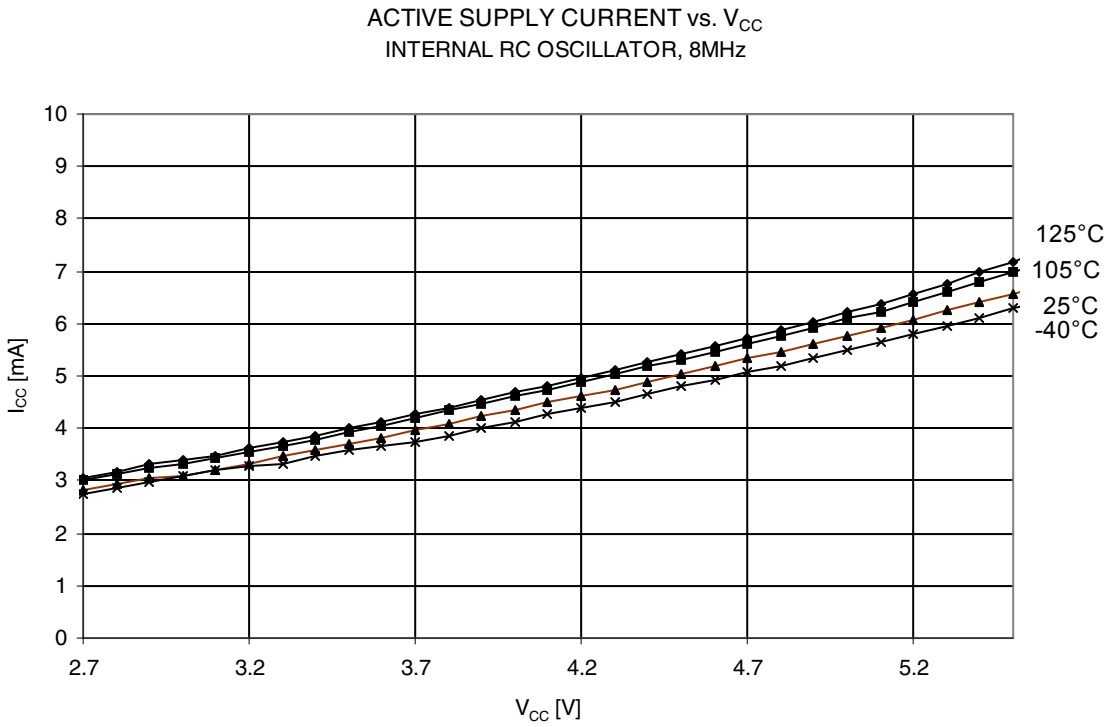
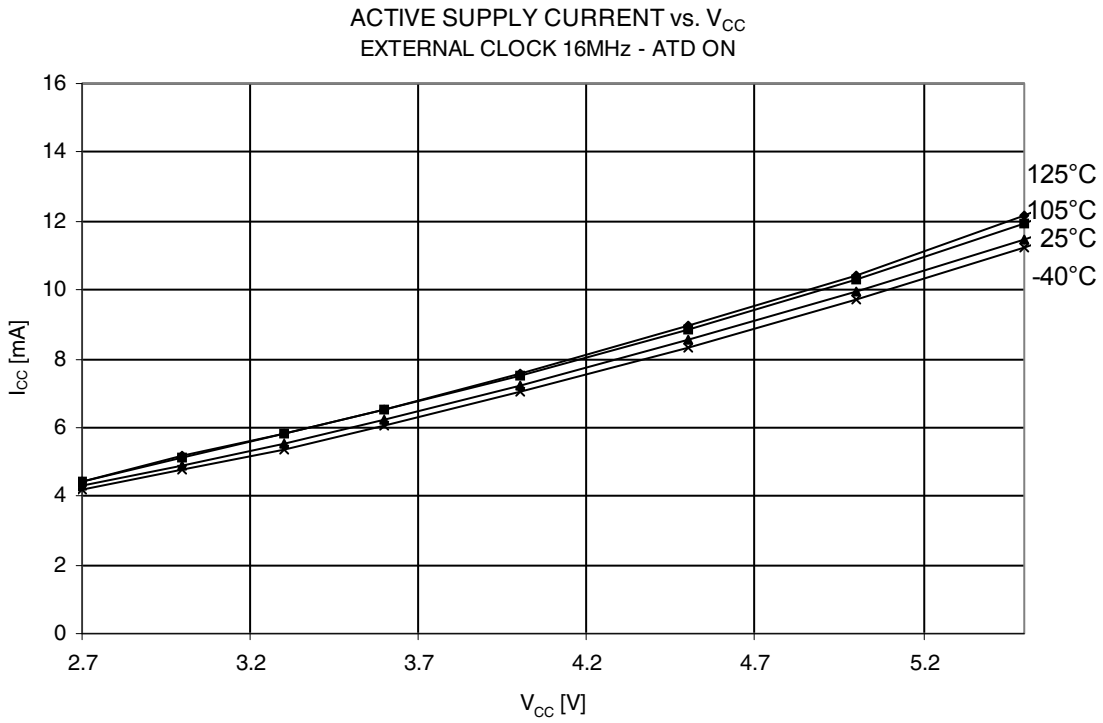


Figure 23-4. Active supply current vs. V_{CC} (external clock, 16MHz).



23.2 Idle Supply Current

Figure 23-5. Idle supply current vs. frequency (0.1MHz - 1.0MHz).

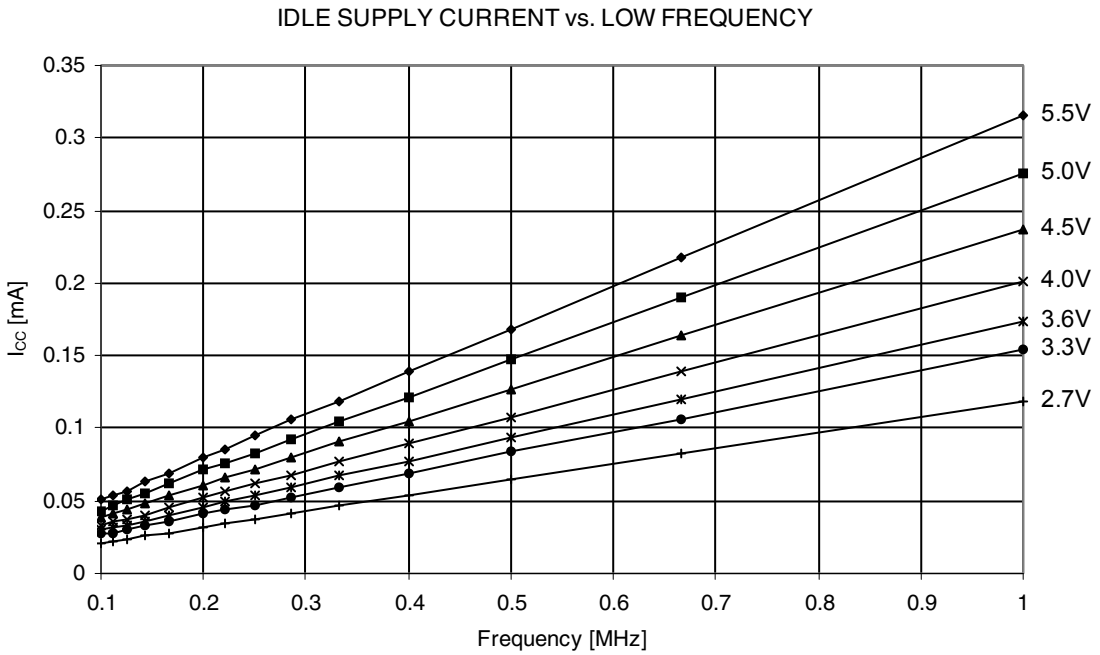


Figure 23-6. Idle supply current vs. frequency (1MHz - 16MHz).

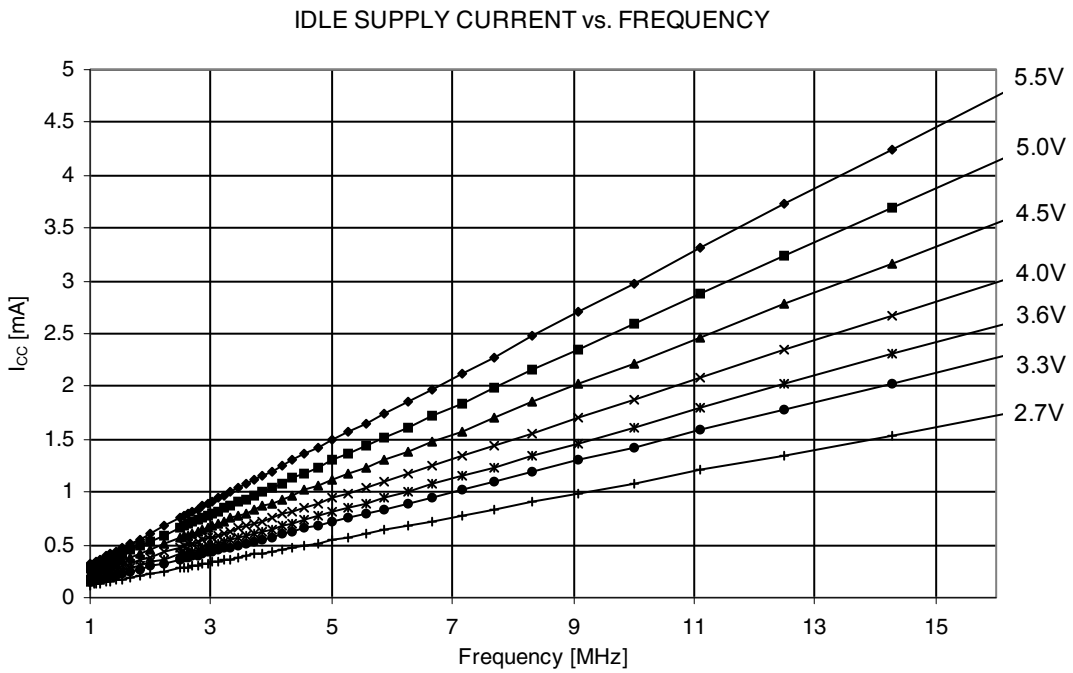


Figure 23-7. Idle supply current vs. V_{CC} (internal RC oscillator, 8MHz).

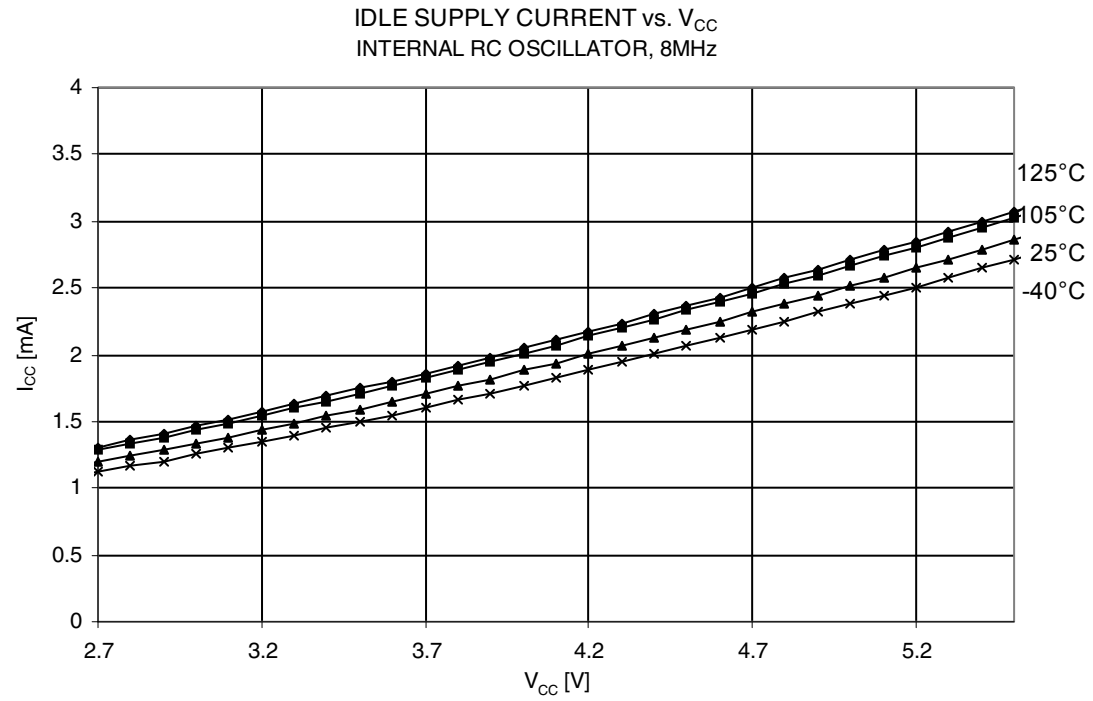
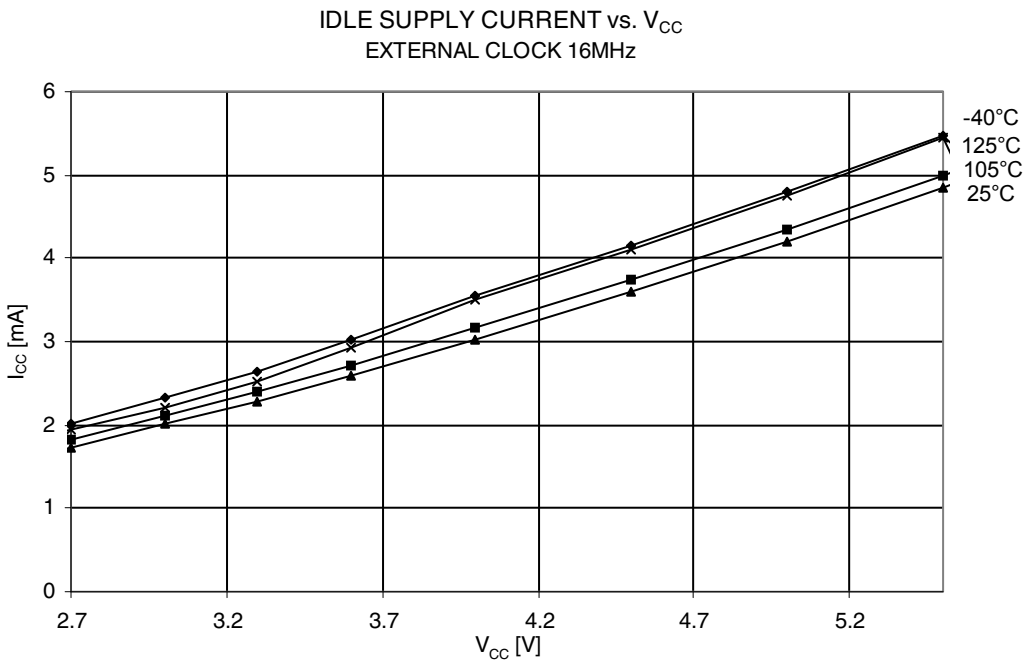


Figure 23-8. Idle supply current vs. V_{CC} (external clock, 16MHz).



23.3 Power-Down Supply Current

Figure 23-9. Power-down supply current vs. V_{CC} (watchdog timer disabled).

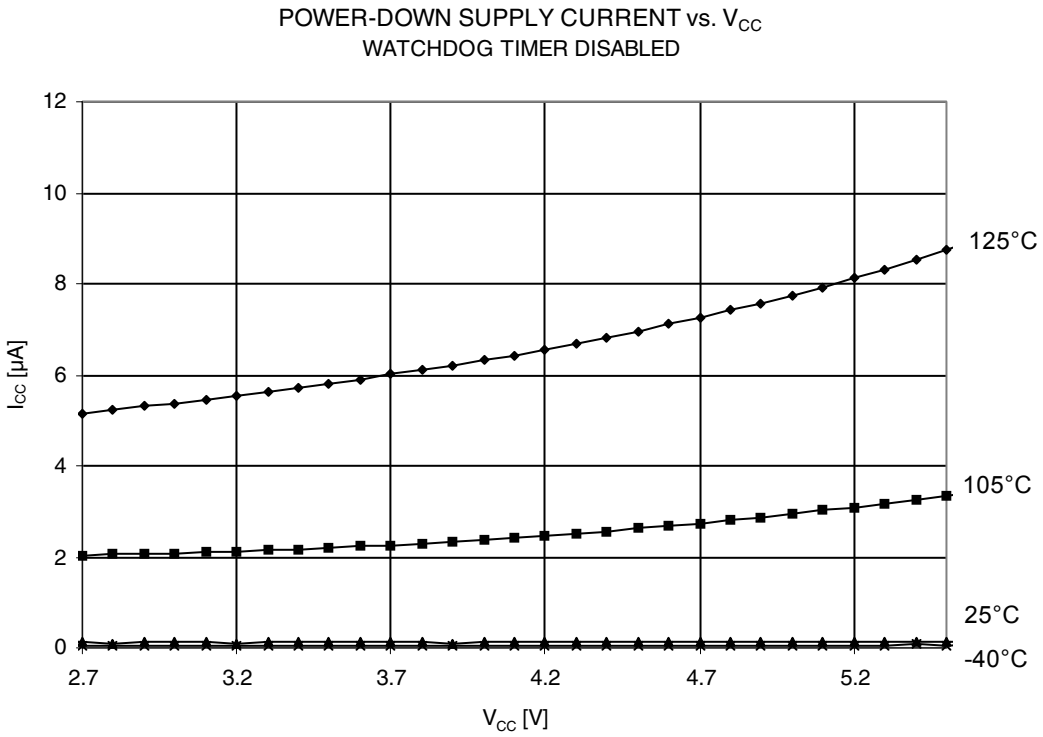
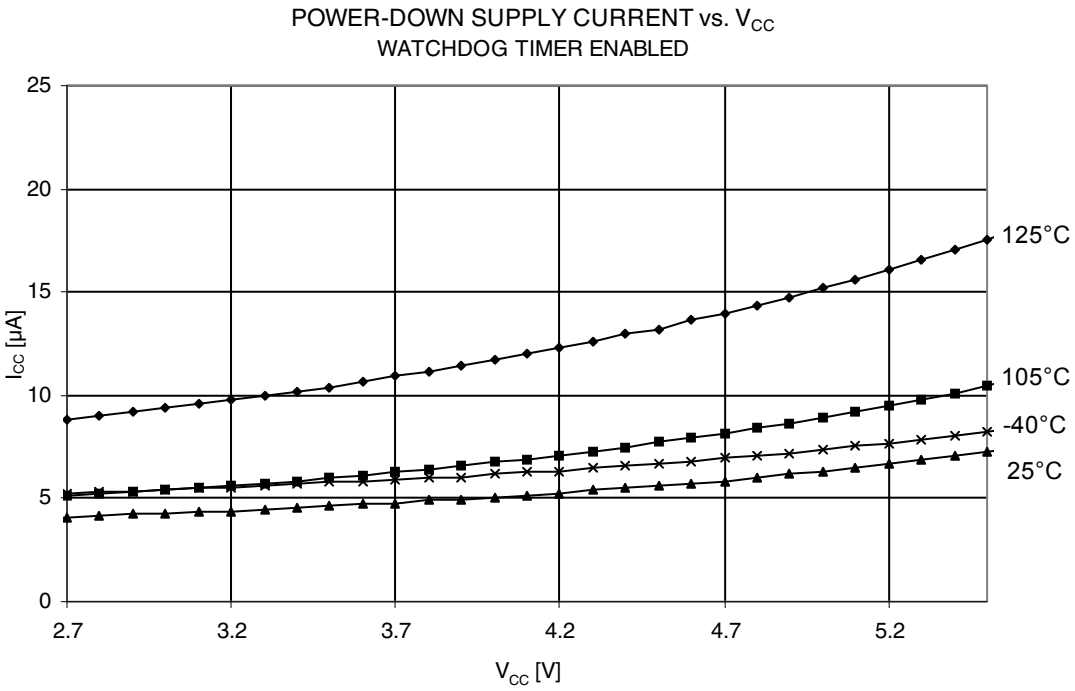


Figure 23-10. Power-down supply current vs. V_{CC} (watchdog timer enabled).



23.4 Pin Pull-up

Figure 23-11. I/O pin pull-up resistor current vs. input voltage (V_{CC} = 5V).

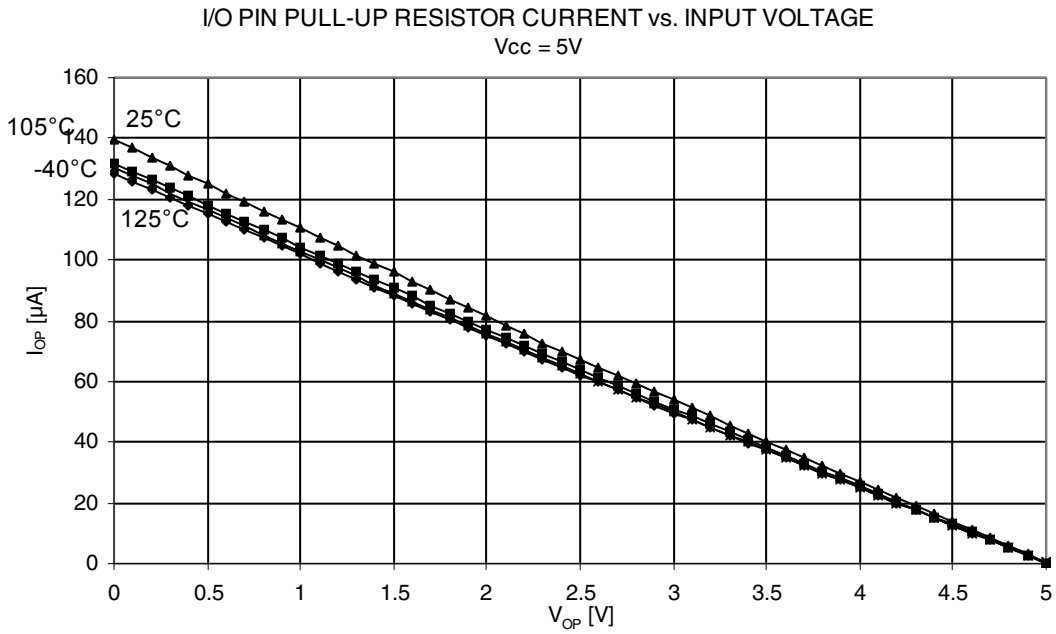


Figure 23-12. I/O pin pull-up resistor current vs. input voltage (V_{CC} = 2.7V).

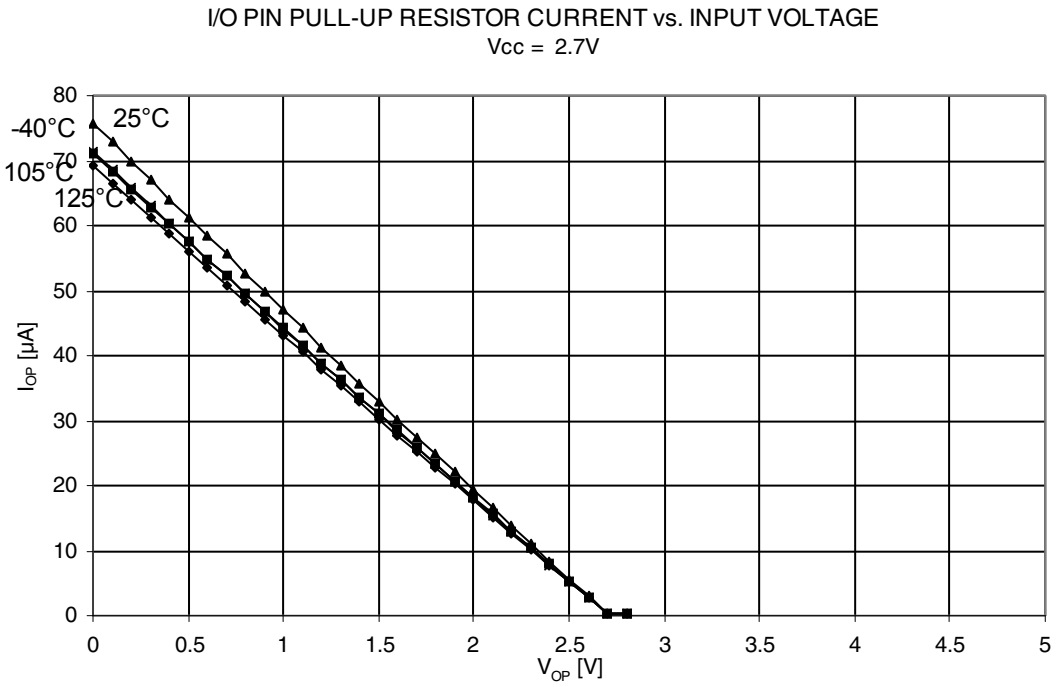


Figure 23-13. I/O pin pull-up resistor current vs. input voltage, PE1 & PE2 pins (V_{CC} = 5V).

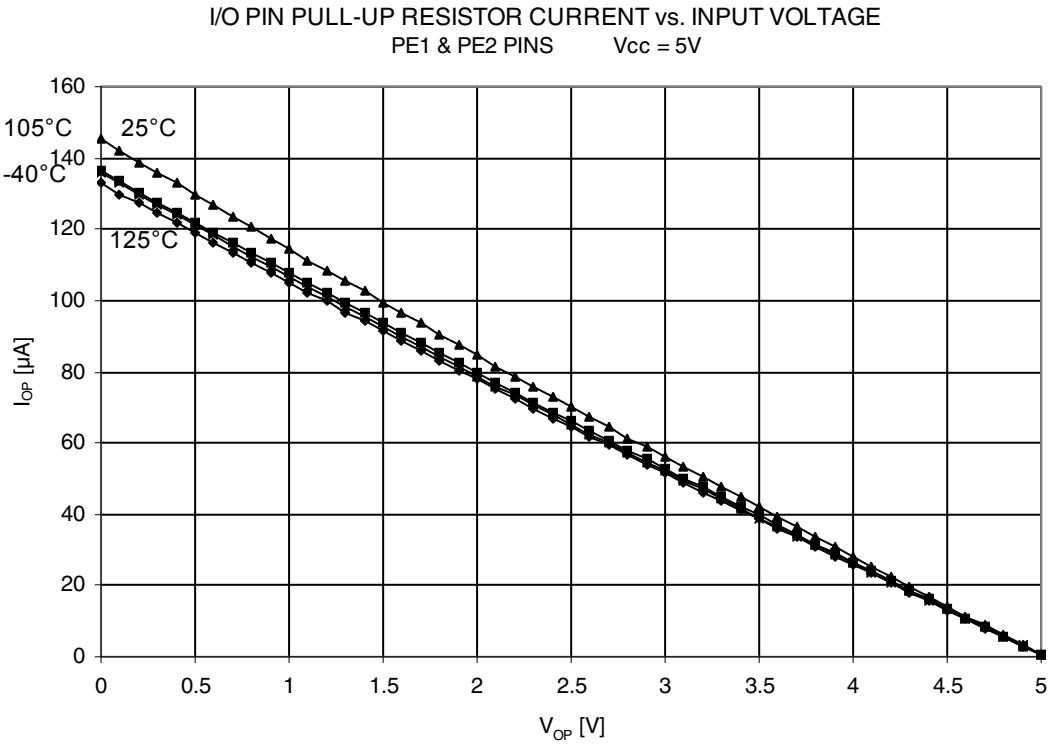


Figure 23-14. I/O pin pull-up resistor current vs. input voltage, PE1 & PE2 pins (V_{CC} = 2.7V).

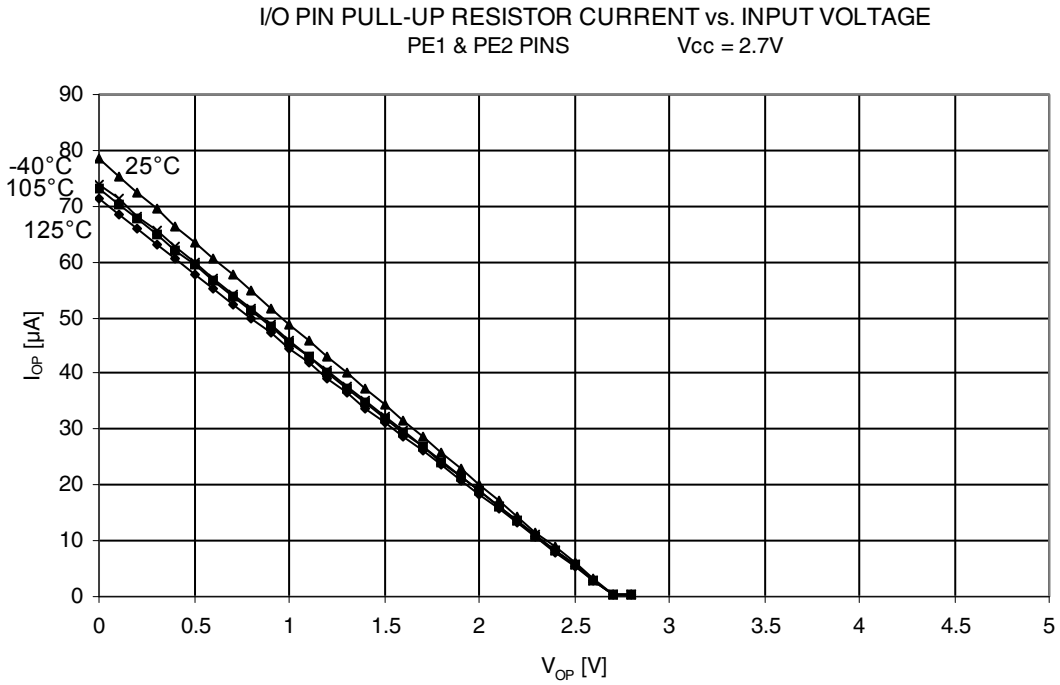


Figure 23-15. Reset pull-up resistor current vs. reset pin voltage (V_{CC} = 5V).

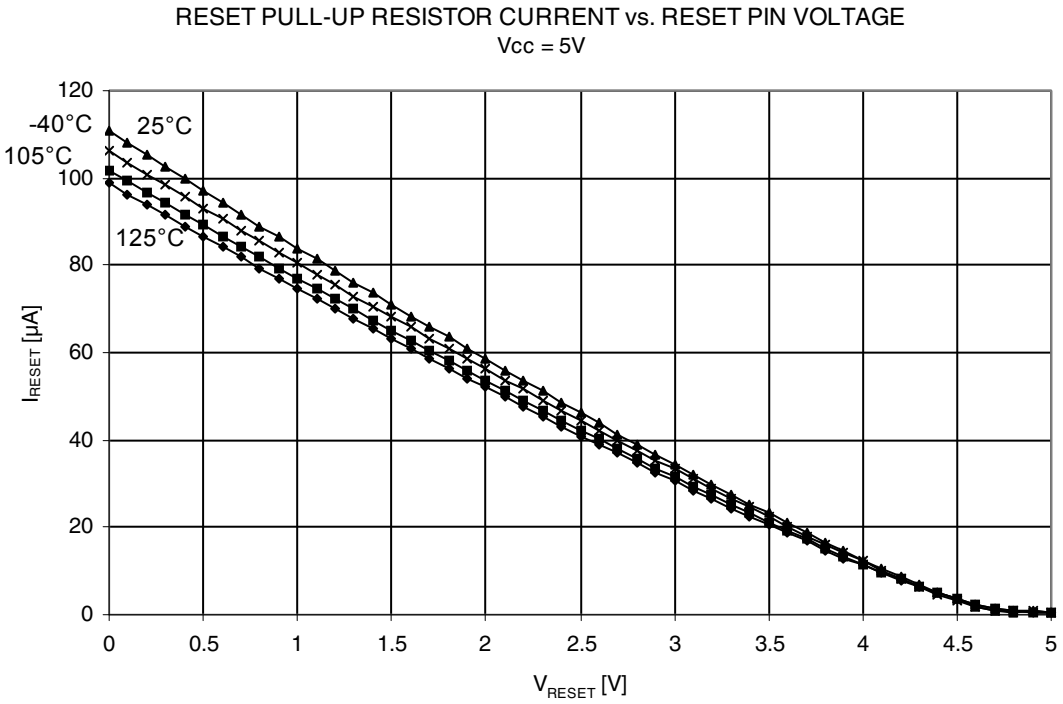
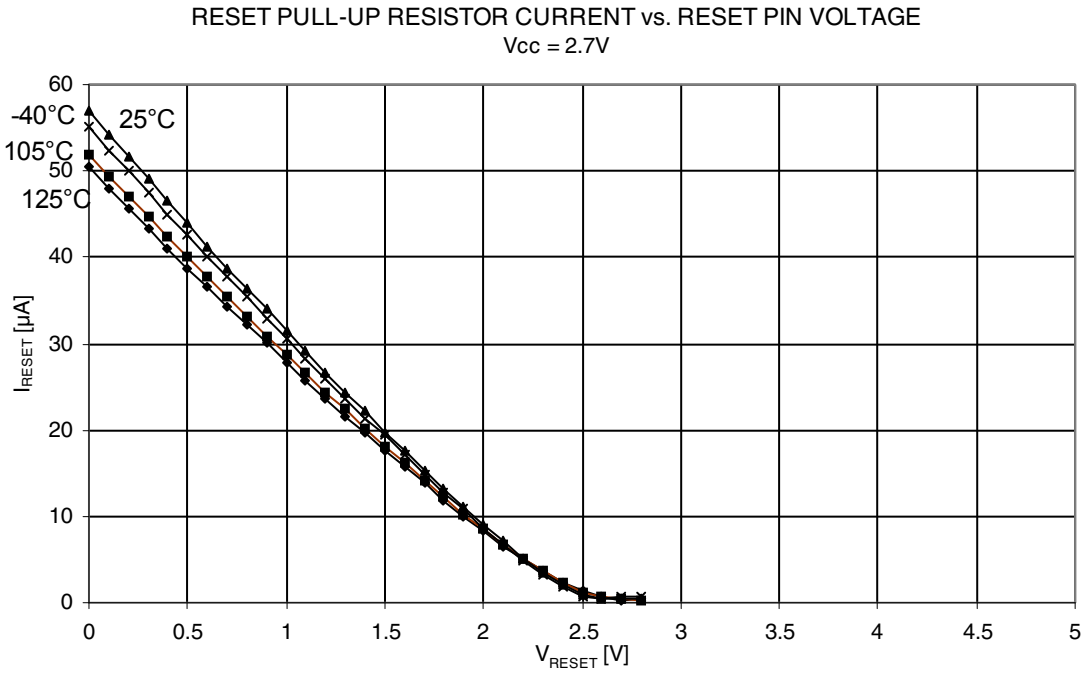


Figure 23-16. Reset pull-up resistor current vs. reset pin voltage (V_{CC} = 2.7V).



23.5 Pin output high voltage

Figure 23-17. I/O pin output voltage vs. source current (V_{CC} = 5V).

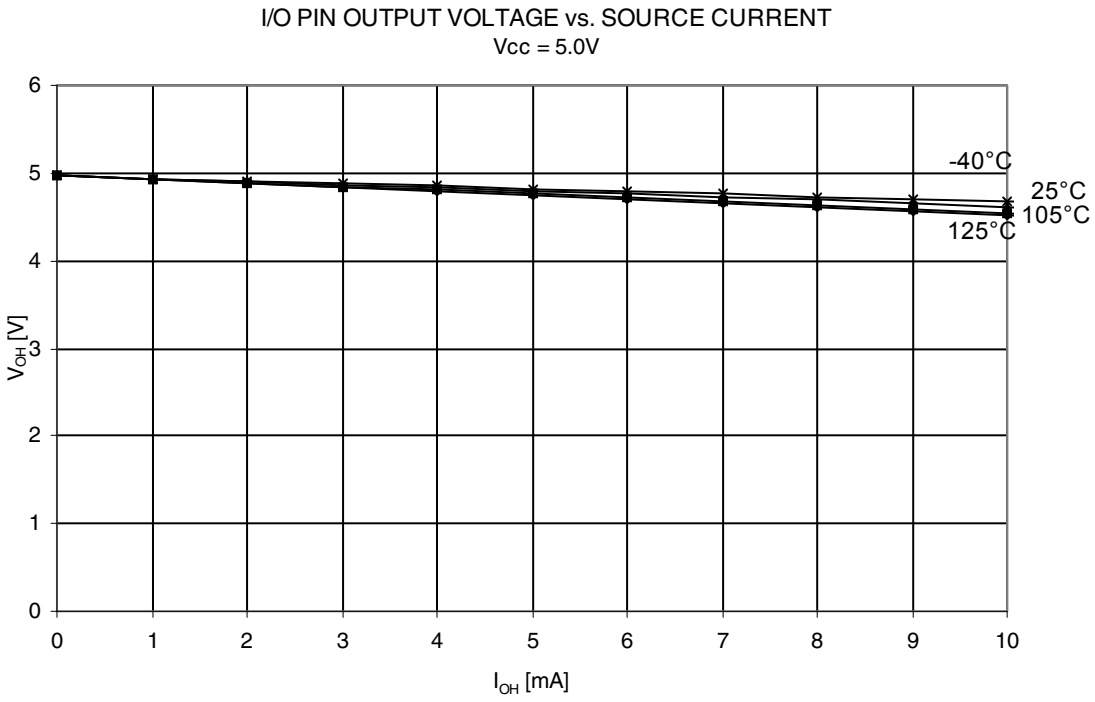
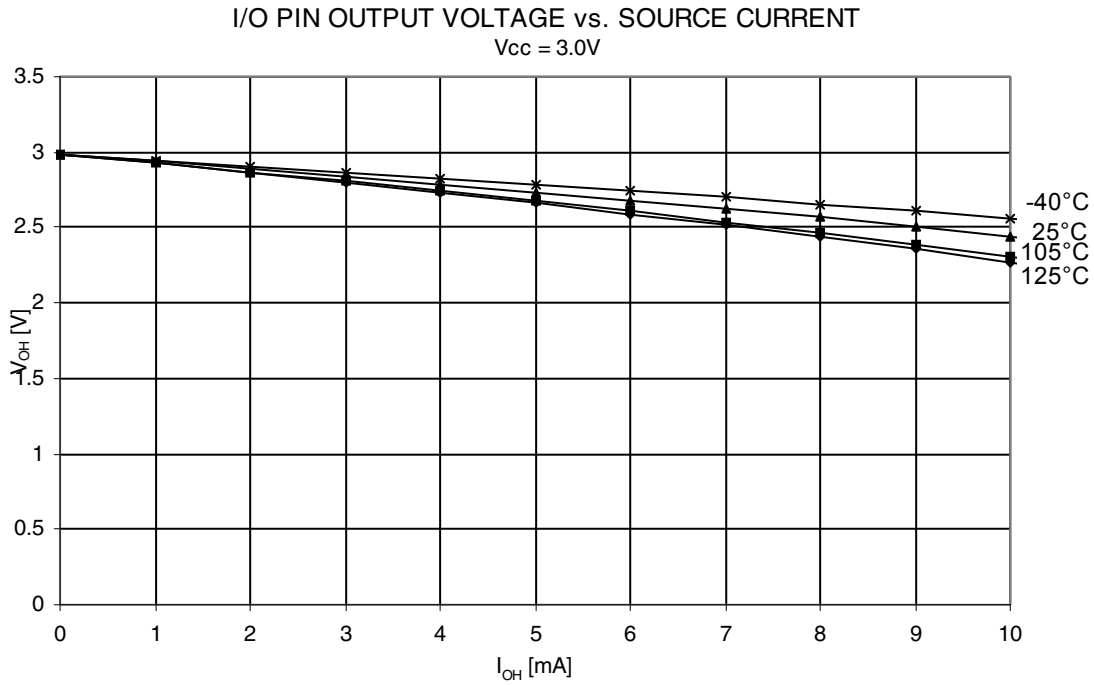


Figure 23-18. I/O pin output voltage vs. source current (V_{CC} = 3V).



23.6 Pin output low voltage

Figure 23-19. I/O pin output voltage vs. sink current (V_{CC} = 5V).

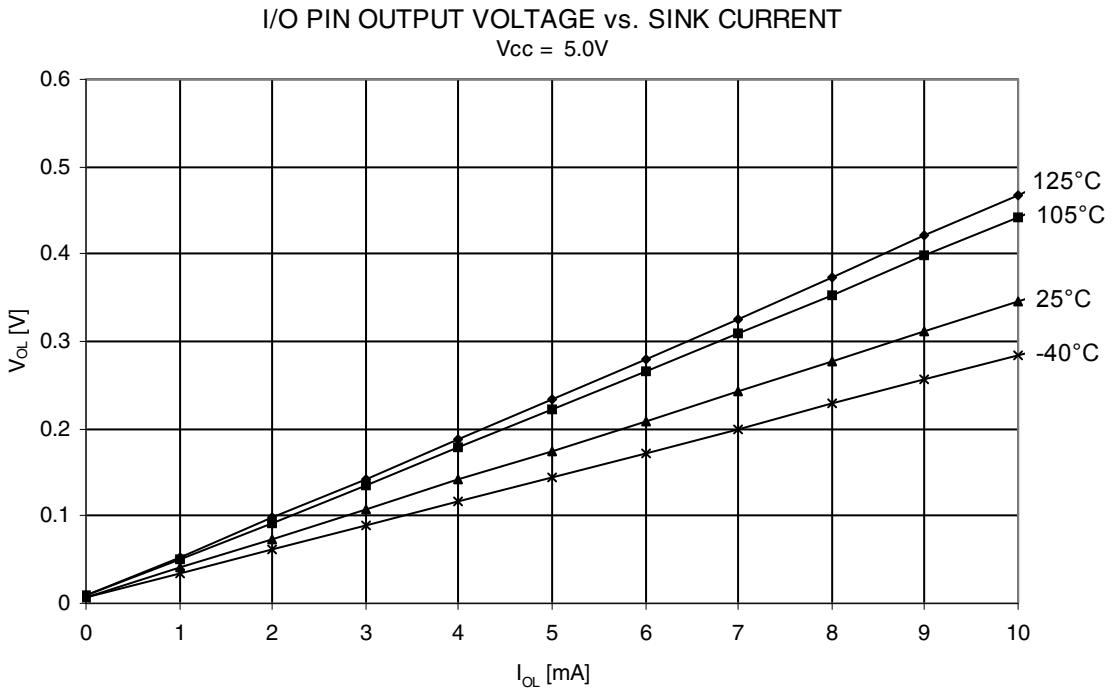
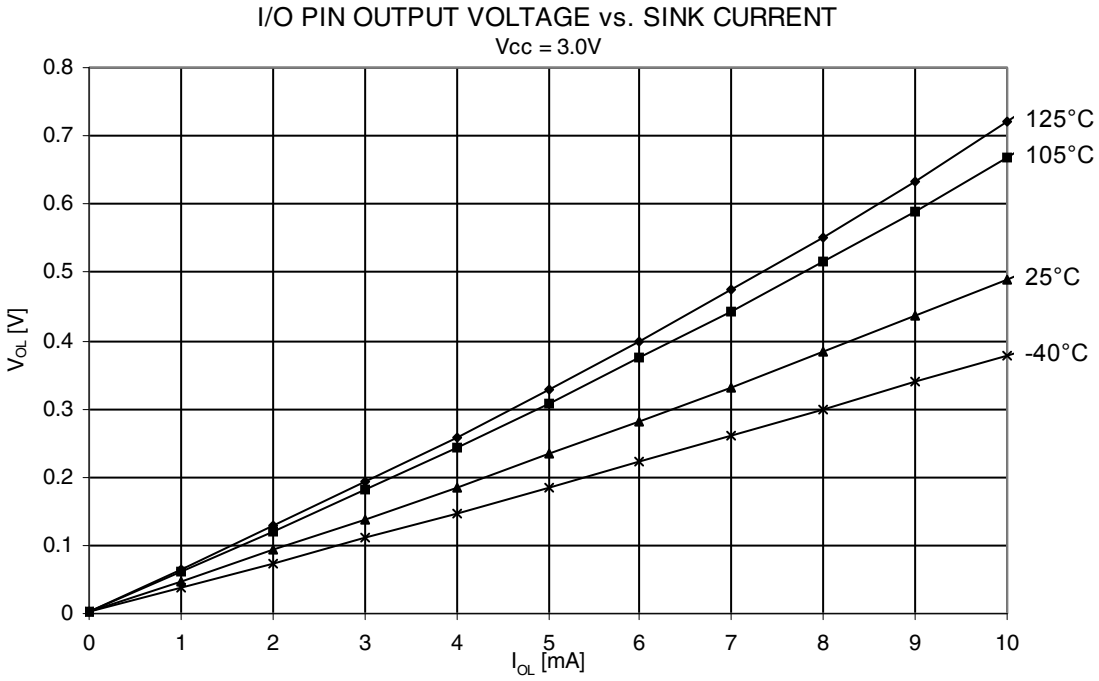


Figure 23-20. I/O pin output voltage vs. sink current (V_{CC} = 3V).



23.7 Pin Thresholds

Figure 23-21. I/O pin input threshold voltage vs. V_{CC} (VIL, I/O pin read As '0').

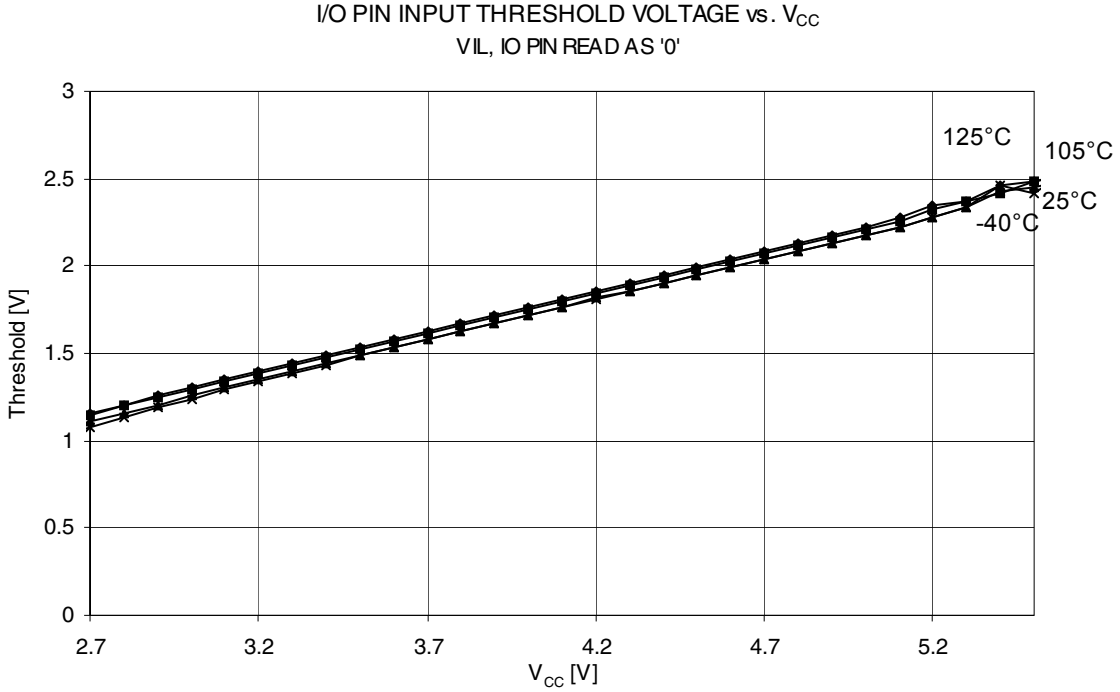
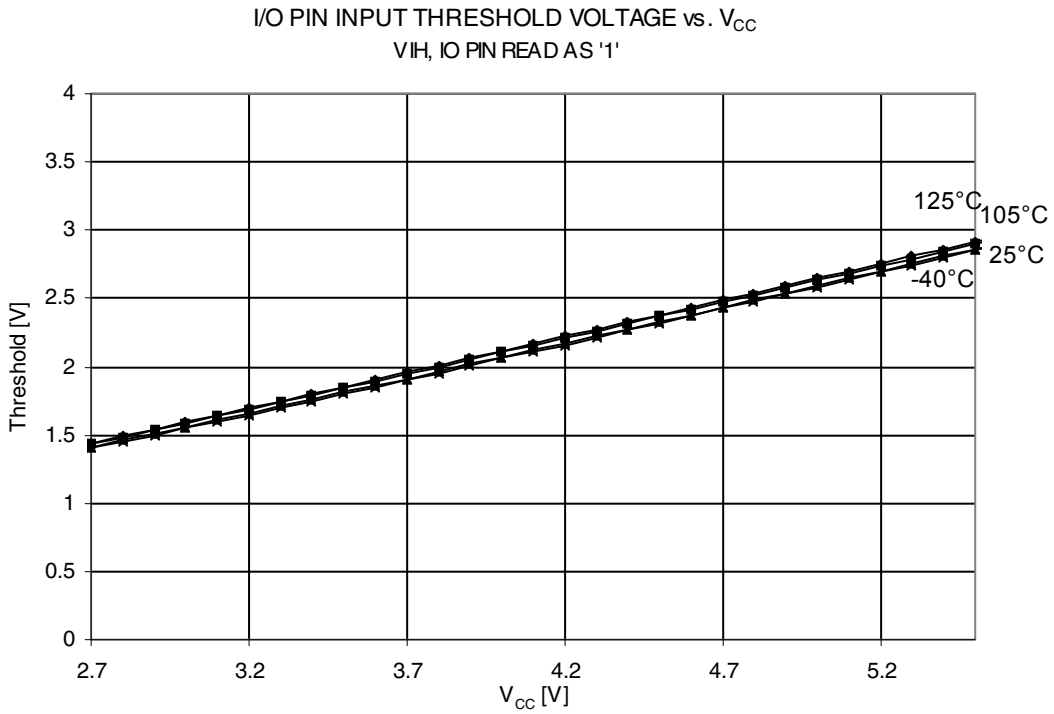


Figure 23-22. I/O pin input threshold voltage vs. V_{CC} (VIH, I/O pin read as '1').



23.8 BOD Thresholds

Figure 23-23. BOD thresholds vs. temperature (BODLEVEL is 4.3V).

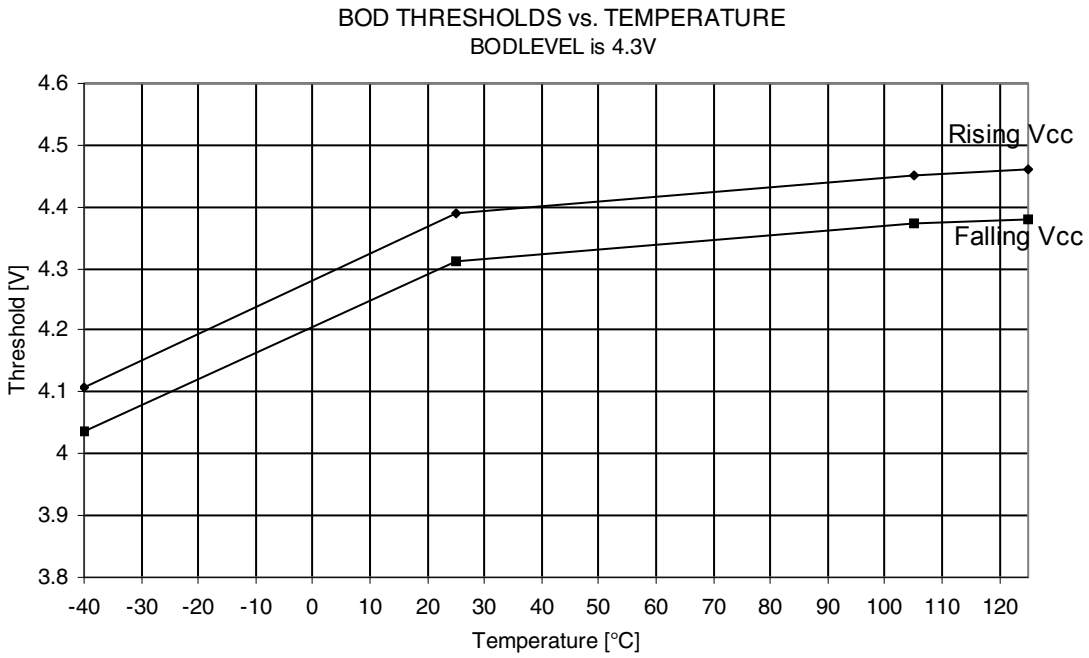
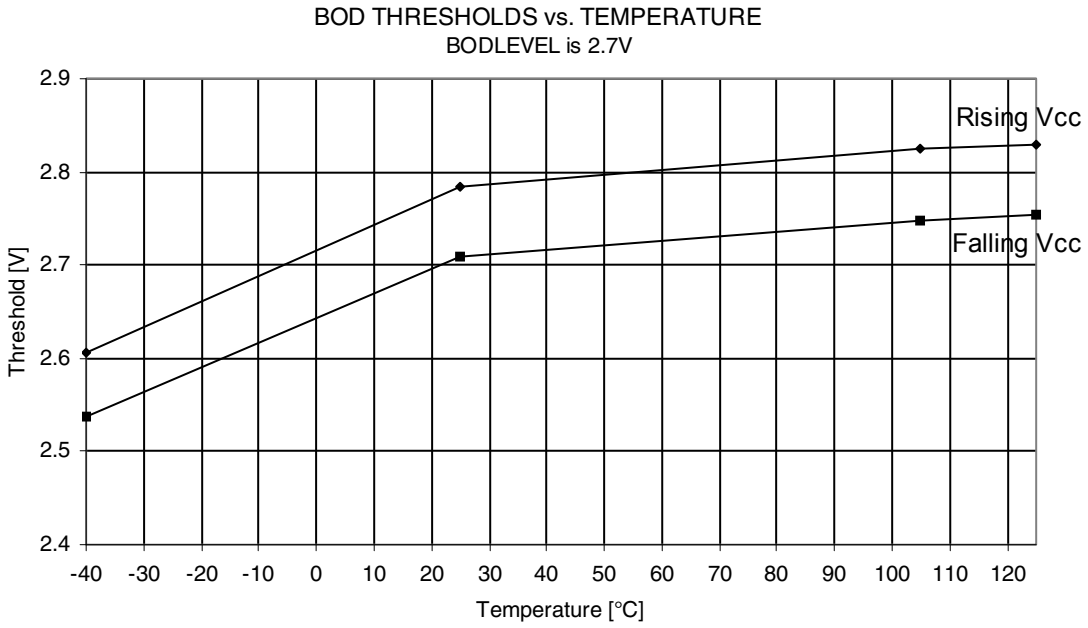


Figure 23-24. BOD thresholds vs. temperature (BODLEVEL is 2.7V).



23.9 Analog Reference

Figure 23-25. V_{REF} voltage vs. V_{CC}.

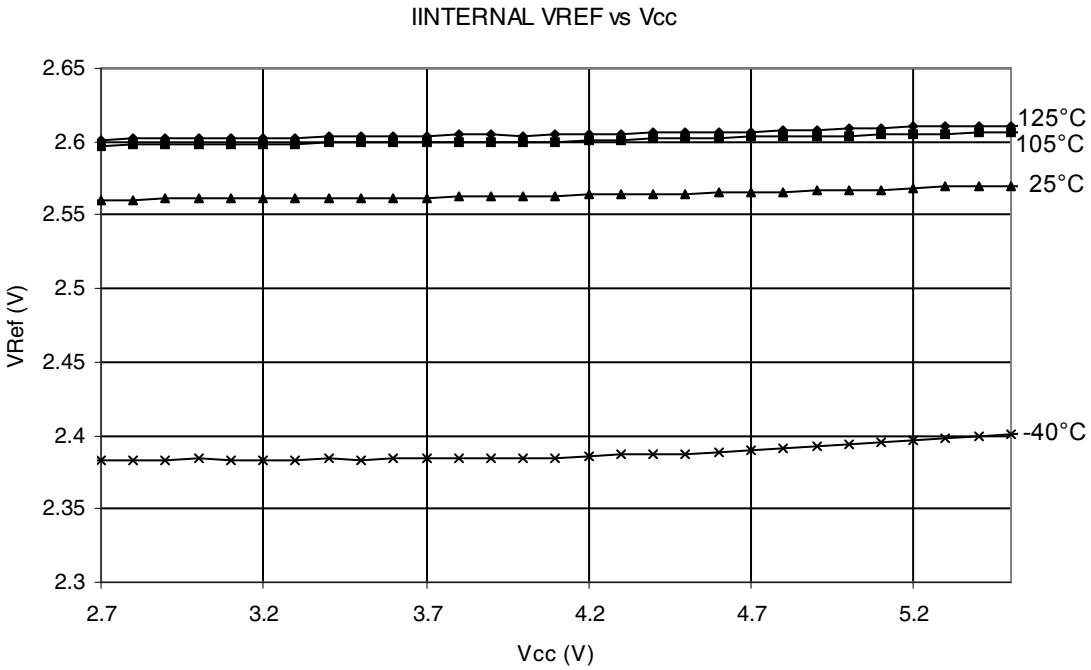
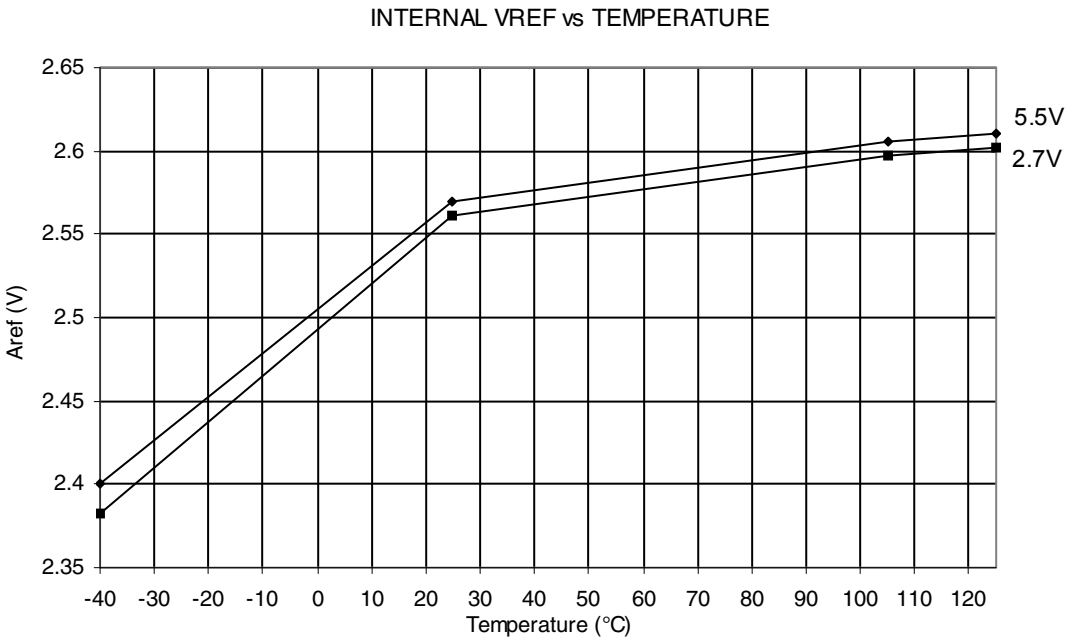


Figure 23-26. V_{REF} voltage vs. temperature.



23.10 Internal Oscillator Speed

Figure 23-27. Watchdog oscillator frequency vs. V_{CC}.

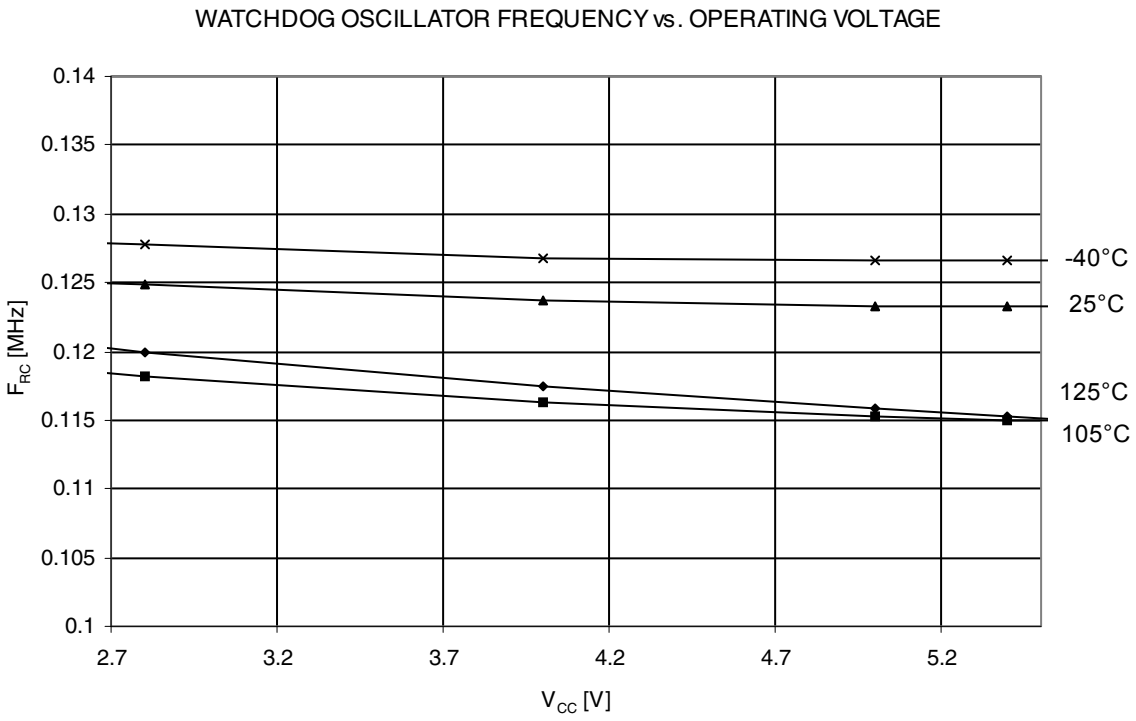


Figure 23-28. Calibrated 8MHz RC oscillator frequency vs. temperature.

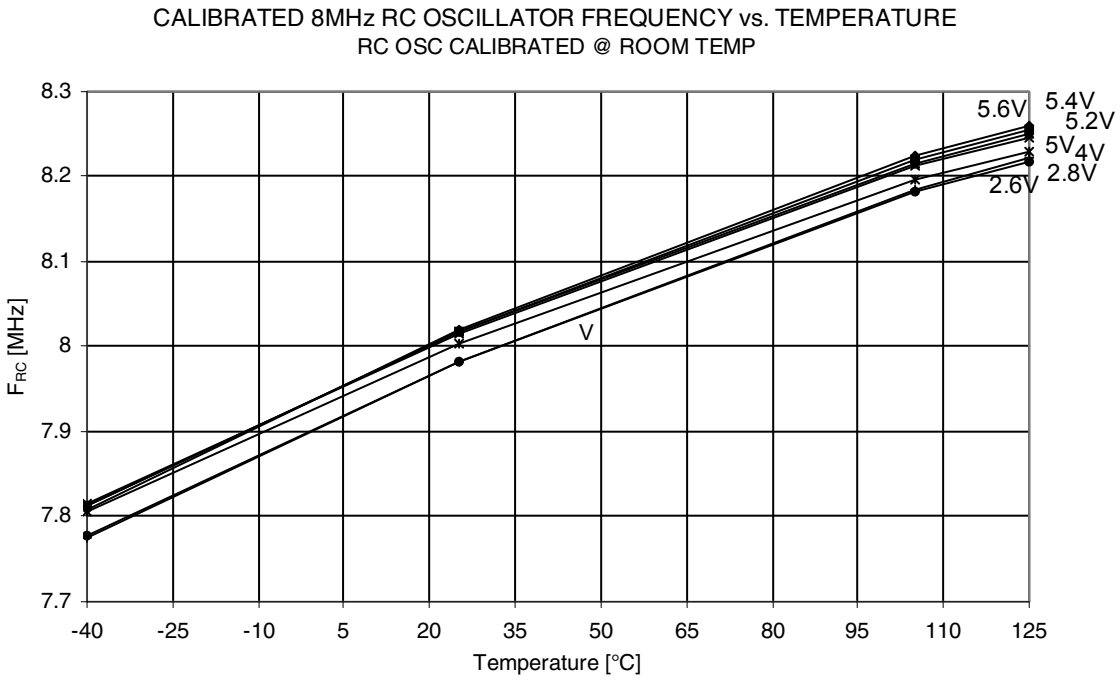


Figure 23-29. Calibrated 8MHz RC oscillator frequency vs. V_{CC}.

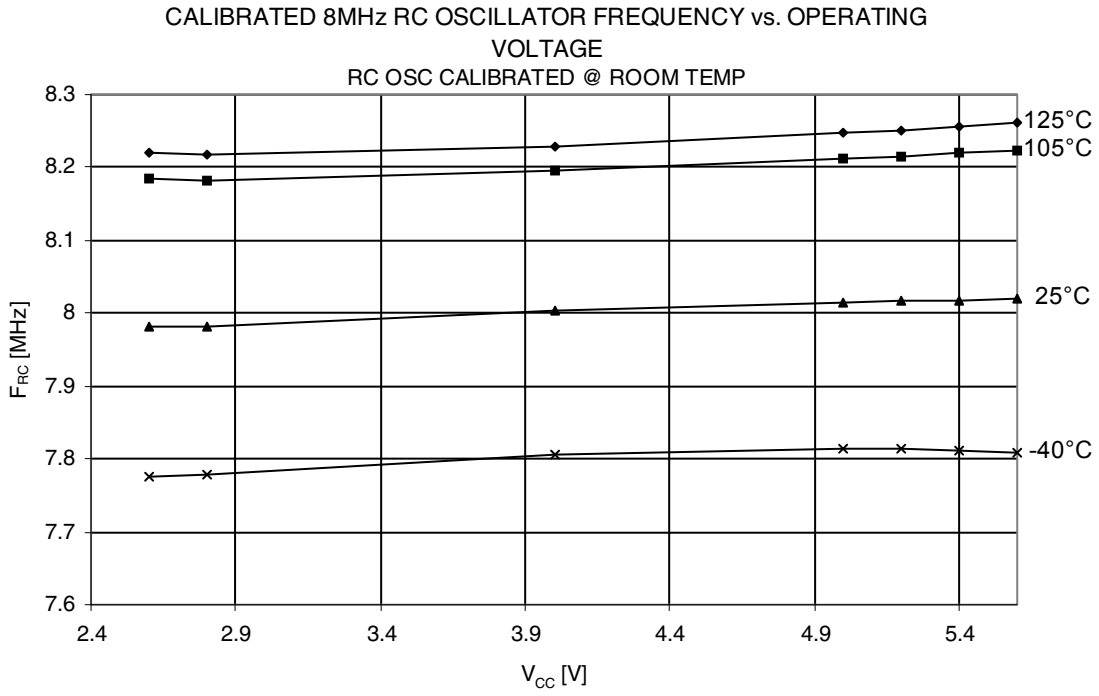
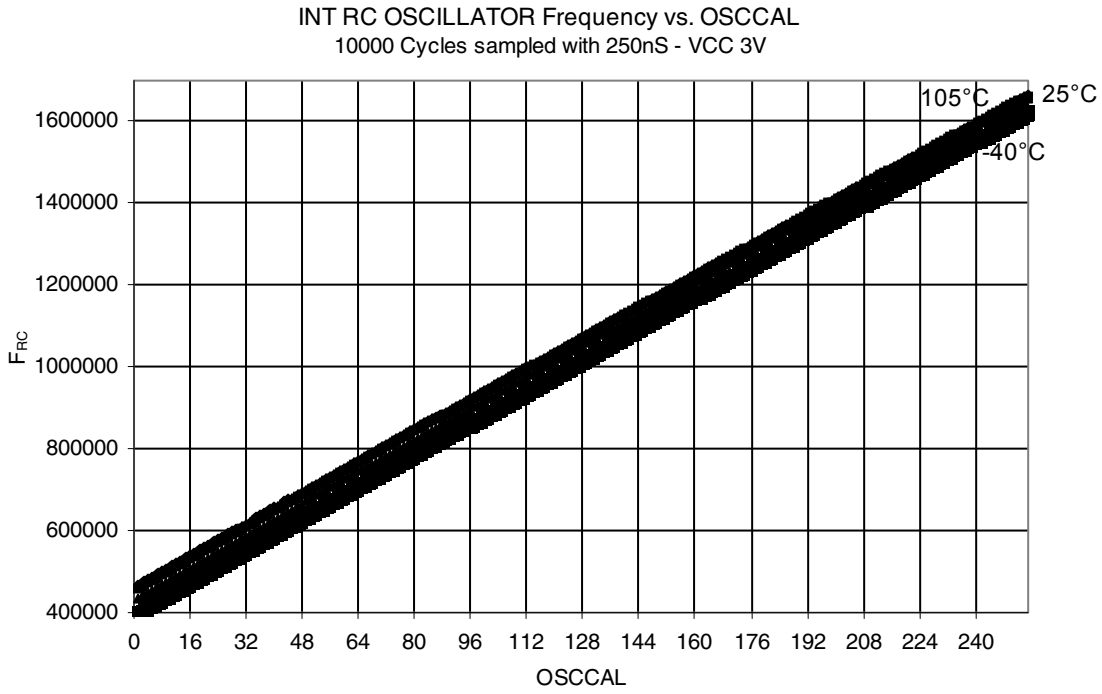


Figure 23-30. Calibrated 8MHz RC oscillator frequency vs. osccal value.



23.11 Current Consumption in Reset

Figure 23-31. Reset supply current vs. V_{CC} (0.1MHz - 1.0MHz, excluding current through the reset pull-up).

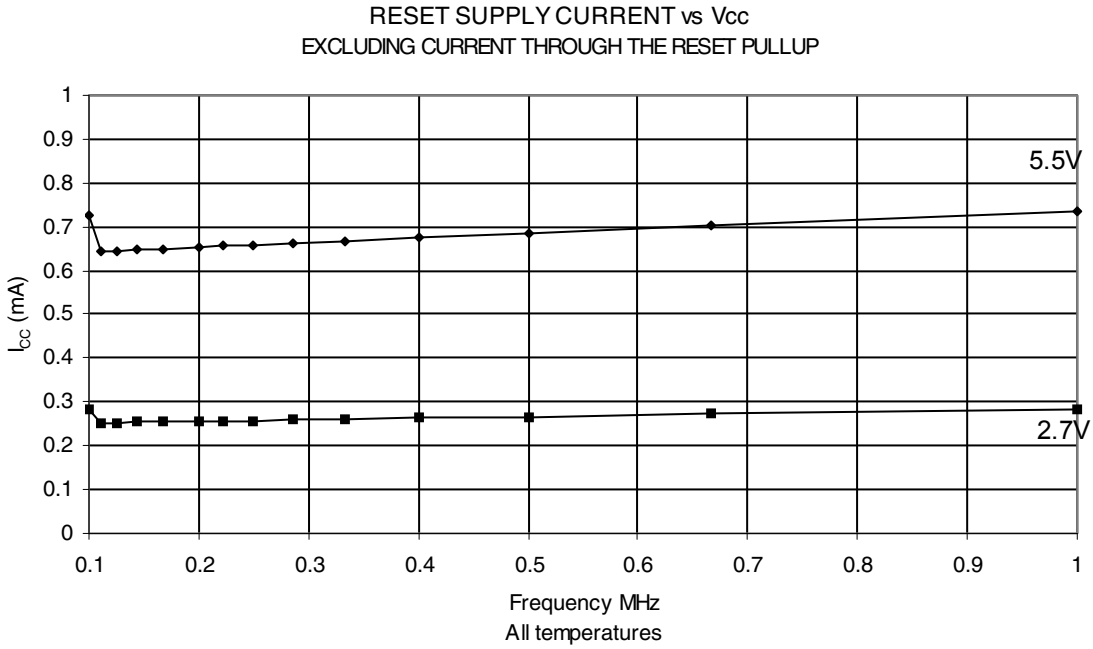
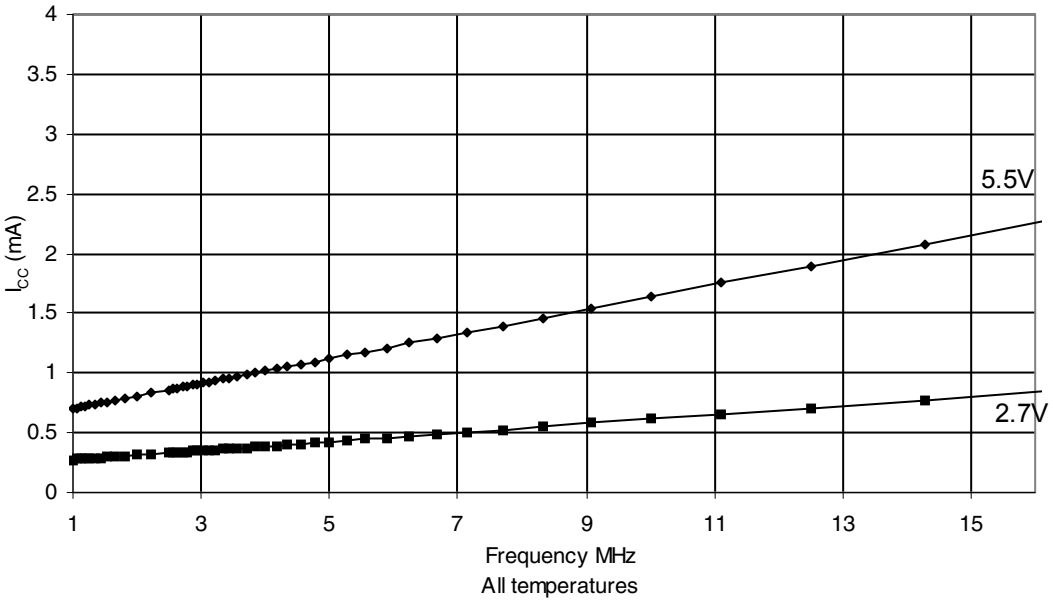


Figure 23-32. Reset supply current vs. V_{CC} (1MHz - 16MHz, excluding current through the reset pull-up).



24. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	Reserved	–	–	–	–	–	–	–	–	
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
(0xF8)	Reserved	–	–	–	–	–	–	–	–	
(0xF7)	Reserved	–	–	–	–	–	–	–	–	
(0xF6)	Reserved	–	–	–	–	–	–	–	–	
(0xF5)	Reserved	–	–	–	–	–	–	–	–	
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	Reserved	–	–	–	–	–	–	–	–	
(0xF2)	Reserved	–	–	–	–	–	–	–	–	
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	Reserved	–	–	–	–	–	–	–	–	
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	Reserved	–	–	–	–	–	–	–	–	
(0xE8)	Reserved	–	–	–	–	–	–	–	–	
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	Reserved	–	–	–	–	–	–	–	–	
(0xE5)	Reserved	–	–	–	–	–	–	–	–	
(0xE4)	Reserved	–	–	–	–	–	–	–	–	
(0xE3)	Reserved	–	–	–	–	–	–	–	–	
(0xE2)	Reserved	–	–	–	–	–	–	–	–	
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	Reserved	–	–	–	–	–	–	–	–	
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(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	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	**Reserved**	–	–	–	–	–	–	–	–	
(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)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAE)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAD)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAC)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAB)	**Reserved**	–	–	–	–	–	–	–	–	
(0xAA)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA9)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA8)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA7)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA6)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA5)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA4)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA3)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA2)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA1)	**Reserved**	–	–	–	–	–	–	–	–	
(0xA0)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9F)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9E)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9D)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9C)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9B)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9A)	**Reserved**	–	–	–	–	–	–	–	–	
(0x99)	**Reserved**	–	–	–	–	–	–	–	–	
(0x98)	**Reserved**	–	–	–	–	–	–	–	–	
(0x97)	**Reserved**	–	–	–	–	–	–	–	–	
(0x96)	**Reserved**	–	–	–	–	–	–	–	–	
(0x95)	**Reserved**	–	–	–	–	–	–	–	–	
(0x94)	**Reserved**	–	–	–	–	–	–	–	–	
(0x93)	**Reserved**	–	–	–	–	–	–	–	–	
(0x9r)	**Reserved**	–	–	–	–	–	–	–	–	
(0x91)	**Reserved**	–	–	–	–	–	–	–	–	
(0x90)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8F)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8E)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8D)	**ICR1H**	ICR115	ICR114	ICR113	ICR112	ICR111	ICR110	ICR19	ICR18	98
(0x8C)	**ICR1L**	ICR17	ICR16	ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	98
(0x8B)	**Reserved**	–	–	–	–	–	–	–	–	
(0x8A)	**TCCR1B**	ICNC1	ICES1	–	WGM13	–	CS12	CS11	CS10	97
(0x89)	**EICRA**	–	–	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	83
(0x88)	**OSCCAL**	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	39
(0x87)	PLLCSR	-	-	PLLF3	PLLF2	PLLF1	PLLF0	PLLE	PLOCK	41
(0x86)	**PRR**	PRPSC2	–	PRPSCR	PRTIM1	–	PRSPI	–	PRADC	49
(0x85)	**CLKSELR**	–	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	43
(0x84)	**CLKCSR**	CLKCCE	–	–	CLKRDY	CLKC3	CLKC2	CLKC1	CLKC0	42
(0x83)	**CLKPR**	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	40
(0x82)	**WDTCSR**	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	59
(0x81)	**BGCCR**		–	–	–	BGCC3	BGCC2	BGCC1	BGCC0	190
(0x80)	**BGCRR**	–	–	–	–	BGCR3	BGCR2	BGCR1	BGCR0	191
(0x7F)	**AC3CON**	AC3EN	AC3IE	AC3IS1	AC3IS0	AC3OEA	AC3M2	AC3M1	AC3M0	199
(0x7E)	**AC2CON**	AC2EN	AC2IE	AC2IS1	AC2IS0	–	AC2M2	AC2M1	AC2M0	198

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	**AC1CON**	AC1EN	AC1IE	AC1IS1	AC1IS0	–	AC1M2	AC1M1	AC1M0	197
(0x7C)	**AC3ECON**	–	–	AC3OI	AC3OE	–	AC3H2	AC3H1	AC3H0	200
(0x7B)	**AC2ECON**	–	–	AC2OI	AC2OE	–	AC2H2	AC2H1	AC2H0	200
(0x7A)	**AC1ECON**	–	–	AC1OI	AC1OE	AC1ICE	AC1H2	AC1H1	AC1H0	200
(0x79)	**AMP0CSR**	AMP0EN	AMP0IS	AMP0G1	AMP0G0	AMP0GS	–	AMP0TS1	AMP0TS0	225
(0x78)	**DIDR1**	–	–	–	–	ACMP1MD	AMP0+D	ADC10D	ADC9D	222
(0x77)	**DIDR0**	ADC8D/ACMP3D	ADC7D/AMP0-D	ADC5D/ACMP2D	ADC4D/ACMP3M	ADC3D/ACMPMD	ADC2D/ACMP2M	ADC1D	ADC0D/ACMP1D	222
(0x76)	**DACON**	DAATE	DATS2	DATS1	DATS0	–	DALA	–	DAEN	228
(0x75)	**Room for analog test registers**									
(0x74)										
(0x73)										
(0x72)										
(0x71)	PASDLY2	PASDLY2[7:0]								139
(0x70)	PCNFE2	PASDLKn2	PASDLKn1	PASDLKn0	PBFMn1	PELEVnA1	PELEVnB1	PISEL0A1	PISEL0B1	137
(0x6F)	POM2	POMV2B3	POMV2B2	POMV2B1	POMV2B0	POMV2A3	POMV2A2	POMV2A1	POMV2A0	142
(0x6E)	PSOC2	POS23	POS22	PSYNC21	PSYNC20	POEN2D	POEN2B	POEN2C	POEN2A	134
(0x6D)	PICR2H	PCST2	–	–	–	PICR2[11:8]				142
(0x6C)	PICR2L	PICR2[7:0]								142
(0x6B)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6A)	PSOC0	PISEL0A1	PISEL0B1	PSYNC01	PSYNC00	–	POEN0B	–	POEN0A	171
(0x69)	PICR0H	PCST0	–	–	–	PICR0[11:8]				177
(0x68)	PICR0L	PICR0[7:0]								177
(0x67)	PFRC2B	PCAE2B	PISEL2B	PELEV2B	PFLTE2B	PRFM2B3	PRFM2B2	PRFM2B1	PRFM2B0	141
(0x66)	PFRC2A	PCAE2A	PISEL2A	PELEV2A	PFLTE2A	PRFM2A3	PRFM2A2	PRFM2A1	PRFM2A0	141
(0x65)	OCR2SAH	–	–	–	–	OCR2SA[11:8]				135
(0x64)	OCR2SAL	OCR2SA[7:0]								135
(0x63)	PFRC0B	PCAE0B	PISEL0B	PELEV0B	PFLTE0B	PRFM0B3	PRFM0B2	PRFM0B1	PRFM0B0	175
(0x62)	PFRC0A	PCAE0A	PISEL0A	PELEV0A	PFLTE0A	PRFM0A3	PRFM0A2	PRFM0A1	PRFM0A0	175
(0x61)	OCR0SAH	–	–	–	–	OCR0SA[11:8]				172
(0x60)	OCR0SAL	OCR0SA[7:0]								172
0x3F (0x5F)	**SREG**	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	**SPH**	–	–	–	–	SP11	SP10	SP9	SP8	12
0x3D (0x5D)	**SPL**	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	**Reserved**	–	–	–	–	–	–	–	–	
0x3B (0x5B)	**TCNT1H**	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18	98
0x3A (0x5A)	**TCNT1L**	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	98
0x39 (0x59)	**DACH**	- / DAC9	- / DAC8	- / DAC7	- / DAC6	- / DAC5	- / DAC4	DAC9 / DAC3	DAC8 / DAC2	229
0x38 (0x58)	**DACL**	DAC7 / DAC1	DAC6 /DAC0	DAC5 / -	DAC4 / -	DAC3 / -	DAC2 / -	DAC1 / -	DAC0 /	229
0x37 (0x57)	**SPMCSR**	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	238
0x36 (0x56)	**SPDR**	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	188
0x35 (0x55)	**MCUCR**	–	–	–	PUD	RSTDIS	CKRC81	IVSEL	IVCE	55 & 74
0x34 (0x54)	**MCUSR**	–	–	–	–	WDRF	BORF	EXTRF	PORF	54
0x33 (0x53)	**SMCR**	–	–	–	–	SM2	SM1	SM0	SE	48
0x32 (0x52)	MSMCR	Monitor Stop Mode Control Register								reserved
0x31 (0x51)	DWDR	DWDR[7:0]								232
0x30 (0x50)	**Reserved**	–	–	–	–	–	–	–	–	
0x2F (0x4F)	OCR2RAH	–	–	–	–	OCR2RA[11:8]				135
0x2E (0x4E)	OCR2RAL	OCR2RA[7:0]								135
0x2D (0x4D)	**ADCH**	- / ADC9	- / ADC8	- / ADC7	- / ADC6	- / ADC5	- / ADC4	ADC9 / ADC3	ADC8 / ADC2	221
0x2C (0x4C)	**ADCL**	ADC7 / ADC1	ADC6 / ADC0	ADC5 / -	ADC4 / -	ADC3 / -	ADC2 / -	ADC1 / -	ADC0 /	221
0x2B (0x4B)	OCR0RAH	–	–	–	–	OCR0RA[11:8]				172
0x2A (0x4A)	OCR0RAL	OCR0RA[7:0]								172
0x29 (0x49)	OCR2RBH	OCR2RB[15:12]				OCR2RB[11:8]				136
0x28 (0x48)	OCR2RBL	OCR2RB[7:0]								136
0x27 (0x47)	OCR2SBH	–	–	–	–	OCR2SB[11:8]				136
0x26 (0x46)	OCR2SBL	OCR2SB[7:0]								136
0x25 (0x45)	OCR0RBH	OCR0RB[15:12]				OCR0RB[11:8]				172
0x24 (0x44)	OCR0RBL	OCR0RB[7:0]								172
0x23 (0x43)	OCR0SBH	–	–	–	–	OCR0SB[11:8]				172
0x22 (0x42)	OCR0SBL	OCR0SB[7:0]								172
0x21 (0x41)	**EIMSK**	–	–	–	–	–	INT2	INT1	INT0	84
0x20 (0x40)	**EIFR**	–	–	–	–	–	INTF2	INTF1	INTF0	84
0x1F (0x3F)	**EEARH**	–	–	–	–	–	–	–	EEAR8	19
0x1E (0x3E)	**EEARL**	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	19
0x1D (0x3D)	**EEDR**	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	19
0x1C (0x3C)	**EECR**	NVMBSY	EEPAGE	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE	19

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	**GPIOR2**	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	26
0x1A (0x3A)	**GPIOR1**	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	26
0x19 (0x39)	**GPIOR0**	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	26
0x18 (0x38)	**SPSR**	SPIF	WCOL	–	–	–	–	–	SPI2X	188
0x17 (0x37)	**SPCR**	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	186
0x16 (0x36)	PCTL2	PPRE21	PPRE20	PBFM2	PAOC2B	PAOC2A	PARUN2	PCCYC2	PRUN2	140
0x15 (0x35)	PCNF2	PFIFTY2	PALOCK2	PLOCK2	PMODE21	PMODE20	POP2	PCLKSEL2	POME2	136
0x14 (0x34)	PIFR2	POAC2B	POAC2A	PSEI2	PEV2B	PEV2A	PRN21	PRN20	PEOP2	144
0x13 (0x33)	PIM2	-	-	PSEIE2	PEVE2B	PEVE2A	-	PEOEPE2	PEOPE2	143
0x12 (0x32)	PCTL0	PPRE01	PPRE00	PBFM01	PAOC0B	PAOC0A	PBFM00	PCCYC0	PRUN0	174
0x11 (0x31)	PCNF0	PFIFTY0	PALOCK0	PLOCK0	PMODE01	PMODE00	POP0	PCLKSEL0	–	173
0x10 (0x30)	PIFR0	POAC0B	POAC0A	–	PEV0B	PEV0A	PRN01	PRN00	PEOP0	178
0x0F (0x2F)	PIM0	-	-	–	PEVE0B	PEVE0A	–	PEOEPE0	PEOPE0	177
0x0E (0x2E)	**PORTE**	–	–	–	–	–	PORTE2	PORTE1	PORTE0	82
0x0D (0x2D)	**DDRE**	–	–	–	–	–	DDE2	DDE1	DDE0	82
0x0C (0x2C)	**PINE**	–	–	–	–	–	PINE2	PINE1	PINE0	82
0x0B (0x2B)	**PORTD**	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	82
0x0A (0x2A)	**DDRD**	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	82
0x09 (0x29)	**PIND**	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	82
0x08 (0x28)	**ADMUX**	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	217
0x07 (0x27)	**ADCSRB**	ADHSM	ADNCDIS	–	ADSSEN	ADTS3	ADTS2	ADTS1	ADTS0	220
0x06 (0x26)	**ADCSRA**	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	219
0x05 (0x25)	**PORTB**	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	81
0x04 (0x24)	**DDRB**	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	82
0x03 (0x23)	**PINB**	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	82
0x02 (0x22)	**TIFR1**	–	–	ICF1	–	–	–	–	TOV1	99
0x01 (0x21)	**TIMSK1**	–	–	ICIE1	–	–	–	–	TOIE1	99
0x00 (0x20)	**ACSR**	AC3IF	AC2IF	AC1IF	–	AC3O	AC2O	AC1O	–	201

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 AT90PWM81/161 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.

25. 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 Å 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 Å 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					
JMP (AT90PWM161)	k	Direct jump	PC ¬ k	None	3
CALL (AT90PWM161)	k	Direct call	PC ¬ k	None	4
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 Å V= 0) then PC ¬ PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N Å 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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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
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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
PUSH	Rr	Push Register on Stack	STACK ¬ Rr	None	2
POP	Rd	Pop Register from Stack	Rd ¬ STACK	None	2

MCU CONTROL INSTRUCTIONS					
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

26. Ordering Information

Speed (MHz)	Power supply	Ordering code	Package	Operation range
16	2.7V - 5.5V	AT90PWM81-16ME	QFN32 (1)	Extended (-40°C to 105°C)
		AT90PWM81-16SE	SO20	
		AT90PWM81-16MF	QFN32 (2)	Extended (-40°C to 125°C)
		AT90PWM81-16SF	SO20	

16	2.7V - 5.5V	AT90PWM161-16MN	QFN32 (3)	Extended (-40°C to 105°C)
		AT90PWM161-16SN	SO20	
		AT90PWM161-16MF	QFN32 (4)	Extended (-40°C to 125°C)
		AT90PWM161-16SF	SO20	

Note: All packages are Pb free, fully LHF

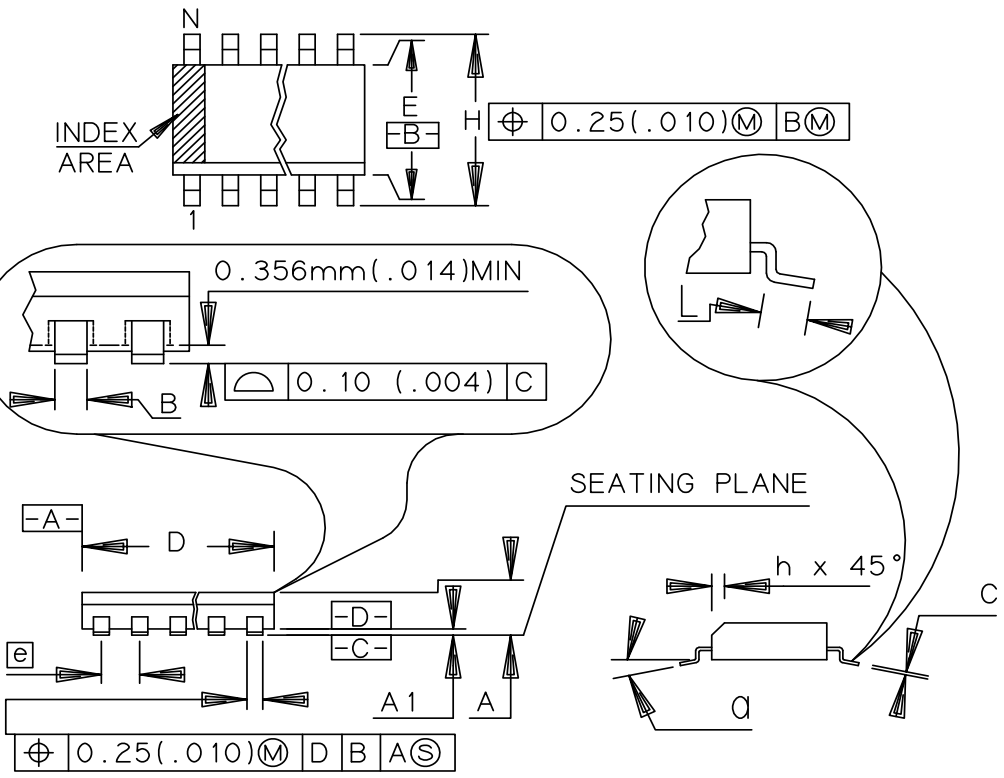
Note: This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

Note: Parts numbers are for shipping in sticks (SO) or in trays (QFN). Thes devices can also be supplied in Tape and Reel. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

Note:
1. Marking on the package is PWM81-MN.
2. Marking on the package is PWM81-MF.
3. Marking on the package is PWM161-MN.
4. Marking on the package is PWM161-MF.

Package type	
SO20	TG, 20-lead, 0.300” body width plastic gull wing small outline package (SOIC)
QFN32	PN, 32-lead, 5.0 x 5.0mm body, 0.50mm pitch quad flat no lead package (QFN)

26.1 SO20

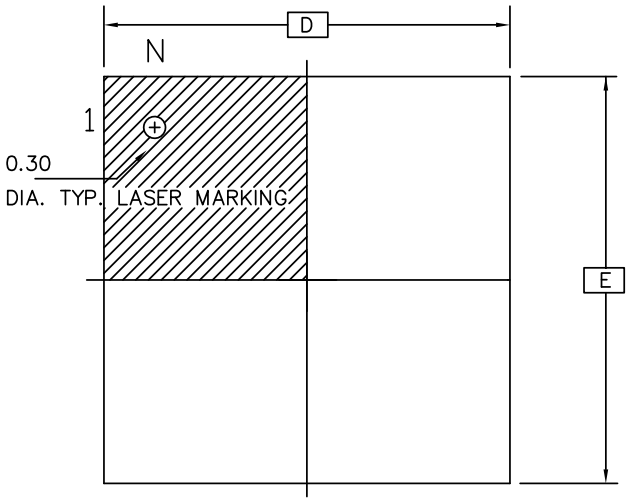


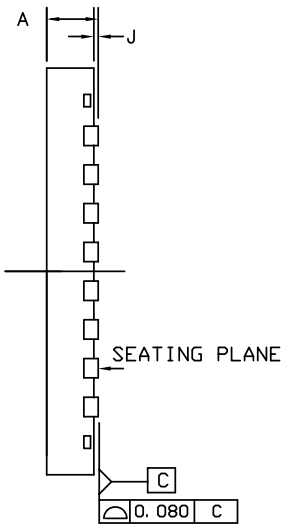
	MM		INCH	
A	2.35	2.65	.093	.104
A1	0.10	0.30	.004	.012
B	0.35	0.49	.014	.019
C	0.23	0.32	.009	.013
D	12.60	13.00	.496	.512
E	7.40	7.60	.291	.299
e	1.27	BSC	.050	BSC
H	10.00	10.65	.394	.419
h	0.25	0.75	.010	.029
L	0.40	1.27	.016	.050
N	20		20	
a	0°		8°	

07/27/07

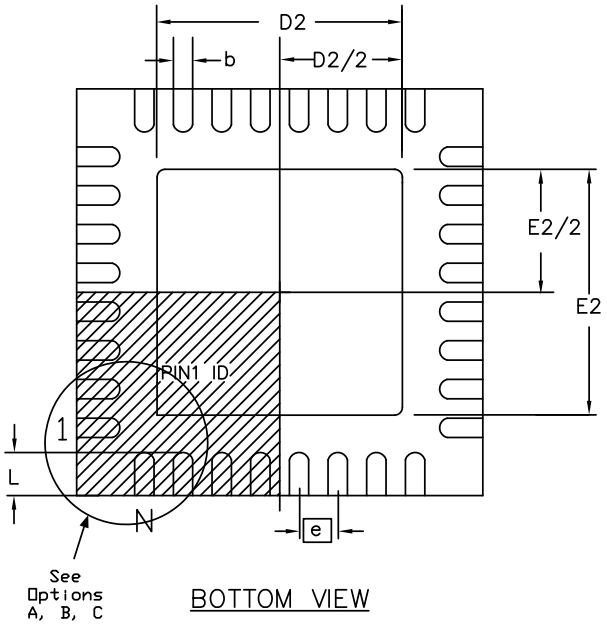
Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	TITLE TG, 20 - Lead, 0.300" Body Width Plastic Gull Wing Small outline Package (SOIC)	DRAWING No. TG	REV. M

26.2 QFN32



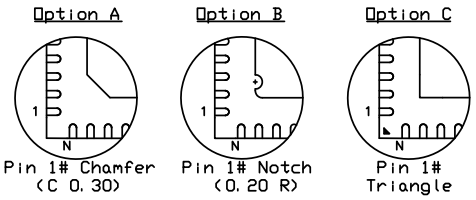


DRAWINGS NOT SCALED



COMMON DIMENSIONS IN MM

SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	----	1.00	
J	0.00	----	0.05	
D/E		5.00 BSC		
D2/E2	1.25	----	3.60	
N		32		
e		0.50 BSC		
L	0.30	0.40	0.50	
b	0.18	0.25	0.30	



Compliant JEDEC Standard MO-220 variation VHHD-2

07/26/07

Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	TITLE PN, 32 - Lead 5.0x5.0 mm Body, 0.50 mm Pitch Quad Flat No Lead Package (QFN)	DRAWING No. PN	REV. G

27. Errata

27.1 Errata AT90PWM81 revA

- **Available on request**

27.2 Errata AT90PWM81 revB

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**
- **PSC output at reset**
- **Flash and EEPROM programming failure if CPU clock is switched**
- **ADC AMPlifier measurement is unstable**
- **ADC measurement reports abnormal values with PSC2-synchronized conversions**
- **Over-consumption in power down sleep mode**

1. **Clock Switch enable & disable**

   After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

   Work around:

   After the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. **Crystal oscillator control with Clock Switch**

   When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

   Work around:

   After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator

3. **BOD disable fuse**

   It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

   Work around:

   If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power supply is below the lowest POR voltage (2.6V).

4. **PSC output at Reset**

   At Reset, the PSC outputs may be set at a value different from the PSC Fuse configuration (Bit 4 of Extended Fuse Byte).

   Work around:

   Initiate PSC output states from source code.

5. **Flash and EEPROM programming failure if CPU clock is switched**

If Clock switching is used in the Application, the memory programming is only possible when the internal RC oscillator is selected as System clock.

If the Application requires a memory programming on a clock source different from the internal RC oscillator, do not switch to this clock source.

Work around:

- Use internal RC oscillator when programming Flash and EEPROM,

or

- Do not use clock switching.

6. **ADC AMPlifier measurement is unstable**

When switching from a single-ended ADC channel to an Amplified channel, noise can appear on ADC conversion.

Work around:

After switching from a single ended to an amplified channel, discard the first ADC conversion.

7. **ADC measurement reports abnormal values with PSC2-synchronized conversions**

When using ADC in synchronized mode, an unexpected extra Single ended conversion can spuriously re-start.This can occur when the End of conversion and the Trigger event occur at the same time.

Work around:

No workaround

8. **Over-consumption in power down sleep mode.**

In Power-down mode, an extra power consumption up to 500µA may occur.

Work around:

No workaround

27.3 Errata AT90PWM81 revC

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**
- **PSC output at Reset**
- **Flash and EEPROM programming failure if CPU clock is switched**
- **ADC AMPlifier measurement is unstable**
- **ADC measurement reports abnormal values with PSC2-synchronized conversions**
- **Over-consumption in power down sleep mode**

1. Clock Switch enable & disable

After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

Work around:

Atter the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. Crystal oscillator control with Clock Switch

When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

Work around:

After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator

3. BOD disable fuse

It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

Work around:

If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power supply is below the lowest POR voltage (2.6V).

4. PSC output at Reset

At Reset, the PSC outputs may be set at a value different from the PSC Fuse configuration (Bit 4 of Extended Fuse Byte).

Work around:

Initiate PSC output states from source code.

5. Flash and EEPROM programming failure if CPU clock is switched

If Clock switching is used in the Application, the memory programming is only possible when the internal RC oscillator is selected as System clock.

If the Application requires a memory programming on a clock source different from the internal RC oscillator, do not switch to this clock source.

Work around:

- Use internal RC oscillator when programming Flash and EEPROM,

or

- Do not use clock switching

6. ADC AMPlifier measurement is unstable

When switching from a single-ended ADC channel to an Amplified channel, noise can appear on ADC conversion.

Work around:

After switching from a single ended to an amplified channel, discard the first ADC conversion.

7. **ADC measurement reports abnormal values with PSC2-synchronized conversions**

When using ADC in synchronized mode, an unexpected extra Single ended conversion can spuriously re-start.This can occur when the End of conversion and the Trigger event occur at the same time.

Work around:

No workaround

8. **Over-consumption in power down sleep mode.**

In Power-down mode, an extra power consumption up to 500µA may occur.

Work around:

No workaround

27.4 Errata AT90PWM81 revD

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**
- **Flash and EEPROM programming failure if CPU clock is switched**
- **ADC Amplifier measurement is unstable**
- **ADC measurement reports abnormal values with PSC2-synchronized conversions**
- **Over-consumption in power down sleep mode**

1. **Clock Switch enable & disable**

After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

Work around:

Atter the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. **Crystal oscillator control with Clock Switch**

When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

Work around:

After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator

3. **BOD disable fuse**

It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

Work around:

If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power supply is below the lowest POR voltage (2.6V).

4. **Flash and EEPROM programming failure if CPU clock is switched**

If Clock switching is used in the Application, the memory programming is only possible when the internal RC oscillator is selected as System clock.

If the Application requires a memory programming on a clock source different from the internal RC oscillator, do not switch to this clock source.

Work around:

- Use internal RC oscillator when programming Flash and EEPROM,

or

- Do not use clock switching

5. **ADC AMPlifier measurement is unstable**

When switching from a single-ended ADC channel to an Amplified channel, noise can appear on ADC conversion.

Work around:

After switching from a single ended to an amplified channel, discard the first ADC conversion.

6. **ADC measurement reports abnormal values with PSC2-synchronized conversions**

When using ADC in synchronized mode, an unexpected extra Single ended conversion can spuriously re-start.This can occur when the End of conversion and the Trigger event occur at the same time.

Work around:

No workaround

7. **Over-consumption in power down sleep mode**

In Power-down mode, an extra power consumption up to 500µA may occur.

Work around:

No workaround

27.5 Errata AT90PWM81 revE

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**

1. **Clock Switch enable & disable**

After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

Work around:

Atter the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. **Crystal oscillator control with Clock Switch**

When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

Work around:

After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator

3. **BOD disable fuse**

It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

Work around:

If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power-supply is below the lowest supply voltage (2.6V).

27.6 Errata AT90PWM161 revA

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**
- **PCSCRRB fuse**
- **Over-consumption in power down sleep mode**

1. **Clock Switch enable & disable**

After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

Work around:

Atter the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. **Crystal oscillator control with Clock Switch**

When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

Work around:

After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator.

3. **BOD disable fuse**

It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

Work around:

If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power-supply is below the lowest supply voltage (2.6V).

4. **PSCRRB fuse**

When this Fuse bit is programmed in Parallel Programming mode, further parallel and ISP programming will not program the device correctly.

Work around:

Program the PSCRRB fuse using ISP mode.

5. **Over-consumption in power down sleep mode**

In Power-down mode, an extra power consumption up to 500µA may occur.

Work around:

No workaround.

27.7 Errata AT90PWM161 revB

- **Clock Switch disable**
- **Crystal oscillator control with Clock Switch**
- **BOD disable fuse**
- **PSCRRB fuse**

1. **Clock Switch enable & disable**

After a “Enable Clock Source” or a “Disable Clock Source” command, the command is still active until the next access of CLKCSR register. If CLKSEL is written with a new value, the corresponding clock will be unintentionnaly enabled or disabled.

Work around:

Atter the Enable or Disable command, write CLKCSR with value 1<<CLKCCE

2. **Crystal oscillator control with Clock Switch**

When a Xtal oscillator is active and CLKSELR is written with a new value for the selection of another clock source (for instance RC or WD) , the Xtal oscillator gain is not correct.

Work around:

After the commands “Enable Clock Source” and “Clock Source Switching”, write back CLKSELR with the values corresponding to the active Xtal oscillator

3. **BOD disable fuse**

It is strongly advised to keep the BOD active. Indeed, the RC oscillator may lock if it is activated when the power suppy goes at a low voltage.

Work around:

If it is mandatory to disable the BOD, do not set the RC oscillator as clock source during reset and makes sure the RC oscillator is never active when the power-supply is below the lowest supply voltage (2.6V).

4. **PSCRRB fuse**

When this Fuse bit is programmed in Parallel Programming mode, further parallel and ISP programming will not program the device correctly.

Work around:

Program the PSCRRB fuse using ISP mode.

28. Datasheet Revision History for AT90PWM81/161

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.

28.1 Rev. 7734A

1. Document creation.

28.2 Rev. 7734B

1. GPIO3 suppressed for compatibility reason.
2. Pinout: PB7 & PD7 swapped.
3. CKSEL values redefined.
4. Clock switching & clock monitoring added.
5. PSCrOUT name changed to PSCOUTR.
6. ADC Auto trigger on PSC synchro improved.
7. Parallel programming updated for 20 pins.
8. Fuses updated: compatibility & potential conflict for reset levels.

28.3 Rev. 7734C

1. Pin out change request.
2. Several improvements on paragraph indent and numbering.
3. P28-29: Device clock option select.
4. P194: BGD bit suppressed.
5. P311-314: Register address changed.

28.4 Rev. 7734D

1. Pin name AGND.
2. PSC reduced support enhanced resolution (Application request).

28.5 Rev. 7734E

1. Speed at 3V, 12Mhz.
2. Add chapter Pin description (PE request).
3. Table 7-1: PE function for 128k RC oscillator is I/O.
4. Details on RC oscillator enable page 30.
5. New warnings on clock switching page 40.
6. Details on CKRC81 page 45.
7. Wake up source PSC not available in PowedDown page 48.
8. Typos on DIDR0/1.
9. ADC sample & hold time on auto conversion.
10. PSC input beheavior during reset precision.
11. Update using the PRR examples with exsisting peripherals.
12. Parallel programing input pins.
13. I/O hysteresis curve.

28.6 Rev. 7734F

1. Clean chapter clock from all “Power save”.
2. Update chapter “Calibrated Internal RC Oscillator” on page 29.
3. Update Table 7-9 on page 35: SUT for PLL.
4. Update chapter Idle Mode page 48.
5. Update figure “PSC Input Module A” on page 119 and “PSC Input Module B” on page 120.
6. Update figure “PSC behavior versus PSCn Input B in Mode 14” on page 132.
7. Update tables 14-16 ”PSC edge & level input Selection” on page 142 & 14-17 ”PSC edge & level input Selection” on page 142.
8. Clean chaper PSC (no more PSC0 & PSC1 register).
9. PSCR registers and bits renamed from “r” to “0”.
10. Update chapter “Digital Input Disable Register 0 – DIDR0” on page 207 & “Digital Input Disable Register 1– DIDR1” on page 208.
11. Update figures on parallel programming :Figure 23-1 on page 261, Figure 23-3 on page 265,Figure 23-4 on page 266,Figure 23-5 on page 268.
12. Suppress chapter ”Parallel Programing Characteristic” after Section 23.7.14, now in “Parallel Programming Characteristics” on page 282.

28.7 Rev. 7734G

1. Update pin out definitions with PE3 as AREF pin: Figures “20 Pin Packages” on page 4, “32Pin Packages” on page 5,Table “Pin out description” on page 7, Chapter “Port E (P32..0) RESET/ XTAL1/ XTAL2/AREF” on page 8 and Chapter “Alternate Functions of Port E” on page 82.
2. Update Table 7-1 on page 28 , for CKSEL 0111, 1100 & 1101.
3. Update figure “Analog to Digital Converter Block Schematic” on page 210.
4. Update Table 19-3 on page 224; warning no more errata.

28.8 Rev. 7734H

1. Udate Product configurationTable 2-1 on page 2.
2. Add chapter “RC Oscillator calibration” on page 31.
3. Update chapter “Internal Voltage Reference” on page 58.
4. Update chapter “On Chip voltage Reference and Temperature sensor overview” on page 192.
5. Update chapter “Temperature Measurement” on page 196.
6. Update Figure “Analog Comparator Block Diagram” on page 201.
7. Update chapter “Reading the Signature Row from Software” on page 250.
8. Update chapter “Calibrated Internal RC Oscillator Accuracy” on page 277.
9. Add chapter “Power consumption estimation with clock prescaling” on page 290.
10. Update chapter “Errata” on page 325.

28.9 Rev. 7734I

1. Remove PE3 I/O function (Only AREF and ADC functions) : Pages 3, 4, 6, 7, 80, 81, 223.
2. Remove the ‘Enable Watchdog in Automatic Reload Mode’ Page 34 and in Table 6-12 on Page 18.
3. Update RC Calibartion section 6.2.2.1 page 30.
4. Remove chapter 6.3.7 on Page 36.

5. Remove chapter 16.3 Band Gap calibration procedure on Page 191.
6. Update Temperature calibration on Page 191.
7. Remove chapter 16.4.3 Two Points Temperature sensor calibration on Page 197.
8. Update Signature Row Addressing on Page249.
9. Update DC Characteristics:
Update table 23-1 Page 275: RC calibartion @25°C
Update Table 23.2 page 272: -40°C in place of -45°C
New Table in 23.2: -40°C to +125°C
10. Update Erratasheet.

28.10 Rev. 7734J

1. Page 2 Table 2-1: QFN32 : 32 Pins.
2. Page 6 Table 3-2: SO24 and QFN20 are removed.
3. Page 30 section 6.2.2.1: RC Osc. is monitored @125°C.
4. Page 51: Table 8.2: BODenable is mandatory.
5. Page 166: removed AT90PWM2/3 comments.
6. Page 267 Table 23-1: User Calibration at +5%.
7. Page 271: Update of ADC Characteristics.
8. Page 273: Add the DAC Characteristics.

28.11 Rev. 7734K

1. Page 193 §16.4.1: removed the last sentence about reading of the temperature sensor during Hot testing.
2. Page 193 §16.4.1: T formula modifed with new TSGAIN.
3. Page 244 Table 21.5: Signature row adressing table updated with right address.

28.12 Rev. 7734L

1. Update Errata Rev E.

28.13 Rev. 7734M

1. Page 204: Figure 18-1 removed REFS2 bit.
2. .Pages 277 to 294: update of Typical characteristics.

28.14 Rev. 7734N

1. Page 52: update of BOD levels.
2. Pages 267, 268, 269, 270: update of Vref, Icc power-down, Icc operating, Icc Idle and Watchdog oscillator characteristics.

28.15 Rev. 7734O

1. Pages 275, 276 Table 23-6: update of ADC characteristics.
2. Page 270: add a new line in Table 23-1 (Calibrated Internal RC oscillator Accuracy).
3. Pages 267, 269, 279: update of Analog comparator characteristics.

28.16 Rev. 7734P

1. Updated “Electrical Characteristics (1)” on page 265 and “AT90PWM81/161 Typical Characteristics” on page 279.

28.17 Rev. 7734Q

1. General update and countless, small corrections.
2. Table 8-1 on page 62 has been updated.
3. The text in “Input Mode Operation” on page 160 has been updated.
4. Changed the overlined text to normal text in the Note below Table 14-1 on page 182.
5. Removed the text/link “13.9” from row #1 in Table 13-5 on page 160.
6. Corrected Figure text and Figure for Figure 17-15 on page 224 and Figure 17-16 on page 224.
7. The number “2” below the list in “Reading the Calibration Byte” on page 260 has been removed.
8. New links have been added.
9. Two places in Table 21-16 on page 263 the text “See Table XXX on page XXX for details” has been replaced by “See Table 21-6 on page 251 for details”.
10. The text “Table 113” two places in Table 4-4 on page 23 has been replaced by “Table 20-7 on page 246”.
11. In Bit 4 – PUD: Pull-up Disable75 the text “Se” has been replaced by “See “Configuring the Pin” on page 69 for more details about this feature.”
12. In Bit 5 – ICIE1: Timer/Counter1, Input Capture Interrupt Enable on page 99, the text “XXX” has been replaced by “Table 8-1 on page 62”.
13. In “Input Mode Operation” on page 120 the text “All” has been replaced by “These modes are listed in Table 12-7”.
14. The figure text in Table 2-2 has been corrected to Pin out description.
15. In the first sentence in “Interrupt Handling” on page 131 the text “(vector...)” has been removed.
16. In Figure 13-2 on page 149 the text “Aralog” has been corrected to “Analog”.
17. In “Prescaling and Conversion Timing” on page 206, section 6, the text “(four XXX to be confirmed)” has been removed.
18. In “ADC Conversion Result” on page 215 the text “Table 82” has been replaced by “Table 17-2 on page 217”.
19. In Table 22-4 on page 271, Note 1, the text “(CPU core, PSC... “ has been removed.
20. “Ordering Information” on page 304 has been updated.
21. Added AT90PWM161 (16K Flash, 1K SRAM) and updated the datasheet accordingly.
22. In chapter “Analog to Digital Converter” on page 47 the text “CROSS REFERENCE MOVED” has been replaced by ““ADC Noise Canceler” on page 210”.
23. In Table 9-2 on page 74 the text “figure” has been replaced by the cross reference “Figure 9-5 on page 73”.
24. In chapter “Alternate Port Functions” on page 73, just below Table 9-2 on page 74 the cross reference Table 9-2 has been added.
25. In chapter “Registers” on page 86 the text “figure” has been replaced by the cross reference Figure 11-1 two places.
26. In chapter “PSOC2 - PSC 2 Synchro and Output Configuration” on page 134, in bullet point “Bit 3:0 – PRFMnx3:0: PSC n Fault Mode”, page 141, the text “PSC Functional Specification” has been replaced by “Table 12-7 on page 120”.

27. In chapter "PFRC0B - PSCR Input B Control Register" on page 175, in bullet point "Bit 3:0 – PRFM0x3:0: PSCR Fault Mode", page 176, the text "PSCR Functional Specification" has been replaced by "Table 13-5 on page 160".
28. In "Bit 2, 1, 0– AC3M2, AC3M1, AC3M0: Analog Comparator 3 Multiplexer register", page 199, the reference "Table 16-4" has been corrected to "Table 16-6".
29. In Table 21-5 on page 250, the reference "Table 113" has been corrected to "Table 20-7 on page 246" two places.
30. In chapter "Signal Names" on page 252, the text "in the following table" has been replaced by the reference "Table 21-8 on page 252".
31. Several cross references have been corrected.
32. The text "The accuracy of this calibration is shown as Factory calibration in Table 24-1 on page 277" on page 30 has been changed into "The accuracy of this calibration is shown as Factory calibration in Table 22-2 on page 270".
33. The first Note in Bit 2– CKRC81: Frequency Selection of the calibrated 8/1MHz RC Oscillator on page 42 is corrected to This bit only can be changed only when the RC oscillator is enabled.
34. Note 1 below Figure 16-1 on page 195 is changed to "Refer to Figure 2-1 on page 3 and Figure 2-2 on page 4 for Analog Comparator pin placement."
35. Figure 16-2 on page 196 has been corrected.
36. In "MISO/ACMP3/ADC8– Bit 6" on page 75 the "DDB0" has been corrected to "DDB6", and the "PORTB0" has been corrected to "PORTB6".
37. In "ADC5/ACMP2/INT1/SCK – Bit 5" on page 76 the "DDD4" has been corrected to "DDB5", and the "PORT" has been corrected to "PORTB5".
38. In "MOSI/ADC3/ACMPM– Bit 4" on page 76 the "DDB1" has been corrected to "DDB4", and the "PORTB1" has been corrected to "PORTB4".
39. Missing information in Table 9-4 on page 77, Table 9-5 on page 77, Table 9-7 on page 79, Table 9-8 on page 80 and Table 9-10 on page 81 has been added.
40. Paragraphs one and two in "In-System Reprogrammable Flash Program Memory" on page 16 have been changed to to include more data regarding the AT90PWM161.
41. The text "TBD" in Table 21-7 on page 251 has been changed into "8B".
42. The text in bullet point number three below Table 21-7 on page 251 has been expanded to include the AT90PWM161.
43. The text "Calibration accuracy" in the heading of Table 22-2 on page 270 has been changed into "Accuracy".
44. The text "AT90PWM161 revA" has been added to the heading in Section 27.5 on page 311.
45. JMP and CALL instructions are added to "BRANCH INSTRUCTIONS" in Section 25. on page 301.
46. "Errata AT90PWM161 revA" on page 312 and "Errata AT90PWM161 revB" on page 313 are added.
47. In "Errata AT90PWM161 revB" on page 313, and in "Errata AT90PWM161 revA" on page 312 the PSCRRB fuse is added.
48. In "Features" on page 227 the bullet points "The DAC could be connected to the negative inputs of the analog comparators and/or to a dedicated output driver" and "Output impedance around 1KOhm" have been removed.
49. The text "AMP3D" has been replaced by "ACMP3D" in "DIDR0 - Digital Input Disable Register 0" on page 202, "DIDR0 - Digital Input Disable Register 0" on page 222, and "Register Summary" on page 297.

50. In “Features” on page 100, the text in one of the bullet pints has been corrected to “Abnormality protection function, emergency input to force all outputs to low level”.
51. @25°C has been removed several places in the table in “DC Characteristics” .
52. The rows describing 105°C on page 269 in the table in “DC Characteristics” have been removed.
53. The values in Table 7-2 on page 53 has been corrected.
54. The text below “Description” in Table 13-8 on page 171 has been corrected.
55. The text below “Description” in Table 13-9 on page 171 has been corrected.
56. A new feature has been added in Section 18.1, page 227.
57. The address for Atmel Japan has been changed.

Table Of Contents

Features *1*

1 Products Configuration *2*

2 Pin Configurations *3*

2.1 Pin Descriptions 6

3 AVR CPU Core *8*

3.1 Introduction 8

3.2 Architectural Overview 8

3.3 ALU – Arithmetic Logic Unit 9

3.4 Status Register 10

3.5 General Purpose Register File 11

3.6 Stack Pointer 12

3.7 Instruction Execution Timing 12

3.8 Reset and Interrupt Handling 13

4 Memories *16*

4.1 In-System Reprogrammable Flash Program Memory 16

4.2 SRAM Data Memory 17

4.3 EEPROM Data Memory 18

4.4 Fuse Bits 22

4.5 I/O Memory 26

4.6 General Purpose I/O Registers 26

5 System Clock and Clock Options *27*

5.1 Clock Systems and their Distribution 27

5.2 Clock Sources 28

5.3 Dynamic Clock Switch 35

5.4 System Clock Prescaler 39

5.5 Register Description 39

6 Power Management and Sleep Modes *45*

6.1 Sleep Modes 45

6.2 Idle Mode 45

6.3 ADC Noise Reduction Mode 46

6.4 Power-down Mode 46

6.5 Standby Mode 46

6.6 Power Reduction Register 46
6.7 Minimizing Power Consumption 47
6.8 Register description 48

7 System Control and Reset 50

7.1 System Control overview 50
7.2 System Control registers 54
7.3 Internal Voltage Reference 55
7.4 Watchdog Timer 56

8 Interrupts 62

8.1 Interrupt Vectors in AT90PWM81/161 62

9 I/O-Ports 68

9.1 Introduction 68
9.2 Ports as General Digital I/O 69
9.3 Alternate Port Functions 73
9.4 Register Description for I/O-Ports 81

10 External Interrupts 83

11 Reduced 16-bit Timer/Counter1 85

11.1 Overview 85
11.2 Accessing 16-bit Registers 87
11.3 Timer/Counter Clock Sources 90
11.4 Counter Unit 91
11.5 Input Capture Unit 92
11.6 Modes of Operation 94
11.7 Timer/Counter Timing Diagrams 95
11.8 16-bit Timer/Counter Register Description 97

12 Power Stage Controller – (PSCn) 100

12.1 Features 100
12.2 Overview 100
12.3 PSC Description 101
12.4 Signal Description 103
12.5 Functional Description 104
12.6 Update of Values 110
12.7 Enhanced Resolution 111
12.8 PSC Inputs 114

12.9 PSC Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait 121
12.10 PSC Input Mode 2: Stop signal, Execute Opposite Pulse and Wait 122
12.11 PSC Input Mode 3: Stop signal, Execute Opposite Pulse while Fault active 123
12.12 PSC Input Mode 4: Deactivate outputs without changing timing 124
12.13 PSC Input Mode 5: Stop signal and Insert Dead-Time 124
12.14 PSC Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait 125
12.15 PSC Input Mode 7: Halt PSC and Wait for Software Action 125
12.16 PSC Input Mode 8: Edge Retrigger PSC 126
12.17 PSC Input Mode 9: Fixed Frequency Edge Retrigger PSC 127
12.18 PSC Input Mode 14: Fixed Frequency Edge Retrigger PSC and Deactivate Output 128
12.19 PSC2 Outputs 129
12.20 Analog Synchronization 131
12.21 Interrupt Handling 131
12.22 PSC Synchronization 132
12.23 PSC Clock Sources 133
12.24 Interrupts 134
12.25 PSC Register Definition 134
12.26 PSC2 Specific Register 142

13 Reduced Power Stage Controller – (PSCR) 147

13.1 Features 147
13.2 Overview 147
13.3 PSCR Description 148
13.4 Signal Description 149
13.5 Functional Description 151
13.6 Update of Values 154
13.7 Enhanced resolution 155
13.8 PSCR Inputs 155
13.9 PSCR Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait ... 161
13.10 PSCR Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait .. 162
13.11 PSCR Input Mode 3: Stop signal, Execute Opposite while Fault active 163
13.12 PSCR Input Mode 4: Deactivate outputs without changing timing 164
13.13 PSCR Input Mode 5: Stop signal and Insert Dead-Time 164
13.14 PSCR Input Mode 6: Stop signal, Jump to Opposite Dead-Time and Wait ... 165
13.15 PSCR Input Mode 7: Halt PSCR and Wait for Software Action 165
13.16 PSCR Input Mode 8: Edge Retrigger PSC 166

13.17 PSCR Input Mode 9: Fixed Frequency Edge Retrigger PSC 167
13.18 PSCR Input Mode 14: Fixed Frequency Edge Retrigger PSCR and Deactivate Output 168
13.19 Analog Synchronization 169
13.20 Interrupt Handling 169
13.21 PSC Clock Sources 169
13.22 Interrupts 170
13.23 PSCR Register Definition 171

14 Serial Peripheral Interface – SPI: 180

14.1 Features 180
14.2 Overview 180
14.3 $\overline{SS}$ Pin Functionality 184
14.4 Data Modes 185
14.5 $\overline{SPI}$ registers 186

15 Voltage Reference and Temperature Sensor 189

15.1 Features 189
15.2 On Chip voltage Reference and Temperature sensor overview 189
15.3 Register Description 190
15.4 Temperature Measurement 192

16 Analog Comparator 194

16.1 Features 194
16.2 Overview 194
16.3 Shared pins between Analog Comparator and ADC 196
16.4 Analog Comparator Register Description 196

17 Analog to Digital Converter - ADC 203

17.1 Features 203
17.2 Operation 205
17.3 Starting a Conversion 205
17.4 Prescaling and Conversion Timing 206
17.5 Changing Channel or Reference Selection 209
17.6 ADC Noise Canceler 210
17.7 ADC Conversion Result 215
17.8 ADC Register Description 217
17.9 Amplifier 222
17.10 Amplifier Control Registers 225

18 Digital to Analog Converter - DAC 227

18.1 Features 227

18.2 Operation 228

18.3 Starting a Conversion 228

18.4 DAC Register Description 228

19 debugWIRE On-chip Debug System 231

19.1 Features 231

19.2 Overview 231

19.3 Physical Interface 231

19.4 Software Break Points 232

19.5 Limitations of debugWIRE 232

19.6 debugWIRE Related Register in I/O Memory 232

20 Boot Loader Support – Read-While-Write Self-Programming 233

20.1 Boot Loader Features 233

20.2 Application and Boot Loader Flash Sections 233

20.3 Read-While-Write and No Read-While-Write Flash Sections 233

20.4 Boot Loader Lock Bits 236

20.5 Entering the Boot Loader Program 237

20.6 Addressing the Flash During Self-Programming 239

20.7 Self-Programming the Flash 240

21 Memory Programming 248

21.1 Program And Data Memory Lock Bits 248

21.2 Fuse Bits 249

21.3 Signature Bytes 251

21.4 Calibration Byte 252

21.5 Parallel Programming Parameters, Pin Mapping, and Commands 252

21.6 Serial Programming Pin Mapping 254

21.7 Parallel Programming 254

21.8 Serial Downloading 261

22 Electrical Characteristics (1) 265

22.1 Absolute Maximum Ratings* 265

22.2 DC Characteristics 266

22.3 Clock Drive Characteristics 270

22.4 Maximum Speed vs. V_{CC} 271

22.5 PLL Characteristics 271

22.6 SPI Timing Characteristics 272
22.7 ADC Characteristics 274
22.8 DAC Characteristics 276
22.9 Parallel Programming Characteristics 276

23 AT90PWM81/161 Typical Characteristics 279

23.1 Active Supply Current 280
23.2 Idle Supply Current 282
23.3 Power-Down Supply Current 284
23.4 Pin Pull-up 285
23.5 Pin output high voltage 288
23.6 Pin output low voltage 289
23.7 Pin Thresholds 290
23.8 BOD Thresholds 291
23.9 Analog Reference 292
23.10 Internal Oscillator Speed 293
23.11 Current Consumption in Reset 295

24 Register Summary 297

25 Instruction Set Summary 301

26 Ordering Information 304

26.1 SO20 305
26.2 QFN32 306

27 Errata 307

27.1 Errata AT90PWM81 revA 307
27.2 Errata AT90PWM81 revB 307
27.3 Errata AT90PWM81 revC 308
27.4 Errata AT90PWM81 revD 310
27.5 Errata AT90PWM81 revE 311
27.6 Errata AT90PWM161 revA 312
27.7 Errata AT90PWM161 revB 313

28 Datasheet Revision History for AT90PWM81/161 314

28.1 Rev. 7734A 314
28.2 Rev. 7734B 314
28.3 Rev. 7734C 314
28.4 Rev. 7734D 314

28.5 Rev. 7734E 314
28.6 Rev. 7734F 315
28.7 Rev. 7734G 315
28.8 Rev. 7734H 315
28.9 Rev. 7734I 315
28.10 Rev. 7734J 316
28.11 Rev. 7734K 316
28.12 Rev. 7734L 316
28.13 Rev. 7734M 316
28.14 Rev. 7734N 316
28.15 Rev. 7734O 316
28.16 Rev. 7734P 317
28.17 Rev. 7734Q 317

Table Of Contents i

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
16F, Shin Osaki Kangyo Bldg.
1-6-4 Osaki Shinagawa-ku
Tokyo 104-0032
JAPAN
Tel: (+81) 3-6417-0300
Fax: (+81) 3-6417-0370