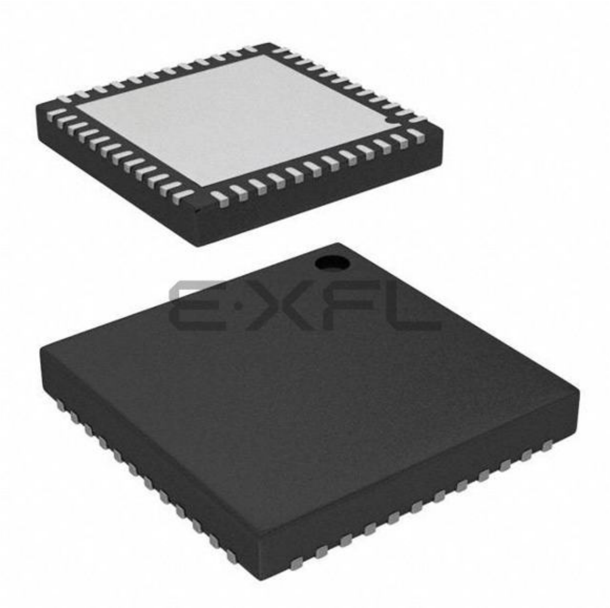


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	15MHz
Connectivity	LINbus, SPI, UART/USART, LINbus-SBC
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	10
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 1x17b Sigma Delta, 1x18b Sigma Delta
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega32hve2-plpw

3. Absolute Maximum Ratings

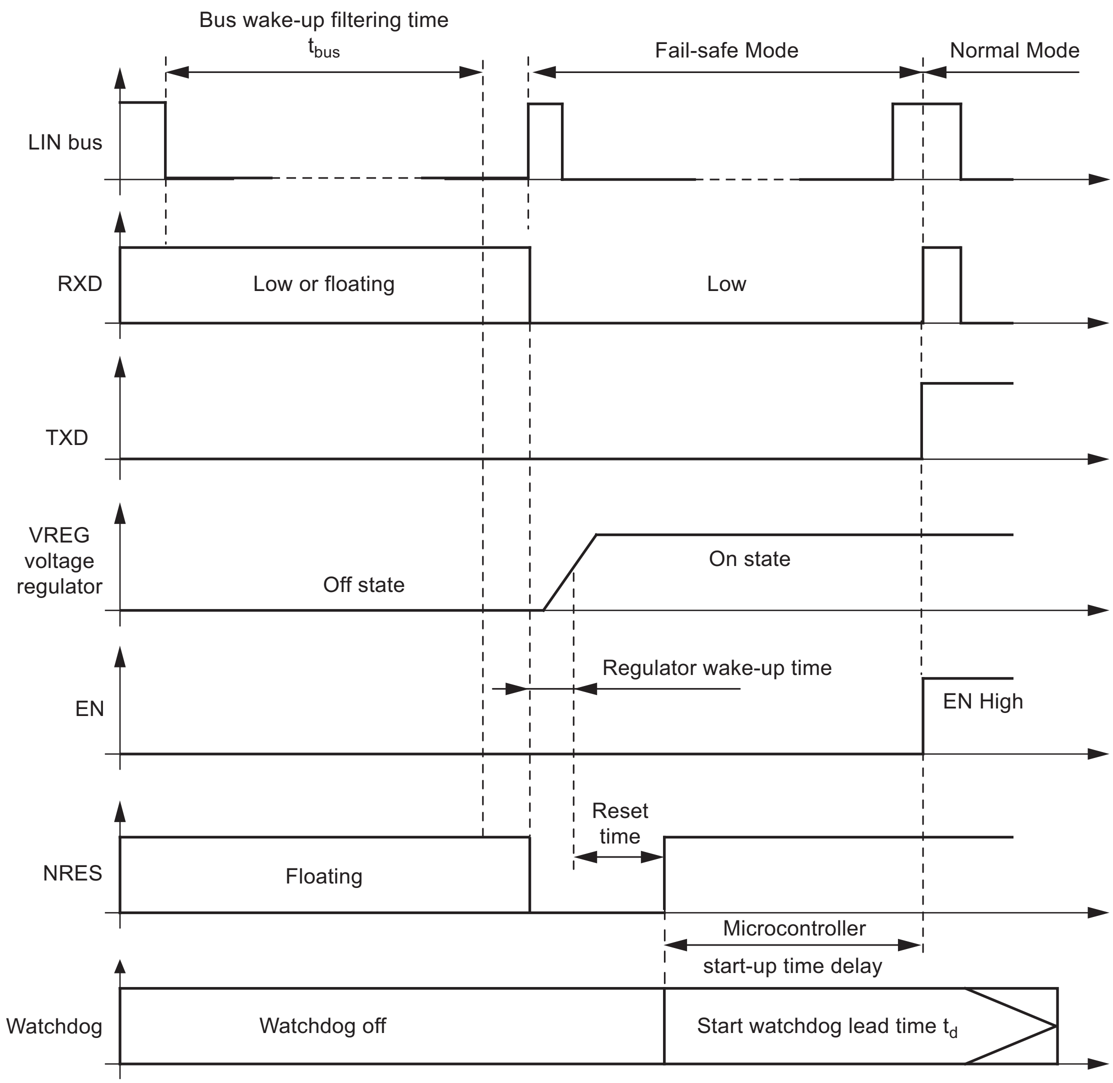
Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Symbol	Min.	Typ.	Max.	Unit
Supply voltage V_S	V_S	–0.3		+40	V
Pulse time ≤ 500ms T_a = 25°C Output current I_{VREG} ≤ 50mA	V_S			+40	V
Pulse time ≤ 2min T_a = 25°C Output current I_{VREG} ≤ 50mA	V_S			27	V
VBAT (with 47Ω/10nF) DC voltage Transient voltage due to ISO7637 3a, 3b (coupling 1nF)		–1 –150		+40 +100	V V
LIN, VBAT - DC voltage		–27		+40	V
Logic pins (RxD, TxD, EN, NRES, NTRIG, WD_OSC, MODE, TM, DIV_ON, SP_MODE, PV1)		–0.3		VREG + 0.5V	V
Pin NV1		–0.3		+0.3	V
Output current NRES	I_{NRES}			+2	mA
PVREG DC voltage VREG DC voltage		–0.3 –0.3		+5.5 +6.5	V V
Logic pins (PA0-PA1, PI, NI, PB0-PB7, PV2, NV2)		–0.5		VCC + 0.5	V
$\overline{\text{RESET}}$		–0.5		+13	V
VREF		–0.5		VCC + 0.5	V
VREFGND Connected via internal metal connection to GND. Do not connect external to GND.		–0.5		+0.5	mA
VCC/AVCC		–0.3		+4.5	V
ESD according to IBEE LIN EMC Test Spec. 1.0 following IEC 61000-4-2 - Pin VS, LIN to GND - Pin VBAT (10nF) to GND		±6			KV
HBM ESD ANSI/ESD-STM5.1 JESD22-A114 AEC-Q100 (002) MIL-STD-883 (M3015.7)		±3			KV
CDM ESD STM 5.3.1		±750			V
MM ESD EIA/JESD22-A115 ESD STM5.2 AEC-Q100 (002)		±200			V
ESD HBM following STM5.1 with 1.5kΩ 100pF - Pin VS, LIN, VBAT to GND		±6			KV

A falling edge at the LIN pin followed by a dominant bus level maintained for a certain time period (t_{bus}) and a rising edge at pin LIN result in a remote wake-up request. The device switches from Sleep Mode to Fail-safe Mode.The VREG regulator is activated, and the internal LIN slave termination resistor is switched on. The remote wake-up request is indicated by a low level at the RXD pin to interrupt the microcontroller (see Figure 7-5).

EN high can be used to switch directly from Sleep/Silent to Fail-safe Mode. If EN is still high after VREG ramp up and undervoltage reset time, the IC switches to the Normal Mode.

Figure 7-5. LIN Wake Up from Sleep Mode



9.2 Worst Case Calculation with R_{WD_OSC} = 51kΩ

The internal oscillator has a tolerance of 20%. This means that t_1 and t_2 can also vary by 20%. The worst case calculation for the watchdog period t_{wd} is calculated as follows.

The ideal watchdog time t_{wd} is between the maximum t_1 and the minimum t_1 plus the minimum t_2.

$t_{1,min} = 0.8 \times t_1 = 16.5ms$, $t_{1,max} = 1.2 \times t_1 = 24.8ms$
$t_{2,min} = 0.8 \times t_2 = 17.3ms$, $t_{2,max} = 1.2 \times t_2 = 26ms$

$t_{wdmax} = t_{1min} + t_{2min} = 16.5ms + 17.3ms = 33.8ms$
$t_{wdmin} = t_{1max} = 24.8ms$

$t_{wd} = 29.3ms \pm 4.5ms\ (\pm 15\%)$

A microcontroller with an oscillator tolerance of ±15% is sufficient to supply the trigger inputs correctly.

Table 9-1. Typical Watchdog Timings

R_{WD_OSC} kΩ	Oscillator Period t_{osc}/µs	Lead Time t_d/ms	Closed Window t_1/ms	Open Window t_2/ms	Trigger Period from Microcontroller t_{wd}/ms	Reset Time t_{nres}/ms
34	13.3	105	14.0	14.7	19.9	4
51	19.61	154.8	20.64	21.67	29.32	4
91	33.54	264.80	35.32	37.06	50.14	4
120	42.84	338.22	45.11	47.34	64.05	4

10. Electrical Characteristics LIN SBC (Continued)

5V < V_S < 27V, –40°C < T_j < 150°C, unless otherwise specified. All values refer to GND pins

No.	Parameters	Test Conditions	Pin	Symbol	Min.	Typ.	Max.	Unit	Type*
9.5	Time delay for mode change from Silent Mode into Normal Mode via EN	$V_{EN} = V_{REG}$	EN	t_{s_n}	5	15	40	µs	A
9.6	Monitoring time for wake-up over LIN-bus		LIN	t_{mon}	6	10	15	ms	A
	LIN Bus Driver AC Parameter with Different Bus Loads Load 1 (small): 1nF, 1kΩ; Load 2 (large): 10nF, 500Ω; R_{RXD} = 5kΩ; C_{RXD} = 20pF; Load 3 (medium): 6.8nF, 660Ω characterized on samples; 10.7 and 10.8 specifies the timing parameters for proper operation of 20Kbit/s, 10.9 and 10.10 at 10.4Kbit/s								
9.7	Duty cycle 1	$TH_{Rec(max)} = 0.744 \times V_S$ $TH_{Dom(max)} = 0.581 \times V_S$ V_S = 7.0V to 18V t_{Bit} = 50µs $D1 = t_{bus\_rec(min)}/(2 \times t_{Bit})$	LIN	D1	0.396				A
9.8	Duty cycle 2	$TH_{Rec(min)} = 0.422 \times V_S$ $TH_{Dom(min)} = 0.284 \times V_S$ V_S = 7.6V to 18V t_{Bit} = 50µs $D2 = t_{bus\_rec(max)}/(2 \times t_{Bit})$	LIN	D2			0.581		A
9.9	Duty cycle 3	$TH_{Rec(max)} = 0.778 \times V_S$ $TH_{Dom(max)} = 0.616 \times V_S$ V_S = 7.0V to 18V t_{Bit} = 96µs $D3 = t_{bus\_rec(min)}/(2 \times t_{Bit})$	LIN	D3	0.417				A
9.10	Duty cycle 4	$TH_{Rec(min)} = 0.389 \times V_S$ $TH_{Dom(min)} = 0.251 \times V_S$ V_S = 7.6V to 18V t_{Bit} = 96µs $D4 = t_{bus\_rec(max)}/(2 \times t_{Bit})$	LIN	D4			0.590		A
9.11	Slope time falling and rising edge at LIN	V_S = 7.0V to 18V	LIN	t_{SLOPE_fall} t_{SLOPE_rise}	3.5		22.5	µs	A
10	Receiver Electrical AC Parameters of the LIN Physical Layer LIN Receiver, RXD Load Conditions (C_{RXD}): 20pF								
10.1	Propagation delay of receiver (Figure 10-1 on page 31)	V_S = 7.0V to 18V $t_{rx\_pd} = \max(t_{rx\_pdr}, t_{rx\_pdf})$	RXD	t_{rx_pd}			6	µs	A
10.2	Symmetry of receiver propagation delay rising edge minus falling edge	V_S = 7.0V to 18V $t_{rx\_sym} = t_{rx\_pdr} - t_{rx\_pdf}$	RXD	t_{rx_sym}	–2		+2	µs	A
11	NRES Open Drain Output Pin								
11.1	Low-level output voltage	V_S ≥ 5.5V I_{NRES} = 1mA	NRES	V_{NRESL}			0.14	V	A
11.2	Low-level output low	10kΩ to 5V V_{REG} = 0V	NRES	V_{NRESLL}			0.14	V	A
11.3	Undervoltage reset time	V_S ≥ 5.5V C_{NRES} = 20pF	NRES	t_{reset}	2	4	6	ms	A
11.4	Reset debounce time for falling edge	V_S ≥ 5.5V C_{NRES} = 20pF	NRES	t_{res_f}	1.5		10	µs	A

*) Type means: A = 100% tested, B = 100% correlation tested, C = Characterized on samples, D = Design parameter

14.4 EEPROM Data Memory

The Atmel® AVR MCU contains 1Kbytes 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 EEPROM programming, see page 183 and page 186 respectively.

14.4.1 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 14-1 on page 45. 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 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.

14.5 I/O Memory

The I/O space definition of the Atmel® AVR MCU is shown in Section 32. "Register Summary" on page 203.

All Atmel AVR MCU I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel AVR MCU 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 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.

14.5.1 General Purpose I/O Registers

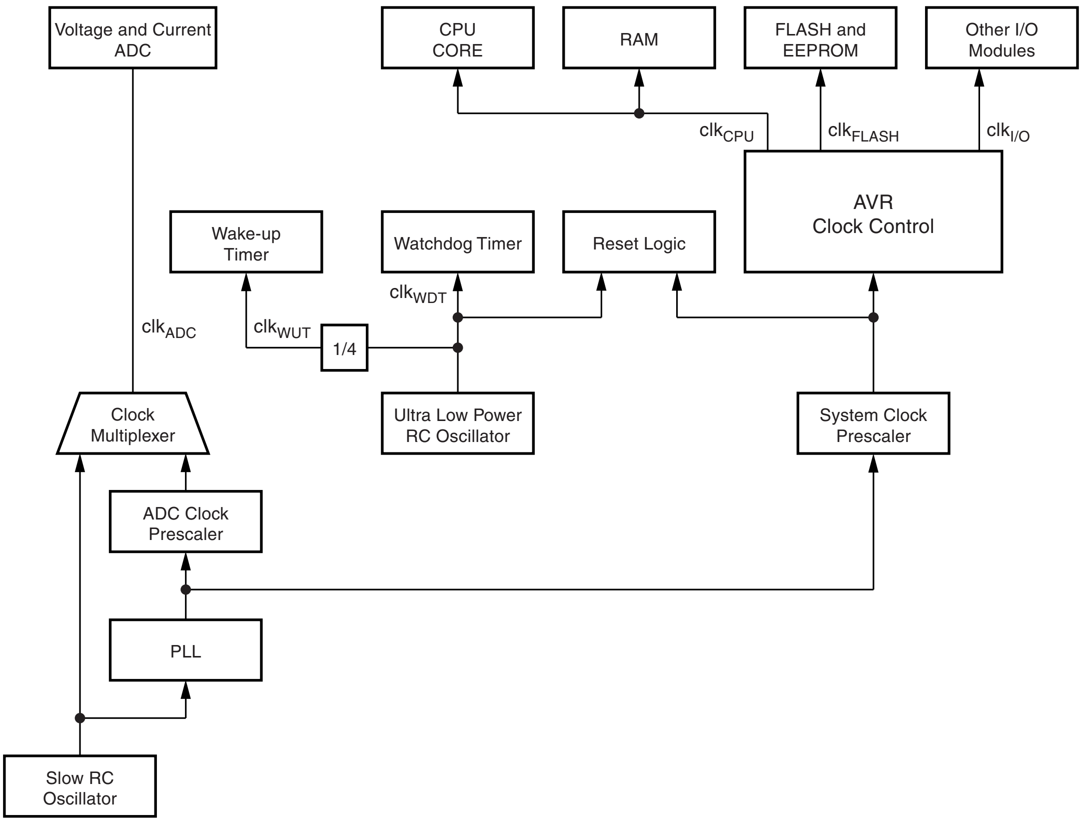
The Atmel AVR MCU contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions. See Section 14.6.4 "GPIOR2 – General Purpose I/O Register 2" on page 47, Section 14.6.5 "GPIOR1 – General Purpose I/O Register 1" on page 47, and Section 14.6.6 "GPIOR0 – General Purpose I/O Register 0" on page 47 for details.

15. System Clock and Clock Options

15.1 Clock Systems and their Distribution

Figure 15-1 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in Section 16. "Power Management and Sleep Modes" on page 53. The clock systems are detailed below.

Figure 15-1. Clock Distribution



15.1.1 CPU Clock – clk_{CPU}

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

15.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules. 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.

15.1.3 Flash Clock – clk_{FLASH}

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

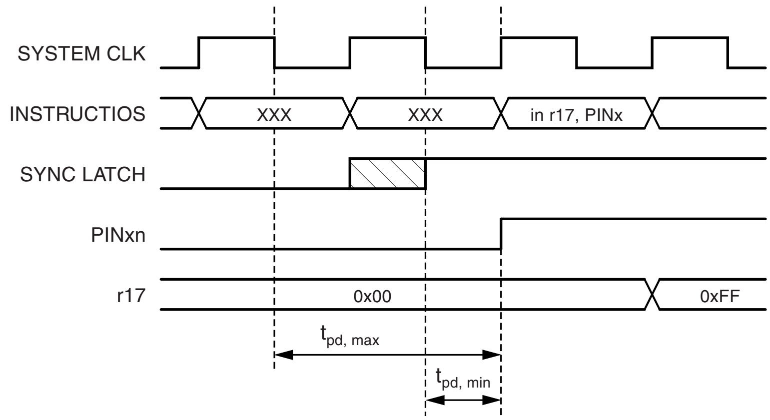
15.1.4 ADC Clock – clk_{ADC}

The Voltage ADC and Current ADC are provided with a dedicated clock domain. The ADCs have two alternate clock sources, selectable by the CKSEL bit in ADCRA, refer to Section 26.6.3 "ADCRA - ADC Control Register A" on page 151 for details.

21.2.4 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 21-2, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 21-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 21-3. 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 tpd,max and tpd,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 21-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 21-4. Synchronization when Reading a Software Assigned Pin Value

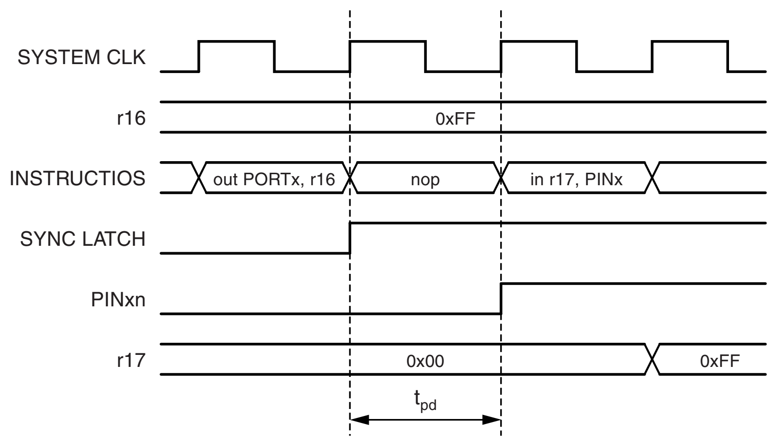


Table 21-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 21-5 on page 83 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 21-2. 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, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

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

21.3.1 Alternate Functions of Port A

The Port A pins with alternate functions are shown in Table 21-3.

Table 21-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA1	ADC1/SGND/PCINT1 (ADC Input 1, Signal Ground or Pin Change Interrupt 1)
PA0	ADC0/SGND/PCINT0 (ADC Input 0, Signal Ground or Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- **ADC0/SGND/PCINT0 - Port A, Bit0**

  ADC0: Voltage ADC Input0. This pin can serve as Input 0 for the Voltage ADC.

  SGND: Voltage ADC SGND. This pin can serve as signal ground for the Voltage ADC.

  PCINT0. Pin Change Interrupt 0. This pin can serve as external interrupt source.
- **ADC1/SGND/PCINT1 - Port A, Bit1**

  ADC1: Voltage ADC Input1. This pin can serve as Input 1 for the Voltage ADC.

  SGND: Voltage ADC SGND. This pin can serve as signal ground for the Voltage ADC.

  PCINT1: Pin Change Interrupt 1. This pin can serve as external interrupt source.

  These pins can serve as external interrupt sourceTable 21-4 relates the alternate functions of Port A to the overriding signals shown in Figure 21-5 on page 83.

Table 21-4. Overriding Signals for Alternate Functions in PA1:PA0

Signal Name	PA1/ADC1/SGND/PCINT1	PA0/ADC0/SGND/PCINT0
PUOE	0	0
PUOV	0	0
DDOE	VAMUX = 001	VAMUX = 010
DDOV	1	1
PVOE	VAMUX = 001	VAMUX = 010
PVOV	0	0
PTOE	-	-
DIEOE	PA1DID | (PCINT1 $\times$ PCIE0)	PA0DID | (PCINT0 $\times$ PCIE0)
DIEOV	$\overline{\text{PA1DID}}$	$\overline{\text{PA0DID}}$
DI	PCINT1 INPUT	PCINT0 INPUT
AIO	ADC1 INPUT SGND INPUT	ADC0 INPUT SGND INPUT

- **CKOUT/PCINT4 - Port B, Bit2**
  CKOUT: Clock output. This pin can serve as clock output pin.
  PCINT4: Pin Change Interrupt 4. This pin can serve as external interrupt source.
- **RXD/PCINT3 - Port B, Bit1**
  RXD: This pin can serve as RXD pin for the LIN interface.
  PCINT3: Pin Change Interrupt 3. This pin can serve as external interrupt source.
- **FH/PCINT2 - Port B, Bit0**
  FH: Force High. When the PBOE0 bit in the PBOV register is set, this pin is forced high.
  PCINT2: Pin Change Interrupt 2. This pin can serve as external interrupt source.

Table 21-6. Overriding Signals for Alternate Functions in PB7:PB4

Signal Name	PB7/MISO/ICP10/ INT0/ PCINT9	PB6/MOSI/PCINT8	PB5/SCK/PCINT7	PB4/$\overline{\text{SS}}$/PCINT6
PUOE	SPE × MASTER	SPE × $\overline{\text{MASTER}}$	SPE × $\overline{\text{MASTER}}$	SPE × $\overline{\text{MASTER}}$
PUOV	PORTB7 × $\overline{\text{PUD}}$	PORTB7 × $\overline{\text{PUD}}$	PORTB7 × $\overline{\text{PUD}}$	PORTB7 × $\overline{\text{PUD}}$
DDOE	SPE × MASTER	SPE × $\overline{\text{MASTER}}$	SPE × $\overline{\text{MASTER}}$	SPE × $\overline{\text{MASTER}}$
DDOV	0	0	0	0
PVOE	SPE × $\overline{\text{MASTER}}$	SPE × MASTER	SPE × MASTER	0
PVOV	SPI SLAVE	SPI MASTER		
PTOE	0	0	0	0
DIEOE	PCINT9 × PCIE | INT0 Enable	PCINT8 × PCIE	PCINT7 × PCIE	PCINT6 × PCIE
DIEOV	1	1	1	1
DI	INT0 ICP10 SPI MASTER PCINT9	SPI SLAVE PCINT8	SCK PCINT7	$\overline{\text{SS}}$ PCINT6
AIO	-	-	-	-

Table 21-7. Overriding Signals for Alternate Functions in PB3:PB0

Signal Name	PB3/TXD/PCINT5	PB2/CKOUT/ PCINT4	PB1/RXD/ PCINT3	PB0/FH/PCINT2
PUOE	LINTXEN	CKOE	LINRXEN	PBOE0
PUOV	LINTXD × PBOE3 × PORTB3	0	PORTB2 × PUD	0
DDOE	LINTXEN	CKOE	LINRXEN	PBOE0
DDOV	$\overline{\text{LINTXD}}$ × $\overline{\text{PBOE3}}$	CKOE	0	1
PVOE	LINTXEN	CKOE	0	PBOE0
PVOV	LINTXD × $\overline{\text{PBOE3}}$	CKOUT	0	1
PTOE	0	0	0	0
DIEOE	PCINT5 × PCIE	(PCINT4 × PCIE) | CKOE	PCINT3	PCINT2 × PCIE
DIEOV	1	PCINT4 × PCIE | $\overline{\text{CKOE}}$	1	1
DI	T1 PCINT5	LINRXD PCINT4	PCINT3	T0 PCINT2
AIO	-	-	-	-

23.7.1 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNTnH/L Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCRnA/B to be initialized to the same value as TCNTn without triggering an interrupt when the Timer/Counter clock is enabled.

23.7.2 Using the Output Compare Unit

Since writing TCNTnH/L will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNTnH/L when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNTnH/L equals the OCRnA/B value, the Compare Match will be missed.

23.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 23-6 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value.

Figure 23-6. Timer/Counter Timing Diagram, no Prescaling

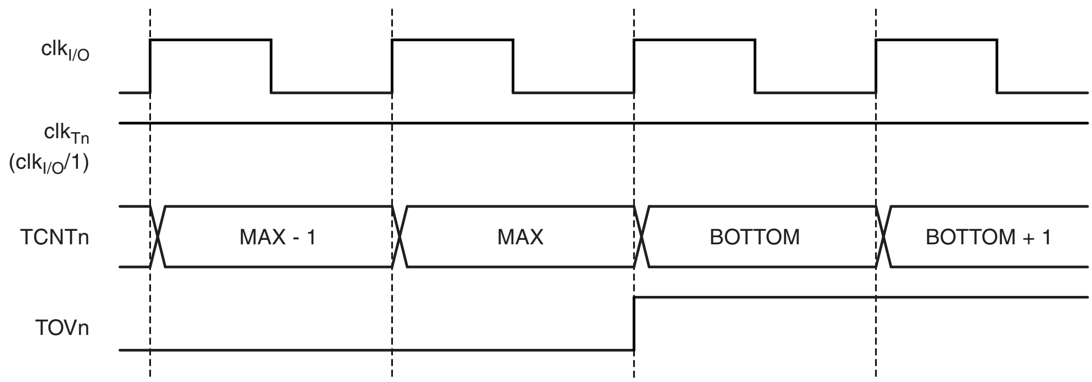
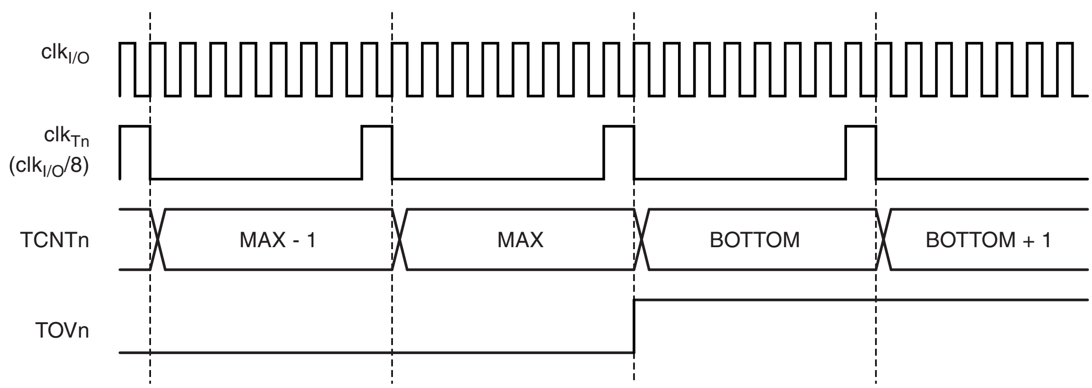


Figure 23-7 shows the same timing data, but with the prescaler enabled.

Figure 23-7. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



23.9 Accessing Registers in 16-bit Mode

In 16-bit mode (the TCWn bit is set to one) the TCNTnH/L and OCRnA/B or TCNTnL/H and OCRnB/A 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. The 16-bit Timer/Counter 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. 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.

There is one exception in the temporary register usage. In the Output Compare mode the 16-bit Output Compare Register OCRnA/B is read without the temporary register, because the Output Compare Register contains a fixed value that is only changed by CPU access. However, in 16-bit Input Capture mode the ICRn register formed by the OCRnA and OCRnB registers must be accessed with the temporary register.

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

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCRnA/B registers.

Assembly Code Example

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

C Code Example

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

Note: 1. See Section 12. “About Code Examples” on page 34

The assembly code example returns the TCNTnH/L 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.

25.5.3 Data Transport

Two types of data may be transported in a frame; signals or diagnostic messages.

- Signals
  Signals are scalar values or byte arrays that are packed into the data field of a frame. A signal is always present at the same position in the data field for all frames with the same identifier.
- Diagnostic messages
  Diagnostic messages are transported in frames with two reserved identifiers. The interpretation of the data field depends on the data field itself as well as the state of the communicating nodes.

25.5.4 Schedule Table

The master task (in the master node) transmits frame headers based on a schedule table. The schedule table specifies the identifiers for each header and the interval between the start of a frame and the start of the following frame. The master application may use different schedule tables and select among them.

25.5.5 Compatibility with LIN 1.3

LIN 2.1 is a super-set of LIN 1.3.

A LIN 2.1 master node can handle clusters consisting of both LIN 1.3 slaves and/or LIN 2.1 slaves. The master will then avoid requesting the new LIN 2.1 features from a LIN 1.3 slave:

- Enhanced checksum,
- Re-configuration and diagnostics,
- Automatic baud rate detection,
- "Response error" status monitoring.

LIN 2.1 slave nodes can not operate with a LIN 1.3 master node (e.g., the LIN1.3 master does not support the enhanced checksum).

The LIN 2.1 physical layer is backwards compatible with the LIN1.3 physical layer. But not the other way around. The LIN 2.1 physical layer sets greater requirements, i.e. a master node using the LIN 2.1 physical layer can operate in a LIN 1.3 cluster.

25.6 LIN / UART Controller

The LIN/UART controller is divided in three main functions:

- Tx LIN Header function,
- Rx LIN Header function,
- LIN Response function.

These functions mainly use two services:

- Rx service,
- Tx service.

Because these two services are basically UART services, the controller is also able to switch into an UART function.

25.6.1 LIN Overview

The LIN/UART controller is designed to match as closely as possible to the LIN software application structure. The LIN software application is developed as independent tasks, several slave tasks and one master task (c.f. Section 25.5.4 on page 118). The Atmel® AVR MCU conforms to this perspective. The only link between the master task and the slave task will be at the cross-over point where the interrupt routine is called once a new identifier is available. Thus, in a master node, housing both master and slave task, the Tx LIN Header function will alert the slave task of an identifier presence. In the same way, in a slave node, the Rx LIN Header function will alert the slave task of an identifier presence.

When the slave task is warned of an identifier presence, it has first to analyze it to know what to do with the response. Hardware flags identify the presence of one of the specific identifiers from 60 (0x3C) up to 63 (0x3F).

26.4.2 Initialization and Settling Time

When the ADCs are enabled (both disabled in advance) an extra initialization time of 30-40 ADC cycles is required until the first conversion is ready. The same initialization time is required when software executes an immediate configuration change command.

When applying new changes the ADC will need to do settling conversions before an actual conversion is ready. If using Automatic Fast/Slow Chopper mode, the settling will automatically be handled in hardware, in other cases the settling must be handled by the software.

If not using Automatic Fast/Slow Chopper mode, settling should be handled in user software by discarding the first two Instantaneous Conversions and the first Accumulation Conversion result after doing a configuration change that requires settling.

For both ADCs, settling is required when enabling the ADC, after changing the decimation ratios, after changing the polarity of the chopper, after changing the sampling clock source and after leaving Automatic Chopper mode configuration.

For the C-ADC, settling time is required when changing the input gain settings.

For the V-ADC, settling time is required when changing conversion channel.

The settling time is summarized in Table 26-2.

Table 26-2. Settling time for the Instantaneous (IC) and Accumulated (AC) Conversion

Chopper Mode	$T_{SETTLING,IC}$	$T_{SETTLING,AC}$
Auto Fast Chopper(1)	$\frac{2}{F_{IC}}$	$\frac{2}{F_{AC}}$
Auto Slow Chopper(2)	$\frac{2}{F_{IC}}$	$\frac{2}{F_{AC}}$
No chopper(3)	$\frac{2}{F_{IC}}$	$\frac{2}{F_{AC}}$

Notes: 1. The first Accumulated Conversion must be discarded in software.
2. The Instantaneous Conversion offset removal has to be performed in Software.
3. Settling should be performed in software when applying configuration changes that require settling.
4. Whenever doing configuration changes the recommended synchronization methods should be used. Otherwise one extra settling conversion has to be added. For details on synchronization, see Section 26.4.1 “Synchronization of Configuration Settings” on page 146.

26.4.3 Sampling Clock

Software can select either the 512kHz PLL clock or the 128kHz Slow RC oscillator as sampling clock for the ADC by writing to the CKSEL bit in Section 26.6.3 “ADCRA - ADC Control Register A” on page 151. When changing clock configuration this will be synchronized in the same way as other configuration settings.

Note that if the PLL has been selected as ADC clock the PLL will keep running even if the CPU has entered sleep modes where the PLL should be automatically disabled. Whenever going to deep sleep modes it is recommended to always use the Slow RC oscillator as sampling clock. This allows the PLL to be automatically switched off which gives minimum power consumption.

If changing to the PLL clock source software should make sure that the PLL has locked to the target frequency before using the conversion data.

When changing sampling clock on the next conversion, the clock change will take affect about 35 ADC clock cycles before the corresponding interrupt is set. Note therefore that the conversion time of the ongoing conversions will be affected.

29.8.4 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 19. "Interrupts" on page 70.

29.8.5 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.

29.8.6 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 "Interrupts" on page 70, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See Section 29.8.12 "Simple Assembly Code Example for a Boot Loader" on page 174 for an example.

29.8.7 Setting the Lock Bits by SPM

To set the Lock bits, wait until the PLL enters LOCK(1), write the desired data to R0, write "X0001001" to SPMCSR and execute SPM within four clock cycles after writing SPMCSR.

Bit	7	6	5	4	3	2	1	0
R0	**1**	**1**	**BLB12**	**BLB11**	**BLB02**	**BLB01**	**LB2**	**LB1**

See following Table 30-2 on page 180 for how the different settings of the Lock bits affect the Flash access.

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

Note: 1. For the PLL lock status, see the PLLCSR register.

29.8.8 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 LBSET and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the LBSET and SPMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The LBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits. When LBSET and SPMEN are cleared, LPM will work as described in the "AVR Instruction Set" description.

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 LBSET and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the LBSET 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 30-4 on page 182 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**

```
	;	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, EEWE
	rjmp		Wait_ee
	;	SPM timed sequence
	out		SPMCSR, spmcrval
	spm
	;	restore SREG (to enable interrupts if originally enabled)
	out		SREG, temp2
	ret
```

30. Memory Programming

30.1 Program And Data Memory Lock Bits

The Atmel® AVR MCU provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 30-2. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 30-1. 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)

“1” means unprogrammed, “0” means programmed

Table 30-2. 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)]
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.

Notes: 1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

Table 30-14. High-voltage Serial Programming Instruction Set for Atmel® AVR MCU (Continued)

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Read EEPROM Byte	SDI	0_**bbbb_bbbb**_00	0_**aaaa_aaaa**_00	0_0000_0000_00	0_0000_0000_00	
	SII	0_0000_1100_00	0_0001_1100_00	0_0110_1000_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**q_qqqq_qqq**0_00	
Write Fuse High Byte	SDI	0_0100_0000_00	0_**hhhh_hhhh**_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write “0” to program the Fuse Bits.
	SII	0_0100_1100_00	0_0010_1100_11	0_0111_0100_00	0_0111_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Write Fuse Low Byte	SDI	0_0100_0000_00	0_**llll_llll**_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write “0” to program the Fuse bit.
	SII	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Write Lock Bit Byte	SDI	0_0010_0000_00	0_cccc_cccc_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write “0” to program the Lock Bit.
	SII	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Read Fuse High Byte	SDI	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00		Reading “0” means the Fuse bit is programmed.
	SII	0_0100_1100_00	0_0111_1000_00	0_0111_1100_00		
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	h_hhhh_hhhx_xx		
Read Fuse Low Byte	SDI	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00		Reading “0” means the Fuse bit is programmed.
	SII	0_0100_1100_00	0_0110_1000_00	0_0110_1100_00		
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**l_llll_lll**x_xx		
Read Lock Bit Byte	SDI	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00		Reading “0” means the Lock bit is programmed.
	SII	0_0100_1100_00	0_0111_1000_00	0_0111_1100_00		
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	c_cccc_cccx_xx		
Read Signature Row Low Byte	SDI	0_0000_1000_00	0_**bbbb_bbbb**_00	0_0000_0000_00	0_0000_0000_00	Repeats Instr 2 4 for each signature low byte address.
	SII	0_0100_1100_00	0_0000_1100_00	0_0110_1000_00	0_0110_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**q_qqqq_qqq**x_xx	
Read Signature Row High Byte	SDI	0_0000_1000_00	0_**aaaa_aaaa**_00	0_0000_0000_00	0_0000_0000_00	Repeats Instr 2 4 for each signature high byte address.
	SII	0_0100_1100_00	0_0001_1100_00	0_0111_1000_00	0_0111_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**p_pppp_ppp**x_xx	
Load “No Operation” Command	SDI	0_0000_0000_00				
	SII	0_0100_1100_00				
	SDO	x_xxxx_xxxx_xx				

Note: 1. **a** = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits, x = don’t care, **c** = Lock Bit Byte, **l** = fuse low byte, **h** = fuse high byte.

Notes: 1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.

2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.

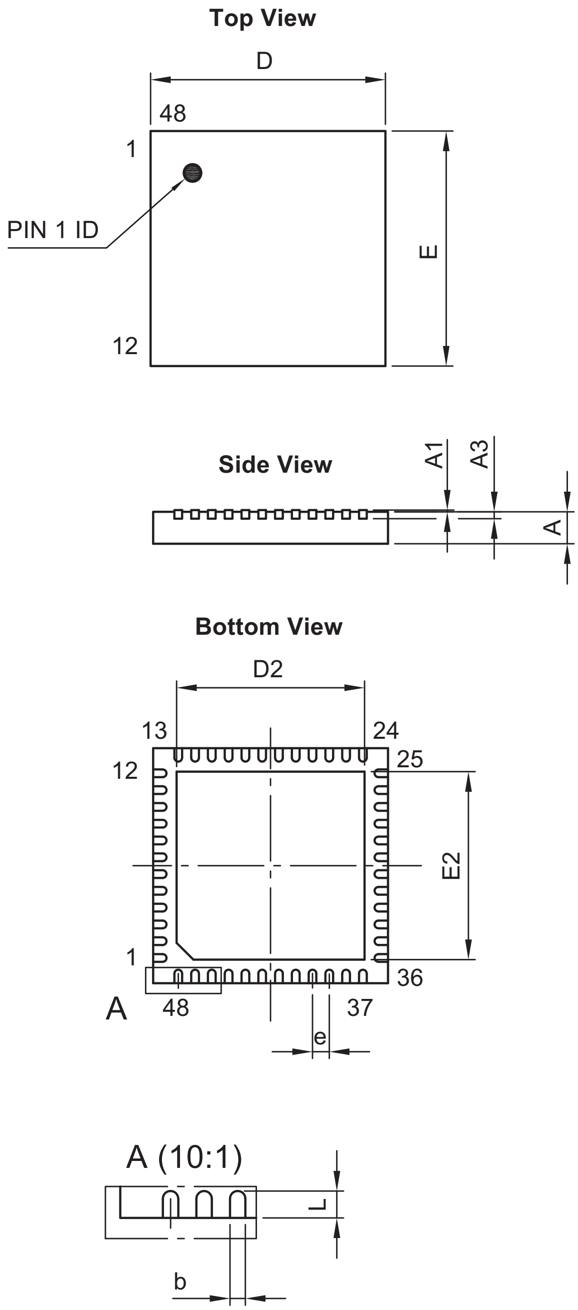
The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in High-voltage Serial Programming, only in SPI Programming.

32. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E (0x2E)	Reserved	–	–	–	–	–	–	–	–	
0x0D (0x2D)	Reserved	–	–	–	–	–	–	–	–	
0x0C (0x2C)	Reserved	–	–	–	–	–	–	–	–	
0x0B (0x2B)	Reserved	–	–	–	–	–	–	–	–	
0x0A (0x2A)	Reserved	–	–	–	–	–	–	–	–	
0x09 (0x29)	Reserved	–	–	–	–	–	–	–	–	
0x08 (0x28)	Reserved	–	–	–	–	–	–	–	–	
0x07 (0x27)	Reserved	–	–	–	–	–	–	–	–	
0x06 (0x26)	Reserved	–	–	–	–	–	–	–	–	
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	88
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	88
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	89
0x02 (0x22)	PORTA	–	–	–	–	–	–	PORTA1	PORTA0	88
0x01 (0x21)	DDRA	–	–	–	–	–	–	DDA1	DDA0	88
0x00 (0x20)	PINA	–	–	–	–	–	–	PINA1	PINA0	88

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00 - $1F 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 the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. 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 $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The Atmel AVR MCU 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 $60 - $FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

36. Packaging Information



technical drawings according to DIN specifications

Dimensions in mm

Standard Singulation process

COMMON DIMENSIONS (Unit of Measure = mm)				
Symbol	**MIN**	**NOM**	**MAX**	**NOTE**
A	0.8	0.9	1	
A1	0.0	0.02	0.05	
A3	0.15	0.2	0.25	
D	6.9	7	7.1	
D2	5.45	5.6	5.75	
E	6.9	7	7.1	
E2	5.45	5.6	5.75	
L	0.3	0.4	0.5	
b	0.16	0.23	0.3	
e		0.5 BSC		

09/07/11

Atmel **Package Drawing Contact:** packagedrawings@atmel.com	**TITLE** **Package: VQFN_7x7_48L** Exposed pad 5.6x5.6	**GPC**	**DRAWING NO.**	**REV.**
			6.543-5130.01-4	3

36.1 Markings

As a minimum, the devices will be marked with the following:

- Date code (year and week number)
- Atmel® part number