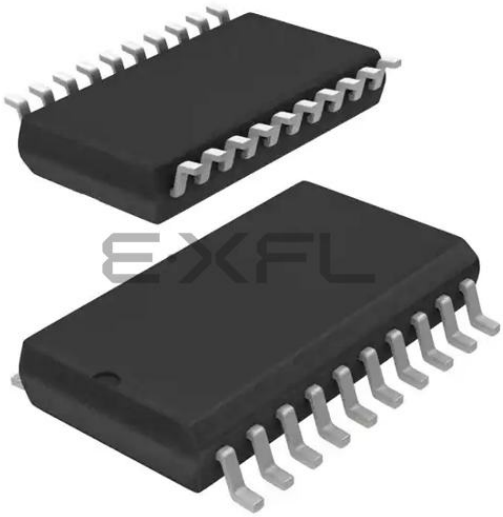


Welcome to E-XFL.COM

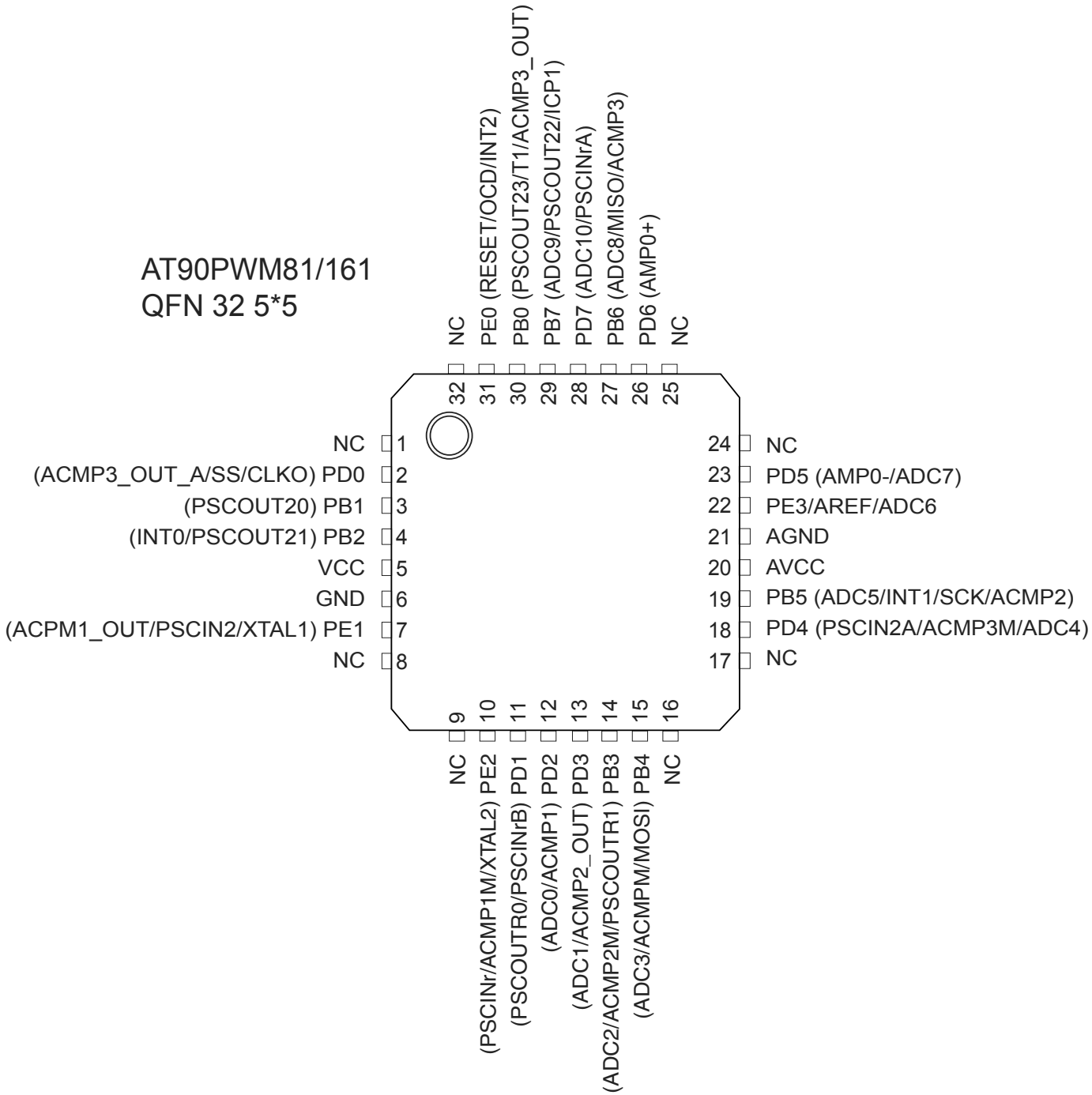
What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	SPI
Peripherals	Brown-out Detect/Reset, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at90pwm81-16sf

Figure 2-2. 32-pin packages.



- **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.

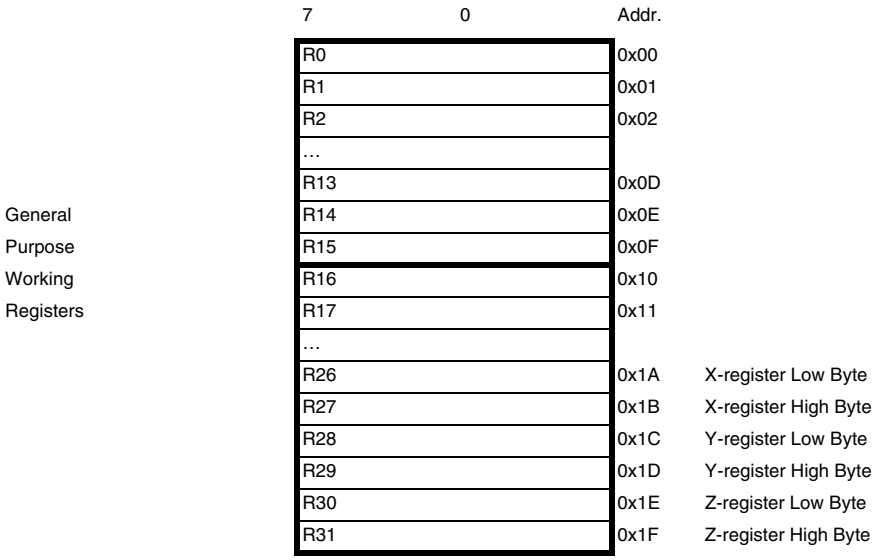
3.5 General Purpose Register File

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.



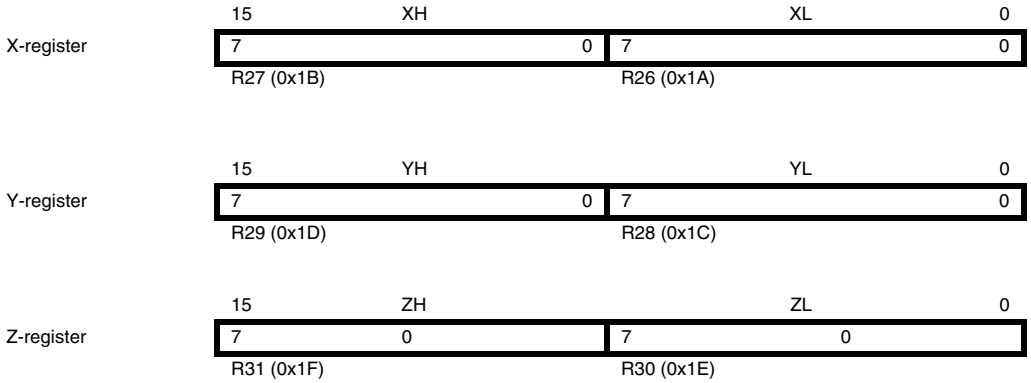
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

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.

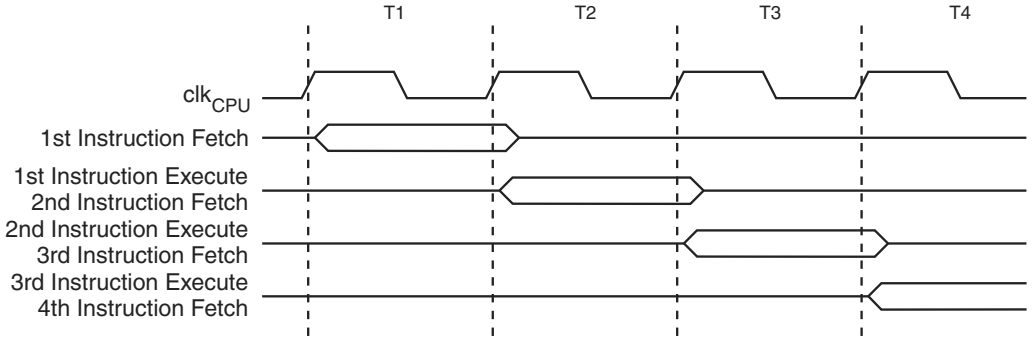
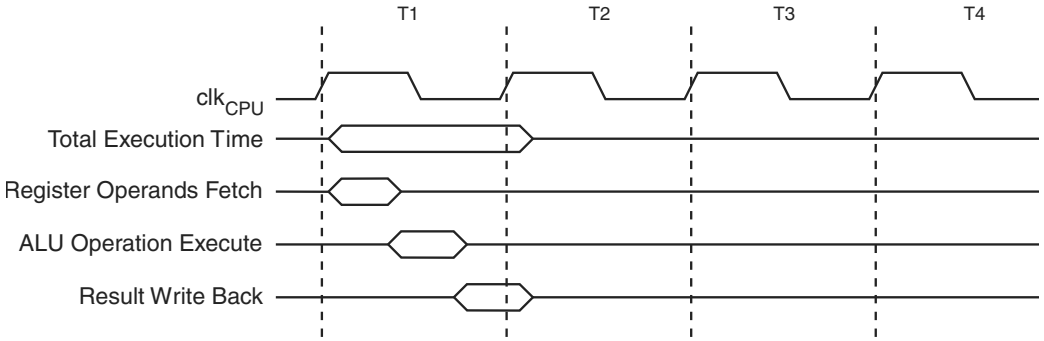


Figure 3-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

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

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.

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.

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.

- **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.

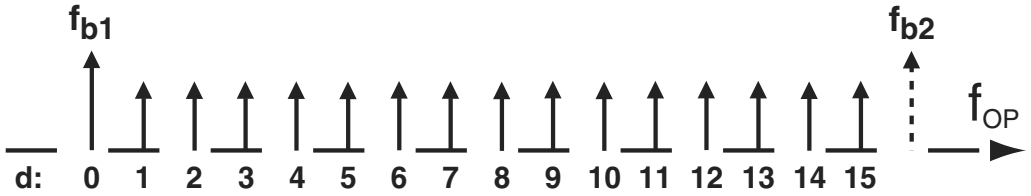
Table 12-6. Distribution of f_{b2} in the modulated frame.

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.

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.

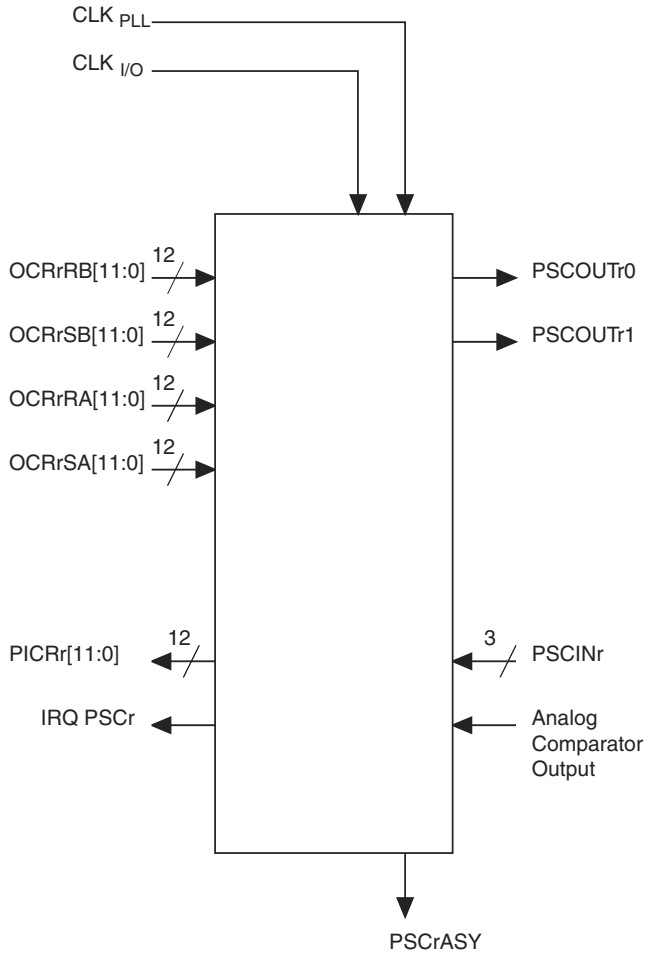
PSCRV gives the state low or high which will be forced on PSC outputs selected by PSC0RB & PSC2RB fuses.

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.

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

14. Serial Peripheral Interface – SPI:

14.1 Features

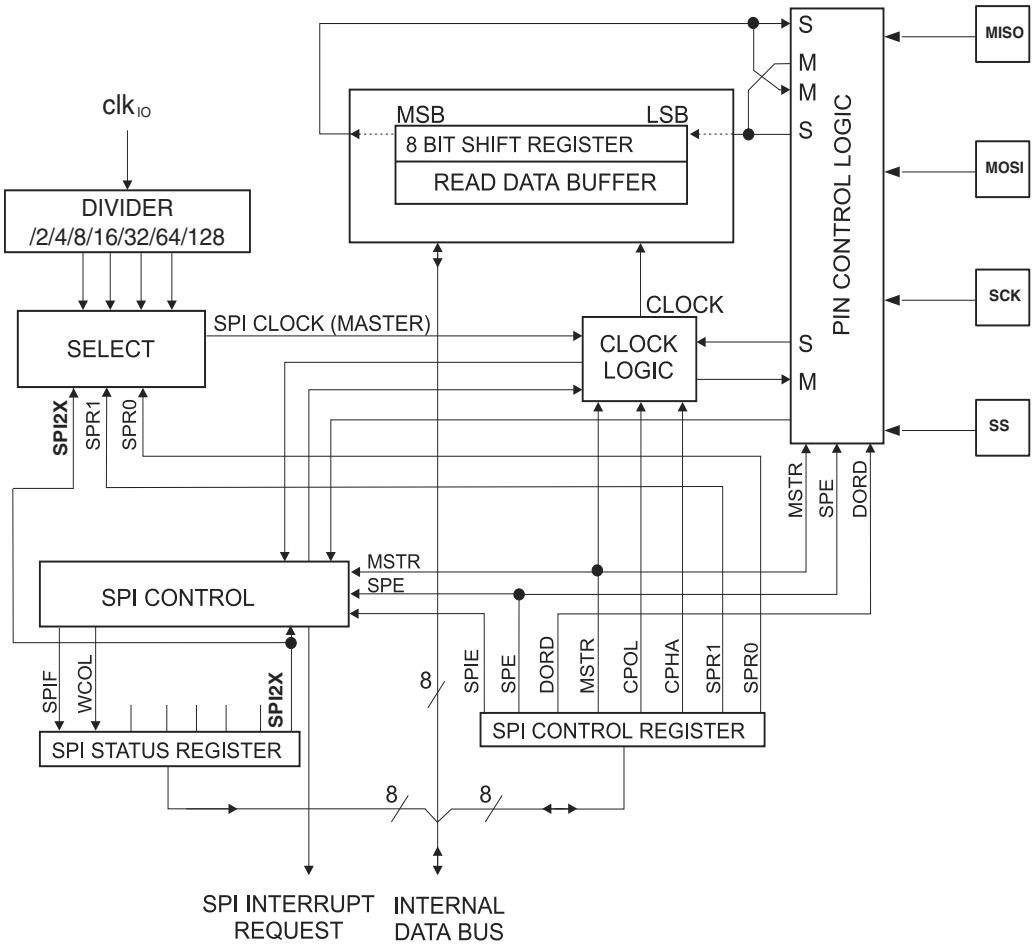
- **Full-duplex, three-wire synchronous data transfer**
- **Master or Slave operation**
- **LSB first or MSB first data transfer**
- **Seven programmable bit rates**
- **End of transmission interrupt flag**
- **Write collision flag protection**
- **Wake-up from idle mode**
- **Double speed (CK/2) Master 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.

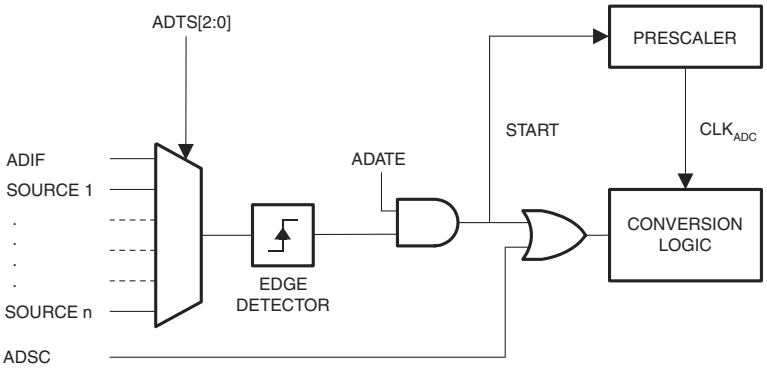
Figure 14-1. SPI block diagram [1].



Note: 1. Refer to Figure 2-1 on page 3, and Table 9-3 on page 75 for SPI pin placement.

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.

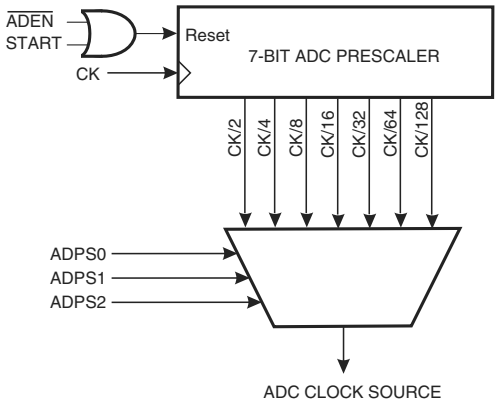


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the 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.

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	

- 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.

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

Figure 23-2. Active supply current vs. frequency (1MHz - 16MHz).

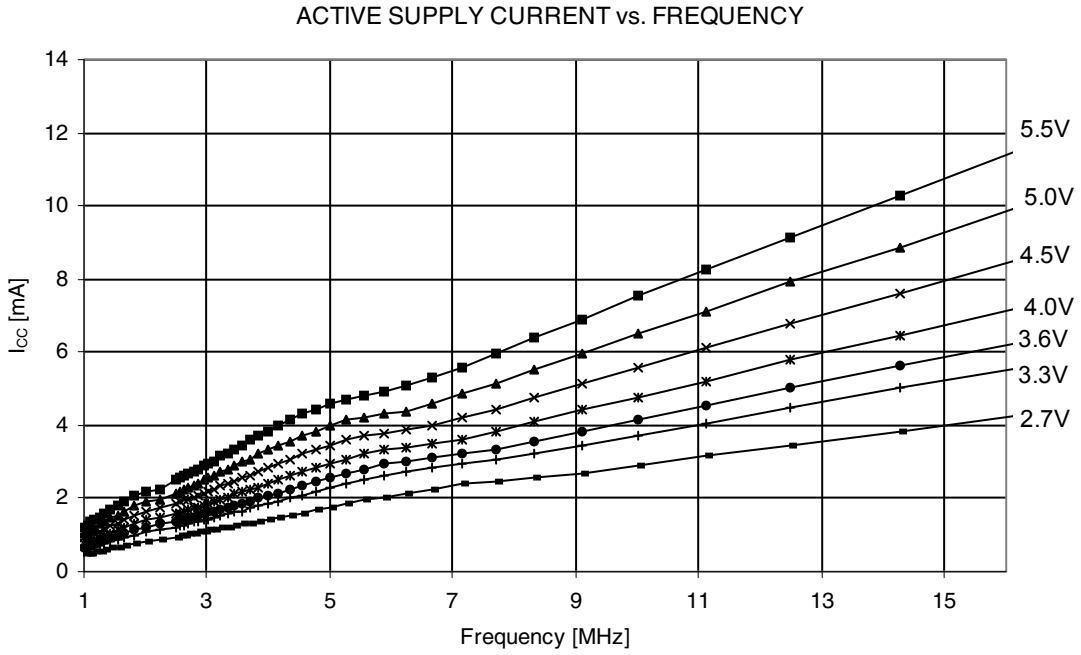
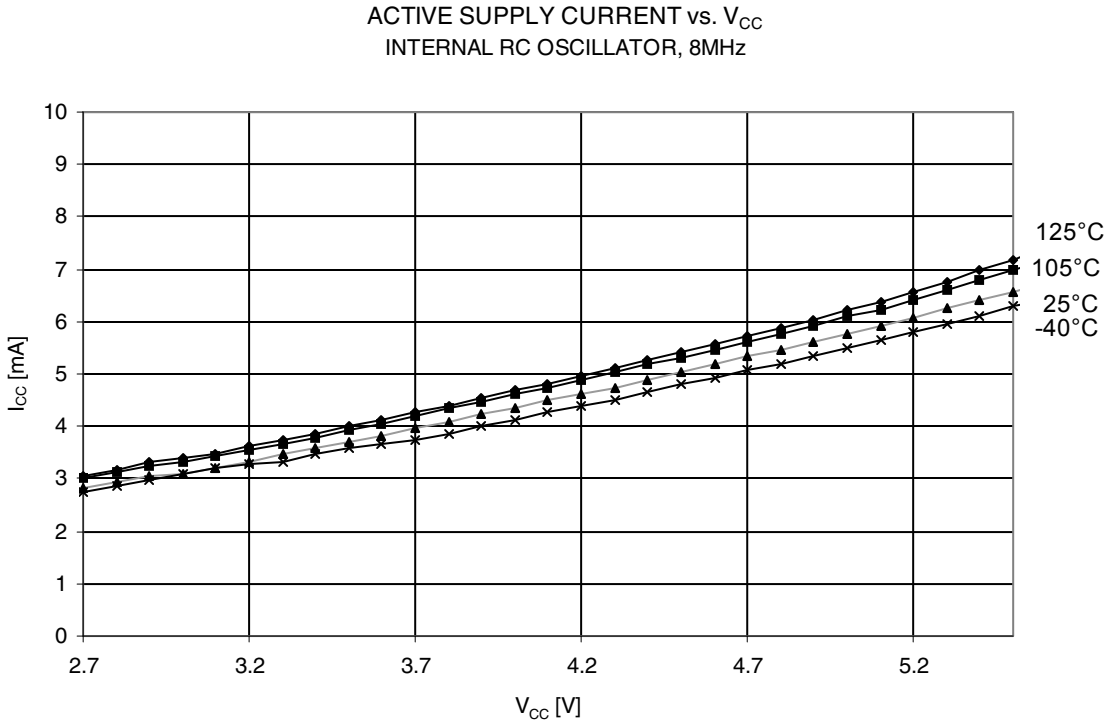
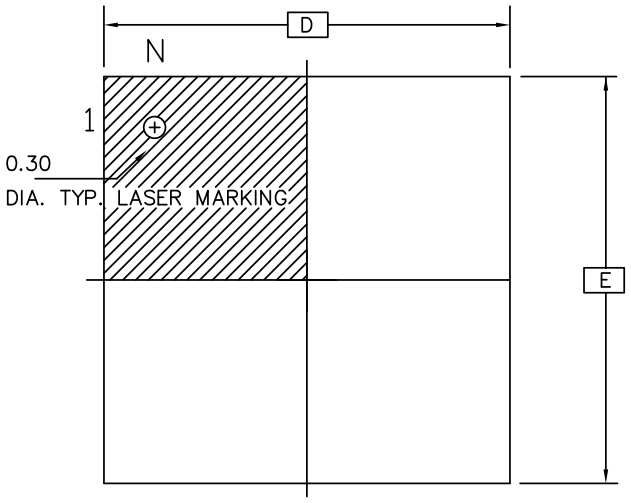
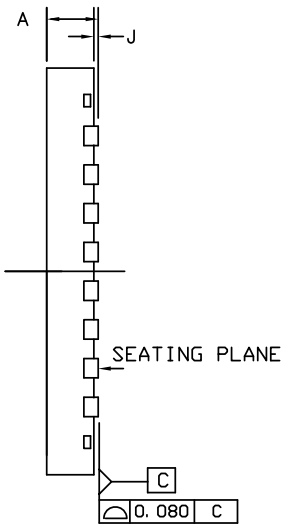


Figure 23-3. Active supply current vs. V_{CC} (internal RC oscillator, 8MHz).

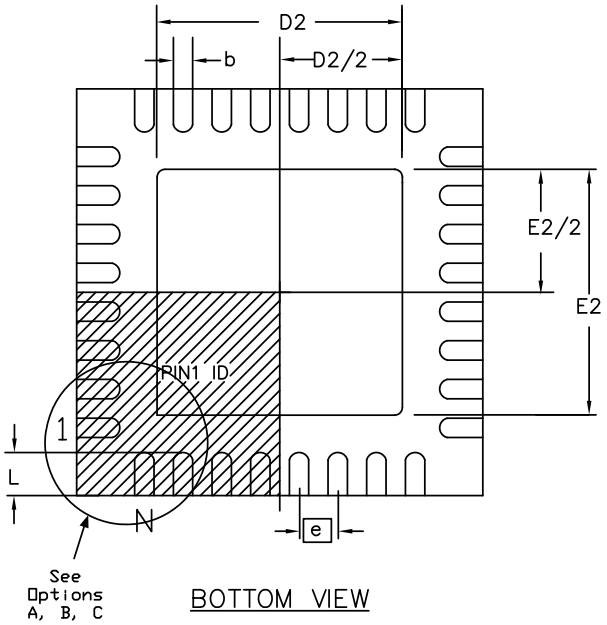


26.2 QFN32



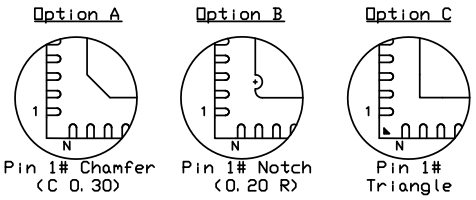


DRAWINGS NOT SCALED



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. 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.

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