# Atmel ATmega32HVE2/ATmega64HVE2

## 8-bit AVR Microcontroller with Precise Analog Frontend for very Accurate Voltage and Current Measurement

**DATASHEET**

## Features

- Single-package fully-integrated
- High precision analog frontend
  - 17bit single-ended voltage-ADC
    - 7 selectable input channels
    - Offset voltage less than ±1LSB
  - 18 bit differential current-ADC with
    - Programmable gain amplifier
    - Comparator Mode
    - Offset voltage less than ±5µV
  - Temperature measurement with external and internal sensors
  - Integrated voltage divider with internal reverse polarity protection for direct sensing of the battery voltage
- Interface
  - LIN physical layer according to LIN 2.0, 2.1 and SAEJ2602-2
    - Fulfils the OEM “Hardware Requirements for LIN in Automotive Applications Rev. 1.1”
    - LIN hardware UART
    - Advanced ESD and EMC performance
    - High-speed Mode up to 115kBaud
- Microcontroller
  - High performance, low power AVR 8-bit microcontroller
    - 32bit math. extension module (+, –, x, /)
- Memory
  - 32K/64K in-system self-programmable flash memory
  - 1K EEPROM / 4K SRAM
- Power
  - Supply voltage –27V to +40V
  - Extreme low power consumption
- Others
  - Package: QFN48, 7x7mm$^2$
  - Temperature range: –40°C to +125°C

# 1. Description

With the ATmega32HVE2/ATmega64HVE2 Atmel® provides an 8-bit AVR® microcontroller with very precise analog frontend for voltage and current measurement and 32bit computing power. The circuit is a complete single-package system solution for applications like, e.g., 12V lead acid or Li-ion battery monitoring or particle filtering in automotive applications.

The device includes 2 dies, the first die (AVR MCU) with the very precise analog frontend consisting of

- a 17bit and a 18bit sigma delta ADC
- programmable gain amplifier with various chopper modes and extreme low offset
- 8-bit microcontroller with 32bit math-extensions module and 32/64Kbytes flash memory

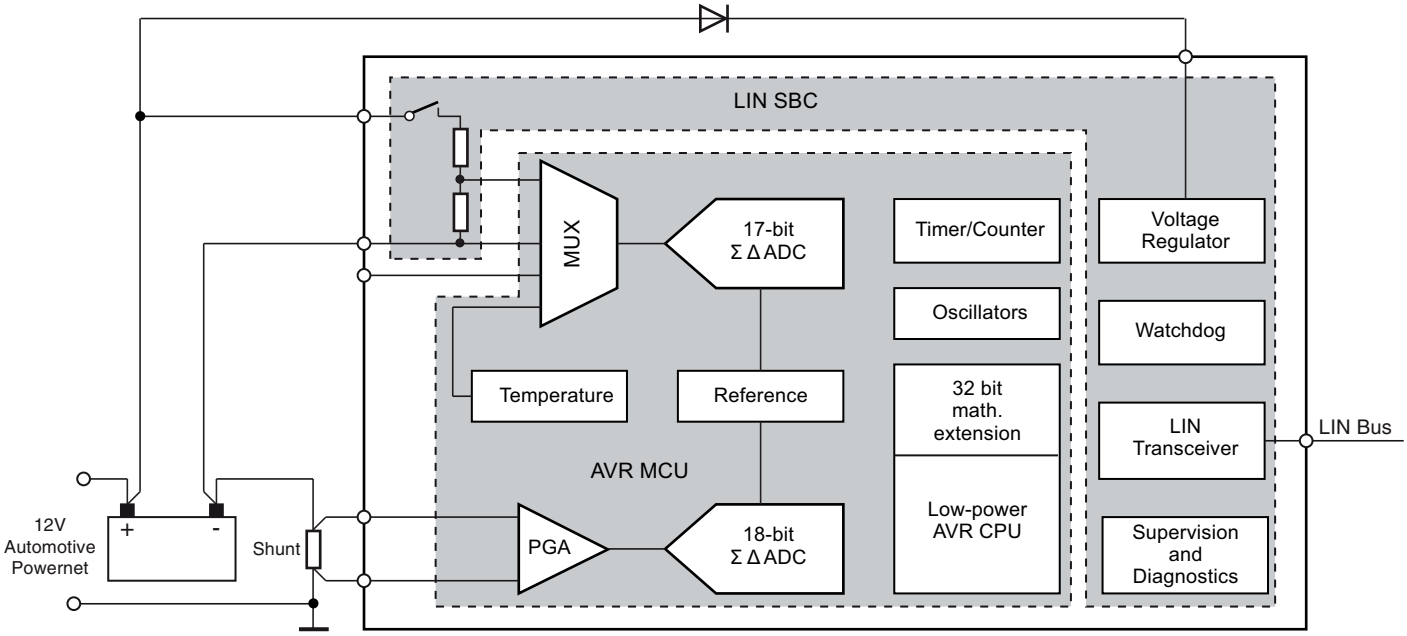
and a LIN(1) system basis chip (LIN SBC) including

- LIN transceiver according to the LIN2.0, 2.1 and SAEJ2602-2 standards
- 3.3V low drop voltage regulator
- window watchdog
- integrated voltage divider with reverse polarity protection for very precise sensing of the battery voltage

The device includes the same LIN SBC die as used in the Atmel ATA6628 LIN system basis chip from Atmel.

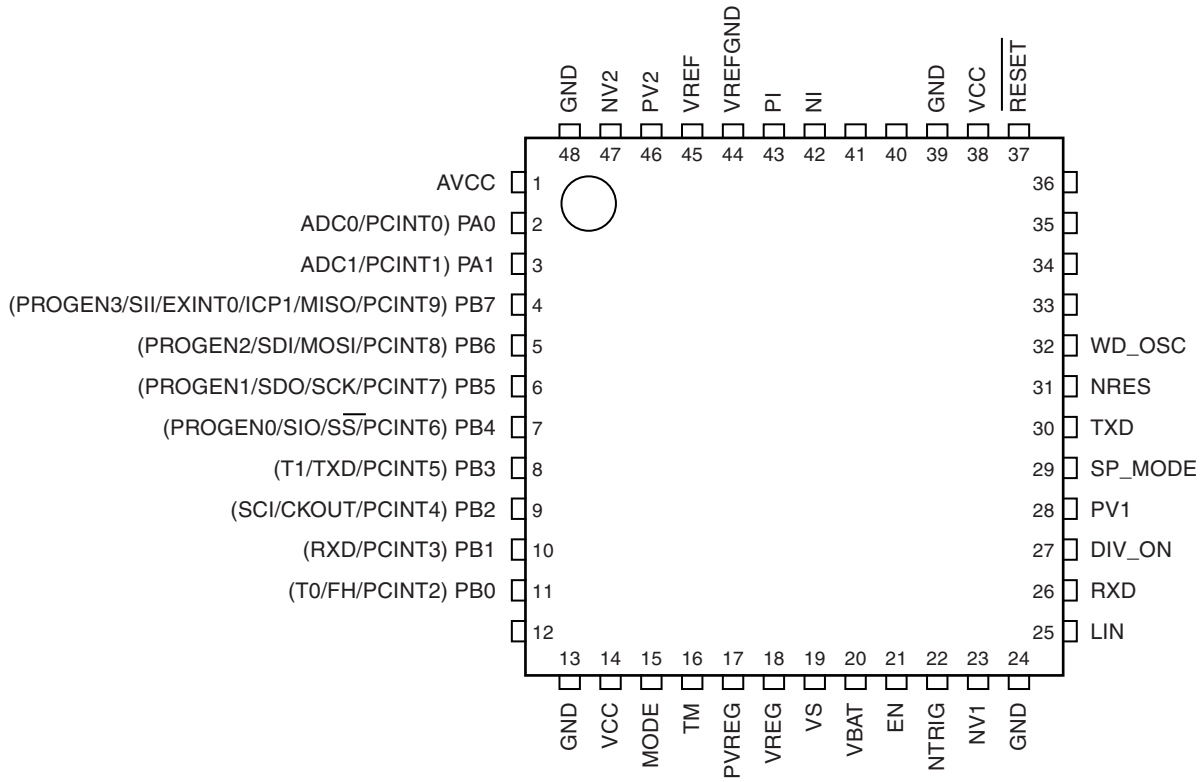
Note: 1. LIN: Local Interconnect Network

**Figure 1-1. Atmel ATmega32HVE2/ATmega64HVE2 Block Diagram**



# 2. Pin Configurations

**Figure 2-1. Pinout QFN-48**




## 2.1 Pin Descriptions

### 2.1.1 VCC

Digital supply voltage.

### 2.1.2 AVCC

Analog supply voltage.

### 2.1.3 VREF

Internal Voltage Reference for external decoupling. For details, see Section 27. “Band Gap Reference and Temperature Sensor” on page 161.

### 2.1.4 VREFGND

Ground for decoupling of Internal Voltage Reference. Do not connect to GND on PCB.

### 2.1.5 GND

Ground

### 2.1.6 Port A (PA1..PA0)

Port A serves as a 2-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A is connected to the input MUX of the Voltage ADC. To avoid any disturbance from Port A pins when doing high accuracy VADC measurements, is not recommended to connect noisy digital signals to these pins.

Port A also serves the functions of various special features of the Atmel® ATmega32HVE2/ATmega64HVE2 as listed in Section 21.3.1 “Alternate Functions of Port A” on page 85.

### 2.1.7 Port B (PB7..0)

Port B is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). 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 Atmel ATmega32HVE2/ATmega64HVE2 as listed in Section 21.3.2 “Alternate Functions of Port B” on page 86.

### 2.1.8 PV2/NV2

Filtered positive/negative input from resistor divider connected to VS. Used by the Voltage ADC to measure the battery pack voltage. For details, see Section 26. “ADC - Analog to Digital Converter” on page 138.

### 2.1.9 PI/NI

Filtered positive/negative input from external current sense resistor. Used by the Current ADC to measure charge/discharge currents flowing in the battery pack. For details, see Section 26. “ADC - Analog to Digital Converter” on page 138.

### 2.1.10 $\overline{\text{RESET}}$/dw

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 Section 31.5 “External Interrupt Characteristics” on page 198. Shorter pulses are not guaranteed to generate a reset. This pin is also used as debugWIRE communication pin.

### 2.1.11 VS

VS represents the power supply to the chip.The LIN operating voltage is VS = 5V to 27V. An undervoltage detection is implemented to disable data transmission if VS falls below $VS_{th}$ in order to avoid false bus messages. After switching on VS, the IC starts in Fail-safe Mode, and the voltage regulator is switched on (i.e., 3.3V/50mA output capability).

The supply current is typically 10µA in Sleep Mode and 40µA in Silent Mode.

### 2.1.12 VREG

The internal 3.3V voltage regulator is capable of driving loads up to 50mA. It is able to supply the microcontroller and other ICs on the PCB and is protected against overloads by means of current limitation and overtemperature shut-down. Furthermore, the output voltage is monitored and will cause a reset signal at the NRES output pin if it drops below a defined threshold Vthun. To boost up the maximum load current, an external NPN transistor may be used, with its base connected to the VREG pin and its emitter connected to PVREG.

### 2.1.13 PVREG

The PVREG is the sense input pin of the 3.3V voltage regulator. For normal applications (i.e. when only using the internal output transistor), this pin must be connected to the VREG pin. If an external boosting transistor is used, the PVREG pin must be connected to the output of this transistor, i.e, its emitter terminal.

### 2.1.14 LIN

A low-side driver with internal current limitation and thermal shutdown and an internal pull-up resistor compliant with the LIN 2.x specification is implemented. The allowed voltage range is between –27V and +40V. Reverse currents from the LIN bus to VS are suppressed, even in the event of GND shifts or battery disconnection. LIN receiver thresholds are compatible with the LIN protocol specification. The fall time from recessive to dominant bus state and the rise time from dominant to recessive bus state are slope controlled.

### 2.1.15 TXD

In Normal Mode the TXD pin is the microcontroller interface used to control the state of the LIN output. TXD must be pulled to ground in order to have a low LIN-bus. If TXD is high or not connected (internal pull-up resistor), the LIN output transistor is turned off, and the bus is in recessive state. During Fail-safe Mode, this pin is used as output and is signalling the fail-safe source. It is current-limited to $I_{TXDwake}$ (Section 10. “Electrical Characteristics LIN SBC” on page 25ff).

### 2.1.16 RXD

This output pin reports the state of the LIN-bus to the microcontroller. LIN high (recessive state) is reported by a high level at RXD; LIN low (dominant state) is reported by a low level at RXD. The output has an internal pull-up resistor with typically 5kΩ to PVREG. The AC characteristics can be defined with an external load capacitor of 20pF. The output is short-circuit protected. RXD is switched off in Unpowered Mode (i.e., VS = 0V). During Fail-safe Mode it is signalling the fail-safe source.

### 2.1.17 EN

The Enable Input pin controls the operation mode of the device. If EN is high, the circuit is in Normal Mode, with transmission paths from TXD to LIN and from LIN to RXD both active. The VREG voltage regulator operates with 3.3V/5V/50mA output capability.

If EN is switched to low while TXD is still high, the device is forced to Silent Mode. No data transmission is then possible, and the current consumption is reduced to $I_{VS}$ typ. 40µA. The VREG regulator has its full functionality.

If EN is switched to low while TXD is low, the device is forced to Sleep Mode. No data transmission is possible, and the voltage regulator is switched off.

### 2.1.18 MODE

With the pin MODE you can enable / disable the watchdog of the LIN SBC. Connect the MODE pin directly or via an external resistor to GND for normal watchdog operation. To debug the software of the connected microcontroller, connect MODE pin to PVREG and the watchdog is switched off.

Note: If you do not use the watchdog from the LIN SBC, connect pin MODE directly to PVREG.

### 2.1.19 TM

The TM pin is used for final production measurements at Atmel®. In normal application, it has to be always connected to GND.

### 2.1.20 NRES

The Reset Output pin, an open drain output, switches to low during VREG undervoltage or a watchdog failure generated by the LIN SBC.

### 2.1.21 WD_OSC

The WD_OSC Output pin provides a typical voltage of 1.2V, which supplies an external resistor with values between 34kΩ and 120kΩ to adjust the watchdog oscillator time. If the watchdog is disabled, this voltage is switched off and you can either tie to GND or leave this pin open. In the ATmega64HVE2 Operating circuit this pin is left open.

### 2.1.22 NTRIG

The NTRIG Input pin is the trigger input for the window watchdog of the LIN SBC. A pull-up resistor is implemented. A negative edge triggers the watchdog. The trigger signal (low) must exceed a minimum time $t_{trigmin}$ to generate a watchdog trigger (see also Section 9. “Watchdog” on page 23).

### 2.1.23 DIV_ON

The DIV_ON pin is a low voltage input. It is used to switch on or off the internal voltage divider PV1 output directly with no time limitation (see Table 2-1). It is switched on if DIV_ON is high or it is switched off if DIV_ON is low. In Sleep Mode the DIV_ON functionality is disabled and PV1 is off. An internal pull-down resistor is implemented.

### 2.1.24 VBATT

The VBAT is a high voltage input pin to supply the internal voltage divider. In an application with battery voltage monitoring, this pin can be connected to $V_{Battery}$ via, e.g., a 47Ω resistor in series and a 10nF capacitor to GND.

### 2.1.25 PV1

PV1 is the voltage divider output of the voltage divider between VBAT and NV1. The divider ratio is 1:24.

For applications with battery monitoring, this pin is directly connected to the ADC of a microcontroller. For buffering the ADC input an external capacitor might be needed. This pin guarantees a voltage and temperature stable output of a VBattery ratio. The PV1 output pin is controlled by the DIV_ON input pin.

**Table 2-1. Table of Voltage Divider**

| Mode of Operation | Input DiV_ON | Voltage Divider Output PV1 |
|---|---|---|
| Fail-safe/Normal/ High-speed/Silent | 0 | Off |
| | 1 | On |
| Sleep | 0 | Off |
| | 1 | Off |

### 2.1.26 SP_MODE

The SP_MODE pin is a low-voltage input. High-speed Mode of the transceiver can be activated via a high level during Normal Mode. Return to LIN 2.x Transceiver Mode with slope control is possible if you switch the SP_MODE pin to low.

### 2.1.27 NV1

This pin is directly connected to the base of the voltage divider. For battery voltage sensing this pin NV1 and pin PV1 should be directly connected to the two inputs of an AD-converter. NV1 should be additional connected to GND.

### 2.1.28 NC

Not connected pins are internally connected to GND.

# 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<br>$T_a$ = 25°C<br>Output current $I_{VREG}$ ≤ 50mA | $V_S$ | | | +40 | V |
| Pulse time ≤ 2min<br>$T_a$ = 25°C<br>Output current $I_{VREG}$ ≤ 50mA | $V_S$ | | | 27 | V |
| VBAT (with 47Ω/10nF)<br>DC voltage<br>Transient voltage due to ISO7637 3a, 3b (coupling 1nF) | | <br>–1<br>–150 | | <br>+40<br>+100 | <br>V<br>V |
| LIN, VBAT<br>- DC voltage | | <br>–27 | | <br>+40 | <br>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<br>VREG DC voltage | | –0.3<br>–0.3 | | +5.5<br>+6.5 | V<br>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<br>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<br>- Pin VS, LIN to GND<br>- Pin VBAT (10nF) to GND | | ±6 | | | KV |
| HBM ESD<br>ANSI/ESD-STM5.1<br>JESD22-A114<br>AEC-Q100 (002)<br>MIL-STD-883 (M3015.7) | | ±3 | | | KV |
| CDM ESD STM 5.3.1 | | ±750 | | | V |
| MM ESD<br>EIA/JESD22-A115<br>ESD STM5.2<br>AEC-Q100 (002) | | ±200 | | | V |
| ESD HBM following STM5.1 with 1.5kΩ 100pF<br>- Pin VS, LIN, VBAT to GND | | ±6 | | | KV |

## 3. Absolute Maximum Ratings (Continued)

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 |
|---|---|---|---|---|---|
| Junction temperature LIN SBC | $T_j$ | –40 | | +150 | °C |
| Junction temperature AVR MCU | $T_j$ | –40 | | +125 | °C |
| Storage temperature | $T_s$ | –55 | | +150 | °C |

## 4. Thermal Characteristics

| Parameters | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| Thermal shutdown of VREG regulator | | 150 | 165 | 170 | °C |
| Thermal shutdown of LIN output | | 150 | 165 | 170 | °C |
| Thermal shutdown hysteresis | | | 10 | | °C |
| Thermal resistance junction to heat slug | $R_{thjc}$ | | 6 | | K/W |
| Thermal resistance junction to ambient(1) | $R_{thja}$ | | 30 | | K/W |

Note: 1. JEDEC Multi-layer PCB, air flow.

# Atmel LIN System Basis Chip (LIN SBC)

## LIN Bus Transceiver with 3.3V Regulator and Watchdog

**PRELIMINARY DATASHEET**

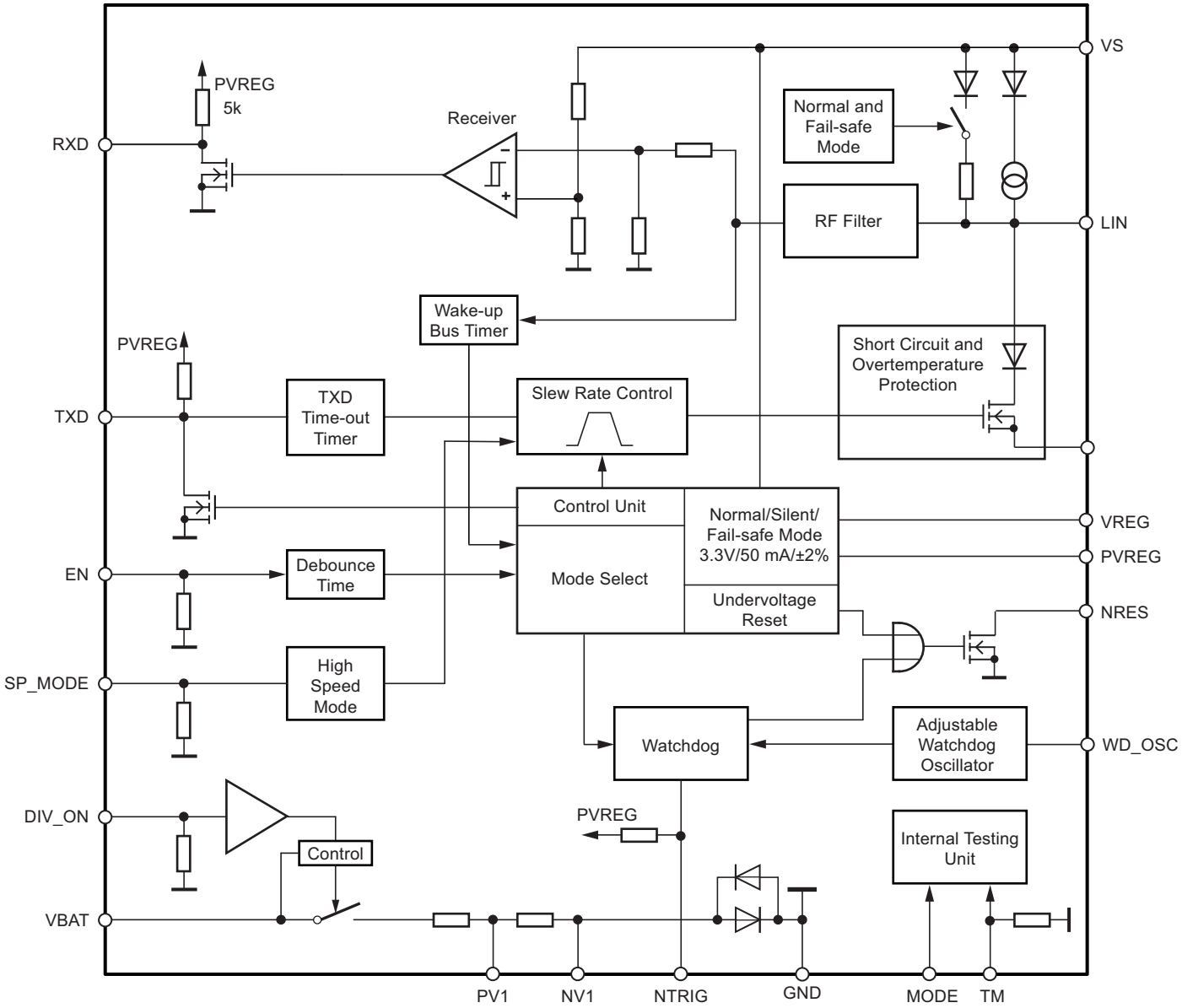
## Features

- Master and Slave Operation Possible
- Supply Voltage –27V to +40V
- Operating Voltage $V_S$ = 5V to 27V
- Typically 10µA Supply Current During Sleep Mode
- Typically 40µA Supply Current in Silent Mode
- Linear Low-drop Voltage Regulator:
  - Normal, Fail-safe, and Silent Mode
    - $V_{REG}$ = 3.3V ±2%
  - In Sleep Mode $V_{REG}$ is Switched Off
- VREG-Undervoltage Detection (4ms Reset Time) and Watchdog Reset Logical Combined at Open Drain Output NRES
- High-speed Mode Up to 115kBaud
- Internal 1:24 Voltage Divider for $V_{Battery}$ Sensing
- Negative Trigger Input for Watchdog
- Boosting the Voltage Regulator Possible with an External NPN Transistor
- LIN Physical Layer According to LIN 2.0, 2.1 and SAEJ2602-2
- Wake-up Capability via LIN-bus
- Bus Pin is Overtemperature and Short-circuit Protected versus GND and Battery
- Adjustable Watchdog Time via External Resistor
- Advanced EMC and ESD Performance
- Fulfills the OEM “Hardware Requirements for LIN in Automotive Applications Rev. 1.1”
- Atmel® ATA6628 LIN SBC inside

# 5. Description

The Atmel® LIN SBC is a fully integrated LIN transceiver, which complies with the LIN 2.0, 2.1 and SAEJ2602-2 specifications. It has a low-drop voltage regulator for 3.3V/50mA output and a window watchdog. The voltage regulator is able to source 50mA, but the output current can be boosted by using an external NPN transistor. This chip combination makes it possible to develop inexpensive, simple, yet powerful slave and master nodes for LIN-bus systems. Atmel LIN SBC is designed to handle the low-speed data communication in vehicles, e.g., in convenience electronics. Improved slope control at the LIN-driver ensures secure data communication up to 20kBaud. Sleep Mode and Silent Mode guarantee very low current consumption.

**Figure 5-1. Block Diagram**



# 6. Functional Description

## 6.1 Pin Functions

For pin functions of the LIN SBC please refer to Section 2.1 “Pin Descriptions” on page 3.

## 6.2 Physical Layer Compatibility

Since the LIN physical layer is independent from higher LIN layers (e.g., the LIN protocol layer), all nodes with a LIN physical layer according to revision 2.x can be mixed with LIN physical layer nodes, which, according to older versions (i.e., LIN 1.0, LIN 1.1, LIN 1.2, LIN 1.3), are without any restrictions.

## 6.3 Wake-u Events from Sleep or Silent Mode

- LIN-bus
- EN pin

## 6.4 Ground Shift

The IC does not affect the LIN-bus in the event of GND disconnection. It is able to handle a ground shift up to 11.5% of VS. This is the mandatory system ground pin.

## 6.5 TXD Dominant Time-out Function

The TXD input has an internal pull-up resistor. An internal timer prevents the bus line from being driven permanently in dominant state. If TXD is forced to low for longer than $t_{DOM}$ > 27ms, the LIN-bus driver is switched to recessive state. Nevertheless, when switching to Sleep Mode, the actual level at the TXD pin is relevant.

To reactivate the LIN bus driver, switch TXD to high (> 10µs).

# 7. Modes of Operation

**Figure 7-1. Modes of Operation**

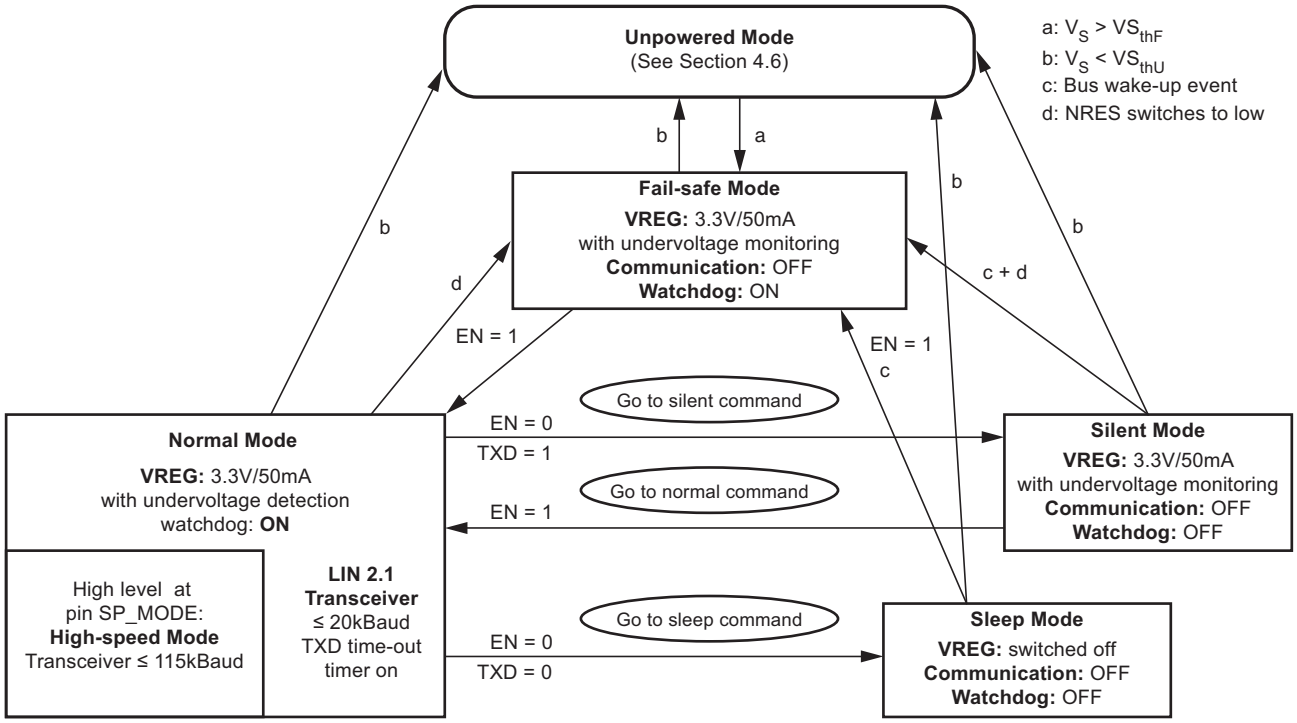



**Table 7-1. Table of Modes**

| Mode of Operation | Transceiver | Pin LIN | $V_{REG}$ | Pin Mode | Watchdog | Pin WD_OSC |
|---|---|---|---|---|---|---|
| Unpowered | Off | Recessive | On | GND | On | On |
| Fail-safe | Off | Recessive | 3.3V | GND | On | 1.23V |
| Normal/ High-speed | On | TXD depending | 3.3V | GND | On | 1.23V |
| Silent | Off | Recessive | 3.3V | GND | Off | 0V |
| Sleep | Off | Recessive | 0V | GND | Off | 0V |

## 7.1 Normal Mode

This is the normal transmitting and receiving mode. The voltage regulator is active and can source up to 50mA. The undervoltage detection is activated. The watchdog needs a trigger signal from NTRIG to avoid resets at NRES. If NRES is switched to low, the IC changes its state to Fail-safe Mode.

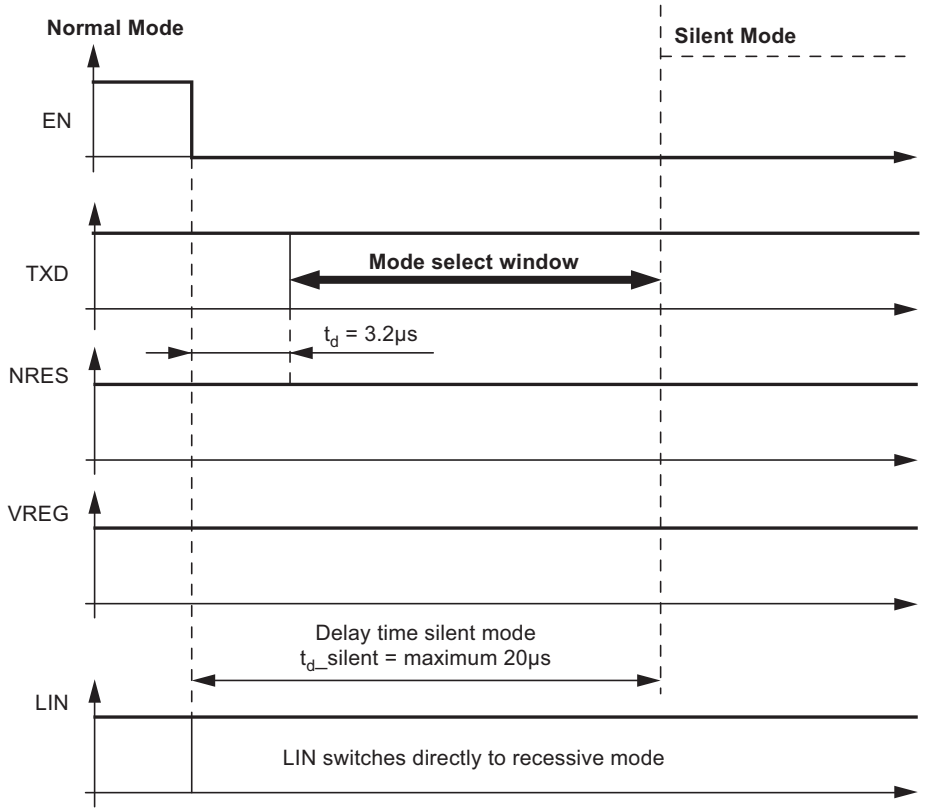
## 7.2 Silent Mode

A falling edge at EN when TXD is high switches the IC into Silent Mode. The TXD Signal has to be logic high during the Mode Select window (see Figure 7-2). The transmission path is disabled in Silent Mode. It is possible to switch on the voltage divider via pin DIV_ON. The overall supply current from $V_{Bat}$ is a combination of the $I_{VSsi}$ = 40µA plus the VREG regulator output current $I_{VREG}$.

The 3.3V regulator with 2% tolerance can source up to 50mA. The internal slave termination between the LIN pin and the VS pin is disabled in Silent Mode to minimize the current consumption in the event that the LIN pin is short-circuited to GND. Only a weak pull-up current (typically 10µA) between the LIN pin and the VS pin is present. Silent Mode can be activated independently from the actual level on the LIN. If an undervoltage condition occurs, NRES is switched to low, and the IC changes its state to Fail-safe Mode.

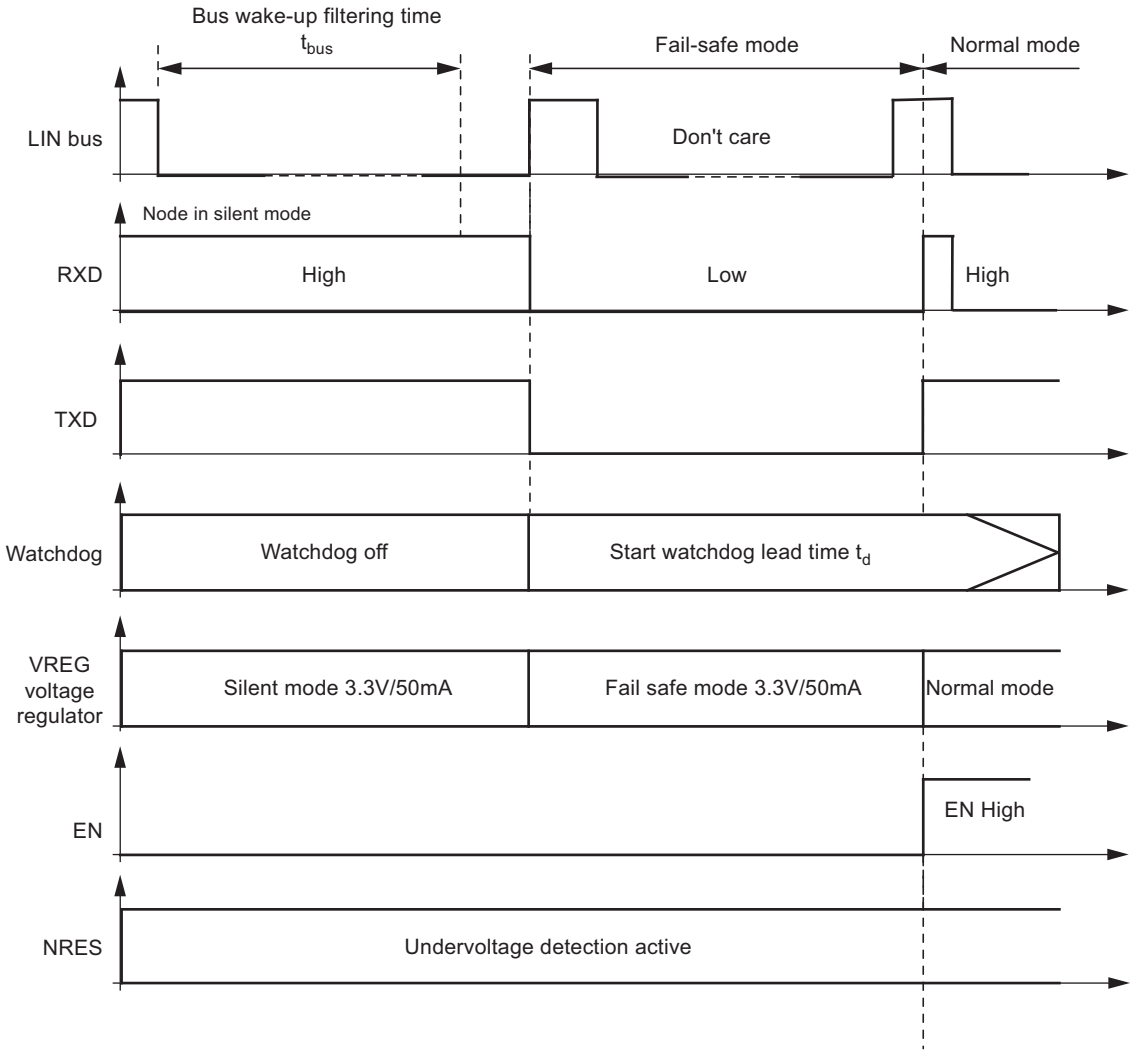
A voltage less than the LIN Pre_Wake detection VLINL at the LIN pin activates the internal LIN receiver and starts the wake-up detection timer.

**Figure 7-2. Switch to Silent Mode**



A falling edge at the LIN pin followed by a dominant bus level maintained for a certain time period ($t_{bus}$) and the following rising edge at the LIN pin (see Figure 7-3) result in a remote wake-up request, which is only possible if TXD is high. The device switches from Silent Mode to Fail-safe Mode. 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 7-3). EN high can be used to switch directly to Normal Mode.

**Figure 7-3. LIN Wake-up from Silent Mode**



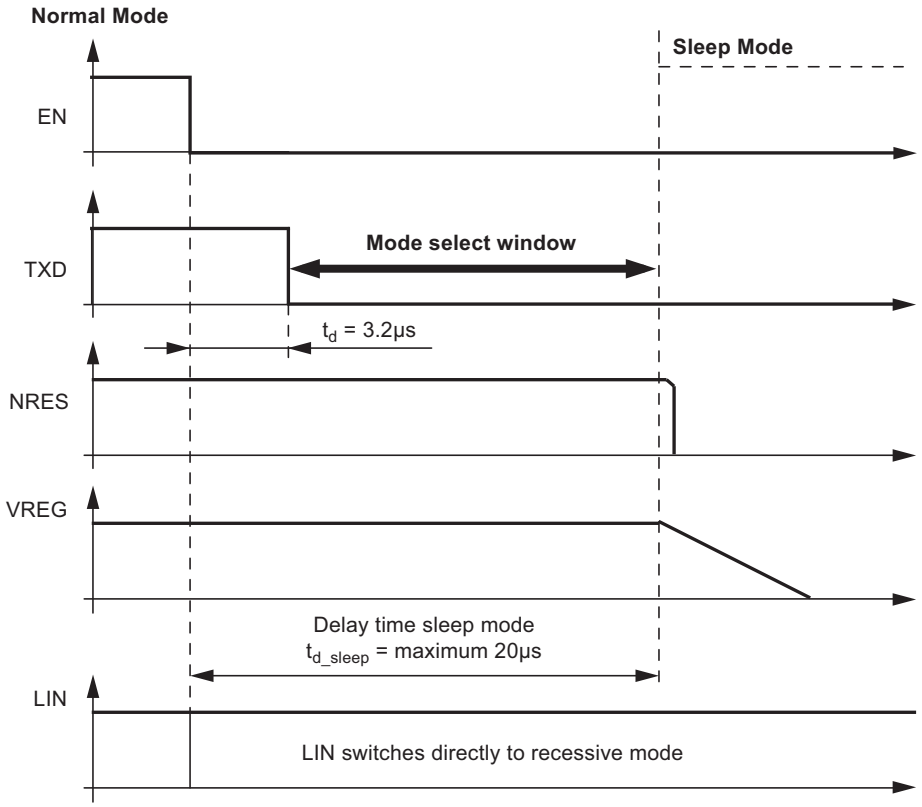
## 7.3 Sleep Mode

A falling edge at EN when TXD is low switches the IC into Sleep Mode. The TXD Signal has to be logic low during the Mode Select window (Figure 7-4). In order to avoid any influence to the LIN-pin during switching into sleep mode it is possible to switch the EN up to 3.2µs earlier to Low than the TXD. Therefore, the best an easiest way are two falling edges at TXD and EN at the same time. The transmission path is disabled in Sleep Mode. The supply current $I_{VSsleep}$ from $V_{Bat}$ is typically 10µA.

The PV1 output and the VREG regulator are switched off. NRES and RXD are low. The internal slave termination between the LIN pin and VS pin is disabled to minimize the current consumption in the event that the LIN pin is short-circuited to GND. Only a weak pull-up current (typically 10µA) between the LIN pin and the VS pin is present. Sleep Mode can be activated independently from the current level on the LIN.

A voltage less than the LIN Pre_Wake detection VLINL at the LIN pin activates the internal LIN receiver and starts the wake-up detection timer.

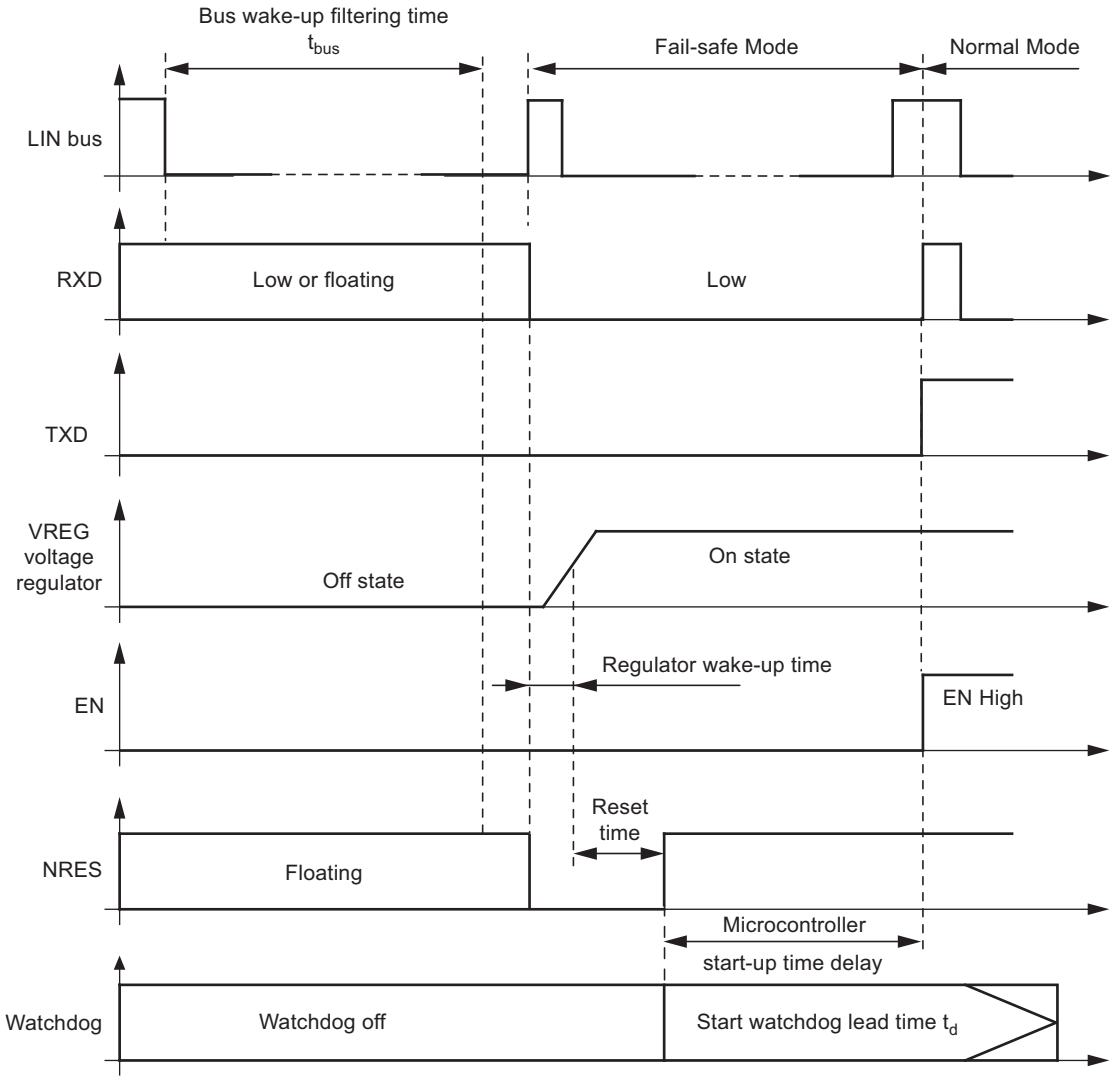
**Figure 7-4. Switch to Sleep Mode**



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**



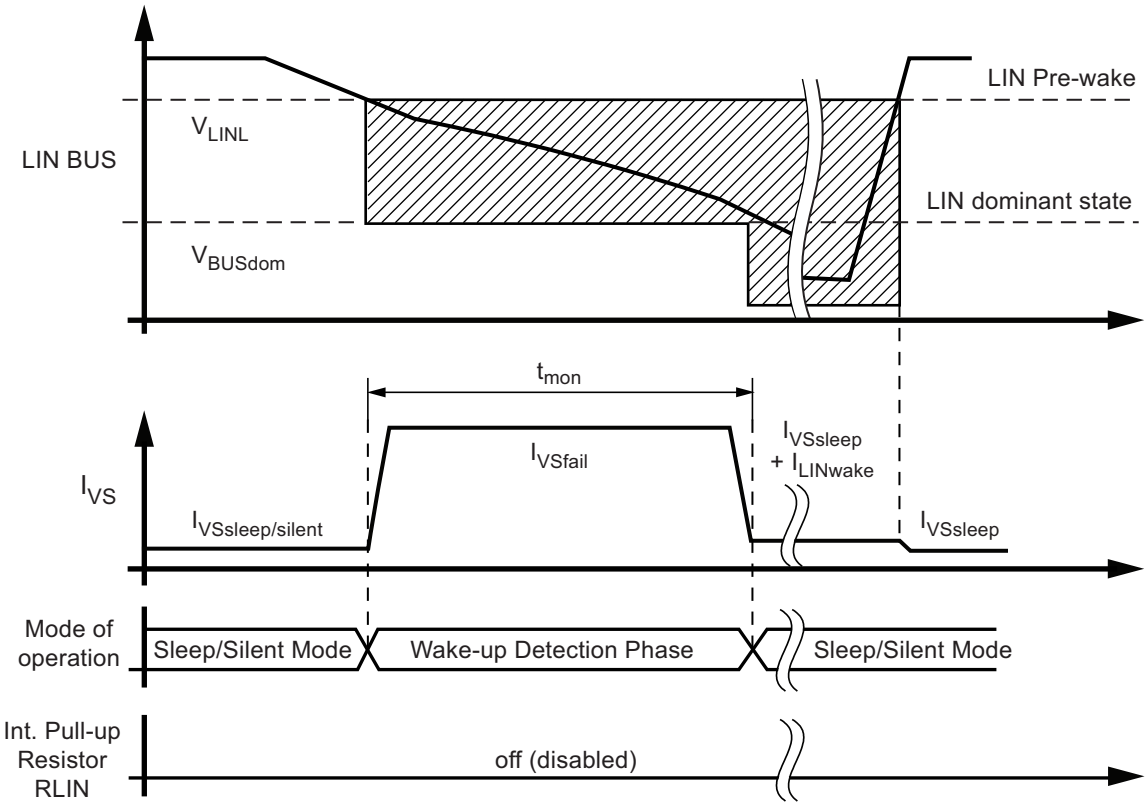
## 7.4 Sleep or Silent Mode: Behavior at a Floating LIN-bus or a Short Circuited LIN to GND

In Sleep or in Silent Mode the device has a very low current consumption even during shortcircuits or floating conditions on the bus. A floating bus can arise if the Master pull-up resistor is missing, e.g., if it is switched off when the LIN- Master is in sleep mode or even if the power supply of the Master node is switched off.

In order to minimize the current consumption $I_{VS}$ in sleep or silent mode during voltage levels at the LIN-pin below the LIN pre-wake threshold, the receiver is activated only for a specific time tmon. If $t_{mon}$ elapses while the voltage at the bus is lower than Pre-wake detection low ($V_{LINL}$) and higher than the LIN dominant level, the receiver is switched off again and the circuit changes back to sleep respectively Silent Mode. The current consumption is then the result of $I_{VSsleep}$ or $I_{VSsilent}$ plus $I_{LINwake}$. If a dominant state is reached on the bus no wake-up will occur. Even if the voltage rises above the Pre-wake detection high ($V_{LINH}$), the IC will stay in sleep respectively silent mode (see Figure 7-6).

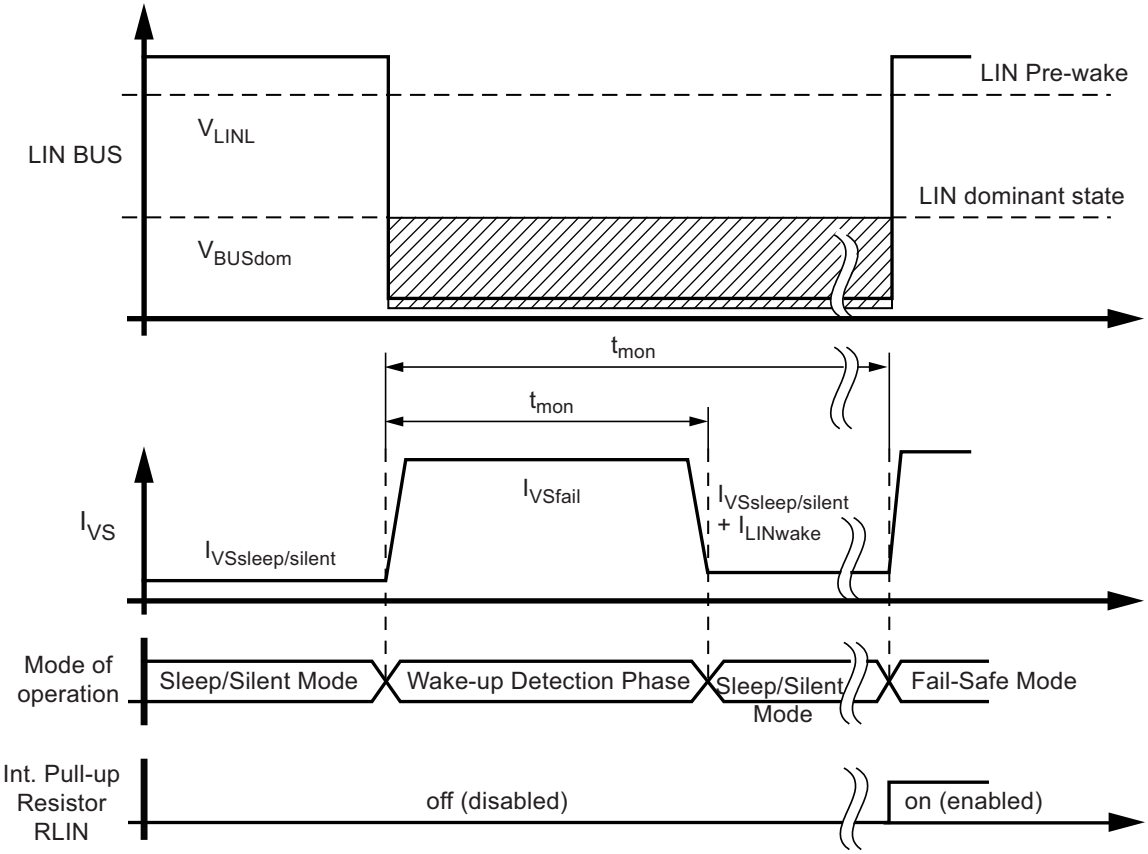
This means the LIN-bus must be above the Pre-wake detection threshold $V_{LINH}$ for a few microseconds before a new LIN wake-up is possible.

**Figure 7-6. Floating LIN-bus During Sleep or Silent Mode**



If the Atmel® LIN SBC is in Sleep or Silent Mode and the voltage level at the LIN-bus is in dominant state ($V_{LIN} < V_{BUSdom}$) for a time period exceeding $t_{mon}$ (during a short circuit at LIN, for example), the IC switches back to Sleep Mode respectively Silent Mode. The $V_S$ current consumption then consists of $I_{VSsleep}$ or $I_{VSsilent}$ plus $I_{LINWAKE}$. After a positive edge at pin LIN the IC switches directly to Fail-safe Mode (see Figure 7-7).

**Figure 7-7. Short Circuit to GND on the LIN-bus During Sleep- or Silent Mode**



## 7.5 Fail-safe Mode

The device automatically switches to Fail-safe Mode at system power-up. The voltage regulator is switched on ($V_{REG}$ = 3.3V/2%/50mA) (see Figure 8-1 on page 22). The NRES output switches to low for $t_{res}$ = 4ms and gives a reset to the microcontroller. LIN communication is switched off. The IC stays in this mode until EN is switched to high. The IC then changes to Normal Mode. A power down of $V_{Bat}$ ($V_S$ < $VS_{thU}$) during Silent or Sleep Mode switches the IC into Fail-safe Mode after power up. A low at NRES switches into Fail-safe Mode directly. During Fail-safe Mode, the TXD pin is an output and signals the fail-safe source. The watchdog is switched on.

The LIN SBC can operate in different Modes, like Normal, Silent, or Sleep Mode. The functionality of these modes is described in Table 7-2.

**Table 7-2. TXD, RXD Depending from Operation Modes**

| Different Modes | TXD | RXD |
|---|---|---|
| Fail-safe Mode | Signalling fail-safe sources (see Table 7-3 and Table 7-4) | |
| Normal Mode | Follows data transmission | |
| Silent Mode | High | High |

A wake-up event from either Silent or Sleep Mode will be signalled to the microcontroller using the two pins RXD and TXD. The coding is shown in Table 7-3.

A wake-up event will lead the IC to the Fail-safe Mode.

**Table 7-3. Signalling Fail-safe Sources**

| Fail-safe Sources | TXD | RXD |
|---|---|---|
| LIN wake-up (pin LIN) | Low | Low |
| $VS_{th}$ (battery) undervoltage detection | High | Low |

**Table 7-4. Signalling in Fail-safe Mode after Reset (NRES was Low), Shows the Reset Source at TXD and RXD Pins**

| Fail-safe Sources | TXD | RXD |
|---|---|---|
| VREG undervoltage at NRES | High | Low |
| Watchdog reset at NRES | High | High |

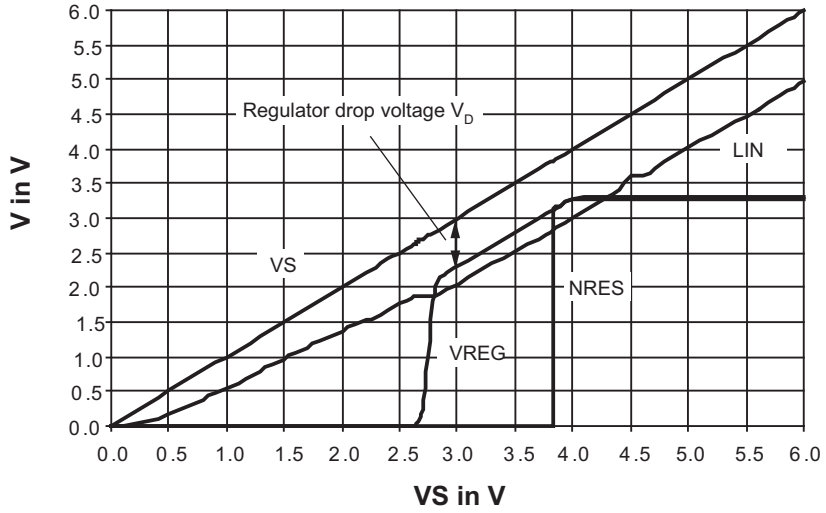
## 7.6 Unpowered Mode

If you connect battery voltage to the application circuit, the voltage at the VS pin increases according to the block capacitor (see Figure 8-1 on page 22). After VS is higher than the VS undervoltage threshold $VS_{th}$, the IC mode changes from Unpowered Mode to Fail-safe Mode. The VREG output voltage reaches its nominal value after $t_{VREG}$. This time, $t_{VREG}$, depends on the VREG capacitor and the load.

The NRES is low for the reset time delay $t_{reset}$. During this time, $t_{reset}$, no mode change is possible.

IF VS drops below $VS_{th}$, then the IC switches to Unpowered Mode. The behavior of VREG, NRES and LIN is shown in Figure 7-8. The watchdog needs to be triggered.

**Figure 7-8. Voltage Regulator: VREG versus VS**




## 7.7 High-speed Mode

If SP_MODE pin is high and the IC is in Normal Mode, the slew rate control is switched off. The slope time of the LIN falling edge is $t_{S_Fall}$ < 2µs. The slope time of the LIN rising edge strongly depends on the LIN capacitive and resistive load. To achieve a high baud rate it is recommended to use a small resistor (500Ω) and a low capacitor. This allows very fast data transmission up to 115kBaud, e.g., for electronic control (ECU) tests and microcontroller program or data download. In this mode superior EMC performance is not guaranteed.

# 8. Wake-up Scenarios from Silent or Sleep Mode

## 8.1 Remote Wake-up via Dominant Bus State

A voltage less than the LIN Pre_Wake detection $V_{LINL}$ at the LIN pin activates the internal LIN receiver and starts the wake-up detection timer.

A falling edge at the LIN pin followed by a dominant bus level $V_{BUSdom}$ maintained for a certain time period ($t_{BUS}$) and a rising edge at pin LIN result in a remote wake-up request. A remote wake up from Silent Mode is possible only if TXD is high. The device switches from Silent or Sleep Mode to Fail-safe Mode. The VREG voltage regulator is/remains activated and the internal slave termination resistor is switched on. The remote wake-up request is indicated by a low level at the RXD pin to generate an interrupt for the microcontroller and a strong pull down at TXD.

## 8.2 Wake-up Source Recognition

The device can distinguish between different wake-up sources (see Table 7-4 on page 19).

The wake-up source can be read on the TXD and RXD pin in Fail-safe Mode. These flags are immediately reset if the microcontroller sets the EN pin to high (see Figure 7-3 on page 14 and Figure 7-5 on page 16) and the IC is in Normal Mode.

## 8.3 Fail-safe Features

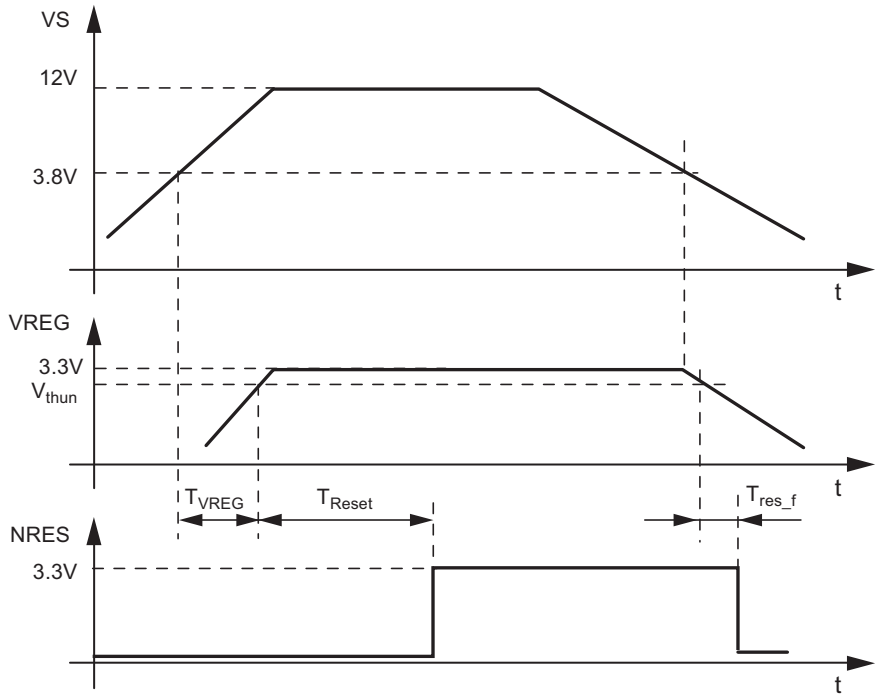
- During a short-circuit at LIN to $V_{Battery}$, the output limits the output current to $I_{BUS_lim}$. Due to the power dissipation, the chip temperature exceeds $T_{LINoff}$, and the LIN output is switched off. The chip cools down and after a hysteresis of $T_{hys}$, switches the output on again. RXD stays on high because LIN is high. During LIN overtemperature switch-off, the VREG regulator works independently.
- During a short-circuit from LIN to GND the IC can be switched into Sleep or Silent Mode and even in this case the current consumption is lower than 45µA in Sleep Mode and lower than 80µA in Silent Mode. If the short-circuit disappears, the IC starts with a remote wake-up.
- Sleep or Silent Mode: During a floating condition on the bus the IC switches back to Sleep Mode/Silent Mode automatically and thereby the current consumption is lower than 45µA/80µA.
- The reverse current is < 2µA at the LIN pin during loss of $V_{Bat}$. This is optimal behavior for bus systems where some slave nodes are supplied from battery or ignition.
- During a short circuit at VREG, the output limits the output current to $I_{VREGlim}$. Because of undervoltage, NRES switches to low and sends a reset to the microcontroller if NRES is connected to the microcontroller. The IC switches into Fail-safe Mode. If the chip temperature exceeds the value $T_{VREGoff}$, the VREG output switches off. The chip cools down and after a hysteresis of $T_{hys}$, switches the output on again. Because of the Fail-safe Mode, the VREG voltage will switch on again although EN is switched off from the microcontroller. The microcontroller can start with its normal operation.
- EN pin provides a pull-down resistor to force the transceiver into recessive mode if EN is disconnected.
- RXD pin is set floating if $V_{Bat}$ is disconnected.
- TXD pin provides a pull-up resistor to force the transceiver into recessive mode if TXD is disconnected.
- If TXD is short-circuited to GND, it is possible to switch to Sleep Mode via ENABLE.
- If the WD_OSC pin has a short circuit to GND and the NTRIG signal has a period time > 27ms a reset is guaranteed.
- If the resistor at the WD_OSC pin is disconnected and the NTRIG signal has a period time < 46ms a reset is guaranteed.
- If there is no NTRIG signal and short circuit at WD_OSC the NRES switches to low after 90ms. For an open circuit (no resistor) at WD_OSC it switches to low after typ. 390ms.

## 8.4 Voltage Regulator

The voltage regulator needs an external capacitor for compensation and for smoothing the disturbances from the microcontroller. It is recommended to use an electrolythic capacitor with C > 1.8µF and a ceramic capacitor with C = 100nF. The values of these capacitors can be varied by the customer, depending on the application.

The main power dissipation of the IC is created from the VREG output current $I_{VREG}$, which is needed for the application.

**Figure 8-1. VREG Voltage Regulator: Ramp-up and Undervoltage Detection**




For microcontroller programming, it may be necessary to supply the VREG output via an external power supply while the $V_S$ Pin of the system basis chip is disconnected. This behavior is no problem for the system basis chip.

# 9. Watchdog

The watchdog anticipates a trigger signal from the microcontroller at the NTRIG (negative edge) input within a time window of $T_{wd}$. The trigger signal must exceed a minimum time $t_{trigmin}$ > 4µs. If a triggering signal is not received, a reset signal will be generated at output NRES. The timing basis of the watchdog is provided by the internal oscillator. Its time period, $T_{osc}$, is adjustable via the external resistor $R_{wd_osc}$ (34kΩ to 120kΩ).

During Silent or Sleep Mode the watchdog is switched off to reduce current consumption.

The minimum time for the first watchdog pulse is required after the undervoltage reset at NRES disappears. It is defined as lead time $t_d$. After wake up from Sleep or Silent Mode, the lead time $t_d$ starts with the negative edge of the RXD output.

## 9.1 Typical Timing Sequence with $R_{WD_OSC}$ = 51kΩ

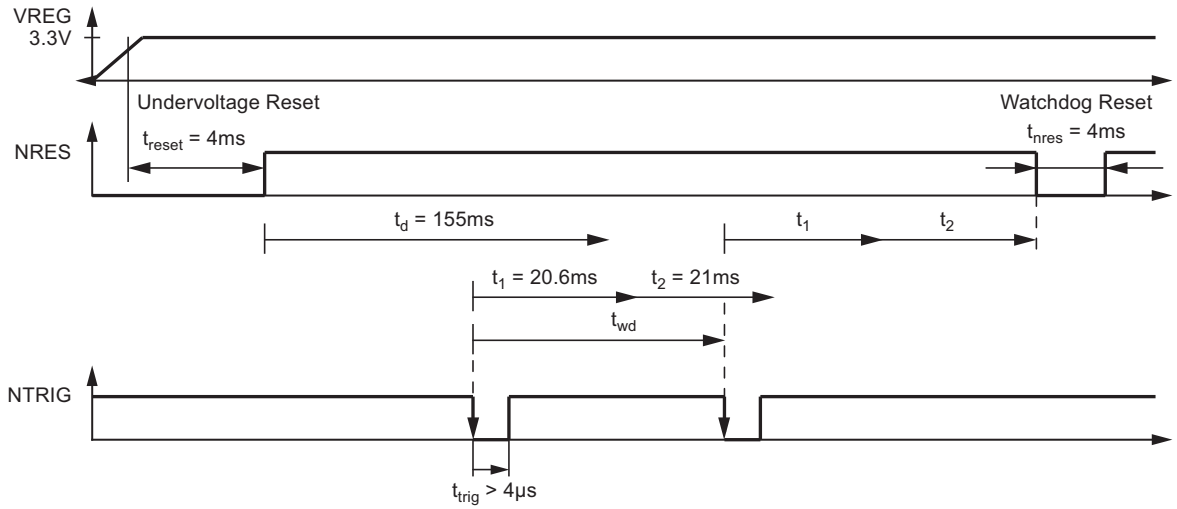
The trigger signal $T_{wd}$ is adjustable between 20ms and 64ms using the external resistor $R_{WD_OSC}$.

For example, with an external resistor of $R_{WD_OSC}$ = 51kΩ ±1%, the typical parameters of the watchdog are as follows:

$t_{osc} = 0.405 \times R_{WD_OSC} - 0.0004 \times (R_{WD_OSC})^2$ ($R_{WD_OSC}$ in kΩ; $t_{osc}$ in µs)
$t_{OSC}$ = 19.6µs due to 51kΩ
$t_d = 7895 \times$ 19.6µs = 155ms
$t_1 = 1053 \times$ 19.6µs = 20.6ms
$t_2 = 1105 \times$ 19.6µs = 21.6ms
$t_{nres}$ = constant = 4ms

After ramping up the battery voltage, the 5V regulator is switched on. The reset output NRES stays low for the time $t_{reset}$ (typically 4ms), then it switches to high, and the watchdog waits for the trigger sequence from the microcontroller. The lead time, $t_d$, follows the reset and is $t_d$ = 155ms. In this time, the first watchdog pulse from the microcontroller is required. If the trigger pulse NTRIG occurs during this time, the time $t_1$ starts immediately. If no trigger signal occurs during the time $t_d$, a watchdog reset with $t_{NRES}$ = 4ms will reset the microcontroller after $t_d$ = 155ms. The times $t_1$ and $t_2$ have a fixed relationship. A triggering signal from the microcontroller is anticipated within the time frame of $t_2$ = 21.6ms. To avoid false triggering from glitches, the trigger pulse must be longer than $t_{TRIG,min}$ > 200ns. This slope serves to restart the watchdog sequence. If the triggering signal fails in this open window $t_2$, the NRES output will be drawn to ground. A triggering signal during the closed window $t_1$ immediately switches NRES to low.

**Figure 9-1. Timing Sequence with $R_{WD_OSC}$ = 51kΩ**



## 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 $t_2$.

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

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

$t_{wd} = 29.3ms \pm 4.5ms$ (±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

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* |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VS Pin | | | | | | | | |
| 1.1 | Nominal DC voltage range | | VS | $V_S$ | 5 | | 27 | V | A |
| 1.2 | Supply current in Sleep Mode | Sleep Mode<br>$V_{LIN}$ > $V_S$ – 0.5V<br>$V_S$ < 14V | VS | $I_{VSsleep}$ | 3 | 10 | 14 | µA | A |
| | | Sleep Mode, $V_{LIN}$ = 0V<br>Bus shorted to GND<br>$V_S$ < 14V | VS | $I_{VSsleep_short}$ | 6 | 17 | 30 | µA | A |
| 1.3 | Supply current in Silent Mode | Bus recessive<br>$V_S$ < 14V ($T_j$ = 25°C)<br>Without load at VREG | VS | $I_{VSsi}$ | 20 | 35 | 45 | µA | A |
| | | Bus recessive<br>$V_S$ < 14V ($T_j$ = 125°C)<br>Without load at VREG | VS | $I_{VSsi}$ | 25 | 40 | 50 | µA | A |
| | | Silent Mode<br>$V_S$ < 14V<br>Bus shorted to GND<br>Without load at VREG | VS | $I_{VSsi_short}$ | 25 | 50 | 80 | µA | A |
| 1.4 | Supply current in Normal Mode | Bus recessive<br>$V_S$ < 14V<br>Without load at VREG | VS | $I_{VSrec}$ | 0.3 | | 0.8 | mA | A |
| 1.5 | Supply current in Normal Mode | Bus recessive<br>$V_S$ < 14V<br>$V_{REG}$ load current 50mA | VS | $I_{VSdom}$ | 50 | | 53 | mA | A |
| 1.6 | Supply current in Fail-safe Mode | Bus recessive, RXD is low<br>$V_S$ < 14V<br>Without load at VREG | VS | $I_{VSfail}$ | 0.8 | | 1.5 | mA | A |
| 1.7 | VS undervoltage threshold | Switch to Unpowered Mode | VS | $V_{SthU}$ | 4.1 | 4.4 | 4.7 | V | A |
| | | Switch to Fail-safe Mode | VS | $V_{SthF}$ | 4.4 | 4.7 | 4.9 | V | A |
| 1.8 | VS undervoltage threshold hysteresis | | VS | $V_{Sth_hys}$ | | 0.3 | | V | A |
| 2 | RXD Output Pin | | | | | | | | |
| 2.1 | Low-level output sink current | Normal Mode<br>$V_{LIN}$ = 0V<br>$V_{RXD}$ = 0.4V | RXD | $I_{RXD}$ | 1.3 | 2.5 | 8 | mA | A |
| 2.2 | Low-level output voltage | $I_{RXD}$ = 1mA | RXD | $V_{RXDL}$ | | | 0.4 | V | A |
| 2.3 | Internal resistor to PVREG | | RXD | $R_{RXD}$ | 3 | 5 | 7 | kΩ | A |

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

## 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* |
|---|---|---|---|---|---|---|---|---|---|
| 3 | TXD Input/Output Pin | | | | | | | | |
| 3.1 | Low-level voltage input | | TXD | $V_{TXDL}$ | –0.3 | | +0.8 | V | A |
| 3.2 | High-level voltage input | | TXD | $V_{TXDH}$ | 2 | | $V_{REG}$ + 0.3V | V | A |
| 3.3 | Pull-up resistor | $V_{TXD}$ = 0V | TXD | $R_{TXD}$ | 125 | 250 | 400 | kΩ | A |
| 3.4 | High-level leakage current | $V_{TXD} = V_{REG}$ | TXD | $I_{TXD}$ | –3 | | +3 | µA | A |
| 3.5 | Low-level output sink current | Fail-safe Mode, wake up<br>$V_{LIN} = V_S$<br>$V_{WAKE}$ = 0V<br>$V_{TXD}$ = 0.4V | TXD | $I_{TXDwake}$ | 2 | 2.5 | 8 | mA | A |
| 4 | EN Input Pin | | | | | | | | |
| 4.1 | Low-level voltage input | | EN | $V_{ENL}$ | –0.3 | | +0.8 | V | A |
| 4.2 | High-level voltage input | | EN | $V_{ENH}$ | 2 | | $V_{REG}$ + 0.3V | V | A |
| 4.3 | Pull-down resistor | $V_{EN} = V_{REG}$ | EN | $R_{EN}$ | 50 | 125 | 200 | kΩ | A |
| 4.4 | Low-level input current | $V_{EN}$ = 0V | EN | $I_{EN}$ | –3 | | +3 | µA | A |
| 5 | NTRIG Watchdog Input Pin | | | | | | | | |
| 5.1 | Low-level voltage input | | NTRIG | $V_{NTRIGL}$ | –0.3 | | +0.8 | V | A |
| 5.2 | High-level voltage input | | NTRIG | $V_{NTRIGH}$ | 2 | | $V_{REG}$ + 0.3V | V | A |
| 5.3 | Pull-up resistor | $V_{NTRIG}$ = 0V | NTRIG | $R_{NTRIG}$ | 125 | 250 | 400 | kΩ | A |
| 5.4 | High-level leakage current | $V_{NTRIG} = V_{REG}$ | NTRIG | $I_{NTRIG}$ | –3 | | +3 | µA | A |
| 6 | Mode Input Pin | | | | | | | | |
| 6.1 | Low-level voltage input | | MODE | $V_{MODEL}$ | –0.3 | | +0.8 | V | A |
| 6.2 | High-level voltage input | | MODE | $V_{MODEH}$ | 2 | | $V_{REG}$ + 0.3V | V | A |
| 6.3 | High-level leakage current | $V_{MODE} = V_{REG}$ or<br>$V_{MODE}$ = 0V | MODE | $I_{MODE}$ | –3 | | +3 | µA | A |
| 7 | LIN Bus Driver | | | | | | | | |
| 7.1 | Driver recessive output voltage | Load1/Load2 | LIN | $V_{BUSrec}$ | $0.9 \times V_S$ | | $V_S$ | V | A |
| 7.2 | Driver dominant voltage | $V_{VS}$ = 7V<br>$R_{load}$ = 500Ω | LIN | $V_{_LoSUP}$ | | | 1.2 | V | A |
| 7.3 | Driver dominant voltage | $V_{VS}$ = 18V<br>$R_{load}$ = 500Ω | LIN | $V_{_HiSUP}$ | | | 2 | V | A |
| 7.4 | Driver dominant voltage | $V_{VS}$ = 7.0V<br>$R_{load}$ = 1000Ω | LIN | $V_{_LoSUP_1k}$ | 0.6 | | | V | A |
| 7.5 | Driver dominant voltage | $V_{VS}$ = 18V<br>$R_{load}$ = 1000Ω | LIN | $V_{_HiSUP_1k}$ | 0.8 | | | V | A |
| 7.6 | Pull-up resistor to VS | The serial diode is mandatory | LIN | $R_{LIN}$ | 20 | 30 | 47 | kΩ | A |
| 7.7 | Voltage drop at the serial diodes | In pull-up path with $R_{slave}$<br>$I_{SerDiode}$ = 10mA | LIN | $V_{SerDiode}$ | 0.4 | | 1.0 | V | D |

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

## 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* |
|---|---|---|---|---|---|---|---|---|---|
| 7.8 | LIN current limitation $V_{BUS} = V_{Bat_max}$ | | LIN | $I_{BUS_LIM}$ | 70 | 120 | 200 | mA | A |
| 7.9 | Input leakage current at the receiver including pull-up resistor as specified | Input leakage current<br>Driver off<br>$V_{BUS} = 0V$<br>$V_{Bat} = 12V$ | LIN | $I_{BUS_PAS_dom}$ | –1 | –0.35 | | mA | A |
| 7.10 | Leakage current LIN recessive | Driver off<br>$8V < V_{Bat} < 18V$<br>$8V < V_{BUS} < 18V$<br>$V_{BUS} \geq V_{Bat}$ | LIN | $I_{BUS_PAS_rec}$ | | 10 | 20 | µA | A |
| 7.11 | Leakage current at GND loss, control unit disconnected from ground. Loss of local ground must not affect communication in the residual network. | $GND_{Device} = V_S$<br>$V_{Bat} = 12V$<br>$0V < V_{BUS} < 18V$ | LIN | $I_{BUS_NO_gnd}$ | –10 | +0.5 | +10 | µA | A |
| 7.12 | Leakage current at loss of battery. Node has to sustain the current that can flow under this condition. Bus must remain operational under this condition. | $V_{Bat}$ disconnected<br>$V_{SUP_Device} = GND$<br>$0V < V_{BUS} < 18V$ | LIN | $I_{BUS_NO_bat}$ | | 0.1 | 2 | µA | A |
| 7.13 | Capacitance on pin LIN to GND | | LIN | $C_{LIN}$ | | | 20 | pF | D |
| 8 | LIN Bus Receiver | | | | | | | | |
| 8.1 | Center of receiver threshold | $V_{BUS_CNT} = (V_{th_dom} + V_{th_rec})/2$ | LIN | $V_{BUS_CNT}$ | $0.475 \times V_S$ | $0.5 \times V_S$ | $0.525 \times V_S$ | V | A |
| 8.2 | Receiver dominant state | $V_{EN} = V_{REG}$ | LIN | $V_{BUSdom}$ | | | $0.4 \times V_S$ | V | A |
| 8.3 | Receiver recessive state | $V_{EN} = V_{REG}$ | LIN | $V_{BUSrec}$ | $0.6 \times V_S$ | | | V | A |
| 8.4 | Receiver input hysteresis | $V_{hys} = V_{th_rec} – V_{th_dom}$ | LIN | $V_{BUShys}$ | $0.028 \times V_S$ | $0.1 \times V_S$ | $0.175 \times V_S$ | V | A |
| 8.5 | Pre_Wake detection LIN High-level input voltage | | LIN | $V_{LINH}$ | $V_S – 2V$ | | $V_S + 0.3V$ | V | A |
| 8.6 | Pre_Wake detection LIN Low-level input voltage | Activates the LIN receiver | LIN | $V_{LINL}$ | –27 | | $V_S – 3.3V$ | V | A |
| 9 | Internal Timers | | | | | | | | |
| 9.1 | Dominant time for wake-up via LIN-bus | $V_{LIN} = 0V$ | LIN | $t_{bus}$ | 30 | 90 | 150 | µs | A |
| 9.2 | Time delay for mode change from Fail-safe into Normal Mode via EN pin | $V_{EN} = V_{REG}$ | EN | $t_{norm}$ | 5 | 15 | 20 | µs | A |
| 9.3 | Time delay for mode change from Normal Mode to Sleep Mode via EN pin | $V_{EN} = 0V$ | EN | $t_{sleep}$ | 5 | 15 | 20 | µs | A |
| 9.4 | TXD dominant time-out timer | $V_{TXD} = 0V$ | TXD | $t_{dom}$ | 27 | 55 | 70 | ms | A |

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

# 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**<br>Load 1 (small): 1nF, 1kΩ; Load 2 (large): 10nF, 500Ω; $R_{RXD}$ = 5kΩ; $C_{RXD}$ = 20pF;<br>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 | $TH_{Rec(max)} = 0.744 \times V_S$<br>$TH_{Dom(max)} = 0.581 \times V_S$<br>$V_S$ = 7.0V to 18V<br>$t_{Bit}$ = 50µs<br>D1 = $t_{bus_rec(min)}/(2 \times t_{Bit})$ | LIN | D1 | 0.396 | | | | A |
| 9.8 | Duty cycle 2 | $TH_{Rec(min)} = 0.422 \times V_S$<br>$TH_{Dom(min)} = 0.284 \times V_S$<br>$V_S$ = 7.6V to 18V<br>$t_{Bit}$ = 50µs<br>D2 = $t_{bus_rec(max)}/(2 \times t_{Bit})$ | LIN | D2 | | | 0.581 | | A |
| 9.9 | Duty cycle 3 | $TH_{Rec(max)} = 0.778 \times V_S$<br>$TH_{Dom(max)} = 0.616 \times V_S$<br>$V_S$ = 7.0V to 18V<br>$t_{Bit}$ = 96µs<br>D3 = $t_{bus_rec(min)}/(2 \times t_{Bit})$ | LIN | D3 | 0.417 | | | | A |
| 9.10 | Duty cycle 4 | $TH_{Rec(min)} = 0.389 \times V_S$<br>$TH_{Dom(min)} = 0.251 \times V_S$<br>$V_S$ = 7.6V to 18V<br>$t_{Bit}$ = 96µs<br>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 | $t_{SLOPE_fall}$<br>$t_{SLOPE_rise}$ | 3.5 | | 22.5 | µs | A |
| 10 | Receiver Electrical AC Parameters of the LIN Physical Layer<br>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<br>$t_{rx_pd}$ = 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<br>$t_{rx_sym} = t_{rx_pdr} - t_{rx_pdf}$ | RXD | $t_{rx_sym}$ | –2 | | +2 | µs | A |
| 11 | NRES Open Drain Output Pin | | | | | | | | |
| 11.1 | Low-level output voltage | $V_S \geq 5.5V$<br>$I_{NRES}$ = 1mA | NRES | $V_{NRESL}$ | | | 0.14 | V | A |
| 11.2 | Low-level output low | 10kΩ to 5V<br>$V_{REG}$ = 0V | NRES | $V_{NRESLL}$ | | | 0.14 | V | A |
| 11.3 | Undervoltage reset time | $V_S \geq 5.5V$<br>$C_{NRES}$ = 20pF | NRES | $t_{reset}$ | 2 | 4 | 6 | ms | A |
| 11.4 | Reset debounce time for falling edge | $V_S \geq 5.5V$<br>$C_{NRES}$ = 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

## 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* |
|---|---|---|---|---|---|---|---|---|---|
| 11.5 | Switch off leakage current | $V_{NRES}$ = 5.5V | NRES | | –3 | | +3 | µA | A |
| 12 | Watchdog Oscillator | | | | | | | | |
| 12.1 | Voltage at WD_OSC in Normal or Fail-safe Mode | $I_{WD_OSC}$ = –200µA<br>$V_{VS}$ ≥ 4V | WD_OSC | $V_{WD_OSC}$ | 1.13 | 1.23 | 1.33 | V | A |
| 12.2 | Possible values of resistor | Resistor ±1% | WD_OSC | $R_{OSC}$ | 34 | | 120 | kΩ | A |
| 12.3 | Oscillator period | $R_{OSC}$ = 34kΩ | | $t_{OSC}$ | 10.65 | 13.3 | 15.97 | µs | A |
| 12.4 | Oscillator period | $R_{OSC}$ = 51kΩ | | $t_{OSC}$ | 15.68 | 19.6 | 23.52 | µs | A |
| 12.5 | Oscillator period | $R_{OSC}$ = 91kΩ | | $t_{OSC}$ | 26.83 | 33.5 | 40.24 | µs | A |
| 12.6 | Oscillator period | $R_{OSC}$ = 120kΩ | | $t_{OSC}$ | 34.2 | 42.8 | 51.4 | µs | A |
| 13 | Watchdog Timing Relative to $t_{OSC}$ | | | | | | | | |
| 13.1 | Watchdog lead time after Reset | | | $t_d$ | | 7895 | | cycles | A |
| 13.2 | Watchdog closed window | | | $t_1$ | | 1053 | | cycles | A |
| 13.3 | Watchdog open window | | | $t_2$ | | 1105 | | cycles | A |
| 13.4 | Watchdog reset time NRES | | NRES | $t_{nres}$ | 3.2 | 4 | 4.8 | ms | A |
| 14 | VREG Voltage Regulator in Normal/Fail-safe and Silent Mode, VREG and PVREG Short-circuited | | | | | | | | |
| 14.1 | Output voltage VREG | 4V < $V_S$ < 18V<br>(0mA to 50mA) | VREG | $VREG_{nor}$ | 3.234 | | 3.366 | V | A |
| 14.2 | Output voltage VREG at low VS | 3V < $V_S$ < 4V | VREG | $VREG_{low}$ | $V_S$ – $V_D$ | | 3.366 | V | A |
| 14.3 | Regulator drop voltage | $V_S$ > 3V, $I_{VREG}$ = –15mA | VS, VREG | $V_D$ | | | 200 | mV | A |
| 14.4 | Regulator drop voltage | $V_S$ > 3V, $I_{VREG}$ = –50mA | VS, VREG | $V_D$ | | 500 | 700 | mV | A |
| 14.5 | Line regulation | 4V < $V_S$ < 18V | VREG | $VREG_{line}$ | | 0.1 | 0.2 | % | A |
| 14.6 | Load regulation | 5mA < $I_{VREG}$ < 50mA | VREG | $VREG_{load}$ | | 0.1 | 0.5 | % | A |
| 14.7 | Power supply ripple rejection | 10Hz to 100kHz<br>$C_{VREG}$ = 10µF<br>$V_S$ = 14V, $I_{VREG}$ = –15mA | VREG | | 50 | | | dB | D |
| 14.8 | Output current limitation | $V_S$ > 4V | VREG | $I_{VREGlim}$ | –240 | –160 | –85 | mA | A |
| 14.9 | Load capacity | 0.2Ω < ESR < 5Ω at 100kHz | VREG | $C_{load}$ | 1.8 | 10 | | µF | D |
| 14.10 | VREG undervoltage threshold | Referred to VREG<br>$V_S$ > 4V | VREG | $V_{thunN}$ | 2.8 | | 3.2 | V | A |
| 14.11 | Hysteresis of undervoltage threshold | Referred to VREG<br>$V_S$ > 4V | VREG | $Vhys_{thun}$ | | 150 | | mV | A |
| 14.12 | Ramp-up time $V_S$ > 4V to $V_{REG}$ = 3.3V | $C_{VREG}$ = 2.2µF<br>$I_{load}$ = –5mA at VREG | VREG | $T_{VREG}$ | | 320 | 500 | µs | A |
| 15 | DIV_ON Input Pin | | | | | | | | |
| 15.1 | Low-level voltage input | | DIV_ON | $V_{DIV_ON}$ | –0.3 | | +0.8 | V | A |
| 15.2 | High-level voltage input | | DIV_ON | $V_{DIV_ON}$ | 2 | | $V_{REG}$ + 0.3 | V | A |
| 15.3 | Pull-down resistor | $V_{DIV_ON}$ = $V_{REG}$ | DIV_ON | $R_{DIV_ON}$ | 125 | 250 | 400 | kΩ | A |
| 15.4 | Low-level input current | $V_{DIV_ON}$ = 0V | DIV_ON | $I_{DIV_ON}$ | –3 | | +3 | µA | A |

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

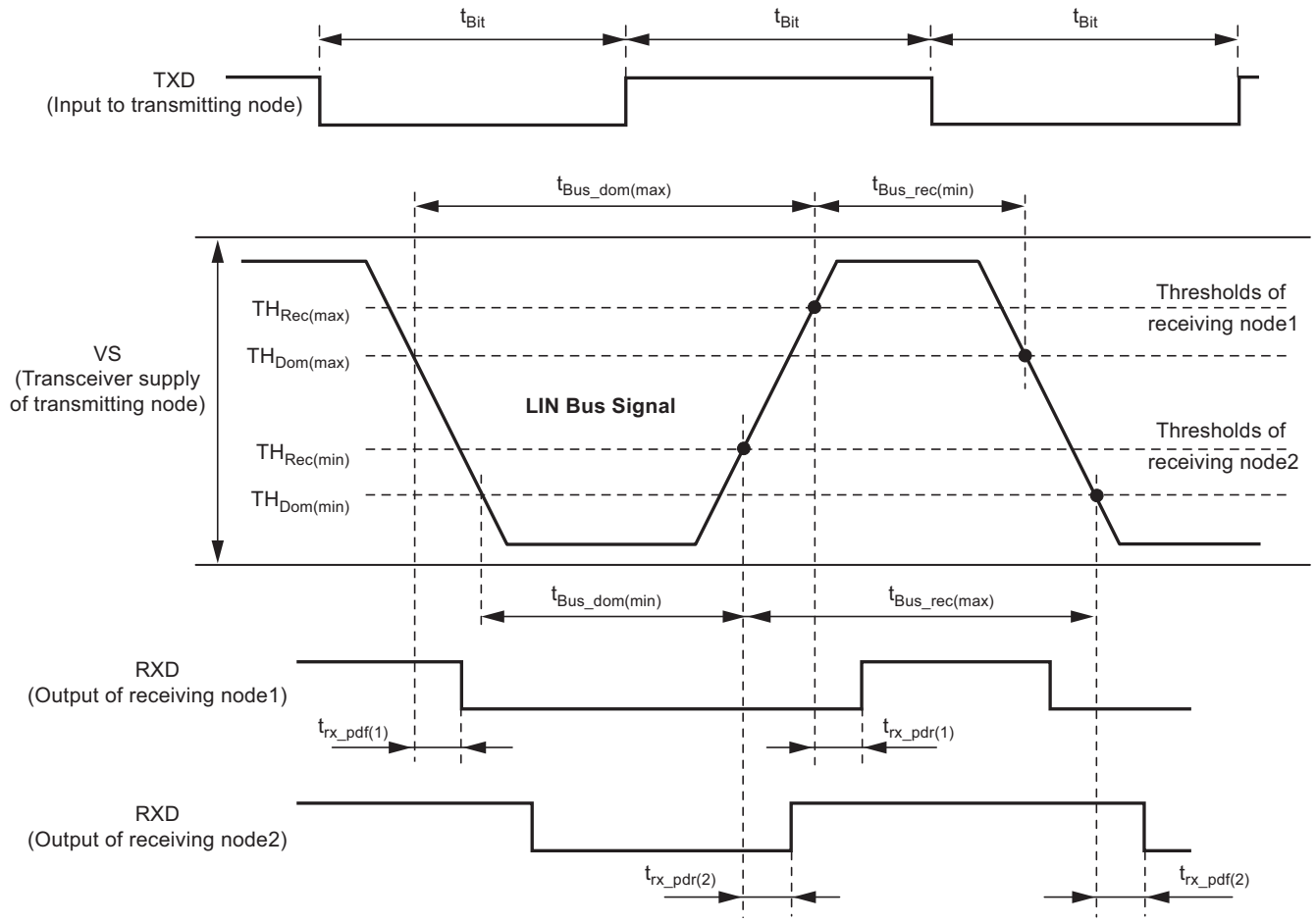
## 10. Electrical Characteristics LIN SBC (Continued)

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* |
|---|---|---|---|---|---|---|---|---|---|
| 16 | SP_MODE Input Pin | | | | | | | | |
| 16.1 | Low-level voltage input | | SP_MODE | $V_{SP_MODE}$ | –0.3 | | +0.8 | V | A |
| 16.2 | High-level voltage input | | SP_MODE | $V_{SP_MODE}$ | 2 | | $V_{REG}$ + 0.3 | V | A |
| 16.3 | Pull-down resistor | $V_{SP_MODE}$ = $V_{REG}$ | SP_MODE | $R_{SP_MODE}$ | 50 | 125 | 200 | kΩ | A |
| 16.4 | Low-level input current | $V_{SP_MODE}$ = 0V | SP_MODE | $I_{SP_MODE}$ | –3 | | +3 | µA | A |
| 17 | LIN Driver in High-speed Mode (VSP_Mode = VREG) | | | | | | | | |
| 17.1 | Transmission Baud rate | $V_S$ = 7V to 18V<br>$R_{LIN}$ = 500Ω, $C_{LIN}$ = 600pF | LIN | SP | 115 | | | kBaud | C |
| 17.2 | Slope time LIN falling edge | $V_S$ = 7V to 18V | LIN | $t_{SL_fall}$ | | 1 | 2 | µs | A |
| 17.3 | Slope time LIN rising edge, depending on RC-load | $V_S$ = 14V<br>$R_{LIN}$ = 500Ω, $C_{LIN}$ = 600pF | LIN | $t_{SL_rise}$ | | 2 | 3 | µs | A |
| 18 | Voltage Divider | | | | | | | | |
| 18.1 | Divider ratio | VBAT 5V to 40V | PV1 | | | 1:24 | | | D |
| 18.2 | Divider ratio error | VS = 5V to 27V | PV1 | | –2 | | +2 | % | A |
| 18.3 | Divider temperature drift | | PV1 | | | 2 | | ppm/°C | C |
| 18.4 | VBAT range of divider linearity | | VBAT | | 5 | | 27 | V | C |
| 18.5 | VBat input current | VBAT = 14V | VBAT | | 100 | | 220 | µA | A |
| 18.6 | Maximum output Voltage at PV1 | | PV1 | | 3 | 3.1 | 3.5 | V | A |
| 18.7 | Pin capacitance | | PV1 | | | 2 | | pF | D |

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

**Figure 10-1. Definition of Bus Timing Characteristics**



# Atmel AVR Microcontroller Unit (AVR MCU)

## 8-bit AVR Microcontroller with 32K/64K Bytes In-System Programmable Flash

**PRELIMINARY DATASHEET**

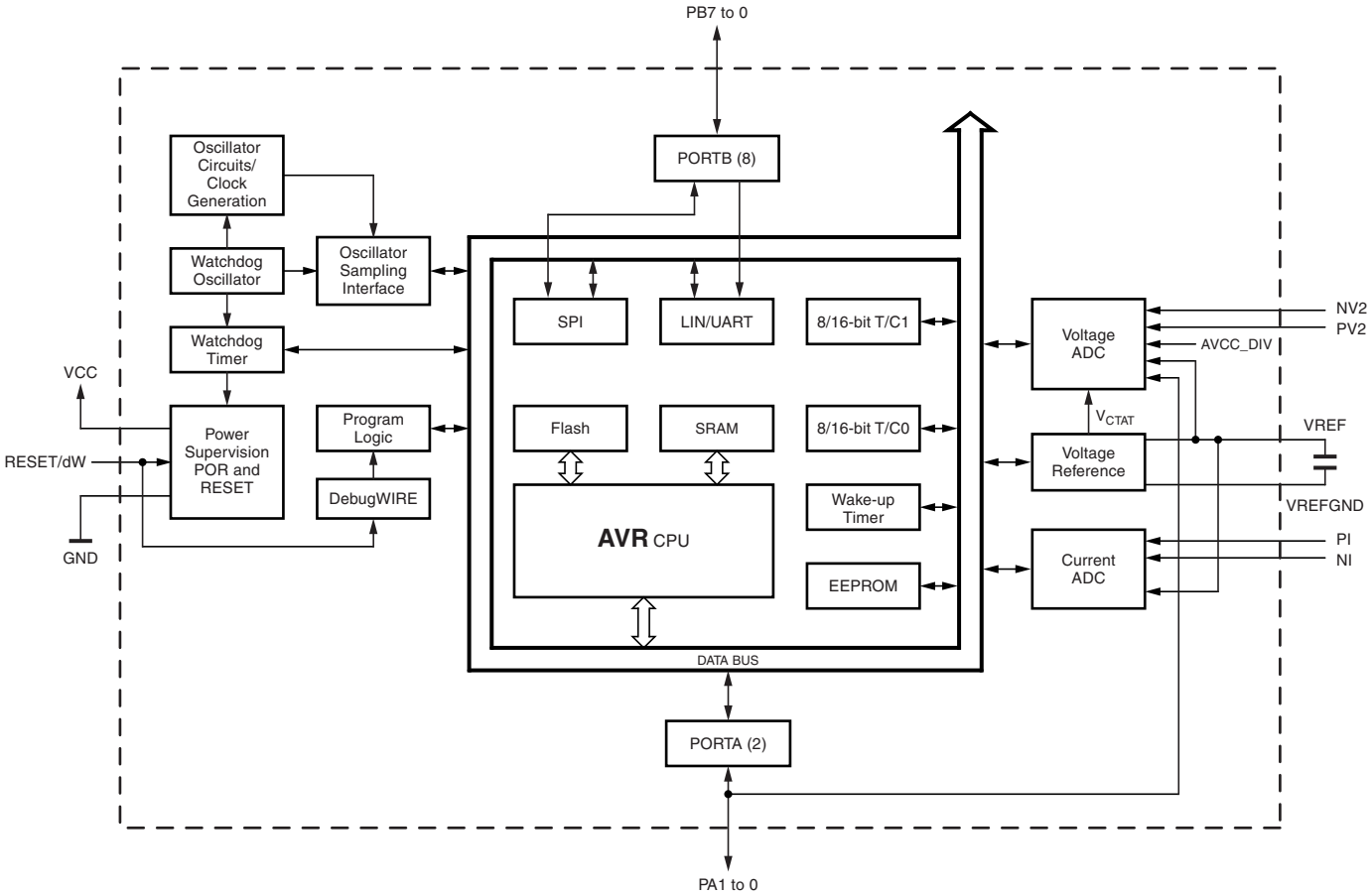
## Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 124 Powerful Instructions - Most Single Clock Cycle Execution
  - Additional Math Extension Instruction set
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 15MIPS Throughput at 15MHz
- High Endurance Non-volatile Memory Segments
  - 32K/64Kbytes of In-System Self-Programmable Flash Program Memory
  - 1Kbyte EEPROM
  - 4Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data Retention: 20 Years at 85°C/100 Years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two Configurable 8 or 16-bit Timers with Separate Prescaler, Optional Input Capture (IC), Compare Mode and CTC
  - LIN UART Serial Communication Interface with Flexible Baud-rate Generator
  - Master/Slave SPI Serial Interface
  - 17-bit 8kS/s Single-ended Voltage-ADC with 7 Selectable Input Channels and Diagnosis Modes
  - 18-bit 8kS/s Differential Current-ADC with Programmable Gain, Comparator Mode and Diagnosis Modes
  - Wake-up Timer
  - Programmable Watchdog Timer with Separate On-chip Oscillator
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Ports
  - Power-on Reset
  - External and Internal Interrupt Sources
  - Sleep Modes:
    - Idle, Power-save and Power-down

# 11. Overview

The Atmel® AVR MCU is a monitoring circuit, e.g., for sensor or car battery applications with focus on high accuracy and low cost. The device contains two high accuracy ADCs and a precision analog voltage and temperature reference, e.g., measurement of battery cell voltage, current and temperature. The device also contains accurate RC oscillators and a PLL, minimizing the external component count. The device contains a dedicated LIN/UART macro module, for data input/output using the LIN protocol. This interface also supports higher data rates than specified in the LIN specification.The device implements low power modes of operation, allowing continuous current monitoring with low current consumption. The feature set makes the Atmel AVR MCU highly suitable in, e.g., car battery monitoring systems focusing on high performance and low cost.

Figure 11-1. Block Diagram




The MCU provides the following features: 32K/64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 1K bytes EEPROM, 4Kbytes SRAM, 32 general purpose working registers, 10 general purpose I/O lines, two flexible Timer/Counters with Input Capture and compare modes, one programmable LIN/UART, two high accuracy Delta Sigma ADCs for voltage, current and temperature measurements, a programmable Watchdog Timer with internal Oscillator, debugWIRE for On-chip debugging, an SPI serial port also used for programming, internal and external interrupts and three software selectable power saving modes. The two Delta Sigma ADCs allow simultaneous measurement of battery voltage and current with very high accuracy, temperature measurements using the internal temperature reference, and continuous monitoring of the battery current with very low current consumption using the Power-save sleep mode.

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

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

The Atmel AVR MCU AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, and On-chip Debugger.

## 12. About Code Examples

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

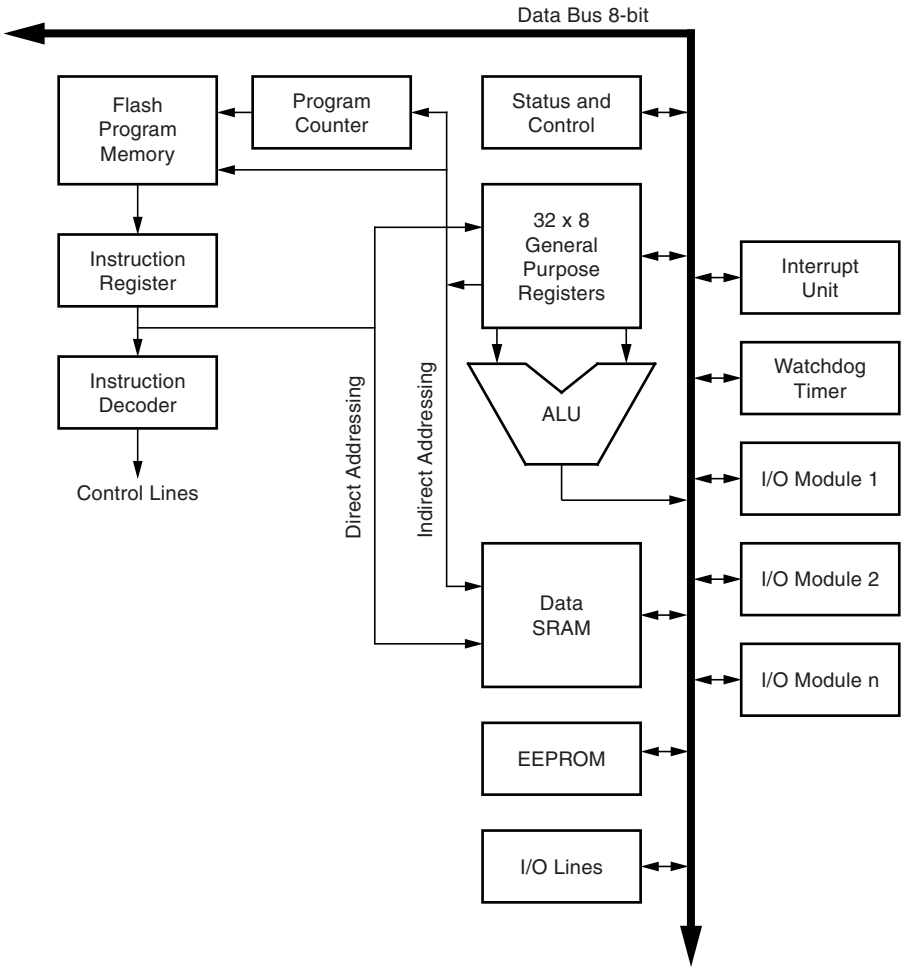
For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

# 13. AVR CPU Core

## 13.1 Overview

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.

Figure 13-1. Block Diagram of the AVR Architecture




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

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

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

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

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

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 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 Atmel® AVR MCU has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

## 13.2 ALU – Arithmetic Logic Unit

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

## 13.3 Status Register

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

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

### 13.3.1 SREG – AVR Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x3F (0x5F) | 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 for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

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

- **Bit 5 – H: Half Carry Flag**

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

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

  The S-bit is always an exclusive or between the negative flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.
- **Bit 3 – V: Two's Complement Overflow Flag**

  The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.
- **Bit 2 – N: Negative Flag**

  The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.
- **Bit 1 – Z: Zero Flag**

  The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.
- **Bit 0 – C: Carry Flag**

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

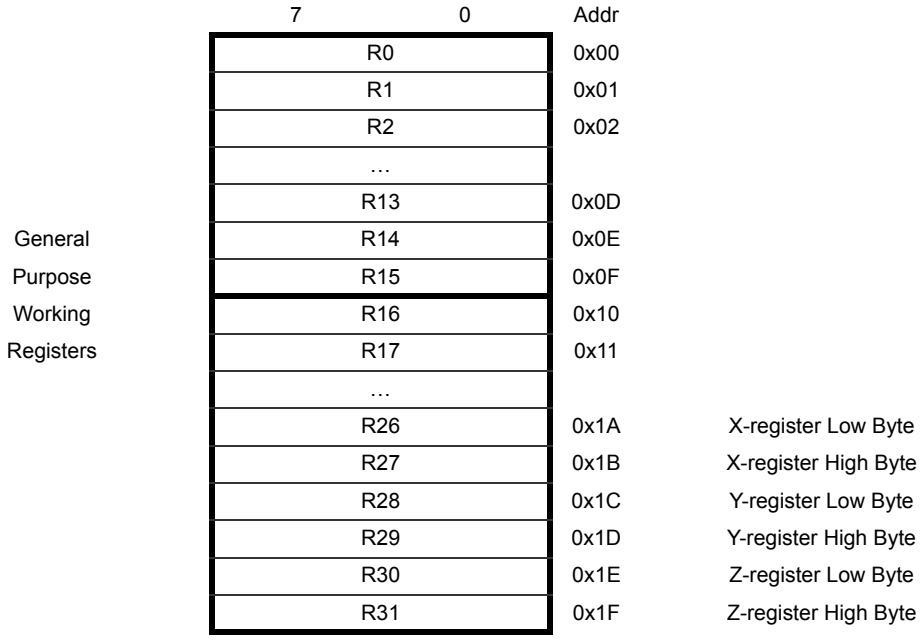
## 13.4 General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 13-2 shows the structure of the 32 general purpose working registers in the CPU.

**Figure 13-2. AVR** CPU General Purpose Working Registers

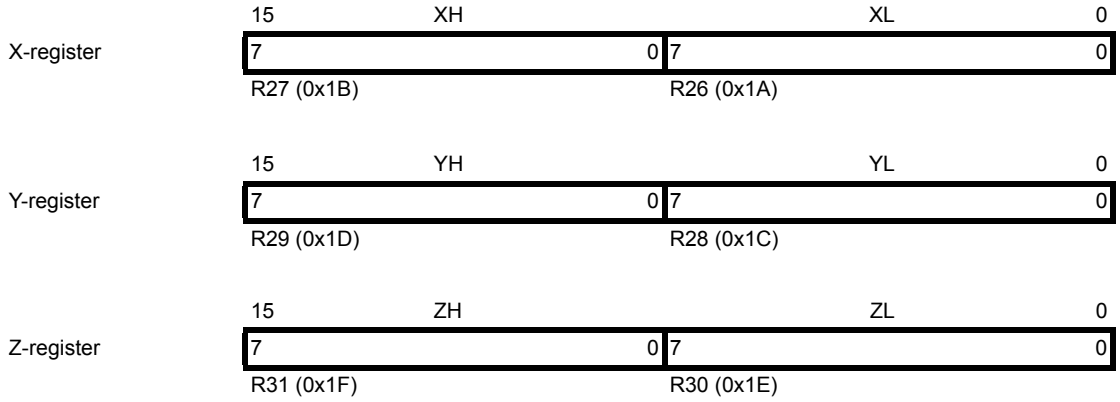



Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions. As shown in Figure 13-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although 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.

### 13.4.1 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 13-3 on page 38.

**Figure 13-3.** The X-, Y-, and Z-registers




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

## 13.5 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.

### 13.5.1 SPH and SPL – Stack Pointer High and Stack Pointer Low

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x3E (0x5E) | **SP15** | **SP14** | **SP13** | **SP12** | **SP11** | **SP10** | **SP9** | **SP8** | **SPH** |
| 0x3D (0x5D) | **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 | |

## 13.6 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 13-4 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 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 13-4. The Parallel Instruction Fetches and Instruction Executions

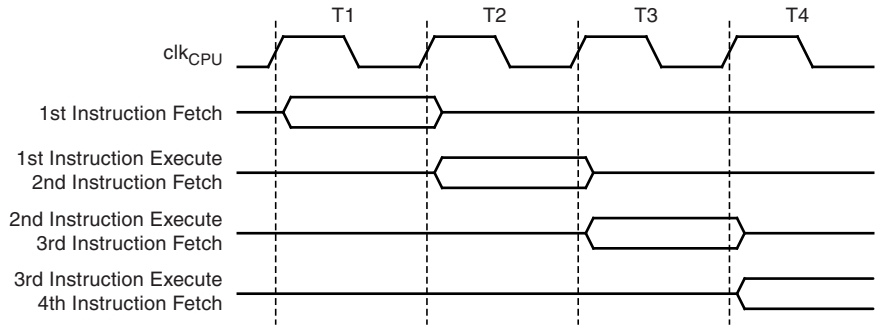
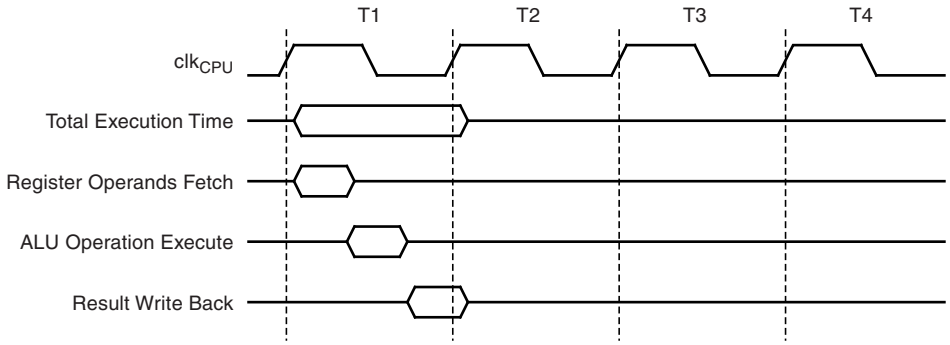



Figure 13-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 13-5. Single Cycle ALU Operation




## 13.7 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.

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 70. 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.

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, EEMPE     ; start EEPROM write
sbi     EECR, EEPE
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<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
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) */
```

### 13.7.1 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.

# 14. AVR Memories

## 14.1 Overview

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

## 14.2 In-System Reprogrammable Flash Program Memory

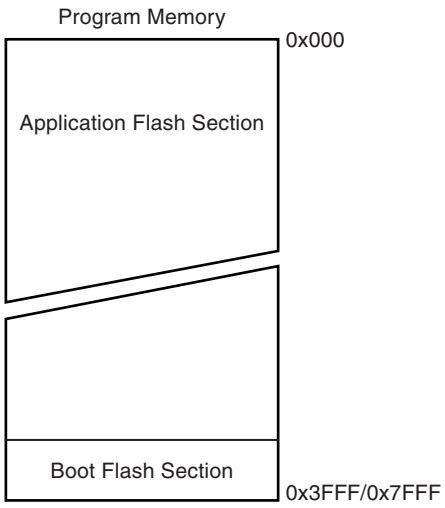
The Atmel AVR MCU contains 32K/64K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 16K x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The Atmel AVR MCU Program Counter (PC) is 14/15 bits wide, thus addressing the 16K/32K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in Section 29. “Boot Loader Support – Read-While-Write Self-Programming” on page 167. Section 30. “Memory Programming” on page 180 contains a detailed description on Flash programming.

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

Timing diagrams for instruction fetch and execution are presented in Section 13.6 “Instruction Execution Timing” on page 39.

Figure 14-1. Program Memory Map




## 14.3 SRAM Data Memory

Figure 14-2 on page 42 shows how the Atmel® AVR MCU SRAM Memory is organized.

The Atmel AVR MCU 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 4352 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 4K locations address the internal data SRAM.

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

The direct addressing reaches the entire data space.

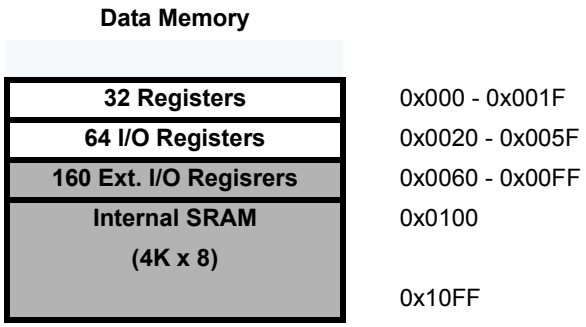
The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

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

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 4K bytes of internal data SRAM in the Atmel AVR MCU are all accessible through all these addressing modes. The Register File is described in Section 13.4 “General Purpose Register File” on page 37.

SRAM data will be unaffected by all other resets than Power-On reset. Note however that if a reset occurs while writing data types larger than 8-bit to the SRAM, the write might only be partially completed. This can leave, e.g., one byte updated in SRAM while the rest of the data word (int, long etc) was not written since the reset canceled the ongoing write operation.

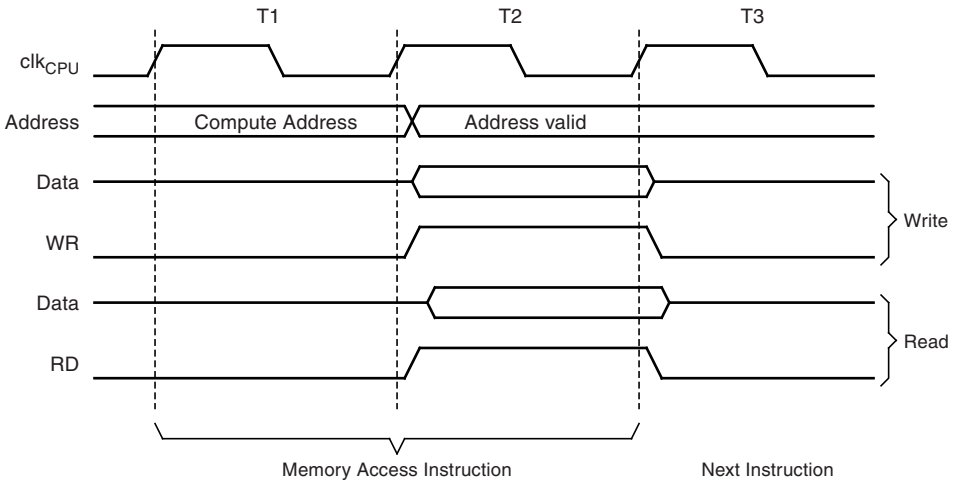
**Figure 14-2.** Data Memory Map




### 14.3.1 Data Memory 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 14-3.

Figure 14-3. On-chip Data SRAM Access Cycles



## 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.

## 14.6 Register Description

### 14.6.1 EEARH and EEARL– The EEPROM Address Register High and Low

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x22 (0x42) | | | | | | | **EEAR9** | **EEAR8** | EEARH |
| 0x21 (0x41) | **EEAR7** | **EEAR6** | **EEAR5** | **EEAR4** | **EEAR3** | **EEAR2** | **EEAR1** | **EEAR0** | EEARL |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | 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 | X | X | |
| | X | X | X | X | X | X | X | X | |

- **Bits 15:10 – Reserved**

  These bits are reserved bits in the Atmel AVR MCU and will always read as zero.

- **Bits 9:0 – EEAR9:0: EEPROM Address**

  The EEPROM Address Registers – EEAR specify the EEPROM address in the 1Kbytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 1023. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

### 14.6.2 EEDR – The EEPROM Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x20 (0x40) | **MSB** | | | | | | | **LSB** | **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.

### 14.6.3 EECR – The EEPROM Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1F (0x3F) | – | – | **EEPM1** | **EEPM0** | **EERIE** | **EEMPE** | **EEPE** | **EERE** | **EECR** |
| Read/Write | R | R | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | X | X | 0 | 0 | X | 0 | |

- **Bits 7:6 – Reserved**

  These bits are reserved bits in the Atmel® AVR MCU and will always read as zero.

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

  The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. 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 14-1. While EEPE 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 14-1. EEPROM Mode Bits**

| EEPM1 | EEPM0 | Typ. Programming Time(1) | Operation |
|---|---|---|---|
| 0 | 0 | 9ms | Erase and Write in one operation (Atomic Operation) |
| 0 | 1 | 4.5ms | Erase Only |
| 1 | 0 | 4.5ms | Write Only |
| 1 | 1 | – | Reserved for future use |

Note: 1. Actual timing depends on frequency of the PLL.

- **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 EEPE is cleared.

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

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

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

  The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is not essential):

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

**Caution:** An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.
When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

**Caution:** A BOD reset during EEPROM write will invalidate the result of the ongoing operation.

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

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

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

  The PLL is used to time the EEPROM accesses and the programming time will therefore depend on the PLL frequency. Table 14-2 lists the typical programming time for EEPROM access from the CPU.

**Table 14-2. EEPROM Programming Time**

| Symbol | Number of PLL Cycles | Typ Programming Time, $f_{PLL}$ = 14.3 MHz |
|---|---|---|
| EEPROM write (from CPU) | 27200 | 3.4ms |

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

Assembly Code Example

```
EEPROM_write:
        ; Wait for completion of previous write
        sbic            EECR,EEPE
        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 EEMPE
        sbi             EECR,EEMPE
        ; Start eeprom write by setting EEPE
        sbi             EECR,EEPE
        ret
```

C Code Example

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

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,EEPE
        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<<EEPE))
                ;
        /* Set up address register */
        EEAR = uiAddress;
        /* Start eeprom read by writing EERE */
        EECR |= (1<<EERE);
        /* Return data from data register */
        return EEDR;
}
```

### 14.6.4 GPIOR2 – General Purpose I/O Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x2B (0x4B) | MSB | | | | | | | LSB | GPIOR2 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

### 14.6.5 GPIOR1 – General Purpose I/O Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x2A (0x4A) | MSB | | | | | | | LSB | GPIOR1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

### 14.6.6 GPIOR0 – General Purpose I/O Register 0

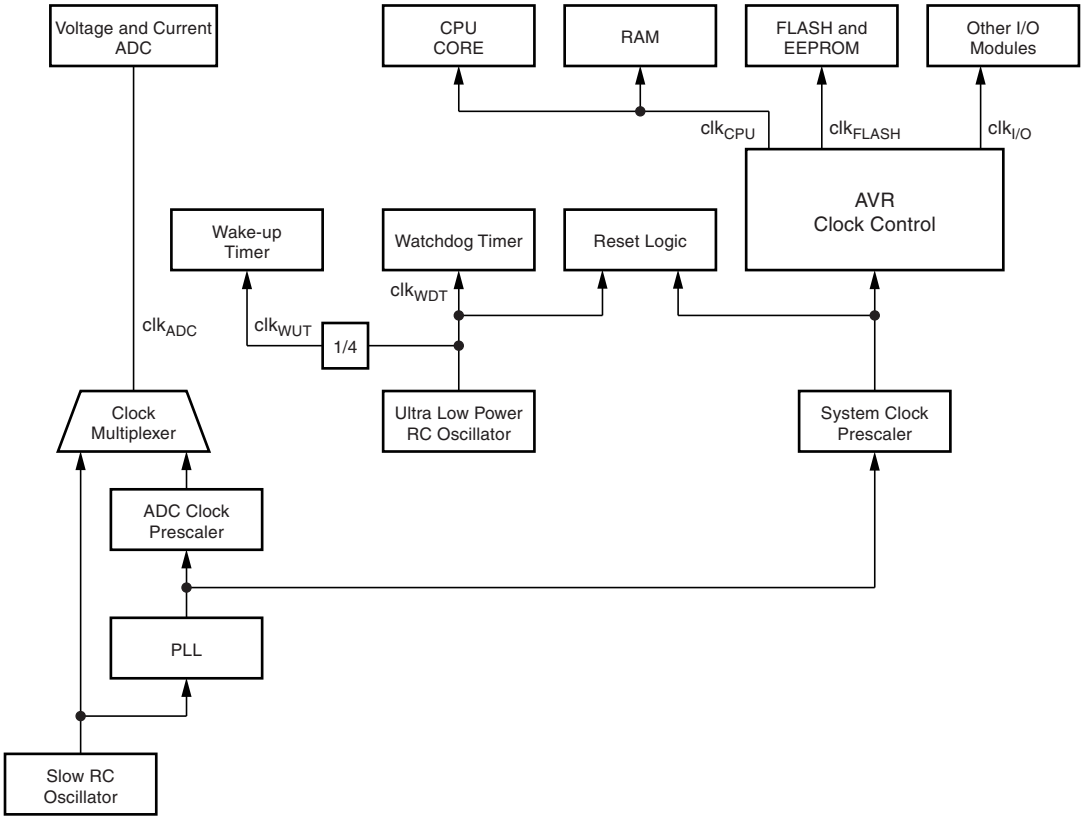
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1E (0x3E) | MSB | | | | | | | LSB | GPIOR0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

# 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.

### 15.1.5 Watchdog Timer and Wake-up Timer Clock – $clk_{WDT}$/$clk_{WUT}$

The Watchdog Timer and Wake-up Timer are provided with dedicated clock domains. This allows operation in all modes. It also allows very low power operation by utilizing an Ultra Low Power RC Oscillator dedicated to this purpose.

## 15.2 Clock Sources

The following section describes the clock sources available in the device. The clocks are input to the AVR clock generator, and routed to the appropriate modules.

The Atmel® AVR MCU has 3 on-chip clock sources used to clock the internal logic. Table 15-1 shows the clock sources and their usage.

**Table 15-1. Clock Sources**

| Clock Source | Usage |
|---|---|
| PLL | The clock source for the CPU, I/O and Flash. This clock divided by 28 is the default clock source for the ADCs. |
| Slow RC Oscillator | The clock source for the PLL, and optional clock source for the ADCs |
| Ultra Low Power RC Oscillator | The clock source for the Watchdog Timer and the Wake-up Timer. |

### 15.2.1 PLL

The PLL input clock is the Slow RC oscillator. The PLL has a fixed multiplication factor of 112, giving an output clock of 14.3 MHz (typical value). When enabled, the PLL will require a settling time before it has locked to the target frequency. During this time, the PLL output frequency will be unstable and inaccurate. Refer to Section 31. “Electrical Characteristics AVR MCU” on page 192ff for details on PLL and Slow RC frequency accuracy.

To allow the CPU to start up almost immediately when the PLL is enabled, the PLL clock will be divided by two to generate the CPU and I/O clock when the PLL is not in lock. When the PLL enters lock, the CPU and I/O clock will switch to the full PLL frequency. Operations that require high accuracy of the PLL clock should not be started until the PLL frequency has reached sufficient accuracy. The PLLCSR[LOCK] bit indicates if the PLL is in lock and could be used as a trigger for high accuracy operations. However, note that even if the PLL is locked, some additional time may be needed after start-up before the clock frequency has stabilized within the required tolerances. This depends on the application requirements.

The LOCK bit in the PLL Control and Status Register (PLLCSR) indicates the lock state of the PLL. A PLL Lock Change Interrupt can be enabled by the PLLCIE bit.

When the CPU wakes up from Power-save or Power-down, the CPU clock source is used to time the start-up, ensuring a valid clock before instruction execution starts. The CPU starts execution before the PLL clock is locked. When waking up from a sleep mode where the PLL is disabled, there is an additional delay of 2 Slow RC clock cycles (CKRC) before the PLL starts up. When the CPU starts from reset, there is an additional delay allowing the supply voltage to reach a stable level before commencing normal operation. The Ultra Low Power RC Oscillator is used for timing this real-time part of the start-up time. Start-up times are determined by the SUT Fuses as shown in Table 15-2.

**Table 15-2. Start-up times for the PLL**

| SUT[1:0] | Start-up times from Power-down and Power-save(1) | Addidtional delay from Reset (Typical values) |
|---|---|---|
| 00(3) | | |
| 01 | 2 CKRC + 6 CK | 14 CK + 16ms(2) |
| 10 | 2 CKRC + 6 CK | 14 CK + 32ms(2) |
| 11(4) | 2 CKRC + 6 CK | 14 CK + 64ms(2) |

Notes:
1. CKRC delay is only added if the PLL is disabled when the wake-up event occurs.
2. Actual value depends on the frequency of the ULP RC oscillator. The typical values of 16ms, 32ms and 64ms correspond to 2K, 4K and 8K cycles of the ULP RC oscillator, repectively.
3. This setting is reserved for test purpose and should not be used in applications.
4. Default setting

### 15.2.2 Slow RC Oscillator

The Slow RC Oscillator provides a 128kHz clock (typical value). The oscillator is factory calibrated, and will operate with no external components. During reset, hardware loads the calibration byte into the SOSCCAL Register and thereby automatically calibrates the Slow RC Oscillator. Refer to Section 15.6.1 “SOSCCALA – Slow RC Oscillator Calibration Register A” on page 51 for details. The Slow RC Oscillator features and automatic temperature compensation, thus removing the need for run-time calibration or frequency prediction. For details on Slow RC frequency drift and other characteristics, please refer to refer to Section 31. “Electrical Characteristics AVR MCU” on page 192ff.

Run-time calibration of the Slow RC Clock is possible, but this is not recommended due to relatively coarse step size compared to the temperature drift of the frequency.

### 15.2.3 Ultra Low Power RC Oscillator

The Ultra Low Power RC Oscillator (ULP Oscillator) provides a 128kHz clock (typical value, refer to Section 31. “Electrical Characteristics AVR MCU” on page 192ff).

## 15.3 Clock Output

The CPU clock divided by 2 can be output to the CLKO pin. The CPU can enable the clock output function by setting the CKOE bit in the MCU Control Register. The clock will not run in any sleep modes.

## 15.4 System Clock Prescaler

The Atmel® AVR MCU has a System Clock Prescaler, used to prescale the PLL clock. The system clock can be divided by setting the Section 15.6.5 “CLKPR – Clock Prescale Register” on page 52, and this enables the user to decrease or increase the system clock frequency as the requirement for power consumption and processing power changes. This system clock will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, $clk_{CPU}$ and $clk_{FLASH}$ are divided by a factor as shown in Table 15-3 on page 52.

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, may be faster than the CPU's clock frequency. It is not possible to determine the state of the prescaler, and the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

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

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

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

## 15.5 ADC Clock Prescaler

The Atmel AVR MCU has an ADC Clock Prescaler which is used to prescale the PLL clock with a fixed division factor of 28 when this clock is selected as the ADC clock source.

## 15.6 Register Description

### 15.6.1 SOSCCALA – Slow RC Oscillator Calibration Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x66) | SCALA7 | SCALA6 | SCALA5 | SCALA4 | SCALA3 | SCALA2 | SCALA1 | SCALA0 | SOSCCALA |
| 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 – SCALA7:0: Slow RC Oscillator Calibration Value A**

  The Slow RC Oscillator Calibration Register A is used to trim the Slow RC Oscillator. The factory-calibrated value is automatically written to this register during chip reset, and should not be changed by the SW. The Slow RC Oscillator Calibration Register A is protected by a timed sequence.

### 15.6.2 SOSCCALB – Slow RC Oscillator Calibration Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x67) | SCALB7 | SCALB6 | SCALB5 | SCALB4 | SCALB3 | SCALB2 | SCALB1 | SCALB0 | SOSCCALB |
| 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 – SCALB7:0: Slow RC Oscillator Calibration Value B**

  The Slow RC Oscillator Calibration Register B is used to trim the Slow RC Oscillator. The factory-calibrated value is automatically written to this register during chip reset, and should not be changed by the SW. The Slow RC Oscillator Calibration Register B is protected by a timed sequence.

### 15.6.3 PLLCSR – PLL Control and Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xD8) | – | – | SWEN | LOCK | – | – | PLLCIF | PLLCIE | PLLCSR |
| Read/Write | R | R | R/W | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:6 – Reserved**

  These bits are reserved and will always read as zero.
- **Bit 5 – SWEN: PLL Software Enable**

  This bit overrides the normal PLL enabling and disabling in Power Save and Power Down sleep modes. This bit is implemented for test purpose and should never be set by the application. When this bit is cleared, the PLL operates as described in Table 16-2 on page 54.
- **Bit 4 – LOCK: PLL Lock**

  If this bit is logic high, the PLL is in lock and the PLL output is used as the system clock. Otherwise, the PLL output is divided by a factor of two to generate the system clock.
- **Bits 3:2 – Reserved**

  These bits are reserved and will always read as zero.
- **Bit 1 – PLLCIF: PLL Lock Change Interrupt Flag**

  This flag is set if an edge on the lock signal is detected. The flag is cleared either by writing a logic one to the bit or by executing the corresponding interrupt routine.
- **Bit 0 – LOCK: PLL Lock Change Interrupt Enable**

  Interrupt enable for the Lock Change Interrupt Flag.

### 15.6.4 MCUCR – MCU Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x35 (0x55) | – | – | CKOE | PUD | – | – | IVSEL | IVCE | MCUCR |
| Read/Write | R | R | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 5 – CKOE: Clock Output**

  When this bit is written to one, the CPU clock divided by 2 is output on the CLKO pin.

### 15.6.5 CLKPR – Clock Prescale Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x61) | CLKPCE | – | – | – | – | – | CLKPS1 | CLKPS0 | CLKPR |
| Read/Write | R/W | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | X(1) | X(1) | |

Note: 1. See CLKPS[1:0 ] 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 CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, or clear the CLKPCE bit.

- **Bits 6:2 – Reserved**

  These bits are reserved and will always read as zero.

- **Bit 1:0 – CLKPS[1:0]: Clock Prescaler Select**

  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 15-3.

  The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “00”. If CKDIV8 is programmed, CLKPS bits are reset to “11”, giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the 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 15-3. System Clock Prescaler Select**

| CLKPS1 | CLKPS0 | Clock Division Factor |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 2 |
| 1 | 0 | 4 |
| 1 | 1 | 8 |

# 16. 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.

## 16.1 Sleep Modes

Figure 15-1 on page 48 presents the different clock systems in the Atmel® AVR MCU, and their distribution. The figure is helpful in selecting an appropriate sleep mode. The different sleep modes and their wake up sources is summarized in Table 16-1, and Figure 16-1 on page 53 shows a sleep mode state diagram.

**Table 16-1. Wake-up Sources for Sleep Modes**

| Mode | Wake-up Sources | | | | | | |
|---|---|---|---|---|---|---|---|
| | Wake-up on Regular Current | WUT | WDT | SPM/EEPROM Ready | C-ADC | V-ADC | Other I/O |
| Idle | X | X | X | X | X | X | X |
| Power-save | X | X | X | | X | X | |
| Power-down | | X | X | | | | |

To enter any of the sleep modes, the SE bit in SMCR, see Section 16.7.1 "SMCR – Sleep Mode Control Register" on page 56, must be written to logic one and a SLEEP instruction must be executed. The SM2..0 bits in the SMCR Register select which sleep mode will be activated by the SLEEP instruction. See Table 16-3 on page 56 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 any sleep mode except Power-off. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

**Figure 16-1. Sleep Mode State Diagram**

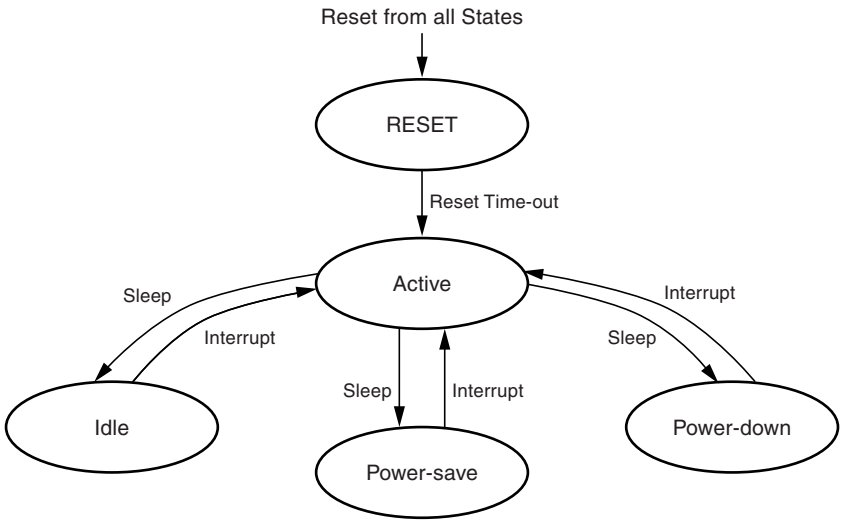


**Table 16-2.** Active Modules in Different Sleep Modes

| Module | Mode | | | |
|---|---|---|---|---|
| | Active | Idle | Power-save | Power-down |
| PLL | X | X | X(1)(2) | |
| RCOSC_ULP | X | X | X | X |
| RCOSC_SLOW | X | X | X | |
| CPU | X | | | |
| Flash | X | | | |
| 8/16-bit Timer | X | X | | |
| LIN UART | X | X | | |
| SPI | X | X | | |
| V-ADC | X | X | X | |
| C-ADC | X | X | X | |
| External Interrupts | X | X | X | X |
| WUT | X | X | X | X |
| WDT | X | X | X | X |

Notes: 1. Depending on ADC clock source setting, refer to Section 26. “ADC - Analog to Digital Converter” on page 138 for details.

2. If PLLCSR[SWEN] bit is set, PLL and SlowRC oscillator will always run in Power-save mode.

## 16.2 Idle Mode

When the SM1..0 bits are written to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing all peripheral functions to continue operating. This sleep mode basically halts $clk_{CPU}$ 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 interrupt.

## 16.3 Power-save Mode

When the SM1..0 bits are written to 11, the SLEEP instruction makes the MCU enter Power-save mode. In this mode, the Voltage ADC, Current ADC, Wake-up Timer (WUT) and Watchdog Timer (WDT) continue operating if enabled.

This mode will be the default mode when application software does not require operation of CPU, Flash or any of the peripheral units running at the PLL clock. If the ADCs are configured to operate on the 512kHz clock or if the PLLSCR[SWEN] bit is set, the PLL will be operating. Refer to Section 26. “ADC - Analog to Digital Converter” on page 138 for details on the ADC clock selection.

If the current through the sense resistor is so small that the Current ADC cannot measure it accurately, Regular Current detection should be enabled to reduce power consumption. The WUT keeps accurately track of the time so that battery self discharge can be calculated.

When waking up from Power-save 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 in Section 15.2 “Clock Sources” on page 49.

## 16.4 Power-down Mode

When the SM1..0 bits are written to 10, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the PLL and Slow RC Oscillator (RCOSC_SLOW) are normally stopped, while the Wake-up Timer (WUT) and Watchdog Timer (WDT) continue operating if enabled. If the PLLCSR[SWEN] bit is set, both SlowRC oscillator and PLL will keep running in this mode.

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 in Section 15.2 “Clock Sources” on page 49

## 16.5 Power Reduction Register

The Power Reduction Register (PRR), see Section 16.7.2 “PRR0 – Power Reduction Register 0” on page 57, 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.

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.

## 16.6 Minimizing Power Consumption

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

### 16.6.1 Wake-up Timer

If the Wake-up Timer is not needed in the application, the module should be turned off. If the Wake-up 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 Section 18. “Wake-up Timer” on page 68 for details on how to configure the Wake-up Timer.

### 16.6.2 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 Section 17.3 “Watchdog Timer” on page 62 for details on how to configure the Watchdog Timer.

### 16.6.3 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 Section 21.2.5 “Digital Input Enable and Sleep Modes” on page 82 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 Register.

### 16.6.4 On-chip Debug System

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.

### 16.6.5 Voltage ADC

If enabled, the V-ADC will consume power independent of sleep mode. To save power, the V-ADC should be disabled by clearing the ADCRE[VADEN] bit when not used. When disabling the VADC or making other VADC configuration changes, make sure that the disable command has been synchronized to the ADC clock domain before entering sleep mode. ADC synchronization is explained in Section 26.4.1 “Synchronization of Configuration Settings” on page 146. See Section 26. “ADC - Analog to Digital Converter” on page 138 for details on V-ADC operation.

### 16.6.6 Current ADC

If enabled, the C-ADC will consume power independent of sleep mode. To save power, the C-ADC should be disabled by clearing the ADCRC[CADEN] bit when not used, or set in Regular Current detection mode. See Section 26. “ADC - Analog to Digital Converter” on page 138 for details on C-ADC operation. When disabling the CADC or making other CADC configuration changes, make sure that the disable command has been synchronized to the ADC clock domain before entering sleep mode. ADC synchronization is explained in Section 26.4.1 “Synchronization of Configuration Settings” on page 146.

### 16.6.7 PLL

If the ADCs are configured to operate on the 512kHz clock, the PLL will be operating in Power-save mode. To minimize power consumption in the Power-save mode, it is therefore recommended to configure the ADCs to operate on the 128kHz clock, allowing the PLL to be disabled. Also note that the PLL will be enabled in all sleep modes if the PLLCSR[SWEN] bit is set. For low power operation it is therefore recommended to clear this bit before entering sleep mode.. Refer to Section 26. “ADC - Analog to Digital Converter” on page 138 for details.

### 16.6.8 Bandgap Voltage Reference

The Bandgap reference will consume power independent of sleep mode. To save power in the Power-down sleep mode, the Bandgap reference can be configured in a special sample mode where the $V_{REF}$ voltage is refreshed occasionally. See Section 27. “Band Gap Reference and Temperature Sensor” on page 161 for details. It is not possible for software to completely disable the bandgap during normal operation.

## 16.7 Register Description

### 16.7.1 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 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x33 (0x53) | – | – | – | – | – | SM1 | SM0 | SE | SMCR |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:3 – Reserved**

  These bits are reserved bits in the Atmel® AVR MCU, and will always read as zero.

- **Bits 2:1 – SM1:0: Sleep Mode Select Bits 1:0**

  These bits select between the available sleep modes as shown in Table 16-3.

**Table 16-3.** Sleep Mode Select

| SM1 | SM0 | Sleep Mode |
|---|---|---|
| 0 | 0 | Idle |
| 0 | 1 | Reserved |
| 1 | 0 | Power-down |
| 1 | 1 | Power-save |

- **Bit 0 – 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.

### 16.7.2 PRR0 – Power Reduction Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x64) | – | – | – | – | PRLIN | PRSPI | PRTIM1 | PRTIM0 | PRR0 |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:4 – Reserved**

  These bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when PRR0 is written.

- **Bit 3 – PRLIN: Power Reduction LIN UART Interface**

  Writing logic one to this bit shuts down the LIN UART Interface by stopping the clock to the module. When waking up the LIN UART again, the LIN UART should be reinitialized to ensure proper operation.

- **Bit 4 – PRSPI: Power Reduction Serial Peripheral Interface**

  Writing logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be reinitialized to ensure proper operation.

- **Bit 1 – PRTIM1: Power Reduction Timer/Counter1**

  Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 0 – PRTIM0: Power Reduction Timer/Counter0**

  Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

# 17. System Control and Reset

## 17.1 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. The circuit diagram in Figure 17-1 on page 59 shows the reset logic. Table 19-1 on page 70 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.

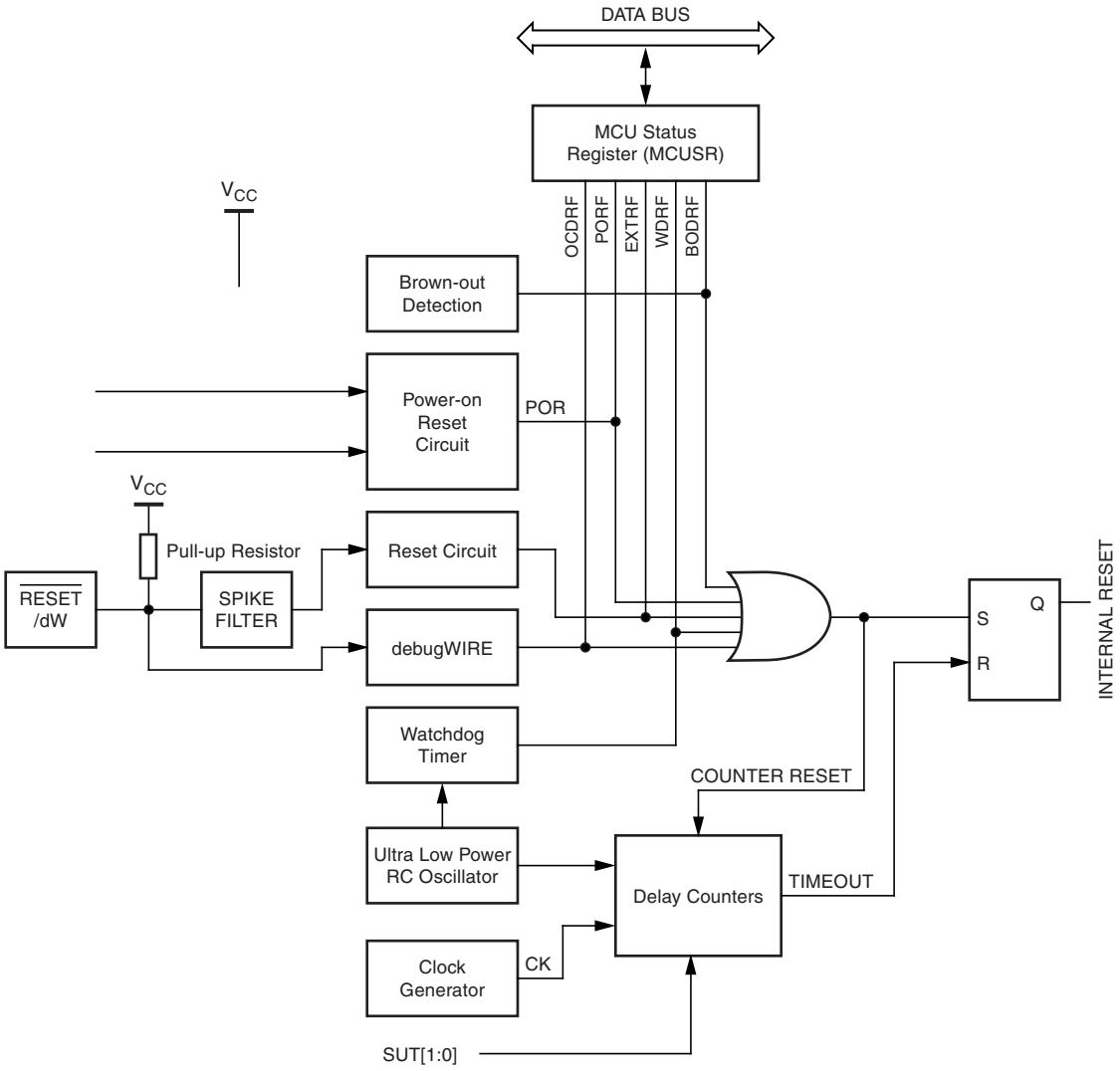
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the voltage regulator to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT Fuses. The different selections for the delay period are presented in Section 15.2 “Clock Sources” on page 49.

## 17.2 Reset Sources

The Atmel® AVR MCU has five sources of reset:

- The Power-on Reset module generates a Power-on Reset when the Voltage Regulator starts up.
- External Reset. The MCU is reset when a low level is present on the RESET pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when $V_{REG}$ is below the Brown-out Reset Threshold, $V_{BOT}$. See Section 17.2.4 “Brown-out Detection” on page 61
- debugWIRE Reset. In On-chip Debug mode, the debugWIRE resets the MCU when giving the Reset command.

**Figure 17-1. Reset Logic**



### 17.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Section 31. “Electrical Characteristics AVR MCU” on page 192ff. 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 17-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to $V_{CC}$**

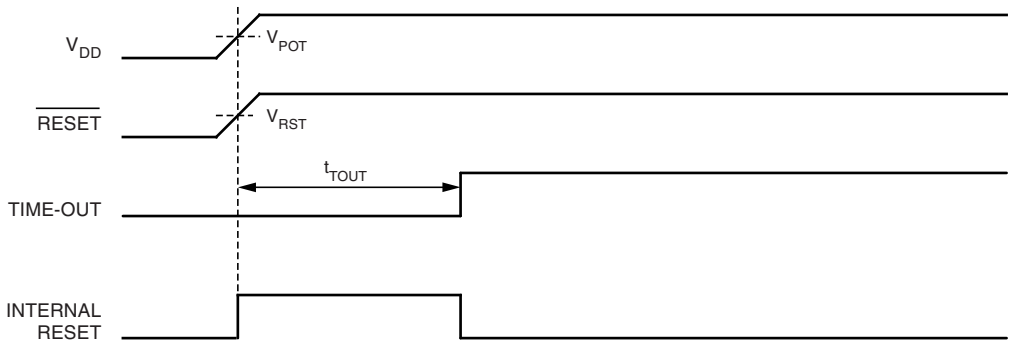
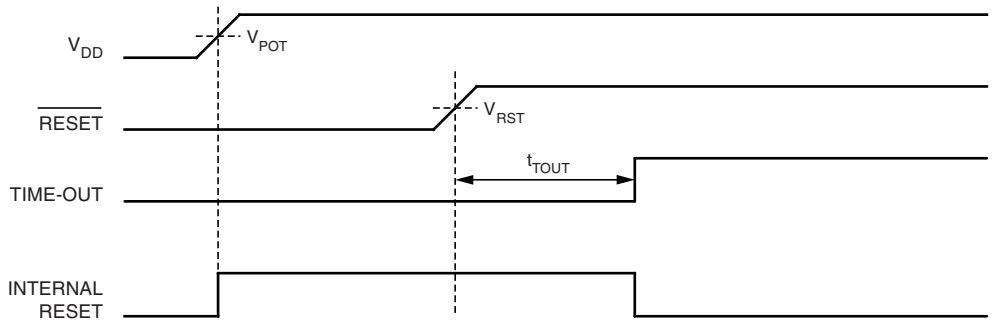


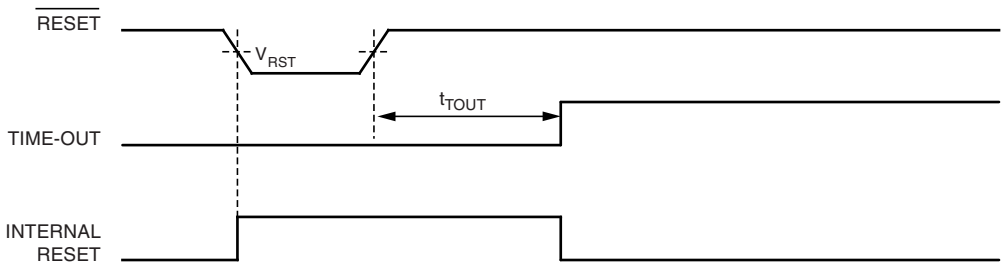

**Figure 17-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally**




### 17.2.2 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 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.

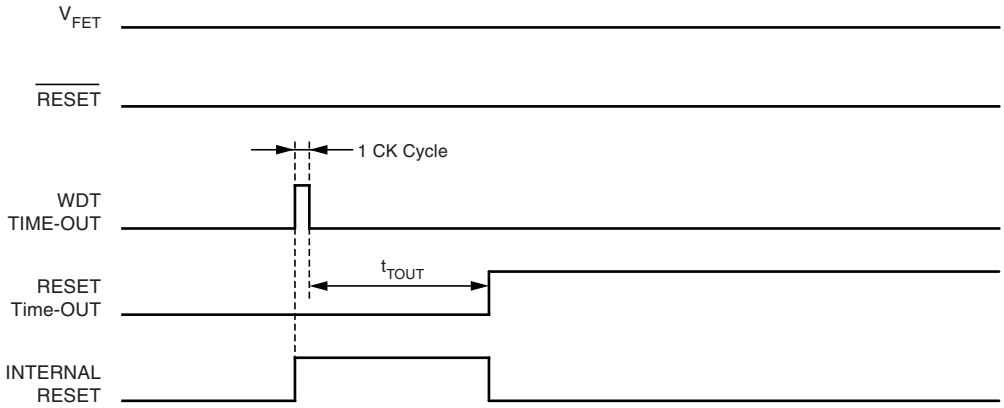
**Figure 17-4. External Reset During Operation**



### 17.2.3 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 62 for details on operation of the Watchdog Timer.

**Figure 17-5. Watchdog Reset During Operation**



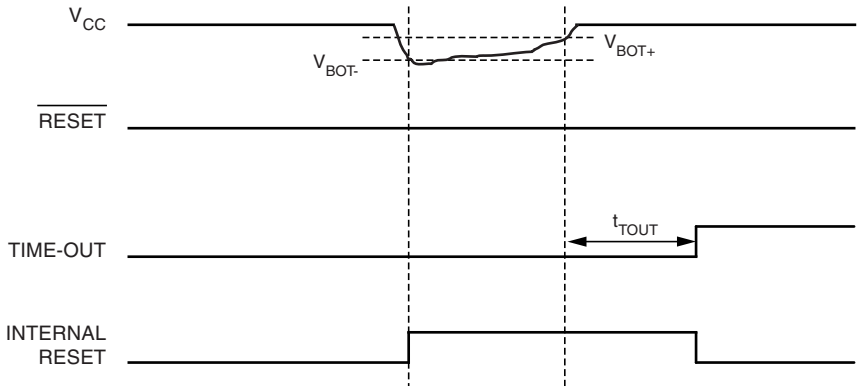

### 17.2.4 Brown-out Detection

The Atmel® AVR MCU has an On-chip Brown-out Detection (BOD) circuit for monitoring the $V_{CC}$ level during operation by comparing it to a trigger level
$V_{BOT} = V_{BOTIDEAL} \times V_{REF}/1.1$. During start-up the VREF value will change, see Section 27. “Band Gap Reference and Temperature Sensor” on page 161. 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 $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD is enabled by setting the BODEN fuse in low fuse byte, see Section 30.2 “Fuse Bits” on page 181. When the fuse is programmed the BOD will be enabled in all modes of operation, except in Power-off mode. For applications that do not have an external VCC monitor to generate a reset in case of low VCC, it is recommended to always enable the BOD in order to guarantee safe operating conditions for the device.

When the BOD is enabled, and $V_{CC}$ decreases to a value below the trigger level ($V_{BOT-}$ in Figure 17-6), the Brown-out Reset is immediately activated. When $V_{CC}$ increases above the trigger level ($V_{BOT+}$ in Figure 17-6), the delay counter starts the MCU after the Time-out period $t_{TOUT}$ has expired.

**Figure 17-6. Brown-out Reset During Operation**



## 17.3 Watchdog Timer

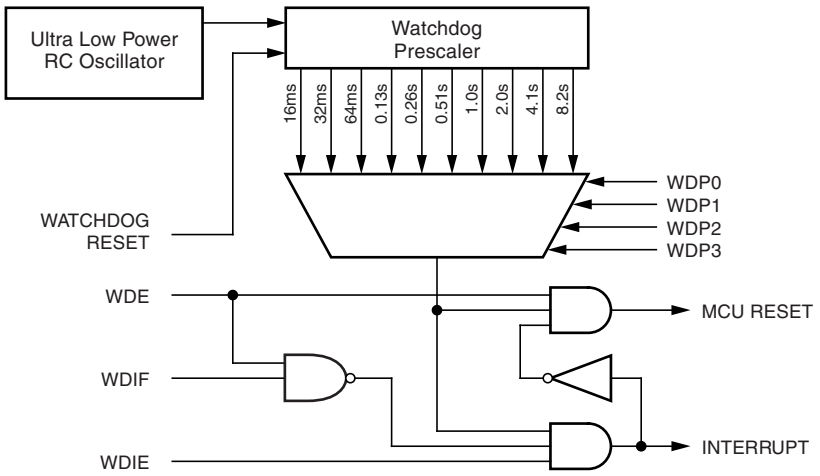
### 17.3.1 Features

- Clocked from separate on-chip oscillator
- 3 operating modes
  - Interrupt
  - System reset
  - Interrupt and system reset
- Selectable time-out period from 16ms to 8s
- Optional locking of watchdog configuration after initial configuration
- Programmable hardware fuse watchdog always on (WDTON) for Fail-safe Mode

### 17.3.2 Overview

The Atmel® AVR MCU has an Enhanced Watchdog Timer (WDT). The WDT counts cycles of the Ultra Low Power RC 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.

**Figure 17-7. Watchdog Timer**




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 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.

As a safe-guard against software run-away, changes to the Watchdog configuration can only be performed with a timed sequence. The sequence for setting or clearing WDE and/or 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.

Assembly Code Example[1]

```
WDT_off:
        ; Turn off global interrupt
        in    r17, SREG ; store SREG value
        cli ; disable interrupts during timed sequence
        ; 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
        in    r16, WDTCSR
        ori   r16, (1<<WDCE) | (1<<WDE)
        out   WDTCSR, r16
        ; Turn off WDT
        ldi   r16, (0<<WDE)
        out   WDTCSR, r16
        ; Restore global interrupt enable setting
        out   SREG, r17 ; restore SREG value (I-bit)
        ret
```

C Code Example[1]

```
void WDT_off(void)
{
        char SREG;
        cSREG = SREG; /* store SREG value */
        __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;
        SREG = cSREG; /* restore SREG value (I-bit) */
}
```

Notes:
1. See Section 12. “About Code Examples” on page 34
2. 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 initialization 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
        in    r17, SREG ; store SREG value
        cli ; disable interrupts during timed sequence
        ; Reset Watchdog Timer
        wdr
        ; Start timed sequence
        in    r16, WDTCSR
        ori   r16, (1<<WDCE) | (1<<WDE)
        out   WDTCSR, r16
        ; --  Got four cycles to set the new values from here -
        ; Set new prescaler(time-out) value = 64Kcycles (~0.5 s)
        ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
        out   WDTCSR, r16
        ; --  Finished setting new values, used 2 cycles -
        ; Restore global interrupt enable setting
        out   SREG, r17 ; restore SREG value (I-bit)
        ret
```

C Code Example[1]

```
void WDT_Prescaler_Change(void)
{
        char SREG;
        cSREG = SREG; /* store SREG value */
        __disable_interrupt();
        __watchdog_reset();
        /* Start timed sequence */
        WDTCSR |= (1<<WDCE) | (1<<WDE);
        /* Set new prescaler(time-out) value = 64Kcycles (~0.5 s) */
        WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
        SREG = cSREG; /* restore SREG value (I-bit) */
}
```

Notes:
1. See Section 12. “About Code Examples” on page 34
2. 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.

As a further safe-guard against software run-away, software has the option to lock the Watchdog configuration from further modification after the initial configuration. The Watchdog configuration will then be locked until the next system reset. To lock the Watchdog configuration, the following algorithm must be followed:

1. In the same operation, write a logic one to WDCLE and WDCL.
2. Within the next four clock cycles, in the same operation, write a logic zero to WDCLE and a logic one to WDCL.

## 17.4 Register Description

### 17.4.1 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 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x34 (0x54) | – | – | – | OCDRF | WDRF | BODRF | EXTRF | PORF | MCUSR |
| Read/Write | R | R | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | See Bit Description | | | | | |

- **Bits 7:5 – Reserved**

  These bits are reserved bits in the Atmel® AVR MCU, and will always read as zero.

- **Bit 4 – OCDRF: OCD Reset Flag**

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

- **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 – BODRF: Brown-out Reset Flag**

  This bit is set if a Brown-out Reset occurs. This 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 the MCUSR as early as possible in the program, and perform the required initialization accordingly. The MCUSR should be cleared once the initialization is completed. If another reset occurs before MCUSR has been cleared, the value of MCUSR after the second reset will show the sources of both the first and second reset.

### 17.4.2 WDTCSR – Watchdog Timer Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x60) | 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 and WDIE are 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 17-1.** Watchdog Timer Configuration

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

Note: 1. WDTON Fuse set to “0” means programmed, “1” means unprogrammed.

- **Bit 5 – WDP3 : Watchdog Timer Prescaler 3**

  The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17-2.
- **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.
- **Bits 2:0 – WDP 2:0: Watchdog Timer Prescaler 2, 1, and 0**

  The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17-2 on page 66.

**Table 17-2.** Watchdog Timer Prescale Select

| WDP3 | WDP2 | WDP1 | WDP0 | Number of WDT Oscillator Cycles | Typical Time-out[1] |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2Kcycles | 16ms |
| 0 | 0 | 0 | 1 | 4Kcycles | 32ms |
| 0 | 0 | 1 | 0 | 8Kcycles | 64ms |
| 0 | 0 | 1 | 1 | 16Kcycles | 0.13s |
| 0 | 1 | 0 | 0 | 32Kcycles | 0.26s |
| 0 | 1 | 0 | 1 | 64Kcycles | 0.51s |
| 0 | 1 | 1 | 0 | 128Kcycles | 1.0s |
| 0 | 1 | 1 | 1 | 256Kcycles | 2.0s |
| 1 | 0 | 0 | 0 | 512Kcycles | 4.1s |
| 1 | 0 | 0 | 1 | 1024Kcycles | 8.2s |
| 1 | 0 | 1 | 0 | Reserved | |
| 1 | 0 | 1 | 1 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | |

Note: 1. The actual timeout value depends on the actual clock period of the Ultra Low Power RC Oscillator, refer to Section 15.2.3 “Ultra Low Power RC Oscillator” on page 50 for details.

### 17.4.3 WDCLR - Watchdog Timer Configuration Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | **WDCL1** | **WDCL0** | **WDCLE** | **WDCLR** |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:3 – Reserved**

  These bits are reserved and will always read as zero.

- **Bits 2:1 – WDCL[1:0]: Watchdog Timer Configuration Lock 1:0**

  The WDTCSR[6:0] register can be locked from any further software updates after initial configuration. Once locked, these bits cannot be accessed until the next hardware reset. This provides a safe method for protecting the register from unintentional modification by software runaway. It is recommended that software configures this register shortly after reset, and then protects the register from further updates. There are two levels for register locking to support all operational modes of the Watchdog timer.

  If using the watchdog timer in Interrupt mode where WDIE must be re-enabled after each timeout, only WDTCSR[5:0] should be locked. When using the Watchdog in pure System reset mode, WDTCSR[6:0] should be locked to prevent the watchdog from switching from System reset mode to Interrupt mode.

  To lock WDTCSR[6:0], the following algorithm must be followed:

  1. In the same operation, write a logic one to WDCLE and WDCL1:0.
  2. Within the next four clock cycles, in the same operation, write a logic zero to WDCLE and a logic one to WDCL1.

  To lock WDTCSR[5:0], the following algorithm must be followed:

  1. In the same operation, write a logic one to WDCLE and WDCL1:0.
  2. Within the next four clock cycles, in the same operation, write a logic zero to WDCLE and WDCL1, and a logic one to WDCL0.

- **Bit 0 – WDCLE: Watchdog Timer Configuration Lock**

  See “Bits 2:1 – WDCL[1:0]: Watchdog Timer Configuration Lock 1:0” on page 67 for description.

# 18. Wake-up Timer

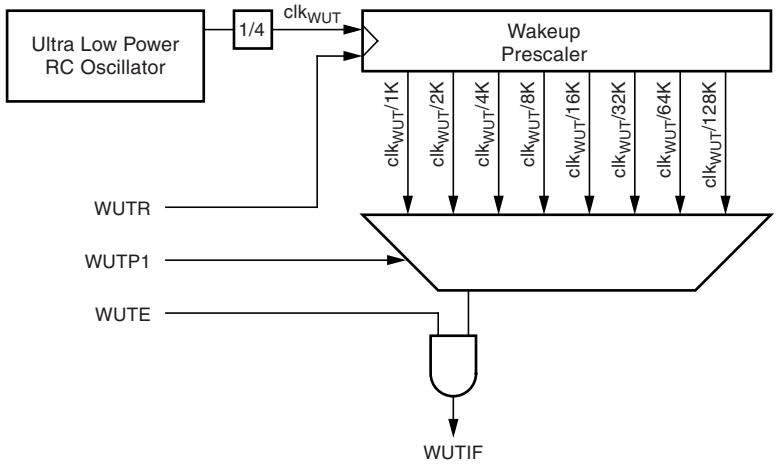
The following section describes the Wake-up Timer in the Atmel® AVR MCU.

- One wake-up timer interrupt
- 8 selectable time-out periods
- Separate clock source

## 18.1 Overview

The Wake-up Timer is clocked from the Ultra Low Power RC Oscillator. By controlling the Wake-up Timer prescaler, the Wake-up interval can be adjusted from 32 ms to 1 s.

**Figure 18-1. Wake-up Timer**




## 18.2 Register Description

### 18.2.1 WUTCSR – Wake-up Timer Control and Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x62) | WUTIF | WUTIE | – | WUTR | WUTE | WUTP2 | WUTP1 | WUTP0 | WUTCSR |
| Read/Write | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – WUTIF: Wake-up Timer Interrupt Flag**

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

- **Bit 6 – WUTIE: Wake-up Timer Interrupt Enable**

  When the WUTIE bit and the I-bit in the Status Register are set (one), the Wake-up Timer interrupt is enabled. The corresponding interrupt is executed if a Wake-up Timer overflow occurs, i.e., when the WUTIF bit is set.

- **Bit 5 – Reserved**

  This bit is reserved and will always read as zero.

- **Bit 4 – WUTR: Wake-up Timer Reset**

  When WUTR bit is written to one, the Wake-up Timer is reset, and starts counting from zero. The WUTR bit is always read as zero.

- **Bit 3 – WUTE: Wake-up Timer Enable**

  When the WUTE bit is set (one) the Wake-up Timer is enabled, and if the WUTE is cleared (zero) the Wake-up Timer function is disabled. It is recommended to reset the Wake-up Timer when enabling it, by simultaneously setting the WUTR and WUTE bits.

- **Bits 2:0 – WUTP2:0: Wake-up Timer Prescaler 2, 1, and 0**

  The WUTP2:0 bits determine the Wake-up Timer prescaling when the Wake-up Timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 18-1. The Wake-up Timer should always be reset when changing these bits.

**Table 18-1.** Wake-up Timer Prescale Select

| WUP2:0 | Number of Ultra Low Power RC Oscillator Cycles | Typical Time-out |
|---|---|---|
| 000 | 4K(4096) | 32ms |
| 001 | 8K(8192) | 64ms |
| 010 | 16K(16384) | 128ms |
| 011 | 32K(32768) | 256ms |
| 100 | 64K(65536) | 512ms |
| 101 | 128K(131072) | 1.0s |
| 110 | 256K(262144) | 2.0s |
| 111 | 512K(524288) | 4.1s |

# 19. Interrupts

## 19.1 Overview

This section describes the specifics of the interrupt handling as performed in the Atmel® AVR MCU. For a general explanation of the AVR interrupt handling, refer to Section 13.7 “Reset and Interrupt Handling” on page 39.

## 19.2 Interrupt Vectors in Atmel AVR MCU

**Table 19-1.** Reset and Interrupt Vectors

| Vector No. | Program Address(1) | Source | Interrupt Definition |
|---|---|---|---|
| 1 | 0x0000 | RESET | External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and debugWIRE Reset |
| 2 | 0x0002 | INT0 | External Interrupt 0 |
| 3 | 0x0004 | PCINT0 | Pin Change Interrupt 0 |
| 4 | 0x0006 | PCINT1 | Pin Change Interrupt 1 |
| 5 | 0x0008 | WDT | Watchdog Time-out Interrupt |
| 6 | 0x000A | WAKEUP | Wake-up Timer Overflow |
| 7 | 0x000C | TIMER1 IC | Timer/Counter 1 input Capture |
| 8 | 0x000E | TIMER1 COMPA | Timer/Counter 1 Compare Match A |
| 9 | 0x0010 | TIMER1 COMPB | Timer/Counter 1 Compare Match B |
| 10 | 0x0012 | TIMER1 OVF | Timer/Counter 1 Overflow |
| 11 | 0x0014 | TIMER0 IC | Timer/Counter 0 input Capture |
| 12 | 0x0016 | TIMER0 COMPA | Timer/Counter 0 Compare Match A |
| 13 | 0x0018 | TIMER0 COMPB | Timer/Counter 0 Compare Match B |
| 14 | 0x001A | TIMER0 OVF | Timer/Counter 0 Overflow |
| 15 | 0x001C | LIN STATUS | LIN Status Interrupt |
| 16 | 0x001E | LIN ERROR | LIN Error Interrupt |
| 17 | 0x0020 | SPI, STC | SPI, Serial Transfer Complete |
| 18 | 0x0022 | VADC CONV | V-ADC Instantaneous Conversion Complete |
| 19 | 0x0024 | VADC ACC | V-ADC Accumulated Conversion Complete |
| 20 | 0x0026 | CADC CONV | C-ADC Instantaneous Conversion Complete |
| 21 | 0x0028 | CADC REG CUR | C-ADC Regular Current |
| 22 | 0x002A | CADC ACC | C-ADC Accumulated Conversion Complete |
| 23 | 0x002C | EE READY | EEPROM Ready |
| 24 | 0x002E | SPM | SPM Ready |
| 25 | 0x0030 | PLL | PLL Lock Change Interrupt |

Notes:
1. 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.
2. When the BOOTRST Fuses are programmed, the device will jump to the Boot Loader address at reset, see Section 29. “Boot Loader Support – Read-While-Write Self-Programming” on page 167.

Table 19-2 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 19-2.** Reset and Interrupt Vectors Placement(1)

| BOOTRST | IVSEL | Reset Address | Interrupt Vectors Start Address |
|---|---|---|---|
| 1 | 0 | 0x0000 | 0x0002 |
| 1 | 1 | 0x0000 | Boot Reset Address + 0x0002 |
| 0 | 0 | Boot Reset Address | 0x0002 |
| 0 | 1 | Boot Reset Address | Boot Reset Address + 0x0002 |

Note: 1. The Boot Reset Addresses for the Atmel® AVR MCU are shown in Section 29.8.13 on page 176 and Section 29.8.14 on page 177, respectively. 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 Atmel AVR MCU is:

```
Address     Labels    Code                                Comments
0x0000                jmp         RESET                 ; Reset Handler
0x0002                jmp         INT0                  ; External Interrupt 0 Handler
0x0004                jmp         PCINT0                ; Pin Change Interrupt 0 Handler
0x0006                jmp         PCINT1                ; Pin Change Interrupt 1 Handler
0x0008                jmp         WDT                   ; Watchdog Time-out Interrupt
0x000A                jmp         WAKE_UP               ; Wake-up Timer Overflow Handler
0x000C                jmp         TIM1_IC               ; Timer1 Input Capture Handler
0x000E                jmp         TIM1_COMPA            ; Timer1 Compare A Handler
0x0010                jmp         TIM1_COMPB            ; Timer1 Compare B Handler
0x0012                jmp         TIM1_OVF              ; Timer1 Overflow Handler
0x0014                jmp         TIM0_IC               ; Timer0 Input Capture Handler
0x0016                jmp         TIM0_COMPA            ; Timer0 CompareA Handler
0x0018                jmp         TIM0_COMPB            ; Timer0 CompareB Handler
0x001A                jmp         TIM0_OVF              ; Timer0 Overflow Handler
0x001C                jmp         LIN_STATUS            ; LIN Status Handler
0x001E                jmp         LIN_ERROR             ; LIN Error Handler
0x0020                jmp         SPI, STC              ; SPI, Serial Transfer Complete Handler
0x0022                jmp         VADC_CONV             ; V-ADC Instantaneous Conversion Complete
                                                          Handler
0x0024                jmp         VADC_ACC              ; V-ADC Accumulated Conversion Complete
                                                          Handler
0x0026                jmp         CADC CONV             ; C-ADC Instantaneous Conversion Complete
                                                          Handler
0x0028                jmp         CADC_REC_CUR          ; C-ADC Regular Current Handler
0x002A                jmp         CADC_ACC              ; C-ADC Accumulated Conversion Complete
                                                          Handler
0x002C                jmp         EE_RDY                ; EEPROM Ready Handler
0x002E                jmp         SPM_RDY               ; Store Program Memory Ready Handler
0x0030                jmp         PLL                   ; PLL Lock Change Interrupt Handler
0x0031      RESET:    ldi         r16, high(RAMEND)     ; Main program start
0x0032                out         SPH,r16               ; Set Stack Pointer to top of RAM
0x0033                ldi         r16, low(RAMEND)
0x0034                out         SPL,r16
0x0035                sei                               ; Enable interrupts
0x0036                <instr>     xxx
0x0037      ...       ...         ...
;
```

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 is:

```
Address     Labels      Code        Comments
0x0000      RESET:      ldi         r16,high(RAMEND     ; Main program start
0x0001                  out         SPH,r16             ; Set Stack Pointer to top
                                                        ; of RAM
0x0002                  ldi         r16,low(RAMEND)
0x0003                  out                             SPL,r16
0x0004                  sei                             ; Enable interrupts
0x0005                  <instr>     xxx
;
.org 0x4C02
0x4C02                  jmp         INT0                ; External Interrupt 0
                                                        ; Handler
...                     ...         ...                 ;
0x4C30                  jmp         PLL_LCHNG           ; PLL Lock Change 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 is:

```
Address     Labels      Code        Comments
.org 0x0002
0x0002                  jmp         INT0                ; External Interrupt 0
                                                        ; Handler
...                     ...         ...                 ;
0x0030                  jmp         PLL_LCHNG           ; PLL Lock Change Handler
;
.org 0x4C00
0x4C00      RESET:      ldi         r16,high(RAMEND)    ; Main program start
0x4C01                  out         SPH,r16             ; Set Stack Pointer to top
                                                        ; of RAM
0x4C02                  ldi         r16,low(RAMEND)
0x4C03                  out                             SPL,r16
0x4C04                  sei                             ; Enable interrupts
0x4C05                  <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 is:

```
Address     Labels      Code        Comments
;
.org 0x4C00
0x4C00                  jmp         RESET               ; Reset Handler
0x4C02                  jmp         INT0                ; External Interrupt 0
                                                        ; Handler
...                     ...         ...                 ;
0x4C30                  jmp         PLL_LCHNG           ; PLL Lock Change Handler
;
0x4C2E      RESET:      ldi         r16,high(RAMEND)    ; Main program start
0x4C2F                  out         SPH,r16             ; Set Stack Pointer to top
                                                        ; of RAM
0x4C30                  ldi         r16,low(RAMEND)
0x4C31                  out         SPL,r16
0x4C32                  sei                             ; Enable interrupts
0x4C33                  <instr>     xxx
```

## 19.3 Moving Interrupts Between Application and Boot Space

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

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);
}
```

## 19.4 Register Description

### 19.4.1 MCUCR – MCU Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x35 (0x55) | JTD | – | – | PUD | – | – | IVSEL | IVCE | MCUCR |
| Read/Write | R/W | R | R | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

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

  When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to Section 29. “Boot Loader Support – Read-While-Write Self-Programming” on page 167 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 Section 29. “Boot Loader Support – Read-While-Write Self-Programming” on page 167 for details on Boot Lock bits.

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

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

# 20. External Interrupts

## 20.1 Overview

The External Interrupts are triggered by the INT0 pin or any of the PCINT pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin change interrupt PCI1 will trigger if any enabled PCINT9:2 pin toggles and a Pin change interrupt PCI0 will trigger if any enabled PCINT1:0 pin toggles. PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT pins are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

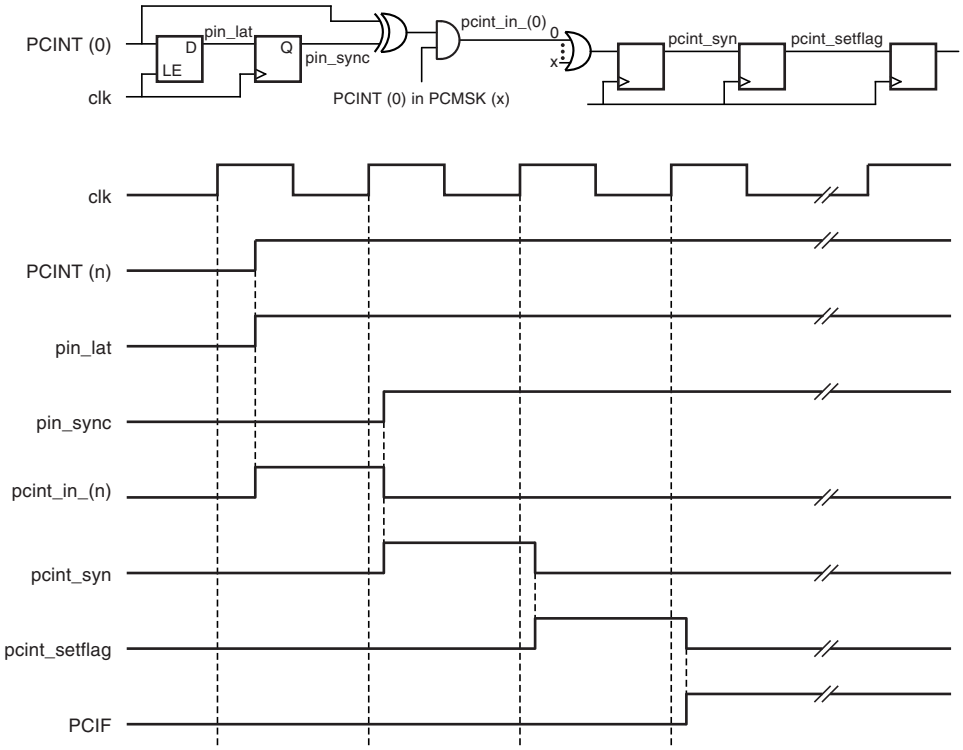
The INT0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A – EICRA. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 requires the presence of an I/O clock, described in Section 15.1 “Clock Systems and their Distribution” on page 48. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. 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, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT Fuses as described in Section 15. “System Clock and Clock Options” on page 48.

## 20.2 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 20-1.

**Figure 20-1. Pin Change Interrupt**



## 20.3 Register Description

### 20.3.1 EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x69) | – | – | – | – | – | – | ISC01 | ISC00 | EICRA |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

  The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 20-1. The value on the INT0 pin is 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. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

**Table 20-1.** Interrupt 0 Sense Control

| ISC01 | ISC00 | Description |
|---|---|---|
| 0 | 0 | The low level of INT0 generates an interrupt request. |
| 0 | 1 | Any logical change on INT0 generates an interrupt request. |
| 1 | 0 | The falling edge of INT0 generates an interrupt request. |
| 1 | 1 | The rising edge of INT0 generates an interrupt request. |

### 20.3.2 EIMSK – External Interrupt Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1D (0x3D) | – | – | – | – | – | – | – | INT0 | EIMSK |
| Read/Write | R | R | R | R | R | R | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:1 – Reserved**

  These bits are reserved and will always read as zero.

- **Bit 0 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

### 20.3.3 EIFR – External Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1C (0x3C) | – | – | – | – | – | – | – | INTF0 | EIFR |
| Read/Write | R | R | R | R | R | R | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:1 – Reserved**

  These bits are reserved and will always read as zero.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

### 20.3.4 PCICR – Pin Change Interrupt Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x68) | – | – | – | – | – | – | PCIE1 | PCIE0 | PCICR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:2 – Reserved**

  These bits are reserved bits ins the AVR MCU, and will always read as zero.

- **Bit 1 – PCIE1: Pin Change Interrupt Enable 1**

  When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT9:2 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT9:2 pins are enabled individually by the PCMSK1 Register.

- **Bit 0 – PCIE0: Pin Change Interrupt Enable 0**

  When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT1:0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT1:0 pins are enabled individually by the PCMSK0 Register.

### 20.3.5 PCIFR – Pin Change Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1B (0x3B) | – | – | – | – | – | – | PCIF1 | PCIF0 | PCIFR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:2 – Reserved**

  These bits are reserved bits ins the Atmel® AVR MCU, and will always read as zero.

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

  When a logic change on any PCINT9:2 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

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

  When a logic change on any PCINT1:0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

### 20.3.6 PCMSK1 – Pin Change Mask Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x6C) | PCINT9 | PCINT8 | PCINT7 | PCINT6 | PCINT5 | PCINT4 | PCINT3 | PCINT2 | PCMSK1 |
| 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 – PCINT9:2: Pin Change Enable Mask 9:2**

  These bits select whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT9:2 is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT9:2 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

### 20.3.7 PCMSK0 – Pin Change Mask Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x6B) | – | – | – | – | – | – | PCINT1 | PCINT0 | PCMSK0 |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:2 – Reserved**

  These bits are reserved bits ins the Atmel AVR MCU, and will always read as zero.

- **Bit 1:0 – PCINT1:0: Pin Change Enable Mask 1:0**

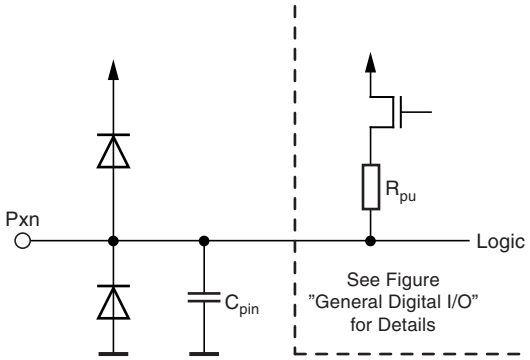
  Each PCINT1:0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT1:0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT1:0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

# 21. I/O-Ports

## 21.1 Overview

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). 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 21-1. Refer to Section 31. “Electrical Characteristics AVR MCU” on page 192ff for a complete list of parameters.

**Figure 21-1. 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 Section 21.4 “Register Description” on page 88.

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.

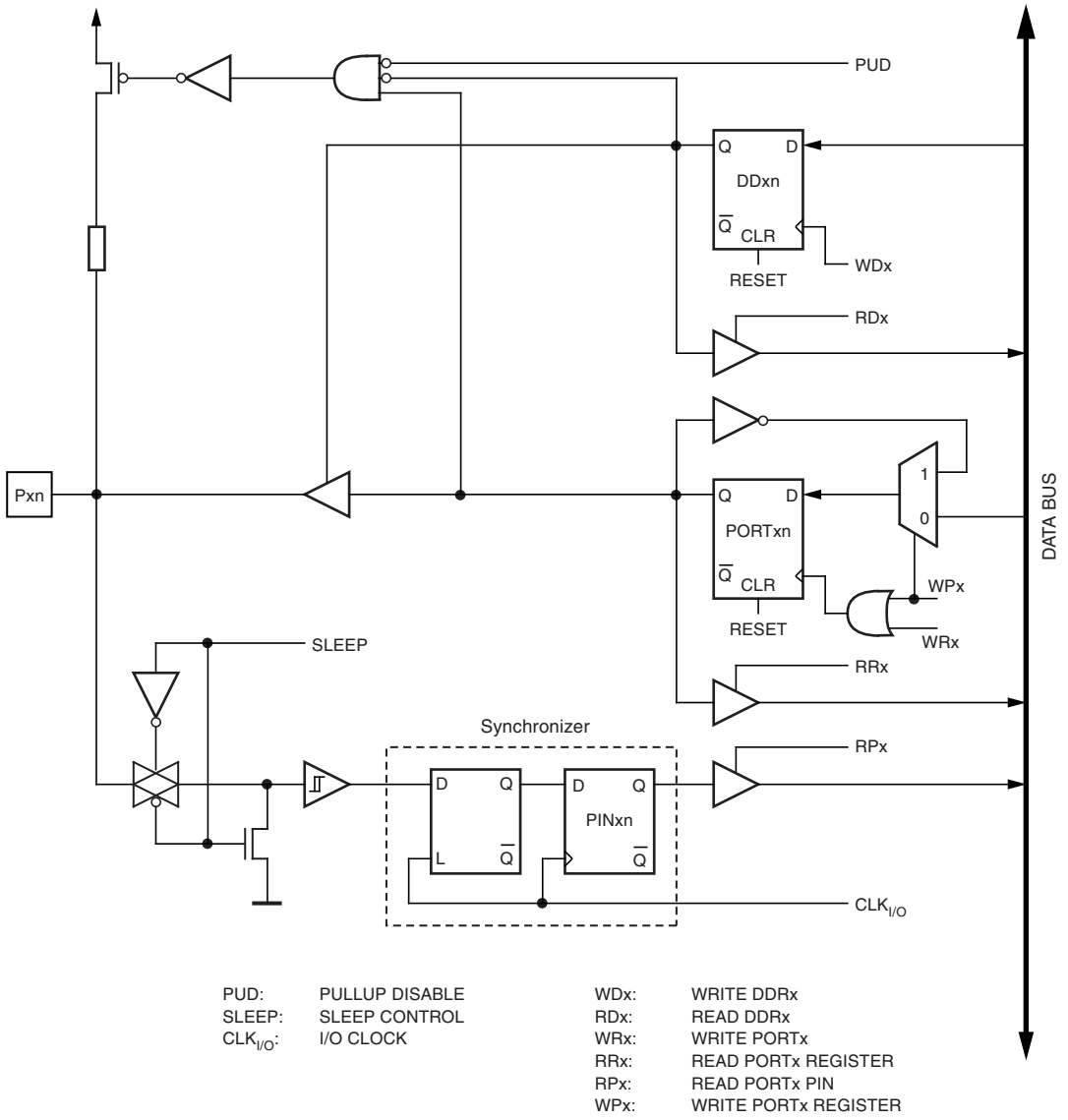
Using the I/O port as General Digital I/O is described in Section 21.2 “Ports as General Digital I/O” on page 79. Many port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 21.3 “Alternate Port Functions” on page 83. 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.

## 21.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 21-2 shows a functional description of one I/O-port pin, here generically called Pxn.

**Figure 21-2. 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.

### 21.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 21.4 “Register Description” on page 88, 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).

### 21.2.2 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.

### 21.2.3 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 21-1 summarizes the control signals for the pin value.

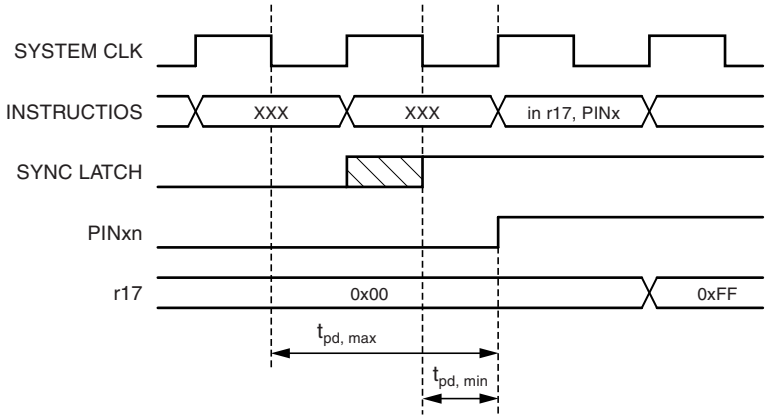
**Table 21-1.** Port Pin Configurations

| DDxn | PORTxn | PUD (in MCUCR) | I/O | Pull-up | Comment |
|---|---|---|---|---|---|
| 0 | 0 | X | Input | No | 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) |

### 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.

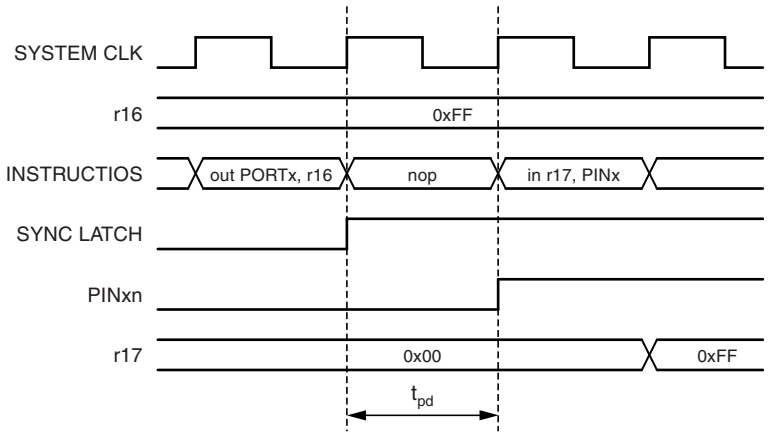
**Figure 21-3. Synchronization when Reading an Externally Applied Pin Value**




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**



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

### 21.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 21-2 on page 79, 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-save mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{REG}/2$.

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 Section 21.3 “Alternate Port Functions” on page 83.

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 mode, as the clamping in these sleep mode produces the requested logic change.

### 21.2.6 Unconnected Pins

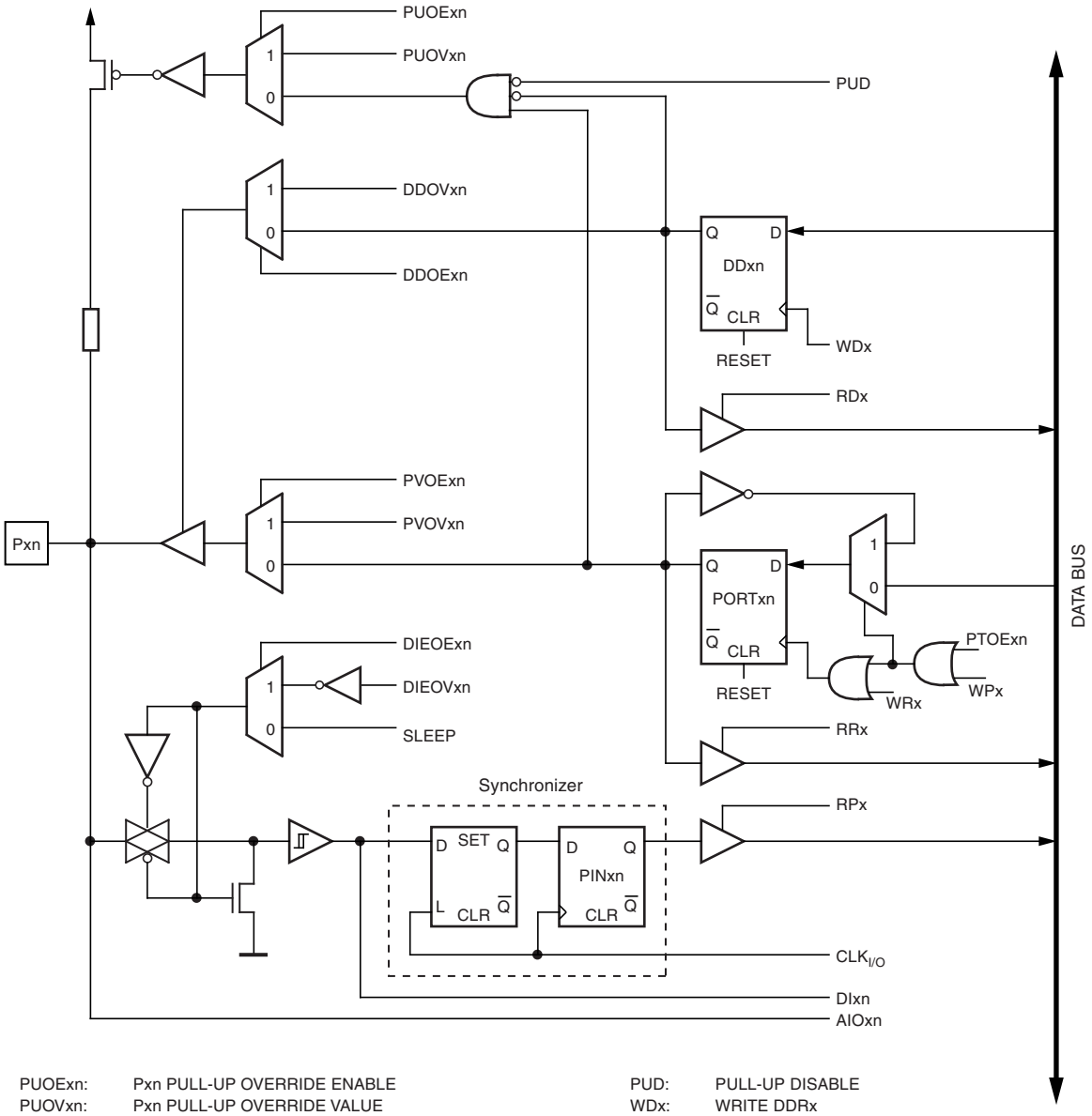
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to $V_{CC}$ or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

## 21.3 Alternate Port Functions

Many port pins have alternate functions in addition to being general digital I/Os. Figure 21-5 shows how the port pin control signals from the simplified Figure 21-2 on page 79 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 21-5. Alternate Port Functions(1)**




| | | | |
|---|---|---|---|
| PUOExn: | Pxn PULL-UP OVERRIDE ENABLE | PUD: | PULL-UP 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 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<br>(ADC Input 1, Signal Ground or Pin Change Interrupt 1) |
| PA0 | ADC0/SGND/PCINT0<br>(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 × PCIE0) | PA0DID \| (PCINT0 × PCIE0) |
| DIEOV | $\overline{\text{PA1DID}}$ | $\overline{\text{PA0DID}}$ |
| DI | PCINT1 INPUT | PCINT0 INPUT |
| AIO | ADC1 INPUT<br>SGND INPUT | ADC0 INPUT<br>SGND INPUT |

### 21.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 21-5.

**Table 21-5. Port B Pins Alternate Functions**

| Port Pin | Alternate Functions |
|---|---|
| PB7 | MISO/ICP10/INT0/PCINT9 (SPI Bus Master Input/Slave Output, Timer1 Input Capture Source0, External Interrupt or Pin Change Interrupt 9) |
| PB6 | MOSI/PCINT8 (SPI Bus Master Output/Slave Input or Pin Change Interrupt 8) |
| PB5 | SCK/PCINT7 (SPI Bus Serial Clock or Pin Change Interrupt 7) |
| PB4 | $\overline{SS}$/PCINT6 (SPI Bus Slave Select input or Pin Change Interrupt 6) |
| PB3 | TXD/PCINT5 (LIN TXD or Pin Change Interrupt 5) |
| PB2 | CKOUT/PCINT4 (Clock Output or Pin Change Interrupt 4) |
| PB1 | RXD/PCINT3 (LIN RXD or Pin Change Interrupt 3) |
| PB0 | FH/PCINT2 (Force High or Pin Change Interrupt 2) |

The alternate pin configuration is as follows:

- **MISO/ICP10/INT0/PCINT9 - Port B, Bit7**

  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 DDB7. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB7. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB7 bit. When not operating in SPI mode, this pin can serve as an external interrupt source.

  ICP10, Timer1 Input capture source 0. The PB7 pin can serve as input capture source for Timer1.

  INT0, External Interrupt source 0: The PB7 pin can serve as an external interrupt source to the MCU.

  PCINT9: Pin Change Interrupt 9. This pin can serve as external interrupt source.

- **MOSI/PCINT8 - Port B, Bit6**

  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 DDB6. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB6. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB6 bit. When not operating in SPI mode, this pin can serve as an external interrupt source.

  PCINT8: Pin Change Interrupt 8. This pin can serve as external interrupt source.

- **SCK/PCINT7 - Port B, Bit5**

  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 by the SPI to be an input, the pull-up can still be controlled by the PORTB5 bit.

  PCINT7: Pin Change Interrupt 7. This pin can serve as external interrupt source.

- **$\overline{SS}$/PCINT6 - Port B, Bit4**

  SS, Slave Select input: When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. 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 DDB4. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB4 bit. When not operating in SPI mode, this pin can serve as Clock Output, CPU Clock divided by 2. See ”Clock Output” on page 28.

  PCINT6: Pin Change Interrupt 6. This pin can serve as external interrupt source.

- **TXD/PCINT5 - Port B, Bit3**

  TXD: This pin can serve as TXD pin for the LIN interface.

  PCINT5: Pin Change Interrupt 5. This pin can serve as external interrupt source.

- **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/$\overline{SS}$/PCINT6 |
|---|---|---|---|---|
| PUOE | SPE × MASTER | SPE × $\overline{MASTER}$ | SPE × $\overline{MASTER}$ | SPE × $\overline{MASTER}$ |
| PUOV | PORTB7 × $\overline{PUD}$ | PORTB7 × $\overline{PUD}$ | PORTB7 × $\overline{PUD}$ | PORTB7 × $\overline{PUD}$ |
| DDOE | SPE × MASTER | SPE × $\overline{MASTER}$ | SPE × $\overline{MASTER}$ | SPE × $\overline{MASTER}$ |
| DDOV | 0 | 0 | 0 | 0 |
| PVOE | SPE × $\overline{MASTER}$ | 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<br>ICP10<br>SPI MASTER<br>PCINT9 | SPI SLAVE<br>PCINT8 | SCK<br>PCINT7 | $\overline{SS}$<br>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 | $\overline{LINTXD}$ × $\overline{PBOE3}$ | CKOE | 0 | 1 |
| PVOE | LINTXEN | CKOE | 0 | PBOE0 |
| PVOV | LINTXD × $\overline{PBOE3}$ | CKOUT | 0 | 1 |
| PTOE | 0 | 0 | 0 | 0 |
| DIEOE | PCINT5 × PCIE | (PCINT4 × PCIE) \| CKOE | PCINT3 | PCINT2 × PCIE |
| DIEOV | 1 | PCINT4 × PCIE \| $\overline{CKOE}$ | 1 | 1 |
| DI | T1<br>PCINT5 | LINRXD<br>PCINT4 | PCINT3 | T0<br>PCINT2 |
| AIO | - | - | - | - |

## 21.4 Register Description

### 21.4.1 MCUCR – MCU Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x35 (0x55) | – | – | CKOE | PUD | – | – | IVSEL | IVCE | MCUCR |
| Read/Write | R | R | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 4 – PUD: Pull-up Disable**
  When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 21.2.1 “Configuring the Pin” on page 80 for more details about this feature.

### 21.4.2 PORTA – Port A Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x02 (0x22) | – | – | – | – | – | – | PORTA1 | PORTA0 | PORTA |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

### 21.4.3 DDRA – Port A Data Direction Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x01 (0x21) | – | – | – | – | – | – | DDA1 | DDA0 | DDRA |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

### 21.4.4 PINA – Port A Input Pins Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x00 (0x20) | – | – | – | – | – | – | PINA1 | PINA0 | PINA |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

### 21.4.5 PORTB – Port B Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x05 (0x25) | 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 | |

### 21.4.6 DDRB – Port B Data Direction Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x04 (0x24) | 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 | |

### 21.4.7 PINB – Port B Input Pins Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x03 (0x23) | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 | PINB |
| Read/Write | R | R | R | R | R | R | R | R | |
| Initial Value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

### 21.4.8 PBOV – Port B Override

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xDC) | PBOVCE | – | – | – | PBOE3 | – | – | PBOE0 | PBOV |
| Read/Write | R/W | R | R | R | R/W | R | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – PBOVCE: Port B Override Change Enable**

  The PBOE0 bit can only be changed by a timed sequence:

  1. In the same operation, write one to the PBOVCE bit and zero to all other bits in the PBOV register.
  2. Within the next four clock cycles, write zero to the PBOVCE bit and the desired values to the PBOE0 bit. This must be done in one operation.

- **Bits 6:4 - Reserved**

  These bits are reserved bits and will always read as zero.

- **Bit 3 - PBOE3: Port B Override Enable 3**

  This bit overrides normal driving capabilities for the LIN transmit signal. If this bit is set, the LIN transmit signal may either be using the internal pull-up transistor or tristating the port for the LINTX high value, depending on the PORTB3 setting. In both cases the pin will be driven actively low for the LINTX low value.

  When the PBOE3 bit is cleared, normal LINTX driving capabilities will be restored and the pin will be driven actively high for the LINTX high value and and driven actively low for the LINTX low value. This is regardless of the PORTB3 setting.

  Note that when the LIN module is disabled, the PBOE3 bit has no effect on the port functionality.

  The LINTX driving capabilities is shown in Table 21-8.

**Table 21-8. LINTX Driving Capabilities**

| PBOE3 | PORTB3 | LINTX low | LINTX high |
|---|---|---|---|
| 0 | x | Active low | Active high |
| 1 | 0 | Active low | Tristated |
| 1 | 1 | Active low | Internal pull-up |

- **Bits 2:1 - Reserved**

  These bits are reserved bits and will always read as zero.

- **Bit 0 - PBOE0: Port B Override Enable 0**

  When this bit is set, PB0 is set to one regardless of settings in the PORTB, DDRB and PINB registers. When this bit is cleared, PB0 the overriding is disabled.

# 22. Timer/Counter0 and Timer/Counter1 Prescalers

## 22.1 Overview

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

### 22.1.1 Internal Clock Source

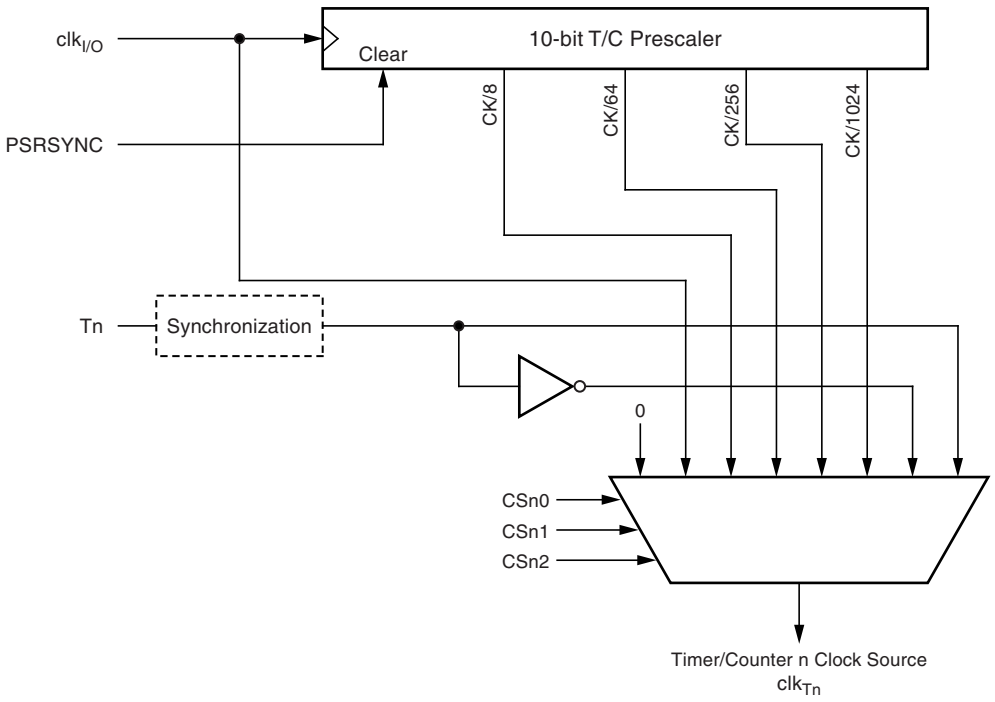
The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

### 22.1.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

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

**Figure 22-1. Prescaler for Timer/Counter**

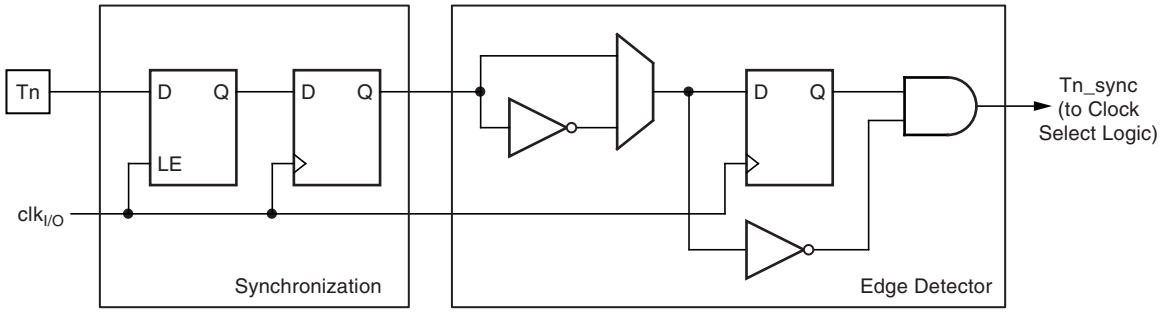


## 22.2 External Clock Source

An external clock source applied to the Tn pin can be used as Timer/Counter clock ($clk_{Tn}$). The Tn 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 22-2 shows a functional equivalent block diagram of the Tn 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 $clk_{Tn}$ pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects. See Table 22-1 on page 92 for details.

**Figure 22-2. Tn Pin Sampling**




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

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

Note: The synchronization logic on the input pins (Tn) is shown in Figure 22-2.

## 22.3 Register Description

### 22.3.1 TCCRnB – Timer/Counter n Control Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | CSn2 | CSn1 | CSn0 | TCCRnB |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 2, 1, 0 – CSn2, CSn1, CSn0: Clock Select0, Bit 2, 1, and 0**

The Clock Select n bits 2, 1, and 0 define the prescaling source of Timer n.

**Table 22-1. Clock Select Bit Description**

| CSn2 | CSn1 | CSn0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped) |
| 0 | 0 | 1 | $clk_{I/O}$/(No prescaling) |
| 0 | 1 | 0 | $clk_{I/O}$/8 (From prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (From prescaler) |
| 1 | 0 | 0 | $clk_{I/O}$/256 (From prescaler) |
| 1 | 0 | 1 | $clk_{I/O}$/1024 (From prescaler) |
| 1 | 1 | 0 | External clock source on Tn pin. Clock on falling edge. |
| 1 | 1 | 1 | External clock source on Tn pin. Clock on rising edge. |

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

### 22.3.2 General Timer/Counter Control Register – GTCCR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TSM | – | – | – | – | – | – | PSRSYNC | GTCCR |
| Read/Write | R/W | R | R | R | R | R | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

  Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRSYNC bit is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero the PSRSYNC bit is cleared by hardware, and the Timer/Counters start counting simultaneously.

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

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

# 23. Timer/Counter(T/C0, T/C1)

## 23.1 Features

- Clear timer on compare match (auto reload)
- Input capture unit
- Four independent interrupt sources (TOVn, OCFnA, OCFnB, ICFn)
- 8-bit mode with two independent output compare units
- 16-bit mode with one independent output compare unit

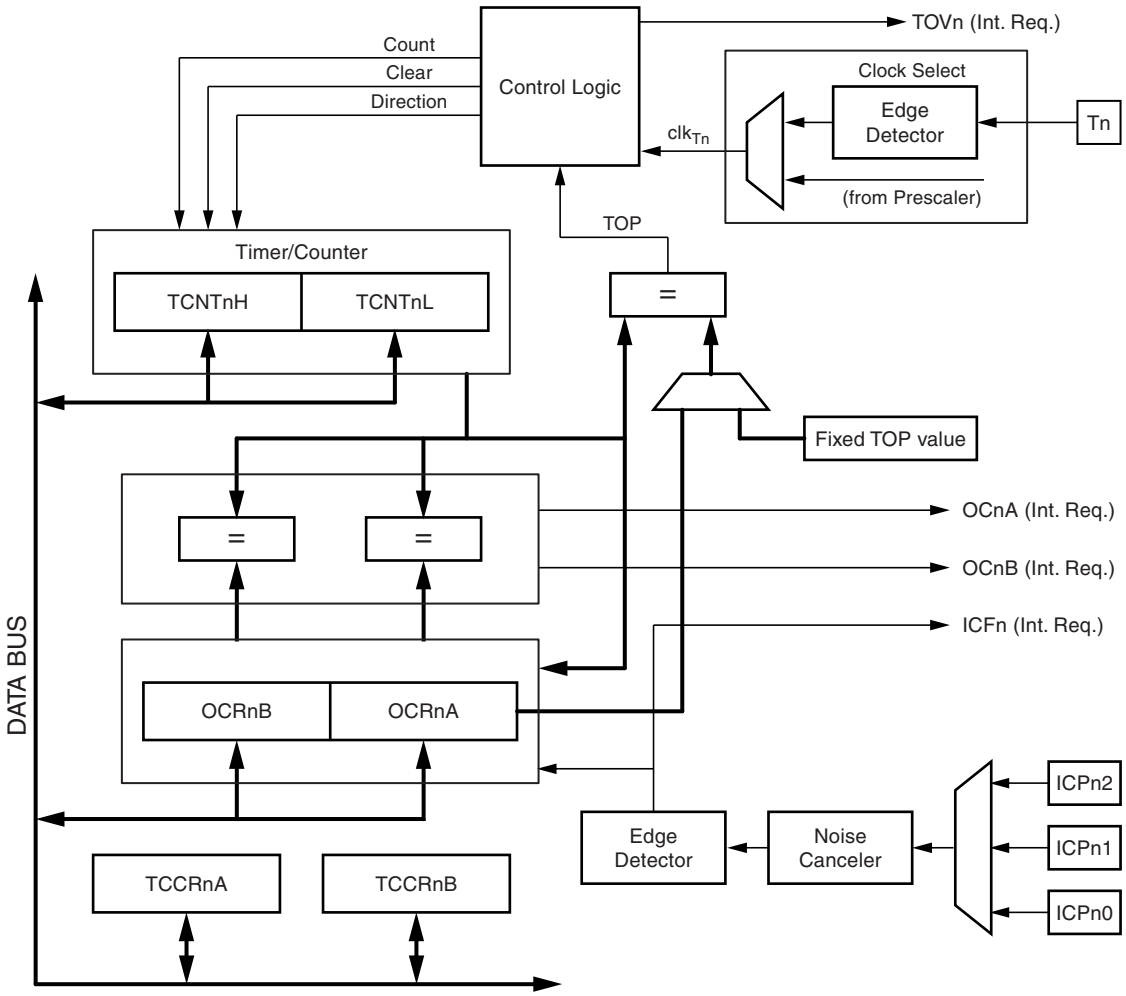
## 23.2 Overview

Timer/Counter n is a general purpose 8-/16-bit Timer/Counter module, with two/one Output Compare units and Input Capture feature.

The Atmel® AVR MCU has two Timer/Counters, Timer/Counter0 and Timer/Counter1. The functionality for both Timer/Counters is described below. Timer/Counter0 and Timer/Counter1 have different Timer/Counter registers, as shown in Section 32. "Register Summary" on page 203.

The Timer/Counter general operation is described in 8-/16-bit mode. A simplified block diagram of the 8-/16-bit Timer/Counter is shown in Figure 23-1. 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 Section 23.10 "Register Description" on page 104.

**Figure 23-1. 8-/16-bit Timer/Counter Block Diagram**



### 23.2.1 Registers

The Timer/Counter Low Byte Register (TCNTnL) and Output Compare Registers (OCRnA and OCRnB) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in Figure 23-1 on page 93) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure.

In 16-bit mode the Timer/Counter consists one more 8-bit register, the Timer/Counter High Byte Register (TCNTnH). Furthermore, there is only one Output Compare Unit in 16-bit mode as the two Output Compare Registers, OCRnA and OCRnB, are combined to one 16-bit Output Compare Register. OCRnA contains the low byte of the word and OCRnB contains the higher byte of the word. When accessing 16-bit registers, special procedures described in Section 23.9 “Accessing Registers in 16-bit Mode” on page 102 must be followed.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock ($clk_{Tn}$).

### 23.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the module number, e.g., Timer/Counter number. A lower case “x” replaces the unit, e.g., OCRnx and ICPnx describes OCRnA/B and ICP1/0x . However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0L for accessing Timer/Counter0 counter value and so on.

The definitions in Table 23-1 are also used extensively throughout the document.

**Table 23-1. Definitions**

| | |
|---|---|
| **BOTTOM** | The counter reaches the BOTTOM when it becomes 0. |
| **MAX** | The counter reaches its MAXimum when it becomes 0xFF (decimal 255) in 8-bit mode or 0xFFFF (decimal 65535) in 16-bit mode. |
| **TOP** | The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF/0xFFFF (MAX) or the value stored in the OCRnA Register. |

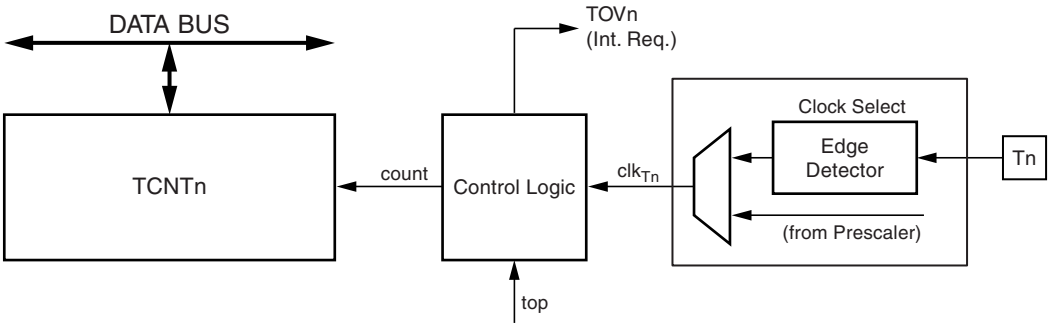
## 23.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source. The Clock Select logic is controlled by the Clock Select (CSn2:0) bits located in the Timer/Counter Control Register n B (TCCRnB), and controls which clock source and edge the Timer/Counter uses to increment its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock ($clk_{Tn}$). For details on clock sources and prescaler, see Section 22. “Timer/Counter0 and Timer/Counter1 Prescalers” on page 90

## 23.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 23-2 on page 95 shows a block diagram of the counter and its surroundings.

**Figure 23-2. Counter Unit Block Diagram**




Signal description (internal signals):

| | |
|---|---|
| **count** | Increment or decrement TCNTn by 1. |
| **$clk_{Tn}$** | Timer/Counter clock, referred to as $clk_{Tn}$ in the following. |
| **top** | Signalize that TCNTn has reached maximum value. |

The counter is incremented at each timer clock ($clk_{Tn}$) until it passes its TOP value and then restarts from BOTTOM. The counting sequence is determined by the setting of the WGMn0 bits located in the Timer/Counter Control Register (TCCRnA). For more details about counting sequences, see Section 23.8 “Timer/Counter Timing Diagrams” on page 100. $clk_{Tn}$ can be generated from an external or internal clock source, selected by the Clock Select bits (CSn2:0). When no clock source is selected (CSn2:0 = 0) the timer is stopped. However, the TCNTn value can be accessed by the CPU, regardless of whether $clk_{Tn}$ is present or not. A CPU write overrides (has priority over) all counter clear or count operations. The Timer/Counter Overflow Flag (TOVn) is set when the counter reaches the maximum value and it can be used for generating a CPU interrupt.

## 23.5 Modes of Operation

The mode of operation is defined by the Timer/Counter Width (TCWn), Input Capture Enable (ICENn) and the Waveform Generation Mode (WGMn0)bits in Section 23.10.1 “TCCRnA – Timer/Counter n Control Register A” on page 104. Table 23-2 on page 95 shows the different Modes of Operation.

**Table 23-2. Modes of Operation**

| Mode | ICENn | TCWn | WGMn0 | Timer/Counter Mode of Operation | TOP | Update of OCRx at | TOV Flag Set on |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal 8-bit Mode | 0xFF | Immediate | MAX (0xFF) |
| 1 | 0 | 0 | 1 | 8-bit CTC | OCRnA | Immediate | MAX (0xFF) |
| 2 | 0 | 1 | 0 | 16-bit Mode | 0xFFFF | Immediate | MAX (0xFFFF) |
| 3 | 0 | 1 | 1 | 16-bit CTC | OCRnB, OCRnA | Immediate | MAX (0xFFFF) |
| 4 | 1 | 0 | 0 | 8-bit Input Capture mode | 0xFF | – | MAX (0xFF) |
| 5 | 1 | 1 | 0 | 16-bit Input Capture mode | 0xFFFF | – | MAX (0xFFFF) |

### 23.5.1 Normal 8-bit Mode

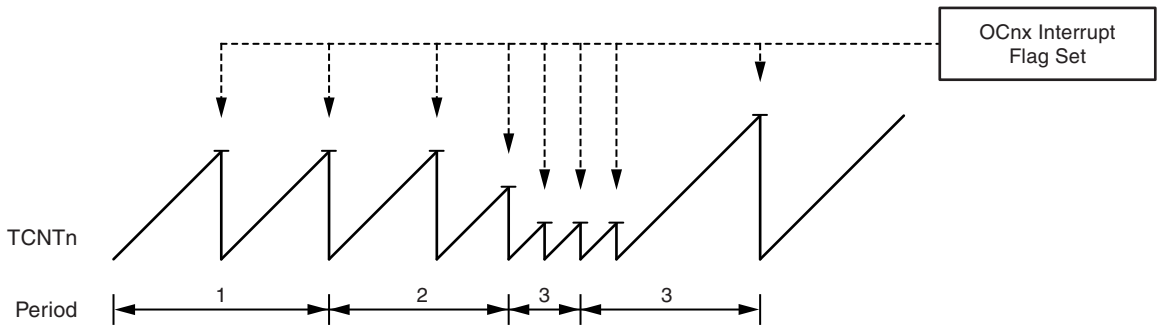
In the Normal mode, the counter (TCNTnL) is incrementing until it overruns when it passes its maximum 8-bit value (MAX = 0xFF) and then restarts from the bottom (0x00), see Table 23-2 on page 95 for bit settings. The Overflow Flag (TOVn) will be set in the same timer clock cycle as the TCNTnL becomes zero. The TOVn Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal 8-bit mode, a new counter value can be written anytime. The Output Compare Unit can be used to generate interrupts at some given time.

### 23.5.2 Clear Timer on Compare Match (CTC) 8-bit Mode

In Clear Timer on Compare or CTC mode, the OCRnA Register is used to manipulate the counter resolution, see Table 23-2 on page 95 for bit settings. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches the OCRnA. The OCRnA defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 23-3 on page 96. The counter value (TCNTn) increases until a Compare Match occurs between TCNTn and OCRnA, and then counter (TCNTn) is cleared.

**Figure 23-3. CTC Mode, Timing Diagram**




An interrupt can be generated each time the counter value reaches the TOP value by using the OCFnA Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCRnA is lower than the current value of TCNTn, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the Compare Match can occur. As for the Normal mode of operation, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

### 23.5.3 16-bit Mode

In 16-bit mode, the counter (TCNTnH/L) is a incrementing until it overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the bottom (0x0000), see Table 23-2 on page 95 for bit settings. The Overflow Flag (TOVn) will be set in the same timer clock cycle as the TCNTnH/L becomes zero. The TOVn Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime. The Output Compare Unit can be used to generate interrupts at some given time.

### 23.5.4 Clear Timer on Compare Match (CTC) 16-bit Mode

In Clear Timer on Compare 16-bit mode, OCRAnA/B Registers are used to manipulate the counter resolution, see Table 23-2 on page 95 for bit settings. In CTC mode the counter is cleared to zero when the counter value (TCNTn) matches OCRnA/B, where OCRnB represents the eight most significant bits and OCRnA represents the eight least significant bits. OCRnA/B defines the top value of 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.

An interrupt can be generated each time the counter reaches the TOP value by using the OCFnA 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 the BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCRnA/B is lower than the current value of TCNTn, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before Compare Match can occur. As for the 16-bit Mode, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

### 23.5.5 8-bit Input Capture Mode

The Timer/Counter can be used in a 8-bit Input Capture mode, see Table 23-2 on page 95 for bit settings. For full description, see Section 23.6 “Input Capture Unit” on page 97.

### 23.5.6 16-bit Input Capture Mode

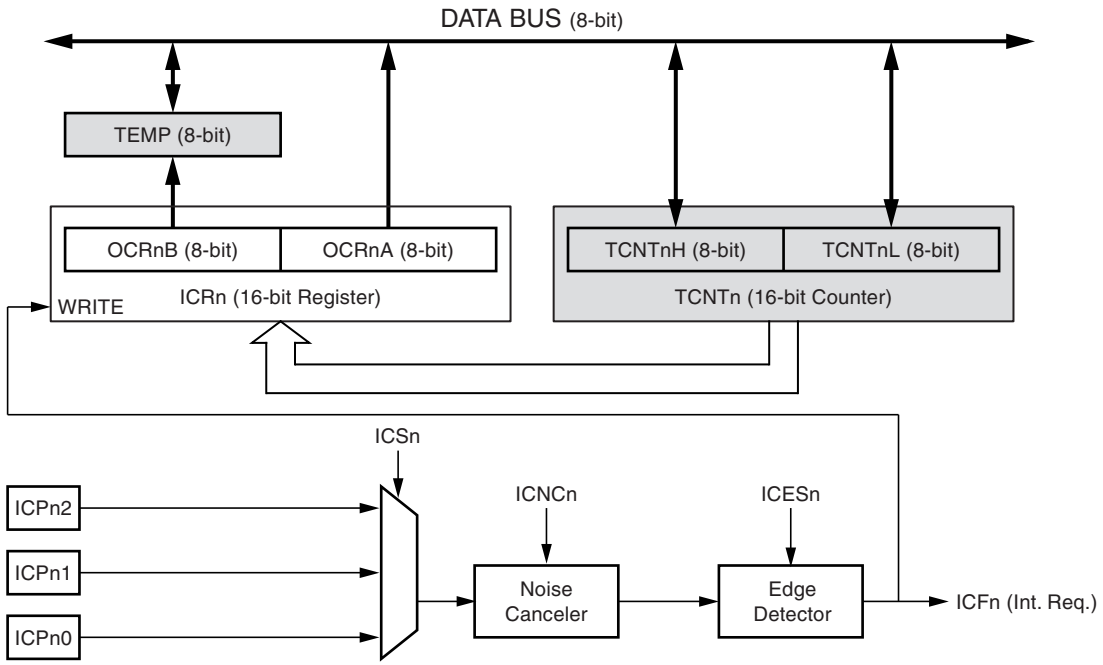
The Timer/Counter can also be used in a 16-bit Input Capture mode, see Table 23-2 on page 95 for bit settings. For full description, see Section 23.6 “Input Capture Unit” on page 97.

## 23.6 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 indicates an event, or multiple events. For Timer/Counter0, the event is triggered by completion of a CADC Instantaneous Conversion (ICP00) or completion of a CADC Accumulated Conversion (ICP01). For Timer/Counter1, the events can be applied by the PB7 pin (ICP10), by the LIN RX Complete signal (ICP11) or by the LIN TX Complete signal (ICP12). 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 23-4 on page 97. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

**Figure 23-4. Input Capture Unit Block Diagram**



The Output Compare Register OCRnA is a dual-purpose register that is also used as an 8-bit Input Capture Register ICRn. In 16-bit Input Capture mode the Output Compare Register OCRnB serves as the high byte of the Input Capture Register ICRn. In 8-bit Input Capture mode the Output Compare Register OCRnB is free to be used as a normal Output Compare Register, but in 16-bit Input Capture mode the Output Compare Unit cannot be used as there are no free Output Compare Register(s). Even though the Input Capture register is called ICRn in this section, it is referring to the Output Compare Register(s). For more information on how to access the 16-bit registers refer to Section 23.9 “Accessing Registers in 16-bit Mode” on page 102.

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

### 23.6.1 Input Capture Trigger Source

The default trigger source for the Input Capture unit is the completion of a CADC Instantaneous Conversion in Timer/Counter0 and the I/O port PB7 in Timer/Counter1. Alternatively can the completion of CADC Accumulated Conversion event be used as trigger source for Timer/Counter0, and the LIN RX and TX Complete events be used as trigger sources for Timer/Counter1. The Input Capture Trigger sources are selected as trigger sources by setting the Input Capture Select bits. Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both Input Capture inputs are sampled using the same technique. 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. An Input Capture on Timer/Counter1 can also be triggered by software by controlling the port of the PB7 pin.

### 23.6.2 Noise Canceler

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

The noise canceler is enabled by setting the Input Capture Noise Canceler (ICNCn) bit in Section 23.10.1 “TCCRnA – Timer/Counter n Control Register A” on page 104. When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICRn Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

### 23.6.3 Using the Input Capture Unit

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

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

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

**Table 23-3. Timer/Counter0 Input Capture Source (ICS)**

| ICS[1:0] | Source |
|---|---|
| 00[1][2] | ICP00: CADC Instantaneous Conversion Complete Interrupt |
| 01[1][2] | ICP01: CADC Accumulated Conversion Complete Interrupt |
| 10 | Reserved for future use |
| 11 | Reserved for future use |

Notes:
1. The noise canceler may filter out this source and it is therefore not recommended to use noise canceler with this source.
2. If this interrupt is chosen as the Input Capture source, an Input Capture event will generate both the chosen interrupt and the Input Capture interrupt. If both interrupts are enabled, the sequence in which the interrupts are handled depends on a number of factors. The application software must therefore allow for both the Input Capture interrupt being handled before the chosen interrupt trigger, and after the chosen interrupt trigger.

**Table 23-4. Timer/Counter1 Input Capture Source (ICS)**

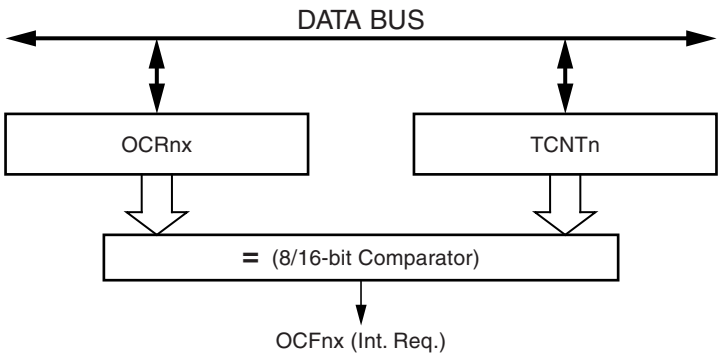
| ICS[1:0] | Source |
|---|---|
| 00 | ICP10: Port PB7 |
| 01[1][2] | ICP11: LIN RX Complete Interrupt |
| 10[1][2] | ICP12: LIN TX Complete Interrupt |
| 11 | Reserved for future use |

Notes:
1. The noise canceler may filter out this source and it is therefore not recommended to use noise canceler with this source.
2. If this interrupt is chosen as the Input Capture source, an Input Capture event will generate both the chosen interrupt and the Input Capture interrupt. If both interrupts are enabled, the sequence in which the interrupts are handled depends on a number of factors. The application software must therefore allow for both the Input Capture interrupt being handled before the chosen interrupt trigger, and after the chosen interrupt trigger.

## 23.7 Output Compare Unit

The comparator continuously compares the Timer/Counter (TCNTn) with the Output Compare Registers (OCRnA and OCRnB), and whenever the Timer/Counter equals to the Output Compare Regisers, the comparator signals a match. A match will set the Output Compare Flag at the next timer clock cycle. In 8-bit mode the match can set either the Output Compare Flag OCFnA or OCFnB, but in 16-bit mode the match can set only the Output Compare Flag OCFnA as there is only one Output Compare Unit. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. Figure 23-5 on page 99 shows a block diagram of the Output Compare unit.

**Figure 23-5. Output Compare Unit, Block Diagram**



### 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**

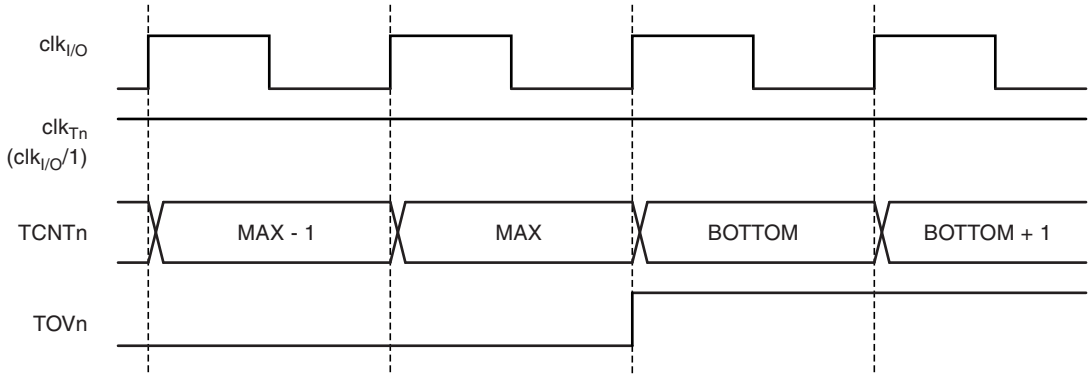



Figure 23-7 shows the same timing data, but with the prescaler enabled.

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

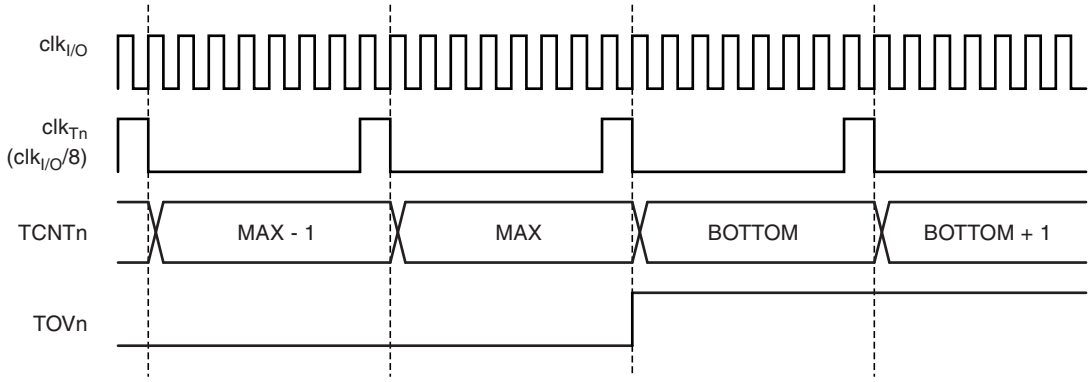


Figure 23-8 shows the setting of OCFnA and OCFnB in Normal mode.

**Figure 23-8. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk_I/O}/8$)**

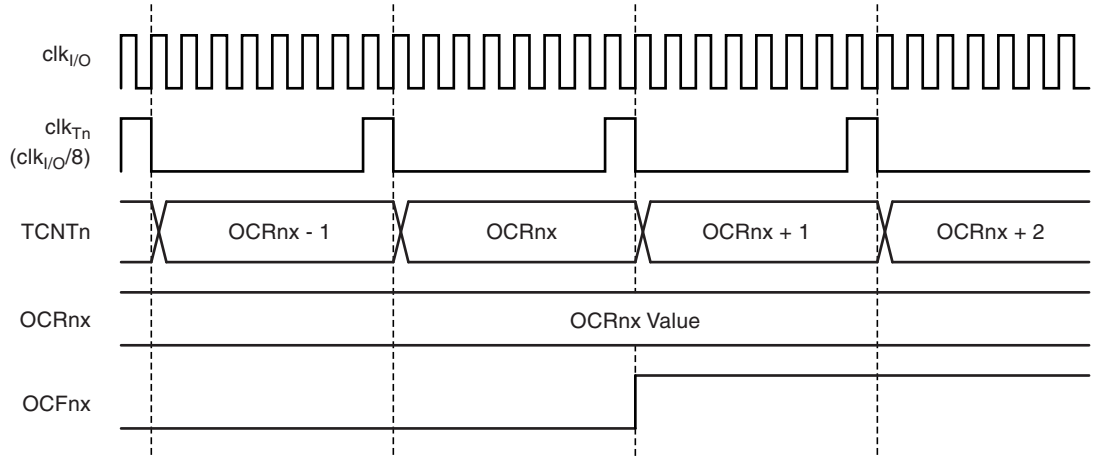
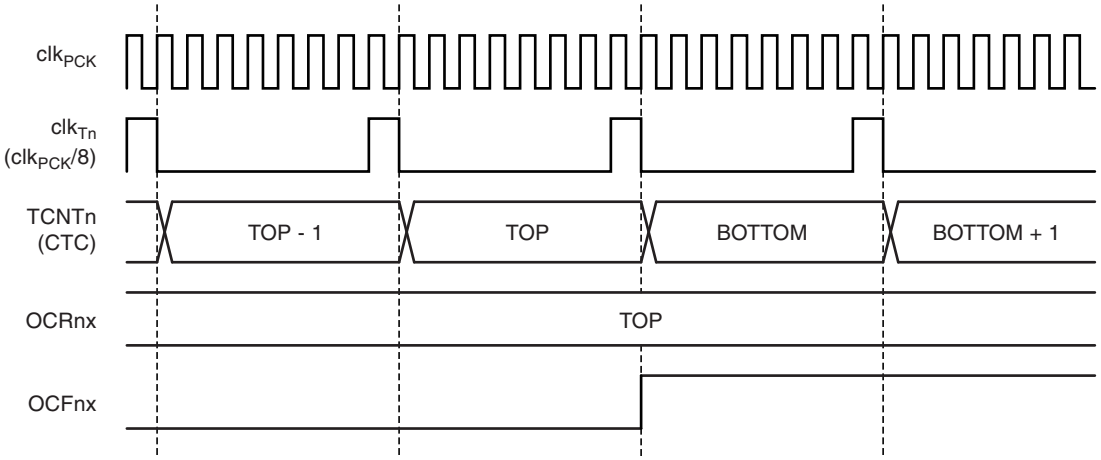



Figure 23-9 shows the setting of OCFnA and the clearing of TCNTn in CTC mode.

**Figure 23-9. Timer/Counter Timing Diagram, CTC mode, 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.

The following code examples show how to do an atomic read of the TCNTn register contents. Reading any of the OCRn register can be done by using the same principle.

Assembly Code Example

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

C Code Example

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

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

The assembly code example returns the TCNTnH/L value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNTnH/L register contents. Writing any of the OCRnA/B registers can be done by using the same principle.

Assembly Code Example

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

C Code Example

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

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

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

### 23.9.1 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.

## 23.10 Register Description

### 23.10.1 TCCRnA – Timer/Counter n Control Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | **TCWn** | **ICENn** | **ICNCn** | **ICESn** | **–** | **–** | **–** | **WGMn0** | **TCCRnA** |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7– TCWn: Timer/Counter Width**

  When this bit is written to one 16-bit mode is selected. Timer/Counter n width is set to 16-bits and the Output Compare Registers OCRnA and OCRnB are combined to form one 16-bit Output Compare Register. Because the 16-bit registers TCNTnH/L and OCRnB/A are accessed by the AVR CPU via the 8-bit data bus, special procedures must be followed. These procedures are described in Section 23.9 “Accessing Registers in 16-bit Mode” on page 102.

- **Bit 6– ICENn: Input Capture Mode Enable**

  The Input Capture Mode is enabled when this bit is written to one.

- **Bit 5 – ICNCn: Input Capture Noise Canceler**

  Setting this bit activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture Source is filtered. The filter function requires four successive equal valued samples of the Input Capture Source for changing its output. The Input Capture is therefore delayed by four System Clock cycles when the noise canceler is enabled.

- **Bit 4 – ICESn: Input Capture Edge Select**

  This bit selects which edge on the Input Capture Source that is used to trigger a capture event. When the ICESn bit is written to zero, a falling (negative) edge is used as trigger, and when the ICESn bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICESn setting, the counter value is copied into the Input Capture Register. The event will also set the Input Capture Flag (ICFn), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

- **Bits 3 – Reserved**

  This bit is reserved in the Atmel® AVR MCU and should always be written to zero.

- **Bits 2:1 – Reserved**

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

- **Bit 0 – WGMn0: Waveform Generation Mode**

  This bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, see Figure 23-6 on page 100. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter) and Clear Timer on Compare Match (CTC) mode (see Section 23.8 “Timer/Counter Timing Diagrams” on page 100).

### 23.10.2 TCCRnC – Timer/Counter n Control Register C

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | ICn1 | ICn0 | TCCRnC |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:2 – Reserved**

  These bits are reserved bits in the Atmel® AVR MCU and will always read as zero.

- **Bit 1:0 – ICS[1:0]: Input Capture Select 1:0**

  These bits control which Input Capture source that should trigger the Timer/Counter Input Capture functionality. To also trigger the Timer/Counter n Input Capture interrupt, the TICIEn bit in the Timer Interrupt Mask Register TIMSK) must be set.

  See Table 23-3 on page 99 and Table 23-4 on page 99 for Input Capture sources.

### 23.10.3 TCNTnL – Timer/Counter n Register Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TCNTnL[7:0] | | | | | | | | TCNTnL |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

The Timer/Counter Register TCNTnL gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNTnL Register blocks (disables) the Compare Match on the following timer clock. Modifying the counter (TCNTnL) while the counter is running, introduces a risk of missing a Compare Match between TCNTnL and the OCRnx Registers. In 16-bit mode the TCNTnL register contains the lower part of the 16-bit Timer/Counter n Register.

### 23.10.4 TCNTnH – Timer/Counter n Register High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TCNTnH[7:0] | | | | | | | | TCNTnH |
| 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 | |

When 16-bit mode is selected (the TCWn bit is set to one) the Timer/Counter Register TCNTnH combined to the Timer/Counter Register TCNTnL gives direct access, both for read and 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 Section 23.9 “Accessing Registers in 16-bit Mode” on page 102.

### 23.10.5 OCRnA – Timer/Counter n Output Compare Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | OCRnA[7:0] | | | | | | | | OCRnA |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNTnL). A match can be used to generate an Output Compare interrupt.

In 16-bit mode the OCRnA register contains the low byte of the 16-bit Output Compare Register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 23.9 “Accessing Registers in 16-bit Mode” on page 102.

### 23.10.6 OCRnB – Timer/Counter n Output Compare Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | OCRnB[7:0] | | | | | | | | OCRnB |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNTnL in 8-bit mode and TCNTnH in 16-bit mode). A match can be used to generate an Output Compare interrupt.

In 16-bit mode the OCRnB register contains the high byte of the 16-bit Output Compare Register. To ensure that both the high and the low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 23.9 “Accessing Registers in 16-bit Mode” on page 102.

### 23.10.7 TIMSKn – Timer/Counter n Interrupt Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | ICIEn | OCIEnB | OCIEnA | TOIEn | TIMSKn |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 3 – ICIEn: Timer/Counter n 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/Counter n Input Capture interrupt is enabled. The corresponding Interrupt Vector (see Section 19. “Interrupts” on page 70) is executed when the ICFn flag, located in TIFRn, is set.

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

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

- **Bit 1 – OCIEnA: Timer/Counter n Output Compare Match A Interrupt Enable**

  When the OCIEnA bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter n Compare Match A interrupt is enabled.

  The corresponding interrupt is executed if a Compare Match in Timer/Counter n occurs, i.e., when the OCFnA bit is set in the Timer/Counter n Interrupt Flag Register – TIFRn.

- **Bit 0 – TOIEn: Timer/Counter n Overflow Interrupt Enable**

  When the TOIEn bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter n Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter n occurs, i.e., when the TOVn bit is set in the Timer/Counter n Interrupt Flag Register – TIFRn.

### 23.10.8 TIFRn – Timer/Counter n Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | ICFn | OCFnB | OCFnA | TOVn | TIFRn |
| 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 – ICFn: Timer/Counter n Input Capture Flag**

  This flag is set when a capture event occurs, according to the setting of ICENn, ICESn and ICSn[1:0] bits in the TCCRnA and TCCRnC Registers.

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

- **Bit 2 – OCFnB: Output Compare Flag n B**

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

  The OCFnB is not set in 16-bit Output Compare mode when the Output Compare Register OCRnB is used as the high byte of the 16-bit Output Compare Register or in 16-bit Input Capture mode when the Output Compare Register OCRnB is used as the high byte of the Input Capture Register.

- **Bit 1– OCFnA: Output Compare Flag n A**

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

  The OCFnA is also set in 16-bit mode when a Compare Match occurs between the Timer/Counter n and 16-bit data in OCRnB/A. The OCFnA is not set in Input Capture mode when the Output Compare Register OCRnA is used as an Input Capture Register.

- **Bit 0 – TOVn: Timer/Counter n Overflow Flag**

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

# 24. SPI – Serial Peripheral Interface

## 24.1 Features

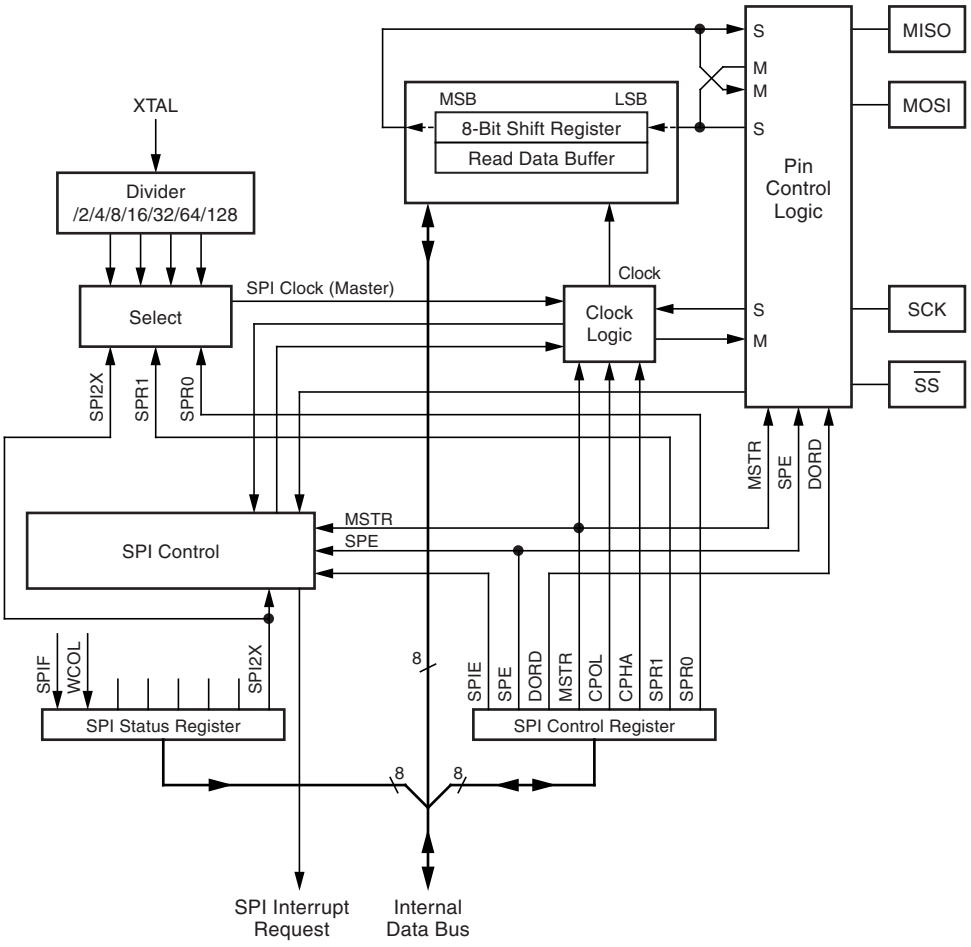
- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- LSB first or MSB first data transfer
- Seven programmable bit rates
- End of transmission interrupt flag
- Write collision protection flag
- Wake-up from Idle Mode
- Double speed (CK/2) Master SPI Mode

## 24.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the Atmel® AVR MCU and peripheral devices or between several AVR devices.

When the SPI is not used, power consumption can be minimized by writing the PRSPI bit in PRR0 to one. See Section 16.7.2 “PRR0 – Power Reduction Register 0” on page 57 for details on how to use the PRSPI bit.

**Figure 24-1. SPI Block Diagram(1)**



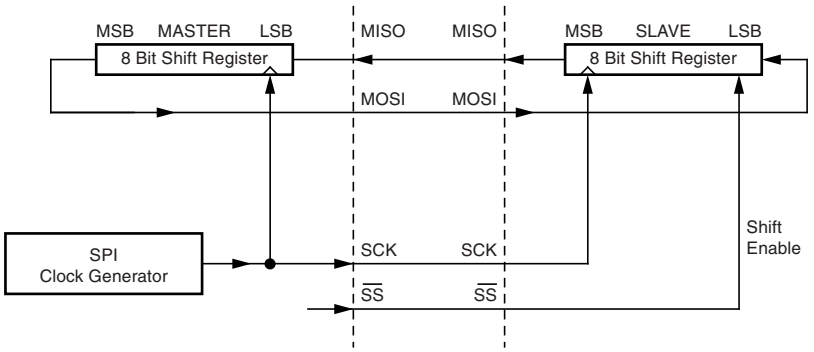

Note: 1. Refer to “Alternate Port Functions” on page 83 for SPI pin placement.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 24-2. 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 24-2. 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 $f_{osc}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 24-1 on page 109. For more details on automatic port overrides, refer to Section 21.3 “Alternate Port Functions” on page 83.

**Table 24-1. 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 Section 21.3.2 “Alternate Functions of Port B” on page 86 for a detailed description of how to define the direction of the user defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. E.g., if MOSI is placed on pin PB5, replace DD_MOSI with DDB5 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. See Section 12. “About Code Examples” on page 34

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

## 24.3 $\overline{SS}$ Pin Functionality

### 24.3.1 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.

### 24.3.2 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.

## 24.4 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 24-3 and Figure 24-4 on page 113. 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 24-3 on page 114 and Table 24-4 on page 114, as done in Table 24-2.

**Table 24-2.** SPI Modes

| SPI Mode | Conditions | Leading Edge | Trailing Edge |
|---|---|---|---|
| 0 | CPOL=0, CPHA=0 | Sample (Rising) | Setup (Falling) |
| 1 | CPOL=0, CPHA=1 | Setup (Rising) | Sample (Falling) |
| 2 | CPOL=1, CPHA=0 | Sample (Falling) | Setup (Rising) |
| 3 | CPOL=1, CPHA=1 | Setup (Falling) | Sample (Rising) |

**Figure 24-3. SPI Transfer Format with CPHA = 0**

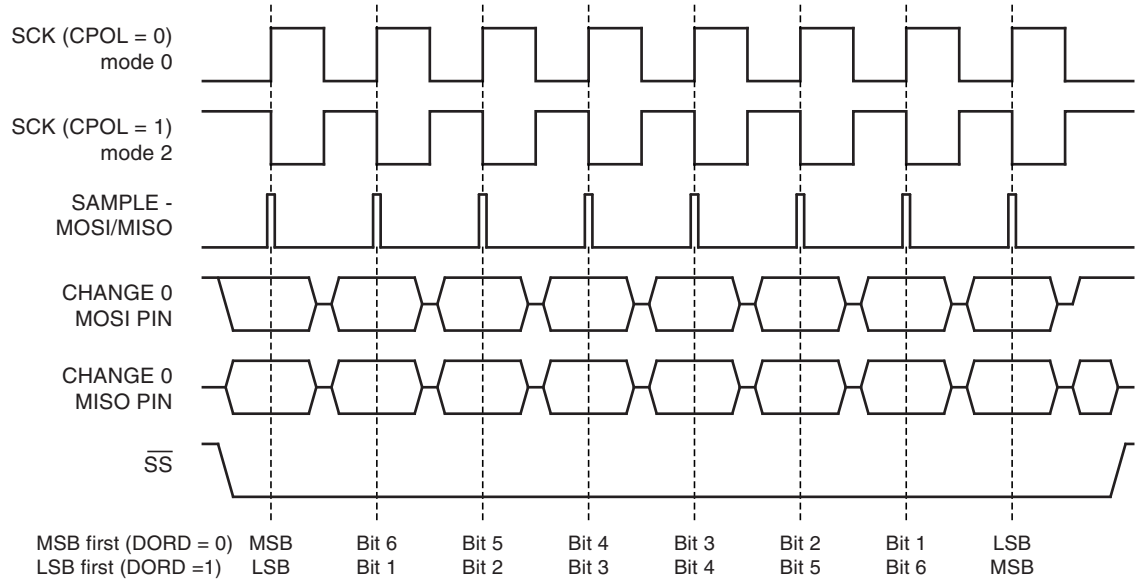
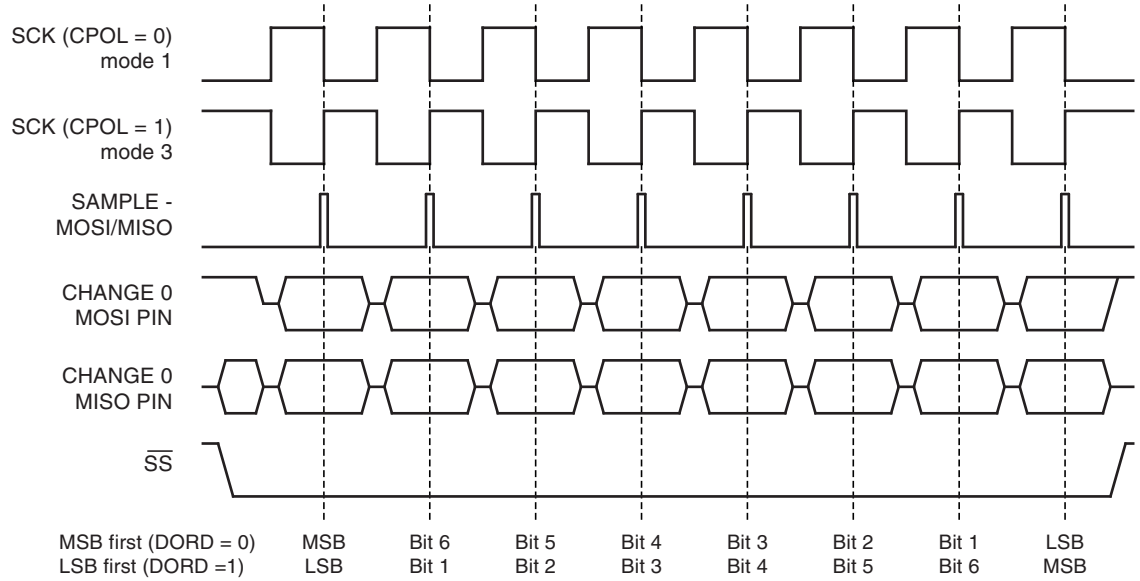



**Figure 24-4. SPI Transfer Format with CPHA = 1**



## 24.5 Register Description

### 24.5.1 SPCR – SPI Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x2C (0x4C) | 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 24-3 and Figure 24-4 for an example. The CPOL functionality is summarized below.

**Table 24-3.** 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 24-3 and Figure 24-4 for an example. The CPHA functionality is summarized below.

**Table 24-4.** 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 Oscillator Clock frequency $f_{osc}$ is shown in the following table:

**Table 24-5.** Relationship Between SCK and the Oscillator Frequency

| SPI2X | SPR1 | SPR0 | SCK Frequency |
|---|---|---|---|
| 0 | 0 | 0 | $f_{osc}/4$ |
| 0 | 0 | 1 | $f_{osc}/16$ |
| 0 | 1 | 0 | $f_{osc}/64$ |
| 0 | 1 | 1 | $f_{osc}/128$ |
| 1 | 0 | 0 | $f_{osc}/2$ |
| 1 | 0 | 1 | $f_{osc}/8$ |
| 1 | 1 | 0 | $f_{osc}/32$ |
| 1 | 1 | 1 | $f_{osc}/64$ |

### 24.5.2 SPSR – SPI Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x2D (0x4D) | **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 – Reserved**

  These bits are reserved bits in the Atmel® AVR MCU 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 24-5 on page 115). 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 $f_{osc}/4$ or lower.

  The SPI interface on the Atmel AVR MCU is also used for program memory and EEPROM downloading or uploading. See Table 30.6 on page 183 for serial programming and verification.

### 24.5.3 SPDR – SPI Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x2E (0x4E) | **MSB** | | | | | | | **LSB** | **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 |

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.

# 25. LIN/UART

## 25.1 Features

- Single master with multiple slaves concept
- Low cost Silicon implementation based on common UART/SCI interface
- Self synchronization with on-chip oscillator in Slave Node
- Deterministic signal transmission with signal propagation time computable in advance
- Low cost single-wire implementation

## 25.2 Overview

The LIN (Local Interconnect Network) is a serial communications protocol which efficiently supports the control of mechatronics nodes in distributed automotive applications.

LIN provides a cost efficient bus communication where the bandwidth and versatility of CAN are not required. If LIN is not required, the controller alternatively can be programmed as Universal Asynchronous serial Receiver and Transmitter (UART).

## 25.3 LIN Features

- Hardware implementation of LIN 2.1 (LIN 1.3 compatibility)
- Small, CPU efficient and independent Master/Slave routines based on “LIN Work Flow Concept” of LIN 2.1 specification
- Automatic LIN header handling and filtering of irrelevant LIN frames
- Automatic LIN response handling
- Extended LIN error detection and signaling
- Hardware frame time-out detection
- Automatic re-synchronization to ensure proper frame integrity
- Fully flexible extended frames support capabilities

## 25.4 UART Features

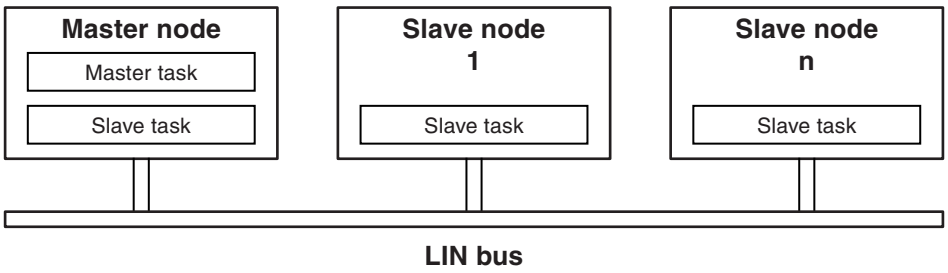
- Full duplex operation (independent serial receive and transmit processes)
- Asynchronous operation
- High resolution baud rate generator
- Hardware support of 8 data bits, odd/even/no parity bit, 1 stop bit frames
- Data over-run and framing error detection

## 25.5 LIN Protocol

### 25.5.1 Master and Slave

A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only.

**Figure 25-1. LIN Cluster with One Master Node and “n” Slave Nodes**

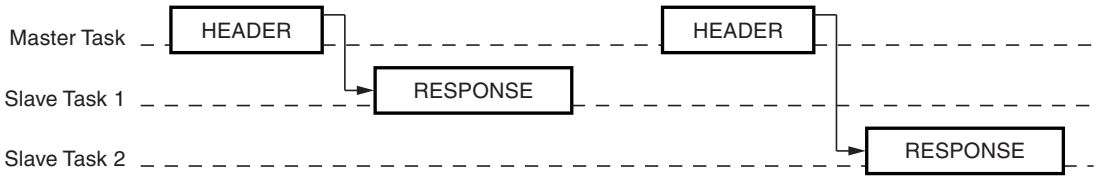



The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transported by each frame. Both the master task and the slave task are parts of the Frame handler

### 25.5.2 Frames

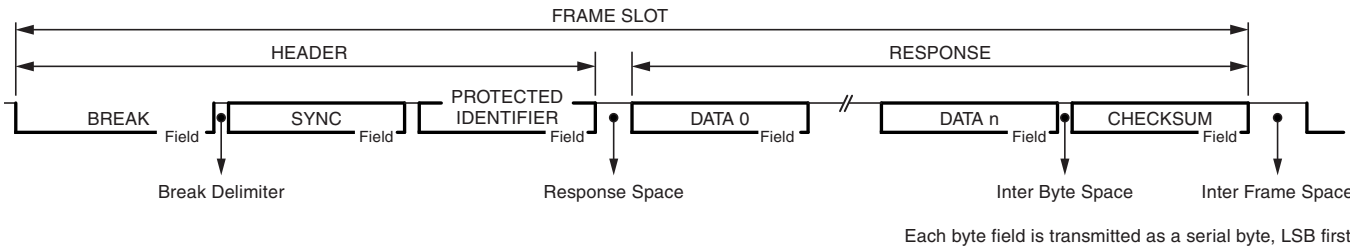
A frame consists of a header (provided by the master task) and a response (provided by a slave task). The header consists of a BREAK and SYNC pattern followed by a PROTECTED IDENTIFIER. The identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the identifier transmits it. The response consists of a DATA field and a CHECKSUM field.

**Figure 25-2. Master and Slave Tasks Behavior in LIN Frame**




The slave tasks waiting for the data associated with the identifier receives the response and uses the data transported after verifying the checksum.

**Figure 25-3. Structure of a LIN Frame**




Each byte field is transmitted as a serial byte, LSB first

### 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).

For LIN communication, only four interrupts need to be managed:

- LIDOK: New LIN identifier available,
- LRXOK: LIN response received,
- LTXOK: LIN response transmitted,
- LERR: LIN Error(s).

The wake-up management can be automated using the UART wake-up capability and a node sending a minimum of 5 low bits (0xF0) for LIN 2.1 and 8 low bits (0x80) for LIN 1.3. Pin change interrupt on LIN wake-up signal can be also used to exit the device of one of its sleep modes.

Extended frame identifiers 62 (0x3E) and 63 (0x3F) are reserved to allow the embedding of user-defined message formats and future LIN formats. The byte transfer mode offered by the UART will ensure the upwards compatibility of LIN slaves with accommodation of the LIN protocol.

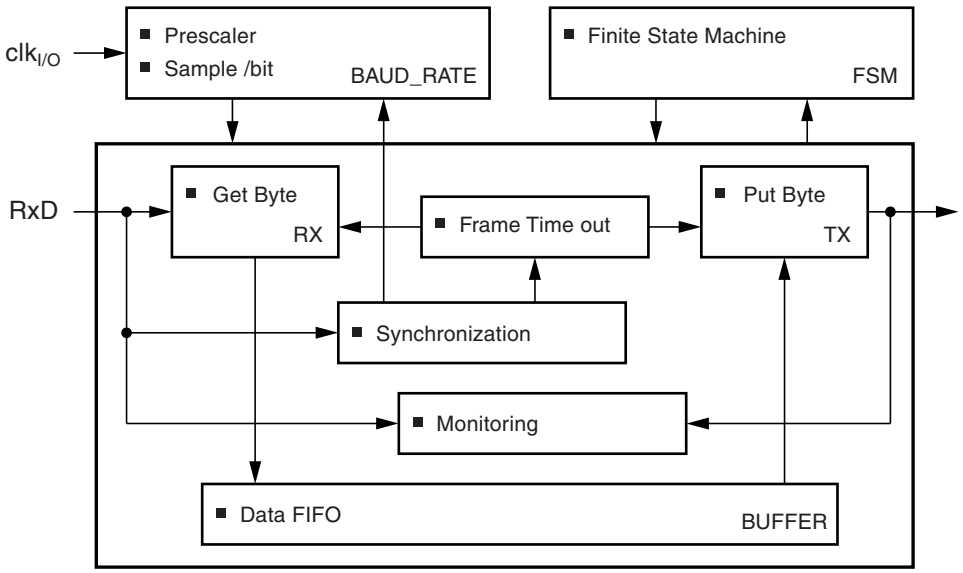
### 25.6.2 UART Overview

The LIN/UART controller can also function as a conventional UART. By default, the UART operates as a full duplex controller. It has local loop back circuitry for test purposes. The UART has the ability to buffer one character for transmit and two for receive. The receive buffer is made of one 8-bit serial register followed by one 8-bit independent buffer register. Automatic flag management is implemented when the application puts or gets characters, thus reducing the software overhead. Because transmit and receive services are independent, the user can save one device pin when one of the two services is not used. The UART has an enhanced baud rate generator providing a maximum error of 2% whatever the clock frequency and the targeted baud rate. The baud rate is given by:

$$\text{UART Baud Rate} = \frac{\text{LIN/UART Clock Freqency}}{(\text{LINBTR} \times (\text{LINDIV} + 1))}$$

### 25.6.3 LIN/UART Controller Structure

**Figure 25-4. LIN/UART Controller Block Diagram**



### 25.6.4 LIN/UART Command Overview

**Figure 25-5. LIN/UART Command Dependencies**

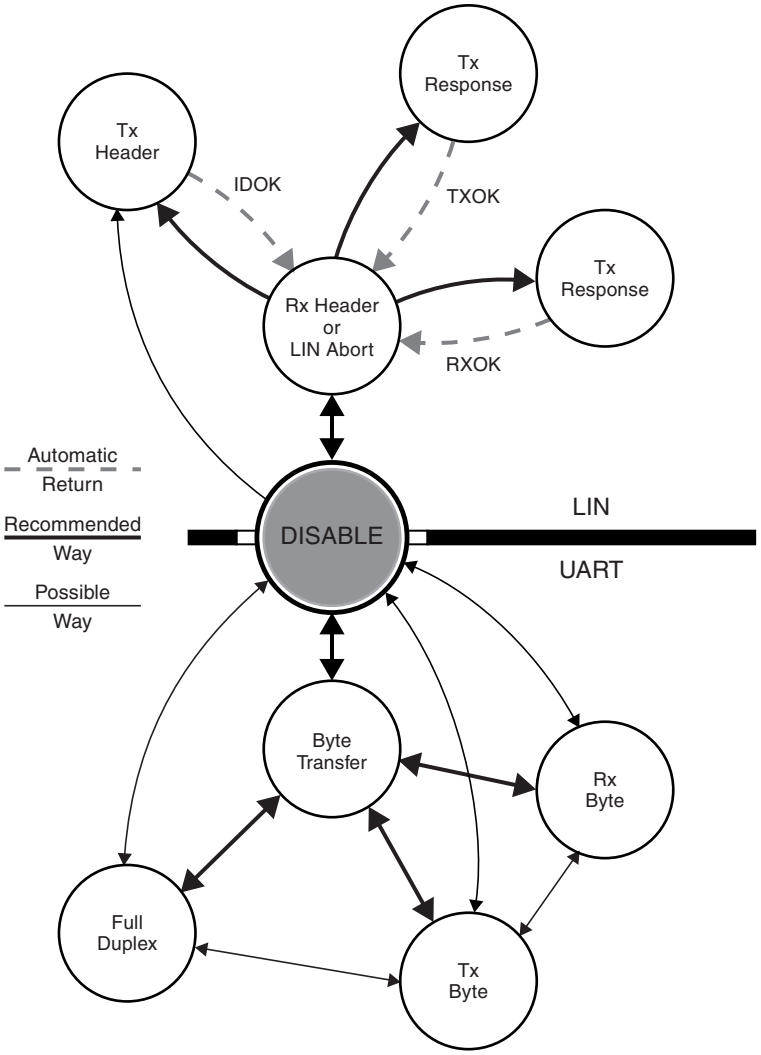



**Table 25-1. LIN/UART Command List**

| LENA | LCMD[2] | LCMD[1] | LCMD[0] | Command | Comment |
|---|---|---|---|---|---|
| 0 | x | x | x | Disable peripheral | |
| 1 | 0 | 0 | 0 | Rx Header - LIN abort | LIN withdrawal |
| | | | 1 | Tx Header | LCMD[2..0]=000 after Tx |
| | | 1 | 0 | Rx Response | LCMD[2..0]=000 after Rx |
| | | | 1 | Tx Response | LCMD[2..0]=000 after Tx |
| | 1 | 0 | 0 | Byte transfer | no CRC, no Time out<br>LTXDL=LRXDL=0<br>(LINDLR: read only register) |
| | | | 1 | Rx Byte | |
| | | 1 | 0 | Tx Byte | |
| | | | 1 | Full duplex | |

### 25.6.5 Enable/Disable

Setting the LENA bit in LINCR register enables the LIN/UART controller. To disable the LIN/UART controller, LENA bit must be written to 0. No wait states are implemented, so, the disable command is taken into account immediately.

### 25.6.6 LIN Commands

Clearing the LCMD[2] bit in LINCR register enables LIN commands.

As shown in Table 25-1 on page 120, four functions controlled by the LCMD[1..0] bits of LINCR register are available (c.f. Figure 25-5 on page 120).

#### 25.6.6.1 Rx Header/LIN Abort Function

This function (or state) is mainly the withdrawal mode of the controller.

When the controller has to execute a master task, this state is the start point before enabling a Tx Header command.

When the controller has only to execute slave tasks, LIN header detection/acquisition is enabled as background function. At the end of such an acquisition (Rx Header function), automatically the appropriate flags are set, and in LIN 1.3, the LINDLR register is set with the uncoded length value.

This state is also the start point before enabling the Tx or the Rx Response command.

A running function (i.e. Tx Header, Tx or Rx Response) can be aborted by clearing LCMD[1..0] bits in LINCR register. In this case, an abort flag - LABORT - in LINERR register will be set to inform the other software tasks. No wait states are implemented, so, the abort command is taken into account immediately.

*Rx Header* function is responsible for:

- The BREAK field detection,
- The hardware re-synchronization analyzing the SYNCH field,
- The reception of the PROTECTED IDENTIFIER field, the parity control and the update of the LINDLR register in case of LIN 1.3,
- The starting of the Frame_Time_Out,
- The checking of the LIN communication integrity.

#### 25.6.6.2 Tx Header Function

In accordance with the LIN protocol, only the master task must enable this function. The header is sent in the appropriate timed slots at the programmed baud rate (c.f. LINBRR and LINBTR registers).

The controller is responsible for:

- The transmission of the BREAK field - 13 dominant bits,
- The transmission of the SYNCH field - character 0x55,
- The transmission of the PROTECTED IDENTIFIER field. It is the full content of the LINIDR register (automatic check bits included).

At the end of this transmission, the controller automatically returns to *Rx Header / LIN Abort* state (i.e. LCMD[1..0] = 00) after setting the appropriate flags. This function leaves the controller in the same setting as after the *Rx Header* function. This means that, in LIN 1.3, the LINDLR register is set with the uncoded length value at the end of the *Tx Header* function.

During this function, the controller is also responsible for:

- The starting of the Frame_Time_Out,
- The checking of the LIN communication integrity.

#### 25.6.6.3 Rx and TX Response Functions

These functions are initiated by the slave task of a LIN node. They must be used after sending an header (master task) or after receiving an header (considered as belonging to the slave task). When the TX Response order is sent, the transmission begins. A Rx Response order can be sent up to the reception of the last serial bit of the first byte (before the stop-bit).

In LIN 1.3, the header slot configures the LINDLR register. In LIN 2.1, the user must configure the LINDLR register, either LRXDL[3..0] for *Rx Response* either LTXDL[3..0] for *Tx Response*.

When the command starts, the controller checks the LIN13 bit of the LINCR register to apply the right rule for computing the checksum. Checksum calculation over the DATA bytes and the PROTECTED IDENTIFIER byte is called enhanced checksum and it is used for communication with LIN 2.1 slaves. Checksum calculation over the DATA bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Note that identifiers 60 (0x3C) to 63 (0x3F) shall always use classic checksum.

At the end of this reception or transmission, the controller automatically returns to *Rx Header / LIN Abort* state (i.e. LCMD[1..0] = 00) after setting the appropriate flags.

If an LIN error occurs, the reception or the transmission is stopped, the appropriate flags are set and the LIN bus is left to recessive state.

During these functions, the controller is responsible for:

- The initialization of the checksum operator,
- The transmission or the reception of ‘n’ data with the update of the checksum calculation,
- The transmission or the checking of the CHECKSUM field,
- The checking of the Frame_Time_Out,
- The checking of the LIN communication integrity.

While the controller is sending or receiving a response, BREAK and SYNCH fields can be detected and the identifier of this new header will be recorded. Of course, specific errors on the previous response will be maintained with this identifier reception.

#### 25.6.6.4 Handling Data of LIN response

A FIFO data buffer is used for data of the LIN response. After setting all parameters in the LINSEL register, repeated accesses to the LINDAT register perform data read or data write (c.f. Section 25.7.13 “Data Management” on page 131).

Note that LRXDL[3..0] and LTXDL[3..0] are not linked to the data access.

### 25.6.7 UART Commands

Setting the LCMD[2] bit in LINENR register enables UART commands.

Tx Byte and Rx Byte services are independent as shown in Table 25-1 on page 120.

- Byte Transfer: the UART is selected but both Rx and Tx services are disabled,
- Rx Byte: only the Rx service is enable but Tx service is disabled,
- Tx Byte: only the Tx service is enable but Rx service is disabled,
- Full Duplex: the UART is selected and both Rx and Tx services are enabled.

This combination of services is controlled by the LCMD[1..0] bits of LINENR register (c.f. Figure 25-5 on page 120).

#### 25.6.7.1 Data Handling

The FIFO used for LIN communication is disabled during UART accesses. LRXDL[3..0] and LTXDL[3..0] values of LINDLR register are then irrelevant. LINDAT register is then used as data register and LINSEL register is not relevant.

#### 25.6.7.2Rx Service

Once this service is enabled, the user is warned of an in-coming character by the LRXOK flag of LINSIR register. Reading LINDAT register automatically clears the flag and makes free the second stage of the buffer. If the user considers that the in-coming character is irrelevant without reading it, he directly can clear the flag (see specific flag management described in Section 25.8.2 on page 133).

The intrinsic structure of the Rx service offers a 2-byte buffer. The fist one is used for serial to parallel conversion, the second one receives the result of the conversion. This second buffer byte is reached reading LINDAT register. If the 2-byte buffer is full, a new in-coming character will overwrite the second one already recorded. An OVRERR error in LINERR register will then accompany this character when read.

A FERR error in LINERR register will be set in case of framing error.

#### 25.6.7.3Tx Service

If this service is enabled, the user sends a character by writing in LINDAT register. Automatically the LTXOK flag of LINSIR register is cleared. It will rise at the end of the serial transmission. If no new character has to be sent, LTXOK flag can be cleared separately (see specific flag management described in Section 25.8.2 on page 133).

There is no transmit buffering.

No error is detected by this service.

## 25.7 LIN / UART Description

### 25.7.1 Reset

The AVR core reset logic signal also resets the LIN/UART controller. Another form of reset exists, a software reset controlled by LSWRES bit in LINCR register. This self-reset bit performs a partial reset as shown in Table 25-2.

**Table 25-2. Reset of LIN/UART Registers**

| Register | Name | Reset Value | LSWRES Value | Comment |
|---|---|---|---|---|
| LIN Control Reg. | LINCR | 0000 0000 $_b$ | 0000 0000 $_b$ | x=unknown<br>u=unchanged |
| LIN Status and Interrupt Reg. | LINSIR | 0000 0000 $_b$ | 0000 0000 $_b$ | |
| LIN Enable Interrupt Reg. | LINENIR | 0000 0000 $_b$ | xxxx 0000 $_b$ | |
| LIN Error Reg. | LINERR | 0000 0000 $_b$ | 0000 0000 $_b$ | |
| LIN Bit Timing Reg. | LINBTR | 0010 0000 $_b$ | 0010 0000 $_b$ | |
| LIN Baud Rate Reg. Low | LINBRRL | 0000 0000 $_b$ | uuuu uuuu $_b$ | |
| LIN Baud Rate Reg. High | LINBRRH | 0000 0000 $_b$ | xxxx uuuu $_b$ | |
| LIN Data Length Reg. | LINDLR | 0000 0000 $_b$ | 0000 0000 $_b$ | |
| LIN Identifier Reg. | LINIDR | 1000 0000 $_b$ | 1000 0000 $_b$ | |
| LIN Data Buffer Selection | LINSEL | 0000 0000 $_b$ | xxxx 0000 $_b$ | |
| LIN Data | LINDAT | 0000 0000 $_b$ | 0000 0000 $_b$ | |

### 25.7.2 LIN Protocol Selection

LIN13 bit in LINCR register is used to select the LIN protocol:

- LIN13 = 0 (default): LIN 2.1 protocol,
- LIN13 = 1: LIN 1.3 protocol.

The controller checks the LIN13 bit in computing the checksum (enhanced checksum in LIN2.1 / classic checksum in LIN 1.3).

This bit is irrelevant for UART commands.

### 25.7.3 Configuration

Depending on the mode (LIN or UART), LCONF[1..0] bits of the LINCR register set the controller in the following configuration (Table 25-3):

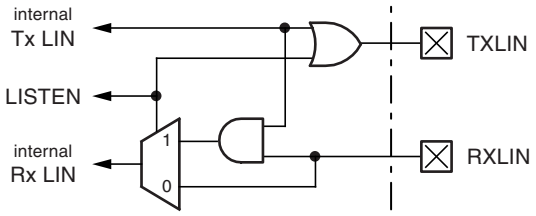
**Table 25-3. Configuration Table versus Mode**

| Mode | LCONF[1..0] | Configuration |
|---|---|---|
| LIN | $00_b$ | LIN standard configuration (default) |
| | $01_b$ | No CRC field detection or transmission |
| | $10_b$ | Frame_Time_Out disable |
| | $11_b$ | Listening mode |
| UART | $00_b$ | 8-bit data, no parity and 1 stop-bit |
| | $01_b$ | 8-bit data, even parity and 1 stop-bit |
| | $10_b$ | 8-bit data, odd parity and 1 stop-bit |
| | $11_b$ | Listening mode, 8-bit data, no parity and 1 stop-bit |

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

**Figure 25-6. Listening Mode**

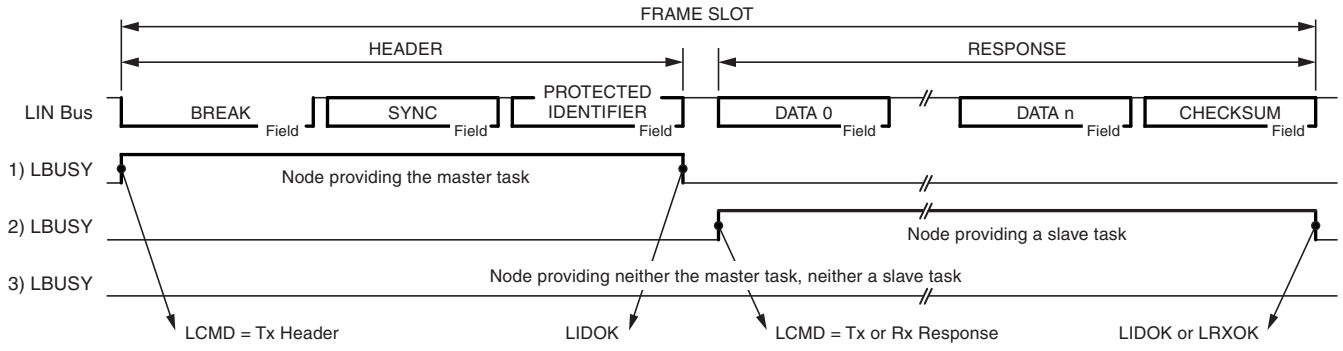



### 25.7.4 Busy Signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

#### 25.7.4.1 Busy Signal in LIN Mode

**Figure 25-7. Busy Signal in LIN Mode**



When the busy signal is set, some registers are locked, user writing is not allowed:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA and LSWRES,
- “LIN Baud Rate Registers” - LINBRRL and LINBRRH,
- “LIN Data Length Register” - LINDLR,
- “LIN Identifier Register” - LINIDR,
- “LIN Data Register” - LINDAT.

If the busy signal is set, the only available commands are:

- LCMD[1..0] = $00_b$, the abort command is taken into account at the end of the byte,
- LENA = 0 and/or LCMD[2] = 0, the kill command is taken into account immediately,
- LSWRES = 1, the reset command is taken into account immediately.

Note that, if another command is entered during busy signal, the new command is not validated and the LOVRERR bit flag of the LINERR register is set. The on-going transfer is not interrupted.

#### 25.7.4.2 Busy Signal in UART Mode

During the byte transmission, the busy signal is set. This locks some registers from being written:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA and LSWRES,
- “LIN Data Register” - LINDAT.

The busy signal is not generated during a byte reception.

### 25.7.5 Bit Timing

#### 25.7.5.1 Baud rate Generator

The baud rate is defined to be the transfer rate in bits per second (bps):

- BAUD: Baud rate (in bps),
- $fclk_{i/o}$: System I/O clock frequency,
- LDIV[11..0]: Contents of LINBRRH and LINBRRL registers - (0-4095), the pre-scaler receives $clk_{i/o}$ as input clock.
- LBT[5..0]: Least significant bits of - LINBTR register- (0-63) is the number of samplings in a LIN or UART bit (default value 32).

Equation for calculating baud rate:

$$BAUD = fclk_{i/o} / LBT[5..0] \times (LDIV[11..0] + 1)$$

Equation for setting LINDIV value:

$$LDIV[11..0] = ( fclk_{i/o} / LBT[5..0] \times BAUD ) - 1$$

Note that in reception a majority vote on three samplings is made.

#### 25.7.5.2 Re-synchronization in LIN Mode

When waiting for Rx Header, LBT[5..0] = 32 in LINBTR register. The re-synchronization begins when the BREAK is detected. If the BREAK size is not in the range (11 bits min., 28 bits max. — 13 bits nominal), the BREAK is refused. The re-synchronization is done by adjusting LBT[5..0] value to the SYNCH field of the received header (0x55). Then the PROTECTED IDENTIFIER is sampled using the new value of LBT[5..0]. The re-synchronization implemented in the controller tolerates a clock deviation of ± 20% and adjusts the baud rate in a ± 2% range.

The new LBT[5..0] value will be used up to the end of the response. Then, the LBT[5..0] will be reset to 32 for the next header.

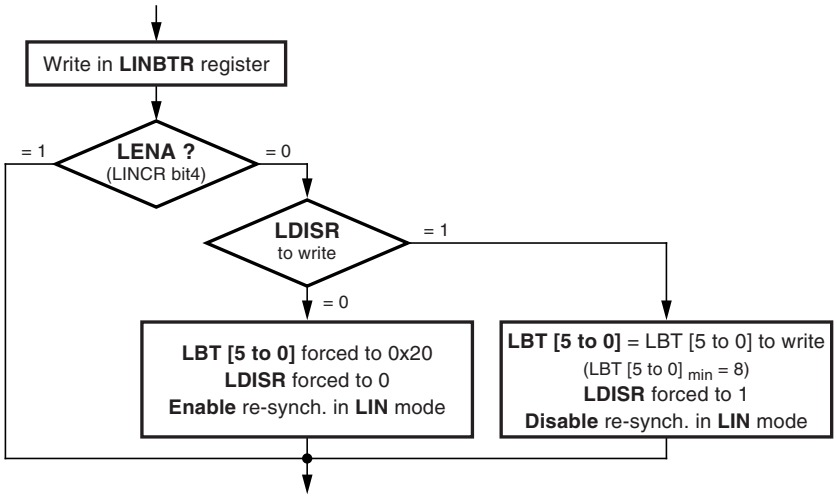
#### 25.7.5.3 Handling LBT[5..0]

LDISR bit of LINBTR register is used to:

- Disable the re-synchronization (for instance in the case of LIN MASTER node),
- To enable the setting of LBT[5..0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5..0] due to the sampling operation.

Note that the LENA bit of LINCR register is important for this handling (see Figure 25-8 on page 126).

**Figure 25-8. Handling LBT[5..0]**




### 25.7.6 Data Length

Section 25.6.6 “LIN Commands” on page 121 describes how to set or how are automatically set the LRXDL[3..0] or LTXDL[3..0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx Response the LRXDL[3..0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx Response the LTXDL[3..0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

#### 25.7.6.1 Data Length in LIN 2.1

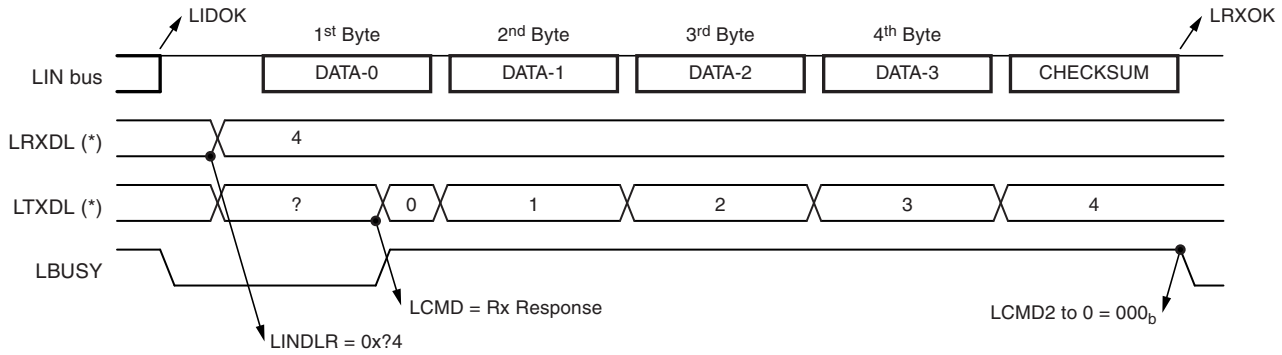
- If LTXDL[3..0]=0 only the CHECKSUM will be sent,
- If LRXDL[3..0]=0 the first byte received will be interpreted as the CHECKSUM,
- If LTXDL[3..0] or LRXDL[3..0] >8, values will be forced to 8 after the command setting and before sending or receiving of the first byte.

#### 25.7.6.2 Data Length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL).
- Via the above mechanism, a length of 0 or >8 is not possible.

### 25.7.6.3Data Length in Rx Response

**Figure 25-9. LIN2.1 - Rx Response - No Error**

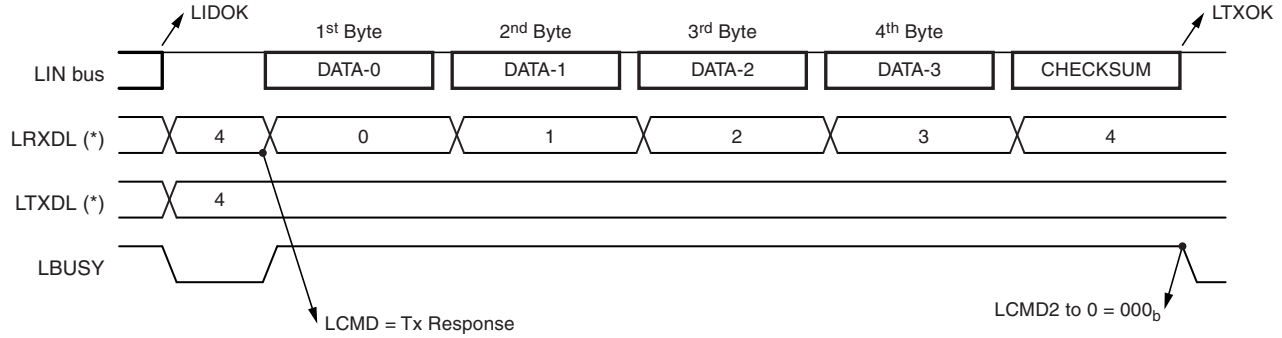



(*): LRXDL and LTXDL updated by user

- The user initializes LRXDL field before setting the Rx Response command,
- After setting the Rx Response command, LTXDL is reset by hardware,
- LRXDL field will remain unchanged during Rx (during busy signal),
- LTXDL field will count the number of received bytes (during busy signal),
- If an error occurs, Rx stops, the corresponding error flag is set and LTXDL will give the number of received bytes without error,
- If no error occurs, LRXOK is set after the reception of the CHECKSUM, LRXDL will be unchanged (and LTXDL = LRXDL).

### 25.7.6.4Data Length in Tx Response

**Figure 25-10.LIN1.3 - Tx Response - No Error**

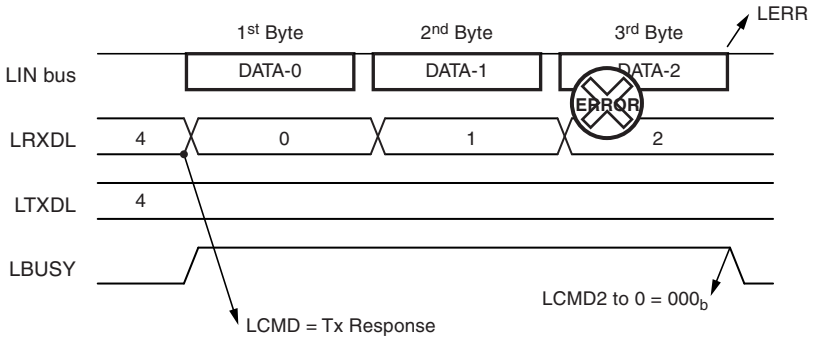



(*): LRXDL and LTXDL updated by Rx Response or Tx Response task

- The user initializes LTXDL field before setting the Tx Response command,
- After setting the Tx Response command, LRXDL is reset by hardware,
- LTXDL will remain unchanged during Tx (during busy signal),
- LRXDL will count the number of transmitted bytes (during busy signal),
- If an error occurs, Tx stops, the corresponding error flag is set and LRXDL will give the number of transmitted bytes without error,
- If no error occurs, LTXOK is set after the transmission of the CHECKSUM, LTXDL will be unchanged (and LRXDL = LTXDL).

### 25.7.6.5Data Length after Error

**Figure 25-11.Tx Response - Error**




Note: Information on response (ex: error on byte) is only available at the end of the serialization/de-serialization of the byte.

### 25.7.6.6Data Length in UART Mode

- The UART mode forces LRXDL and LTXDL to 0 and disables the writing in LINDLR register,
- Note that after reset, LRXDL and LTXDL are also forced to 0.

## 25.7.7 xxOK Flags

There are three xxOK flags in LINSIR register:

- LIDOK: LIN IDentifier OK
  It is set at the end of the header, either by the Tx Header function or by the Rx Header. In LIN 1.3, before generating LIDOK, the controller updates the LRXDL and LTXDL fields in LINDLR register.
  It is not driven in UART mode.
- LRXOK: LIN RX response complete
  It is set at the end of the response by the Rx Response function in LIN mode and once a character is received in UART mode.
- LTXOK: LIN TX response complete
  It is set at the end of the response by the Tx Response function in LIN mode and once a character has been sent in UART mode.

These flags can generate interrupts if the corresponding enable interrupt bit is set in the LINENIR register (see Section 25.7.11 “Interrupts” on page 130).

### 25.7.8 xxERR Flags

LERR bit of the LINSIR register is an logical ‘OR’ of all the bits of LINERR register (see Section 25.7.11 “Interrupts” on page 130). There are eight flags:

- LBERR = LIN Bit ERRor.
  A unit that is sending a bit on the bus also monitors the bus. A LIN bit error will be flagged when the bit value that is monitored is different from the bit value that is sent. After detection of a LIN bit error the transmission is aborted.
- LCERR = LIN Checksum ERRor.
  A LIN checksum error will be flagged if the inverted modulo-256 sum of all received data bytes (and the protected identifier in LIN 2.1) added to the checksum does not result in 0xFF.
- LPERR = LIN Parity ERRor (identifier).
  A LIN parity error in the IDENTIFIER field will be flagged if the value of the parity bits does not match with the identifier value. (See LP[1:0] bits in Section 25.8.8 “LINIDR – LIN Identifier Register” on page 136). A LIN slave application does not distinguish between corrupted parity bits and a corrupted identifier. The hardware does not undertake any correction. However, the LIN slave application has to solve this as:
  - known identifier (parity bits corrupted),
  - or corrupted identifier to be ignored,
  - or new identifier.
- LSERR = LIN Synchronization ERRor.
  A LIN synchronization error will be flagged if a slave detects the edges of the SYNCH field outside the given tolerance.
- LFERR = LIN Framing ERRor.
  A framing error will be flagged if dominant STOP bit is sampled.
  Same function in UART mode.
- LTOERR = LIN Time Out ERRor.
  A time-out error will be flagged if the MESSAGE frame is not fully completed within the maximum length $T_{Frame_Maximum}$ by any slave task upon transmission of the SYNCH and IDENTIFIER fields (see Section 25.7.9 “Frame Time Out” on page 129).
- LOVERR = LIN OVerrun ERRor.
  Overrun error will be flagged if a new command (other than LIN Abort) is entered while ‘Busy signal’ is present.
  In UART mode, an overrun error will be flagged if a received byte overwrites the byte stored in the serial input buffer.
- LABORT
  LIN abort transfer reflects a previous LIN Abort command (LCMD[2..0] = 000) while ‘Busy signal’ is present.

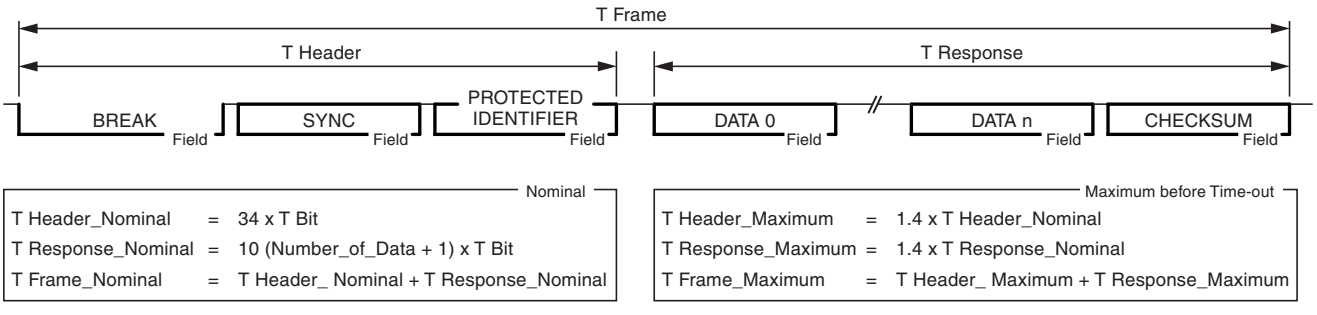
After each LIN error, the LIN controller stops its previous activity and returns to its withdrawal mode (LCMD[2..0] = $000_b$) as illustrated in Figure 25-11 on page 128.

Writing 1 in LERR of LINSIR register resets LERR bit and all the bits of the LINERR register.

### 25.7.9 Frame Time Out

According to the LIN protocol, a frame time-out error is flagged if: $\mathbf{T_{Frame} > T_{Frame_Maximum}}$. This feature is implemented in the LIN/UART controller.

**Figure 25-12.LIN Timing and Frame Time-out**



### 25.7.10 Checksum

The last field of a frame is the checksum.

In LIN 2.1, the checksum contains the inverted eight bit sum with carry over all data bytes and the protected identifier. This calculation is called enhanced checksum.

$$\text{CHECKSUM} = 255 - \left( \text{unsigned char}\left( \left( \sum_{0}^{n} \text{DATA}_n \right) + \text{PROTECTED ID.} \right) + \text{unsigned char}\left( \left( \left( \sum_{0}^{n} \text{DATA}_n \right) + \text{PROTECTED ID.} \right) \gg 8 \right) \right)$$

In LIN 1.3, the checksum contains the inverted eight bit sum with carry over all data bytes. This calculation is called classic checksum.

$$\text{CHECKSUM} = 255 - \left( \text{unsigned char}\left( \sum_{0}^{n} \text{DATA}_n \right) + \text{unsigned char}\left( \left( \sum_{0}^{n} \text{DATA}_n \right) \gg 8 \right) \right)$$

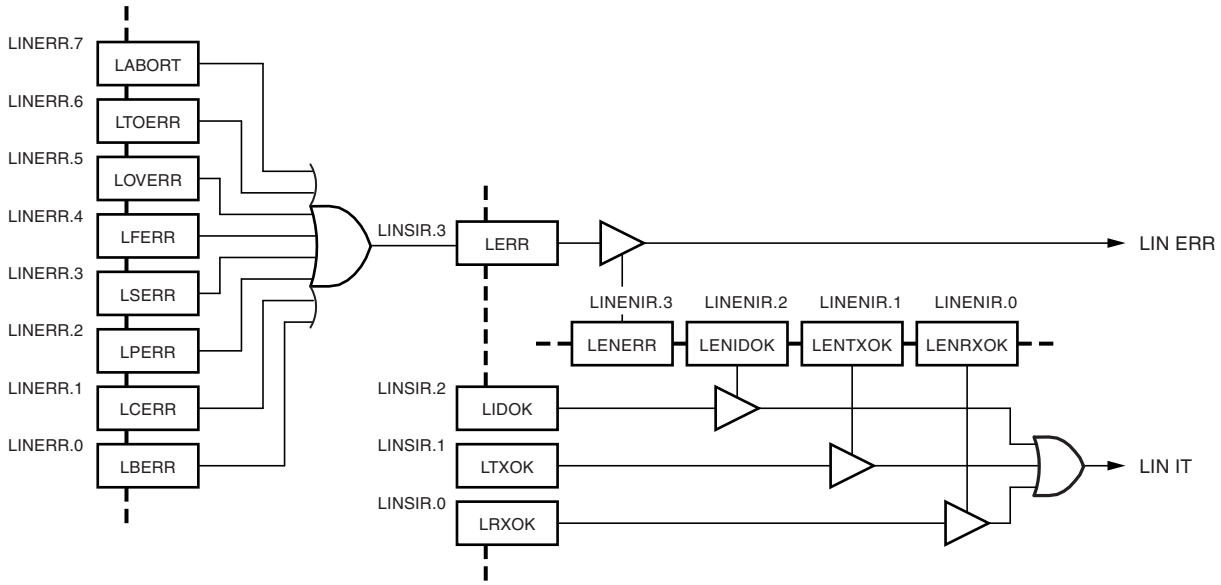
Frame identifiers 60 (0x3C) to 61 (0x3D) shall always use classic checksum

### 25.7.11 Interrupts

As shown in Figure 25-13 on page 130, the four communication flags of the LINSIR register are combined to drive two interrupts. Each of these flags have their respective enable interrupt bit in LINENIR register.

(see Section 25.7.7 “xxOK Flags” on page 128 and Section 25.7.8 “xxERR Flags” on page 129).

**Figure 25-13.LIN Interrupt Mapping**



### 25.7.12 Message Filtering

Message filtering based upon the whole identifier is not implemented. Only a status for frame headers having 0x3C, 0x3D, 0x3E and 0x3F as identifier is available in the LINSIR register.

**Table 25-4. Frame Status**

| LIDST[2..0] | Frame Status |
|---|---|
| $0xx_b$ | No specific identifier |
| $100_b$ | 60 (0x3C) identifier |
| $101_b$ | 61 (0x3D) identifier |
| $110_b$ | 62 (0x3E) identifier |
| $111_b$ | 63 (0x3F) identifier |

The LIN protocol says that a message with an identifier from 60 (0x3C) up to 63 (0x3F) uses a classic checksum (sum over the data bytes only). Software will be responsible for switching correctly the LIN13 bit to provide/check this expected checksum (the insertion of the ID field in the computation of the CRC is set - or not - just after entering the Rx or Tx Response command).

### 25.7.13 Data Management

#### 25.7.13.1LIN FIFO Data Buffer

To preserve register allocation, the LIN data buffer is seen as a FIFO (with address pointer accessible). This FIFO is accessed via the LINDX[2..0] field of LINSEL register through the LINDAT register.

LINDX[2..0], the data index, is the address pointer to the required data byte. The data byte can be read or written. The data index is automatically incremented after each LINDAT access if the LAINC (active low) bit is cleared. A roll-over is implemented, after data index=7 it is data index=0. Otherwise, if LAINC bit is set, the data index needs to be written (updated) before each LINDAT access.

The first byte of a LIN frame is stored at the data index=0, the second one at the data index=1, and so on. Nevertheless, LINSEL must be initialized by the user before use.

#### 25.7.13.2UART Data Register

The LINDAT register is the data register (no buffering - no FIFO). In write access, LINDAT will be for data out and in read access, LINDAT will be for data in.

In UART mode the LINSEL register is unused.

## 25.8 LIN / UART Register Description

### 25.8.1 LINCR – LIN Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC0) | LSWRES | LIN13 | LCONF1 | LCONF0 | LENA | LCMD2 | LCMD1 | LCMD0 | LINCR |
| 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 – LSWRES: Software Reset**
  - 0 = No action,
  - 1 = Software reset (this bit is self-reset at the end of the reset procedure).
- **Bit 6 – LIN13: LIN 1.3 mode**
  - 0 = LIN 2.1 (default),
  - 1 = LIN 1.3.
- **Bit 5:4 – LCONF[1:0]: Configuration**
  1. LIN mode (default = 00):
     - 00 = LIN Standard configuration (listen mode “off”, CRC “on” and Frame_Time_Out “on”,
     - 01 = No CRC, no Time out (listen mode “off”),
     - 10 = No Frame_Time_Out (listen mode “off” and CRC “on”),
     - 11 = Listening mode (CRC “on” and Frame_Time_Out “on”).
  2. UART mode (default = 00):
     - 00 = 8-bit, no parity (listen mode “off”),
     - 01 = 8-bit, even parity (listen mode “off”),
     - 10 = 8-bit, odd parity (listen mode “off”),
     - 11 = Listening mode, 8-bit, no parity.
- **Bit 3 – LENA: Enable**
  - 0 = Disable (both LIN and UART modes),
  - 1 = Enable (both LIN and UART modes).
- **Bit 2:0 – LCMD[2:0]: Command and mode**

  The command is only available if LENA is set.
  - 000 = LIN Rx Header - LIN abort,
  - 001 = LIN Tx Header,
  - 010 = LIN Rx Response,
  - 011 = LIN Tx Response,
  - 100 = UART Rx and Tx Byte disable,
  - 11x = UART Rx Byte enable,
  - 1x1 = UART Tx Byte enable.

### 25.8.2 LINSIR – LIN Status and Interrupt Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC1) | **LIDST2** | **LIDST1** | **LIDST0** | **LBUSY** | **LERR** | **LIDOK** | **LTXOK** | **LRXOK** | **LINSIR** |
| Read/Write | R | R | R | R | $R/W_{one}$ | $R/W_{one}$ | $R/W_{one}$ | $R/W_{one}$ | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:5 – LIDST[2:0]: Identifier Status**
  - 0xx = no specific identifier,
  - 100 = Identifier 60 (0x3C),
  - 101 = Identifier 61 (0x3D),
  - 110 = Identifier 62 (0x3E),
  - 111 = Identifier 63 (0x3F).
- **Bit 4 – LBUSY: Busy Signal**
  - 0 = Not busy,
  - 1 = Busy (receiving or transmitting).
- **Bit 3 – LERR: Error Interrupt**

  It is a logical OR of LINERR register bits. This bit generates an interrupt if its respective enable bit - LENERR - is set in LINENIR.
  - 0 = No error,
  - 1 = An error has occurred.

  The user clears this bit by writing 1 in order to reset this interrupt. Resetting LERR also resets all LINERR bits.

  In UART mode, this bit is also cleared by reading LINDAT.

- **Bit 2 – LIDOK: Identifier Interrupt**

  This bit generates an interrupt if its respective enable bit - LENIDOK - is set in LINENIR.
  - 0 = No identifier,
  - 1 = Slave task: Identifier present, master task: Tx Header complete.

  The user clears this bit by writing 1, in order to reset this interrupt.
- **Bit 1 – LTXOK: Transmit Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENTXOK - is set in LINENIR.
  - 0 = No Tx,
  - 1 = Tx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by writing LINDAT.
- **Bit 0 – LRXOK: Receive Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENRXOK - is set in LINENIR.
  - 0 = No Rx
  - 1 = Rx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by reading LINDAT.

### 25.8.3 LINENIR – LIN Enable Interrupt Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC2) | – | – | – | – | LENERR | LENIDOK | LENTXOK | LENRXOK | LINENIR |
| 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 7:4 – Reserved Bits**
  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINENIR is written.
- **Bit 3 – LENERR: Enable Error Interrupt**
  - 0 = Error interrupt masked,
  - 1 = Error interrupt enabled.
- **Bit 2 – LENIDOK: Enable Identifier Interrupt**
  - 0 = Identifier interrupt masked,
  - 1 = Identifier interrupt enabled.
- **Bit 1 – LENTXOK: Enable Transmit Performed Interrupt**
  - 0 = Transmit performed interrupt masked,
  - 1 = Transmit performed interrupt enabled.
- **Bit 0 – LENRXOK: Enable Receive Performed Interrupt**
  - 0 = Receive performed interrupt masked,
  - 1 = Receive performed interrupt enabled.

### 25.8.4 LINERR – LIN Error Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC3) | LABORT | LTOERR | LOVERR | LFERR | LSERR | LPERR | LCERR | LBERR | LINERR |
| Read/Write | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – LABORT: Abort Flag**
  - 0 = No warning,
  - 1 = LIN abort command occurred.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 6 – LTOERR: Frame_Time_Out Error Flag**
  - 0 = No error,
  - 1 = Frame_Time_Out error.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 5 – LOVERR: Overrun Error Flag**
  - 0 = No error,
  - 1 = Overrun error.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 4 – LFERR: Framing Error Flag**
  - 0 = No error,
  - 1 = Framing error.

  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 3 – LSERR: Synchronization Error Flag**
  - 0 = No error,
  - 1 = Synchronization error.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 2 – LPERR: Parity Error Flag**
  - 0 = No error,
  - 1 = Parity error.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 1 – LCERR: Checksum Error Flag**
  - 0 = No error,
  - 1 = Checksum error.

  This bit is cleared when LERR bit in LINSIR is cleared.
- **Bit 0 – LBERR: Bit Error Flag**
  - 0 = no error,
  - 1 = Bit error.

  This bit is cleared when LERR bit in LINSIR is cleared.

### 25.8.5 LINBTR – LIN Bit Timing Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC4) | **LDISR** | – | **LBT5** | **LBT4** | **LBT3** | **LBT2** | **LBT1** | **LBT0** | **LINBTR** |
| Read/Write | R/W | R | R/(W) | R/(W) | R/(W) | R/(W) | R/(W) | R/(W) | |
| Initial Value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – LDISR: Disable Bit Timing Resynchronization**
  - 0 = Bit timing re-synchronization enabled (default),
  - 1 = Bit timing re-synchronization disabled.
- **Bits 5:0 – LBT[5:0]: LIN Bit Timing**

  Gives the number of samples of a bit.
  sample-time = (1 / $fclk_{i/o}$ ) x (LDIV[11..0] + 1)
  Default value: LBT[6:0]=32 — Min. value: LBT[6:0]=8 — Max. value: LBT[6:0]=63

### 25.8.6 LINBRR – LIN Baud Rate Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC5) | **LDIV7** | **LDIV6** | **LDIV5** | **LDIV4** | **LDIV3** | **LDIV2** | **LDIV1** | **LDIV0** | **LINBRRL** |
| (0xC6) | – | – | – | – | **LDIV11** | **LDIV10** | **LDIV9** | **LDIV8** | **LINBRRH** |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| 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 15:12 – Reserved**

  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINBRR is written.
- **Bits 11:0 – LDIV[11:0]: Scaling of $clk_{i/o}$ Frequency**

  The LDIV value is used to scale the entering $clk_{i/o}$ frequency to achieve appropriate LIN or UART baud rate.

### 25.8.7 LINDLR – LIN Data Length Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC7) | LTXDL3 | LTXDL2 | LTXDL1 | LTXDL0 | LRXDL3 | LRXDL2 | LRXDL1 | LRXDL0 | LINDLR |
| 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 7:4 – LTXDL[3:0]: LIN Transmit Data Length**
  In LIN mode, this field gives the number of bytes to be transmitted (clamped to 8 Max).
  In UART mode this field is unused.
- **Bits 3:0 – LRXDL[3:0]: LIN Receive Data Length**
  In LIN mode, this field gives the number of bytes to be received (clamped to 8 Max).
  In UART mode this field is unused.

### 25.8.8 LINIDR – LIN Identifier Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC8) | LP1 | LP0 | LID5 / LDL1 | LID4 / LDL0 | LID3 | LID2 | LID1 | LID0 | LINIDR |
| 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 7:6 – LP[1:0]: Parity**
  In LIN mode:

```
LP0 = LID4 ^ LID2 ^ LID1 ^ LID0
LP1 = ! ( LID1 ^ LID3 ^ LID4 ^ LID5 )
```

  In UART mode this field is unused.
- **Bits 5:4 – LDL[1:0]: LIN 1.3 Data Length**
  In LIN 1.3 mode:
  - 00 = 2-byte response,
  - 01 = 2-byte response,
  - 10 = 4-byte response,
  - 11 = 8-byte response.

  In UART mode this field is unused.
- **Bits 3:0 – LID[3:0]: LIN 1.3 Identifier**
  In LIN 1.3 mode: 4-bit identifier.
  In UART mode this field is unused.
- **Bits 5:0 – LID[5:0]: LIN 2.1 Identifier**
  In LIN 2.1 mode: 6-bit identifier (no length transported).
  In UART mode this field is unused.

### 25.8.9 LINSEL – LIN Data Buffer Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xC9) | – | – | – | – | LAINC | LINDX2 | LINDX1 | LINDX0 | LINSEL |
| Read/Write | – | – | – | – | R/W | R/W | R/W | R/W | |
| Initial Value | – | – | – | – | 0 | 0 | 0 | 0 | |

- **Bits 7:4 – Reserved Bits**

  These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINSEL is written.

- **Bit 3 – $\overline{\text{LAINC}}$: $\overline{\text{Auto Increment}}$ of Data Buffer Index**

  In LIN mode:

  - 0 = Auto incrementation of FIFO data buffer index (default),
  - 1 = No auto incrementation.

  In UART mode this field is unused.

- **Bits 2:0 – LINDX 2:0: FIFO LIN Data Buffer Index**

  In LIN mode: location (index) of the LIN response data byte into the FIFO data buffer. The FIFO data buffer is accessed through LINDAT.

  In UART mode this field is unused.

### 25.8.10 LINDAT – LIN Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xCA) | LDATA7 | LDATA6 | LDATA5 | LDATA4 | LDATA3 | LDATA2 | LDATA1 | LDATA0 | LINDAT |
| 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 – LDATA[7:0]: LIN Data In / Data out**

  In LIN mode: FIFO data buffer port.

  In UART mode: data register (no data buffer - no FIFO).

  - In Write access, data out.
  - In Read access, data in.

# 26. ADC - Analog to Digital Converter

## 26.1 Features

- Synchronous current and voltage ADC
- Configurable sample clock rate
  - 512kHz PLL or 128kHz slow RC oscillator
- Cascaded decimation with programmable settings
  - Instantaneous measurements with programmable output rate
  - Accumulated measurements with programmable output rate
- Programmable Chopper Mode to cancel offset on both current and voltage measurements
- Programmable gain for current ADC
- 7 selectable input channels for voltage ADC
- Diagnosis Mode

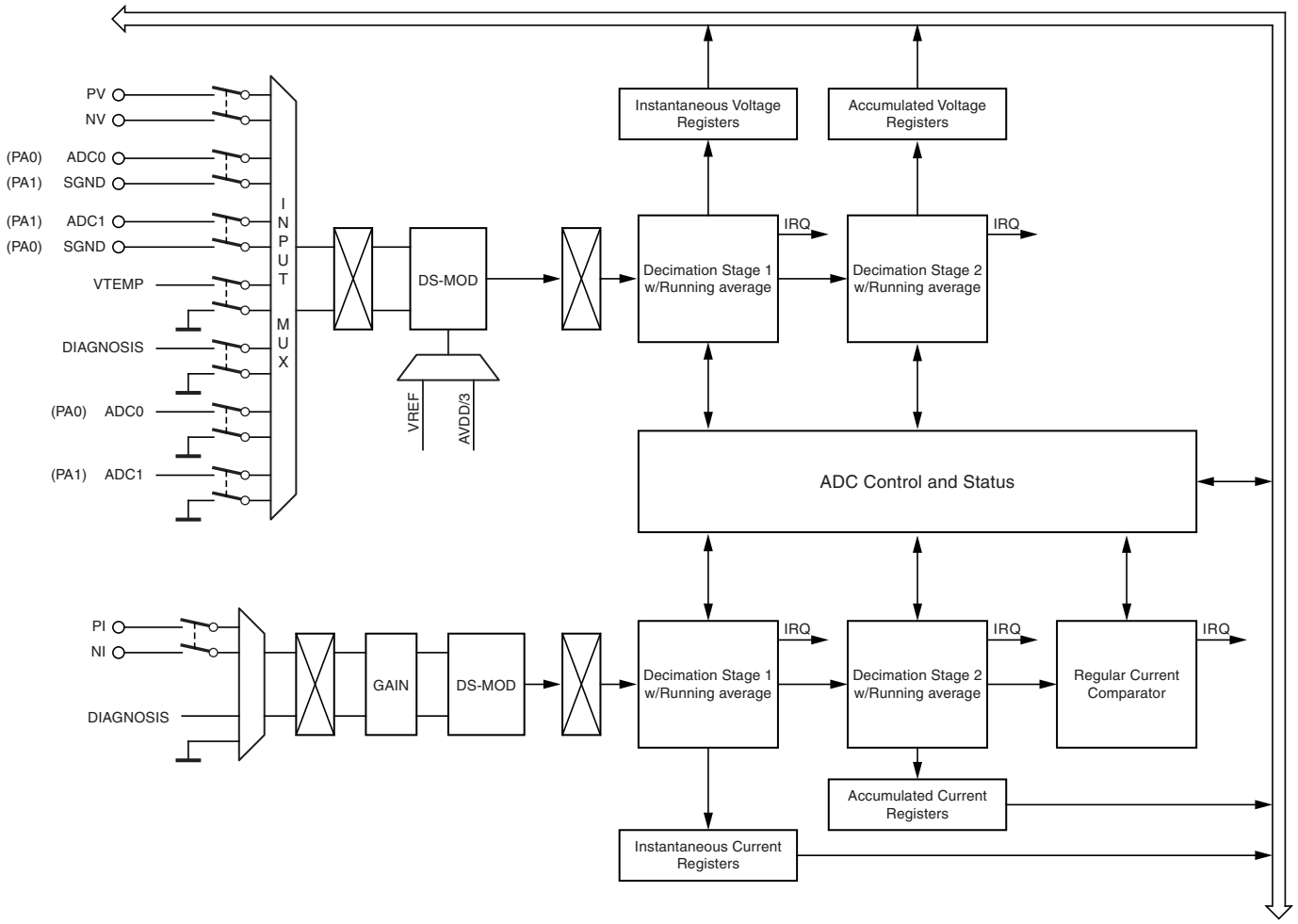
## 26.2 Overview

The Atmel® AVR MCU contains two separate ADCs, a Current ADC (C-ADC) and a Voltage ADC (V-ADC). The C-ADC is dedicated to measure current flowing through an external shunt resistor. The V-ADC is used to measure the battery terminal voltage, external temperature sensor or internal temperature sensor. Note that it also has an additional input for test intended for self diagnosis. An overview of the ADC system is illustrated in Figure 26-1 on page 139.

Both ADCs use Atmel's patented ADC architecture consisting of a Delta Sigma Modulator sampling the input, followed by 2 cascaded decimation filters which give both an Instantaneous Conversion result and an Accumulated Conversion result, trading accuracy with faster data rate. Both filters have programmable decimation ratios to be able to select suitable data rates in different operation modes. Decimation stage 1 outputs an Instantaneous Conversion result with data rate $F_{IC}$. This result is typically used for monitoring rapidly changing inputs with a shorter conversion time. Decimation stage 2 further accumulates the Instantaneous Conversion result, giving a mean average of the input over a longer time period. The Accumulated Conversion will output data with a low output data rate, $F_{AC}$. Further accumulation of the data result should be performed in software.

When both ADCs are enabled, they will sample synchronously. The CPU can enable/disable either ADC whenever it is not needed. To save power it is highly recommended that software disables the ADC whenever it is not used. To ensure synchronous operation, both ADCs have a common ADC controller. The ADC controller contains IO registers for CPU configuration and control. By writing to the ADC IO registers the CPU can enable/disable the ADC individually, configure the conversion ratios and Chopper mode for the ADCs, select input channel for the V-ADC and configure the Gain and Regular Current Detection mode for the C-ADC.

**Figure 26-1. ADC Overview**




Both the C-ADC and V-ADC include chopper functionality to automatically cancel offset. The chopper can be configured to run automatically or it can be controlled directly by software. When running in automatic settings, the chopper will automatically switch the input polarity on regular intervals and calculate a running average to remove the offset. The automatic chopping can be configured to follow the Instantaneous Current or the Accumulated Conversion Complete.

To allow measurements on a wide range of current inputs, the C-ADC has programmable gain settings.

The C-ADC also includes a Regular Current Comparator. With the Regular Current Comparator the system can be configured to enter a low power mode where it wakes up when current exceeds a configurable trigger level.

The V-ADC is connected to seven different sources through the Input Multiplexer. The PV2/NV2 pins are dedicated pins for measuring the scaled battery terminal voltage. ADC0/1 are two general purpose inputs which can be configured for different external configurations. In addition to the external channels there are two internal channels for internal temperature sensor measurement and diagnosis function.

## 26.3 Operation

### 26.3.1 Delta Sigma Modulator

The Delta Sigma modulator will perform oversampling and quantization of the input signal and do noise shaping of the quantization error.

The input for both the C-ADC and the V-ADC will be sampled at either the 512kHz PLL clock or the 128kHz Slow RC oscillator, depending on CKSEL IO bit in Section 26.6.3 “ADCRA - ADC Control Register A” on page 151.

To avoid aliasing when sampling the input, the input needs to be band limited by an external filter. Due to the high oversampling of the Delta Sigma Modulator a first order passive RC filter should be sufficient.

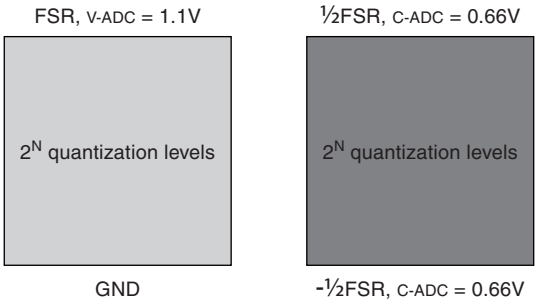
The ADCs use the VREF as reference voltage. For details on this voltage see Section 27. “Band Gap Reference and Temperature Sensor” on page 161. The Voltage Reference, VREF, is used to create the internal quantization range for the Delta Sigma modulator.

The V-ADC uses single ended sampling, and the internal quantization range is from GND to $FSR_{,V\text{-}ADC}$. The V-ADC has no internal gain for the external channels and will quantize the input relative to the Full-Scale Range, $FSR_{,V\text{-}ADC}$.

The C-ADC uses differential signaling to be able to measure both charge and discharge currents. The internal signal range of the Current ADC is $-½FSR_{,C\text{-}ADC}$ to $½FSR_{,C\text{-}ADC}$. The Current ADC has programmable gain and the input range depends on the gain settings. For details on gain settings, see Section 26.3.3 “Programmable Gain” on page 141.

The figure below illustrates the FSR of the V-ADC and the C-ADC

**Figure 26-2. Internal Signal Range with VREF = 1.1V**



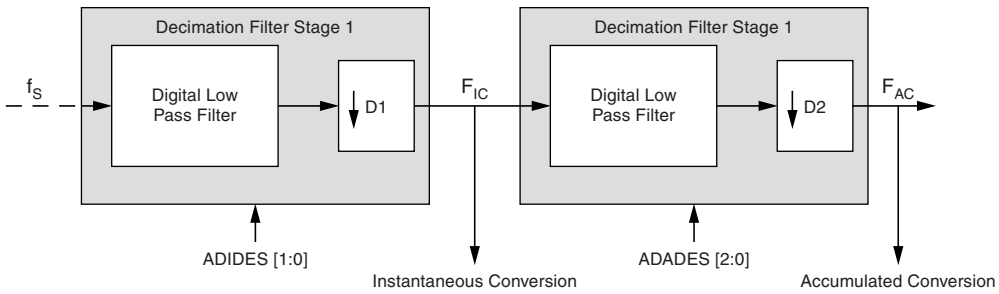

To ensure stable modulator operation, the input voltage/current should be limited to 90% of the modulator Full Scale Range (FSR).

Note that if the input voltage exceeds the allowable input range (FSR), the conversion register will saturate. The conversion will then give max or min values in the data registers.

### 26.3.2 Programmable Decimation Filters

The output of the Delta Sigma modulator contains a noise shaped, oversampled signal representing the input signal together with both in and out-of band components. In order to remove the out-of band noise, the digital ADC contains 2 cascaded decimation filters which will band limit the input with a low-pass filter before down-sampling the signal. The output of both filters can be read by the software.

**Figure 26-3. Decimation Filter**



Each filter has programmable decimation factors that can be adjusted by writing to the ADIDES[1:0] and ADADES[2:0] IO bits in Section 26.6.5 “ADCRC - ADC Control Register C” on page 153. Figure 26-3 shows the decimation filter and their conversion outputs. The first decimation filter will output an Instantaneous Data Conversion every 512, 256, 128 or 64 sampling cycle. The second decimation filter will output an Accumulated Conversion Result every 512, 256, 128, 64, 32, 16, 8 or 4th Instantaneous Conversion. The data output rate $F_{IC}$ and $F_{AC}$ for the Instantaneous and Accumulated Conversion output as a function of the decimation settings is given in the equation below.

$$F_{IC} = \frac{F_S}{64 \times 2^{ADIDES[1:0]}}$$

$$F_{AC} = \frac{F_{IC}}{2^{ADIADES[2:0]+2}}$$

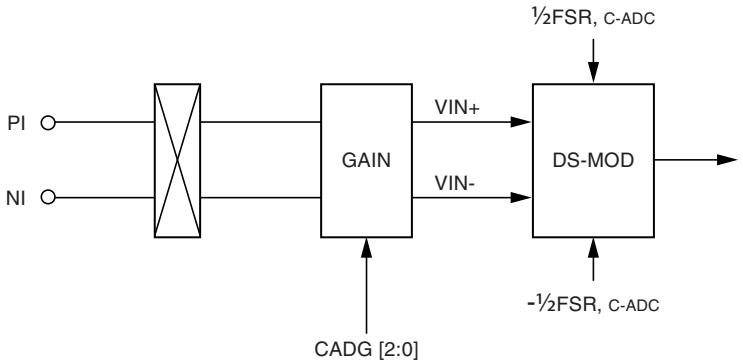
Note that when either Automatic Fast or Slow Chopping is enabled the data rate will be lower. For details on data rates when chopping is enabled, see Section 26.3.4 “Programmable Chopper Control” on page 142.

### 26.3.3 Programmable Gain

The C-ADC has programmable input gain settings to be able to measure a wide range of current inputs. When a small current is flowing the input can be scaled to fit the operation range of the ADC.

The input gain will adjust the signal at the input before it is sampled and quantized by the Delta Sigma modulator. The input gain has 7 programmable gain levels which can be selected by software by writing to the CADG2:0 bits. 4x, 8x, 16x, 32x, 64x, 128x or 256x can be selected.

**Figure 26-4. Analog Gain Stage at Input**




Note that the gain stage will saturate if the input exceeds the range of the modulator. The data result will then give max positive or min negative values in the data registers.

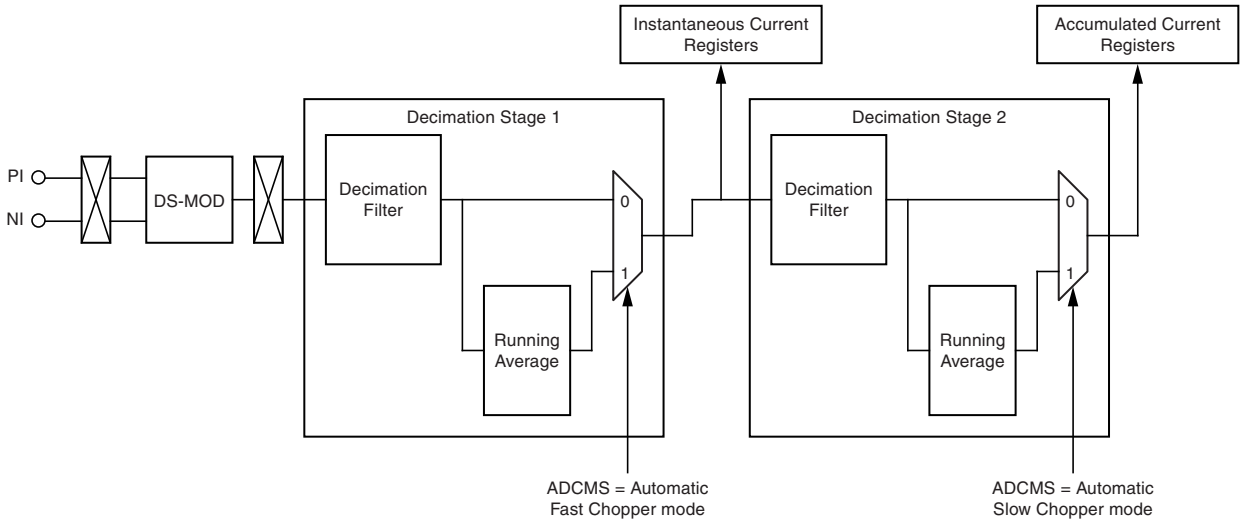
### 26.3.4 Programmable Chopper Control

Both the C-ADC and V-ADC have a chopper feature to cancel offset in the conversion data. The chopper can be configured by writing to the ADCMS1:0 IO bits in the Section 26.6.3 “ADCRA - ADC Control Register A” on page 151. If enabled, the chopper can be configured to run with in either

- Automatic Fast Chopper mode
- Automatic Slow Chopper mode
- Software Polarity Control mode

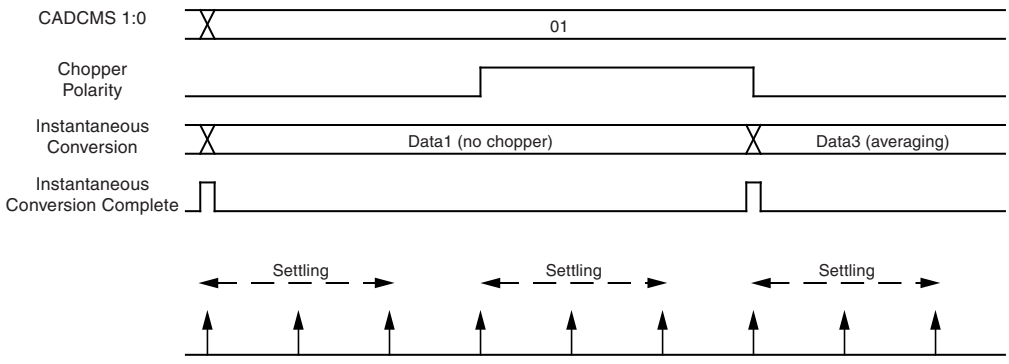
Figure 26-5 gives an overview of the chopper functionality.

**Figure 26-5. Chopper Overview**



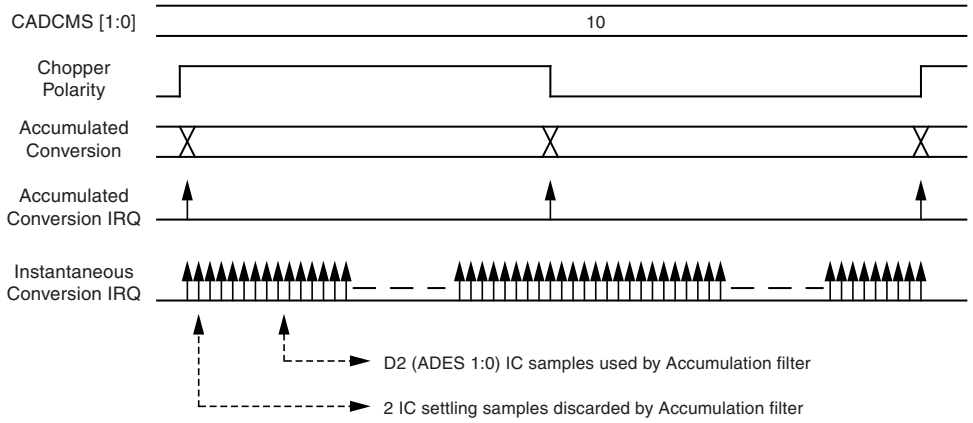

If selecting the Automatic Fast Chopper mode the chopper will switch the polarity of the input on each Instantaneous Conversion and calculate running averaging on the last two conversion results. In Automatic Fast Chopper mode the ADC will automatically perform two settling conversions before using the 3rd Instantaneous Conversion for Running Average, hence the Instantaneous Conversion data rate is reduced by 3x compared to having the Fast Chopper off. To get accurate Accumulation Conversion result the settling conversions in the Instantaneous filter are not used by the second decimation filter stage; hence the data rate in the Accumulated Conversion is also reduced by 3. The Fast Chopper timing is illustrated in Figure 26-6.

**Figure 26-6. Fast Chopper Timing**



If selecting Automatic Slow Chopper mode the chopper will switch polarity on each Accumulated Conversion and calculate running average on the last two conversion results. Note that when the chopper switches the polarity the Instantaneous filter will need 2 settling samples, these settling samples are automatically discarded at the input of the Accumulation filter reducing the data rate of the Accumulated Conversion with the time it takes to convert two Instantaneous Conversion values. The extra conversion delay is illustrated in Figure 26-7.

**Figure 26-7. Slow Chopper Timing**




If selecting Software Polarity Control mode, the CPU can select the chopper polarity by writing to the ADPSEL bit. In this mode software could change the polarity on regular intervals and do settling and running average in software. Note that the ADPSEL bit will be synchronized together with other control settings to the ADC. For details on synchronization, see Section 26.4.1 “Synchronization of Configuration Settings” on page 146.

When enabling Automatic Chopper mode, the first result will take twice as long since an running average value has to be calculated. This is also the case when changing between Automatic Fast and Slow Chopper mode.

**Table 26-1. Data Rates for Instantaneous (IC) and Accumulated (AC) Conversion(1)(3)**

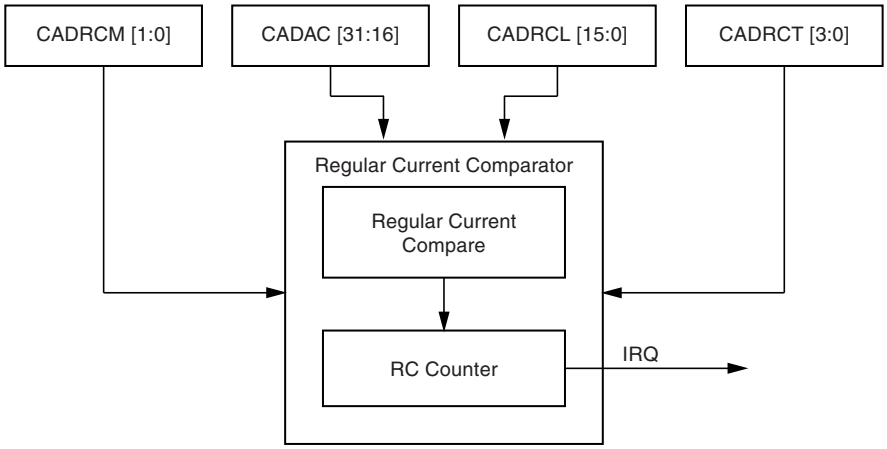
| Chopper mode | $F_{ic}$ | $F_{ac}$ |
|---|---|---|
| Auto fast chopper | $\frac{f_S}{3 \cdot ICDEC}$ | $\frac{f_S}{3 \cdot ICDEC \cdot ACDEC}$ |
| Auto slow chopper(2) | $\frac{f_S}{ICDEC}$ | $\frac{f_S}{(2 \cdot ICDEC) + (ICDEC \cdot ACDEC)}$ |
| No chopper | $\frac{f_S}{ICDEC}$ | $\frac{f_S}{ICDEC \cdot ACDEC}$ |

Notes:
1. Output sampling rate as function of decimation settings and input sampling rate, $f_S$.
2. Settling of Instantaneous is handled in hardware. After the chopper polarity is switched, 2 CADIC result is automatically discarded.
3. ICDEC represents the configured IC decimation ratio. ACDEC represents the configured AC decimation ratio.

### 26.3.5 Programmable Regulator Current Comparator

To be able to minimize CPU workload, the C-ADC can be configured to run with the Regular Current Comparator enabled. Enabling this feature allows the CPU to wake up only when the current is higher than a programmable threshold for a programmable number of samples.

**Figure 26-8. Regulator Current Comparator**




By writing to the Section 26.6.5 “ADCRC - ADC Control Register C” on page 153, Regular Current Comparator can either be enabled or disabled, and the Regular Current Counter mode of operation and timeout can be selected. The value of the comparator threshold can be configured by writing to the Section 26.6.10 “CADRCLH and CADRCLL - C-ADC Regulator Current Comparator Threshold Level” on page 157. If enabled, the Regular Current Comparator will increment the Regular Current Counter when the absolute value of the Accumulated Current measurement exceeds the programmable threshold level. If the Accumulated Current measurement goes below the threshold, the counter will either reset or decrement depending on the configuration. When the counter reaches the configured timeout level, the Regular Current Comparator can be configured to give a Regular Current Detection interrupt to the CPU.

Figure 26-9 illustrates the Regular Current Comparator timing with the counter configured to reset when going below the threshold level.

**Figure 26-9. Regulator Current Timing with Counter Reset**

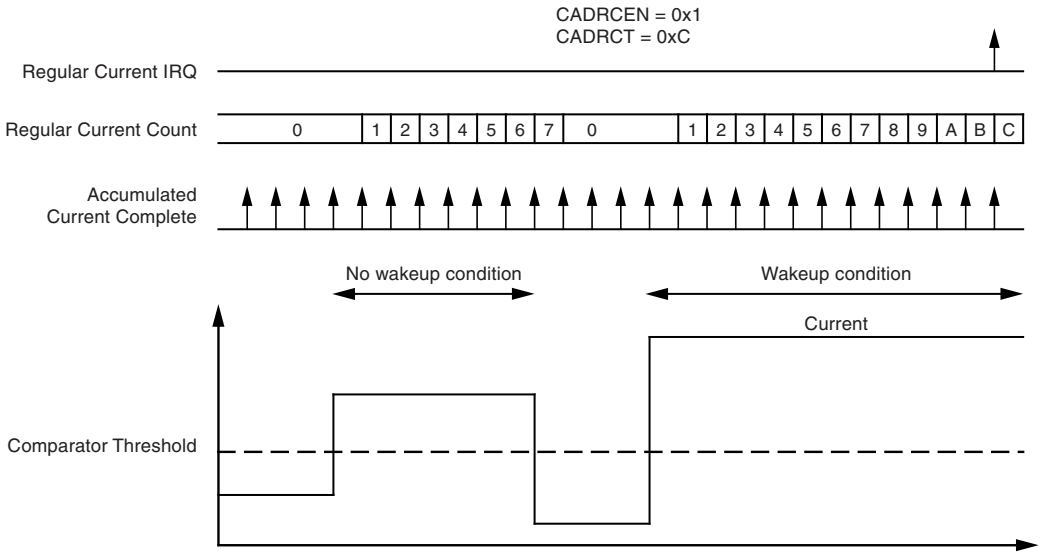
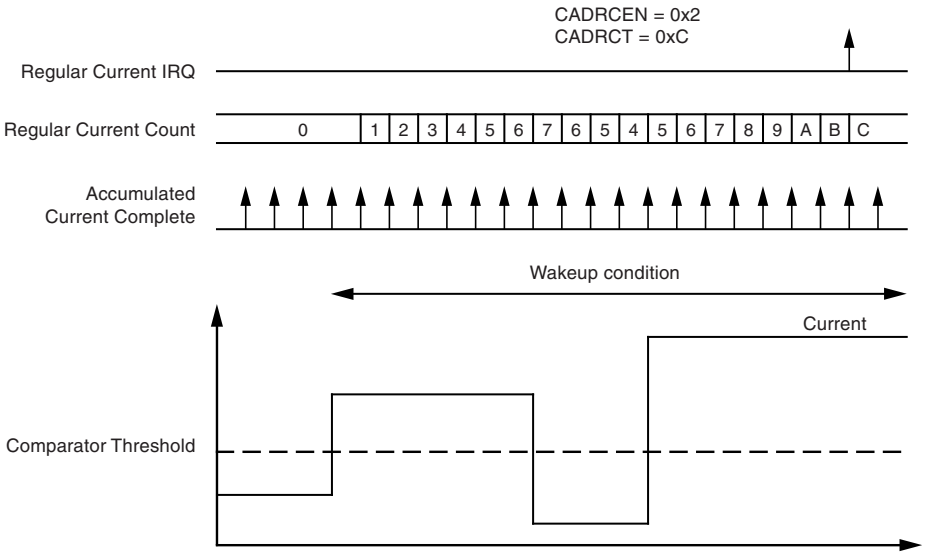


Figure 26-10 illustrates the Regular Current Comparator timing with the counter configured to decrement when going below the threshold level.

**Figure 26-10.Regulator Current Timing with Counter Decrement**




When the Regular Current Comparator has detected a Regular Current condition this will automatically reset the Regular Current Counter. The Regular Current Counter will also reset if software changes configuration from reset to decrement or decrement to reset mode.

If the CADAC conversion result is blocked by a read out busy, the Regular Current Comparator will not be affected.

### 26.3.6 Conversion Result

For the C-ADC the Current Flowing through the external shunt is given by the following equation.

$$\text{Current} = \frac{|\text{FSR}|}{R_{\text{shunt}} \times 2^{N-1} \times \text{Input Gain}} \times \text{Conversion Word}$$

For the V-ADC the following equations is used to calculate the external voltage.

$$\text{Voltage} = \frac{\text{FSR}}{2^{N}} \times \text{Conversion Word}$$

In both expressions N is the number of data bits in the conversion word. The Instantaneous conversion result has 16 bits resolution for both the C-ADC and the V-ADC while the Accumulated conversion has 18 bits for the C-ADC and 17 bits for the V-ADC.

## 26.4 Configuration and Usage

### 26.4.1 Synchronization of Configuration Settings

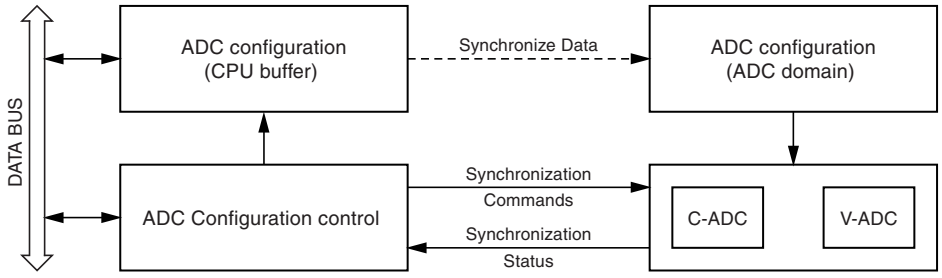
The ADCs operate in a different clock domain than the CPU. For safe synchronization and seamless configuration changes the V-ADC and C-ADC have a common configuration controller. When the CPU writes new configuration data to the ADC, the data are placed in temporary buffers. When all configuration changes have been written to the IO registers, the SCMD1:0 bits should be written to execute the synchronization command. This will start the synchronization of data to the ADC domain. Depending on command written the ADC will either:

- Update to new settings immediately
- Wait for the next Instantaneous Conversion before updating to new settings
- Wait for the next Accumulated Conversion before updating to new settings

While new configuration changes are being synchronized, the temporary buffers are locked for further writing until the data have been synchronized to the ADC domain. During synchronization, the SBSY bit will stay high until synchronization is completed. The CPU can therefore monitor the status of the synchronization by reading this bit.

The CPU/ADC synchronization is illustrated in Figure 26-11.

**Figure 26-11.Synchronization of Configuration Settings**




To ensure safe configuration changes, the following sequence should be used:

1. Check that the SBSY is low
2. Write new configuration settings (ADCSR, ADCRA, ADCRB, ADCRC, …)
3. Write SCMD1:0 to the preferred setting.

If both ADCs are disabled, they can be configured immediately without waiting for an Instantaneous or Accumulated conversion complete. In this case, the synchronization will take place immediately regardless of what is written to the SCMD1:0 bits. Note however that due to synchronization between the different clock domains, 2-3 ADC clock cycles are required before the actual configuration takes place.

If doing an immediate update to new settings, the ADC will automatically reset before applying the new settings.

When software does a configuration change by sending a synchronization command that should do reconfiguration at either next Instantaneous or Accumulated Conversion edge, the synchronization command register SCMD1:0 has to be written with a time margin to the next Instantaneous or Accumulated Conversion edge. To guarantee that the reconfiguration is done at the following edge, software has to write the command at least 35 ADC clock cycles before the next conversion edge is expected. It software writes the command too late, the reconfiguration will not be updated until the next conversion edge.

Depending on usage and the conversion result used by software, different synchronization methods are recommended:

- When either Accumulated Conversion result or both the Instantaneous and the Accumulated conversions are used by software, it is recommended to always synchronize configuration changes on the next Accumulated Conversion edge or immediately.
- When only the Instantaneous Conversion result is used by software, it is recommended to always synchronize configuration changes on the next Accumulated Conversion edge or immediately.
- When software enters or exit software polarity control mode it is recommended to synchronize configuration changes immediately.

### 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.

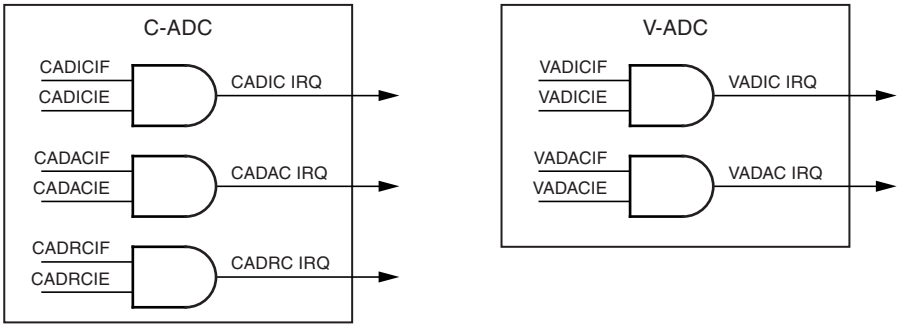
If changing to the PLL clock source software should make sure that the PLL has locked to the target frequency before using the conversion data.

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.

### 26.4.4 Interrupts

The two ADCs have five interrupt sources. Each interrupt source can be disabled individually. The interrupts are shown in the figure below.

**Figure 26-12.ADC Interrupts**




Both the V-ADC and the C-ADC can be configured to issue an interrupt on each Instantaneous and Accumulated conversion. In addition the C-ADC can be configured to issue an interrupt when a Regular Current Detection occurs.

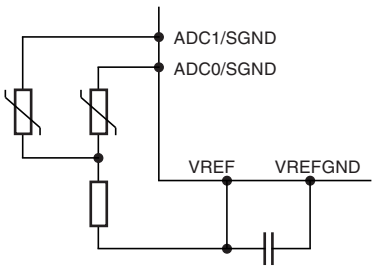
When the C-ADC is configured to run in with the Regular Current Comparator enabled, the Instantaneous and Accumulated conversion complete interrupts are still available. To avoid waking up whenever there is no Regular Current condition, the C-ADC should have only the Regular Current Detection Interrupt enabled in this mode.

### 26.4.5 Configuring ADC1 and ADC0 for V-ADC Operation

When one of the ADC0 or ADC1 is used as analog input to the V-ADC, either ADC0 or ADC1 can be used as signal ground (SGND). The use of ADC1 and ADC0 as SGND is efficient for the thermistor configuration shown in Figure 26-13. Both thermistors, are connected through a common divider resistor to ADC0 and ADC1 respectively. Both ADC0 and ADC1 have very high input impedance when used as ADC inputs, which makes it possible to connect two thermistors in the configuration shown in Figure 26-13.

When measuring the ADC0/ADC1 channel in this configuration the ADC1/ADC0 is automatically switched to SGND.

**Figure 26-13.Thermistor Configuration**




In addition to the thermistor configuration in Figure 26-13, ADC0/ADC1 can also be configured to measure the pin voltage without using the SGND configuration.

It is recommended to set the PA0DID and PA1DID bits to avoid high current consumption on the digital input buffer.

When measuring at ADC0, software should configure ADC0 as an input. When measuring at ADC1, software should configure ADC1 as an input.

### 26.4.6 Configuration Changes and Sleep Mode

To ensure the lowest power consumption in Power-down, it is recommended to disable both the CADC and the VADC by clearing the CADEN and VADEN bits before entering sleep.

When configuring the ADC, the actual configuration will not be completed until synchronization has been completed. To ensure correct operation it is not recommended to enter sleep until the synchronization has been completed and the SBSY bit in Section 26.6.1 “ADSCSRA - ADC Synchronization Control and Status Register A” on page 149 is cleared

## 26.5 Diagnosis Mode

Both the V-ADC and the C-ADC features diagnosis capability.

The V-ADC has a separate diagnosis input channel to check that the Voltage Reference is within its range. The voltage on this channel is VREF/2. When this channel is selected AVDD/3 should be selected as reference voltage. In addition the V-ADC has pull-up functionality on the PV2 and NV2 pins that can be enabled to check for opens and shorts on these pins. The pull-up is enabled by configuring the VADPDM[1:0] bits in ADCRE. Please note that this pull-up resistor is connected to VREF.

The C-ADC has a diagnosis input channel with a fixed input voltage of 3/40 × VREF that can be used to check the C-ADC. In addition it feature pull-up functionality on the PI and NI pins that can be enabled to check for opens on these pins. The pull-up is enabled by configuring the CADPDM[1:0] bits in ADCRD. Please note that this pull-up resistor is connected to VREF.

## 26.6 Register Description

### 26.6.1 ADSCSRA - ADC Synchronization Control and Status Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE0) | – | – | – | – | – | SBSY | **SCMD1** | **SCMD0** | **ADSCSRA** |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:3 - Reserved**

  These bits are reserved and will always read as zero.

- **Bit 2 - SBSY: Synchronization Busy**

  This bit shows the status of the synchronization command from the CPU to the ADC domain. When this bit is set synchronization is busy, and the configuration changes are pending. When this bit is cleared synchronization is ready and the ADC is ready to be reconfigured.

  When the SBSY is high software writing to the following ADC registers are blocked by hardware; ADSCSR, ADCRA, ADCRB, ADCRC, ADCRD, ADCRE and CADRCLH and CARDRLL.

- **Bit 1:0 - SCMD[1:0]: Synchronization Command**

  By writing to these bits a synchronization command is issued from the CPU domain to the ADC domain. The ADC will update the new settings based on the synchronization command. Table 26-3 shows the synchronization commands from the CPU to the ADC.

  When writing to these bits while the SBSY bit is low, a synchronization command is issued and SBSY will go high. Any writing to these bits while the SBSY bit is high will be ignored. When SBSY is high the ongoing command will be present in the SCMD1:0 bits. When SBSY goes low the SCMD1:0 bits are automatically cleared.

**Table 26-3.** CPU Synchronization Commands

| SCMD[1:0] | Synchronization Command |
|---|---|
| 00 | No Synchronization |
| 01 | Reset and Synchronize |
| 10 | Synchronize on next Instantaneous Conversion |
| 11 | Synchronize on next Accumulated Conversion |

### 26.6.2 ADSCSRB - ADC Synchronization Control and Status Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE1) | – | VADICPS | VADACRB | VADICRB | – | CADICPS | CADACRB | CADICRB | ADSCSRB |
| Read/Write | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 - Reserved**

  This bit is reserved and will always read as zero.

- **Bit 6 - VADICPS: V-ADC Instantaneous Conversion Polarity Status**

  When the Automatic Chopper is configured to run in Automatic Slow Chopping mode, this bit shows the polarity of the conversion present in the V-ADC Instantaneous conversion registers, VADICL and VADICH. This bit should be read before reading out the Instantaneous conversion result to ensure correct polarity information. In other modes this bit will always read as zero. For details on chopping, see Section 26.3.4 “Programmable Chopper Control” on page 142.

- **Bit 5 - VADACRB: VADAC Data Read Out Busy**

  This bit will be set when reading either the VADAC0, VADAC1 or VADAC2 data registers and cleared when reading the VADAC3 data register. When the VADACRB bit is set it means that a data read out is busy and during this time V-ADC Accumulated conversion registers are blocked for further update. If a read out is busy when a new V-ADC Accumulated conversion value is ready, both the new conversion value and setting of the Interrupt Flag will be lost.

- **Bit 4 - VADICRB: VADIC Data Busy Read**

  This bit will be set when reading the VADICL data register and cleared when reading the VADICH data register. When the VADICRB bit is set it means that a data read out is busy and during this time V-ADC Instantaneous Voltage registers are blocked for further update. If a read out is busy when a new V-ADC Instantaneous conversion value is ready, both the new conversion value and setting of the Interrupt Flag will be lost.

- **Bit 3 - Reserved**

  This bit is reserved bits and will always read as zero.

- **Bit 2 - CADICPS: C-ADC Instantaneous Conversion Polarity Status**

  When the Automatic Chopper is configured to run in Automatic Slow Chopping mode, this bit shows the polarity of the conversion present in the C-ADC Instantaneous conversion registers, CADICL and CADICH. This bit should be read before reading out the Instantaneous conversion result to ensure correct polarity information. In other modes this bit will always read as zero. For details on chopping, see Section 26.3.4 “Programmable Chopper Control” on page 142.

- **Bit 1 - CADACRB: CADAC Data Read Out Busy**

  This bit will be set when reading either the CADAC0, CADAC1 or CADAC2 data registers and cleared when reading the CADAC3 data register. When the CADACRB bit is set it means that a data read out is busy and during this time C-ADC Accumulate conversion registers are blocked for further update. If a read out is busy when a new C-ADC Accumulated conversion value is ready, both the new conversion value and setting of the Interrupt Flag will be lost.

- **Bit 0 - CADICRB: CADIC Data Read Out Busy**

  This bit will be set when reading the CADICL data register and cleared when reading the CADICH data register. When the CADICRB bit is set it means that a data read out is busy and during this time C-ADC Instantaneous conversion registers are blocked for further update. If a read out is busy when a new C-ADC Instantaneous  Current value is ready, both the new conversion value and setting of the Interrupt Flag will be lost.

### 26.6.3 ADCRA - ADC Control Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE2) | – | – | – | – | ADPSEL | ADCMS[1:0] | | CKSEL | ADCRA |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:4 - Reserved**

  Theser bits are reserved and will always read as zero.

- **Bit 4 - ADPSEL: ADC Polarity Select**

  This bit is used to control the polarity when running Software Polarity Control operation. To write this bit to '1', the ADCMS1:0 bits have to be written to '11' at the same time, otherwise this bit will not be set.

- **Bit 2:1 - ADCMS[1:0]: C-ADC Chopper Mode Select**

  These bits are used to configure the chopper for the ADCs according to Table 26-4.

**Table 26-4. Chopper Mode Selection**

| ADCMS[1:0] | Chopper Mode Select |
|---|---|
| 00 | Chopping disabled |
| 01 | Automatic Fast Chopping |
| 10 | Automatic Slow Chopping |
| 11 | Software Polarity Control |

- **Bit 0 - CKSEL: Sampling Clock Select**

This bit selects the sampling clock for both ADCs. By writing this bit to one the Slow RC oscillator will be used as sampling clock. Table 26-5 shows the sampling clock selection for the ADC.

Note that to avoid getting wrong data result by changing the clock in the middle of a conversion, ADC clock changing is synchronized along with other configuration changes.

**Table 26-5. Sampling Clock Selection**

| CKSEL | ADC Clock Source |
|---|---|
| 0 | PLL (512kHz output) as sampling clock |
| 1 | Slow RC Oscillator as sampling clock |

### 26.6.4 ADCRB - ADC Control Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE3) | – | – | | ADIDES[1:0] | | ADADES[2:0] | | | ADCRB |
| 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 7:5 - Reserved**

  Theser bits are reserved and will always read as zero.

- **Bit 4:3 - ADIDES[1:0]: Instantaneous Decimation Ratio Select**

  These bits determine the Decimation Ratio for the Instantaneous Conversion output. The same settings apply for both the C-ADC and the V-ADC.

**Table 26-6. Instantaneous Decimation Ratios**

| ADIDES[1:0] | Instantaneous Decimation Ratio |
|---|---|
| 00 | 512 |
| 01 | 256 |
| 10 | 128 |
| 11 | 64 |

- **Bit 2:0 - ADADES[2:0]: Accumulated Decimation Ratio Select**

  These bits determine the Decimation Ratio for the Accumulated Conversion output. The same settings apply for both the C-ADC and the V-ADC.

**Table 26-7. Accumulated Decimation Ratios**

| ADADES[2:0] | Accumulated Decimation Ratio |
|---|---|
| 000 | 512 |
| 001 | 256 |
| 010 | 128 |
| 011 | 64 |
| 100 | 32 |
| 101 | 16 |
| 110 | 8 |
| 111 | 4 |

### 26.6.5 ADCRC - ADC Control Register C

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE4) | CADEN | – | CADRCM[1:0] | | CADRCT[3:0] | | | | ADCRC |
| 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 - CADEN: C-ADC Enable**

  This bit is used to enable the C-ADC. When this bit is set to one the C-ADC will be enabled. When clearing this bit the C-ADC will be disabled.

- **Bit 6 - Reserved**

  This bit is reserved and will always read as zero.

- **Bit 5:4 - CADRCM[1:0]: C-ADC Regular Current Comparator Mode**

  These bits are used to enable the Regular Current Comparator and to configure the Regular Current Counter.

**Table 26-8. Regular Current Comparator Mode**

| CADRCM[1:0] | Regular Current Comparator Mode |
|---|---|
| 00 | Comparator Disabled |
| 01 | Comparator Enabled.<br>The Regular Current Counter counts up if Accumulated Current measurement is above threshold and it is reset if the Accumulated Current goes below threshold. |
| 10 | Comparator enabled.<br>The Regular Current Counter counts up if Accumulated Current measurement is above threshold and down towards 0 if Accumulated Current goes below threshold. |
| 11 | Reserved |

- **Bit 7-4 - CADRCT[3:0]: C-ADC Regular Current Count Threshold**

  These bits determine number of Accumulated Current measurements that must be above the Regular Current Threshold Level before a Regular Current Condition occurs.

**Table 26-9. Regular Current Count Threshold**

| CADRCT[3:0] | Number of sample above threshold |
|---|---|
| 0000 | 1 |
| 0001 | 2 |
| 0010 | 3 |
| ... | ... |
| 1111 | 16 |

### 26.6.6 ADCRD - ADC Control Register D

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE5) | – | – | CADG[2:0] | | | CADPDM[1:0] | | CADDSEL | ADCRD |
| 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 7:6 - Reserved**
  These bits are reserved and will always read as zero.
- **Bit 5:3 - CADG[2:0]: C-ADC Gain**
  These bits determine the gain in the analog input stage of the C-ADC according to Table 26-10.

**Table 26-10. Input Gain**

| CADG[2:0] | Input Gain |
|---|---|
| 000 | 4x |
| 001 | 8x |
| 010 | 16x |
| 011 | 32x |
| 100 | 64x |
| 101 | 128x |
| 110 | 256x |
| 111 | Reserved |

- **Bit 2:1 - CADPDM[1:0]: C-ADC Pin Diagnostics Mode**
  These bits are used to enable diagnosis pull-up voltages for hence the PI and NI input pins of the C-ADC according to Table 26-11.

**Table 26-11. Pin Diagnosis Mode**

| CADPDM[1:0] | Description |
|---|---|
| 00 | Pull-up disabled |
| 01 | Pull-up on PI pin enabled |
| 10 | Pull-up on NI pin enabled |
| 11 | Pull-up on both PI/NI pins enabled |

- **Bit 0 - CADDSEL: C-ADC Diagnosis Channel Select**
  When this bit is set, the C-ADC will select a diagnosis voltage input corresponding to 3/40VREF. When using this channel the C-ADC gain, CADG[1:0] bits, should be configured with 4x setting. When this bit is cleared the PI and NI will be used as inputs for the C-ADC.

### 26.6.7 ADCRE - ADC Control Register E

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE6) | VADEN | – | VADREFS | VADPDM[1:0] | | VAMUX[2:0] | | | ADCRE |
| 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 - VADEN: V-ADC Enable**

  This bit is used to enable the V-ADC. When this bit is set to one the V-ADC will be enabled. When clearing this bit the V-ADC will be disabled .

- **Bit 6 - Reserved**

  This bit is reserved and is always read as zero.

- **Bit 5 - VADREFS: V-ADC Reference Select**

  This bit is used to select the Reference Voltage for the V-ADC according to Table 26-12.

**Table 26-12. V-ADC Reference Select**

| VADREFS | Description |
|---|---|
| 0 | VREF as reference |
| 1 | AVDD/3 as reference (for diagnosis purpose) |

- **Bit 4:3 - VADPDM[1:0]: V-ADC Pin Diagnostics Mode**

These bits are used to enable diagnosis pull-up voltages for hence the PV2 and NV2 input pins of the V-ADC according to Table 26-13.

**Table 26-13. Pin Diagnosis Mode**

| VADPDM[1:0] | Description |
|---|---|
| 00 | Pull-up disabled |
| 01 | Pull-up on PV2 pin enabled |
| 10 | Pull-up on NV2 pin enabled |
| 11 | Pull-up on both PV2/NV2 pins enabled |

- **Bit 2:0 - VAMUX[2:0]: V-ADC Channel Selection Bits**

This VAMUX bits determine the V-ADC channel selection according to Table 26-14.

**Table 26-14. V-ADC Channel Selection**

| VAMUX[2:0] | Channel Selected |
|---|---|
| 000 | PV2 - NV2 |
| 001 | ADC0 - SGND(1) |
| 010 | ADC1 - SGND(2) |
| 011 | VTEMP - GND |
| 100 | DIAGNOSIS (VREF/2) - GND |
| 101 | ADC0 - GND |
| 110 | ADC1 - GND |
| 111 | Reserved |

Notes: 1. ADC1 will automatically be configured as SGND.
2. ADC0 will automatically be configured as SGND.

### 26.6.8 ADIFR - ADC Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE7) | – | – | VADACIF | VADICIF | – | CADRCIF | CADACIF | CADICIF | ADIER |
| Read/Write | R | R | R/W | R/W | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:6 - Reserved**

  These bits are reserved and will always read as zero.

- **Bit 5 - VADACIF: Accumulated Conversion Interrupt Flag**

  This bit is set when the V-ADC completes an Accumulated Conversion and the V-ADC Accumulated Conversion registers are updated. The V-ADC Accumulated Conversion Complete Interrupt is executed if the VADACIE bit and the I-bit in SREG is set. VADACIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, VADACIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADIFR, a pending interrupt can be lost.

- **Bit 4 - VADICIF: Instantaneous Conversion Interrupt Flag**

  This bit is set when the V-ADC completes an Instantaneous Conversion and the V-ADC Instantaneous Conversion registers are updated. The V-ADC Instantaneous Conversion Complete Interrupt is executed if the VADICIE bit and the I-bit in SREG is set. VADICIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, VADICIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADIFR, a pending interrupt can be lost.

- **Bit 3 - Reserved**

  This bit is reserved and will always read as zero.

- **Bit 2 - CADRCIF: Regulator Current Interrupt Flag**

  This bit is set when a C-ADC has detected that a Regular Current Condition has occurred. The C-ADC Regular Current Interrupt is executed if the CADRCIE bit and the I-bit in SREG is set. CADRCIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, CADRCIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADIFR, a pending interrupt can be lost.

- **Bit 1 - CADACIF: Accumulated Conversion Interrupt Flag**

  This bit is set when the C-ADC completes an Accumulated Conversion and the C-ADC Accumulated Conversion registers are updated. The C-ADC Accumulated Conversion Complete Interrupt is executed if the CADACIE bit and the I-bit in SREG is set. CADACIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, CADACIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADIFR, a pending interrupt can be lost.

- **Bit 0 - CADICIF: Instantaneous Conversion Interrupt Flag**

  This bit is set when a C-ADC completes an Instantaneous Conversion and the C-ADC Instantaneous Conversion registers are updated. The C-ADC Instantaneous Conversion Complete Interrupt is executed if the CADICIE bit and the I-bit in SREG is set. CADICIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, CADICIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADIFR, a pending interrupt can be lost.

### 26.6.9 ADIMR - ADC Interrupt Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xE8) | – | – | VADACIE | VADICIE | – | CADRCIE | CADACIE | CADICIE | ADIMR |
| Read/Write | R | R | R/W | R/W | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:6 - Reserved**

  These bits are reserved and will always read as zero.
- **Bit 5 - VADACIE: Accumulated Conversion Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the V-ADC Accumulate Conversion Complete Interrupt is activated.
- **Bit 4 - VADICIE: Instantaneous Conversion Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the V-ADC Instantaneous Conversion Complete Interrupt is activated.
- **Bit 3 - Reserved**

  This bit is reserved and will always read as zero.
- **Bit 2 - CADRCIE: Regular Current Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the C-ADC Regular Current Detection Interrupt is activated.
- **Bit 1 - CADACIE: Accumulated Conversion Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the C-ADC Accumulated Conversion Complete Interrupt is activated.
- **Bit 0 - CADICIE: Instantaneous Conversion Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the C-ADC Instantaneous Conversion Complete Interrupt is activated.

### 26.6.10 CADRCLH and CADRCLL - C-ADC Regulator Current Comparator Threshold Level

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xEA) | CADRCL[15:8] | | | | | | | | CADRCLH |
| (0xE9) | CADRCL[7:0] | | | | | | | | CADRCLL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | 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 | |

- **Bit 15:0 - CADRCL[15:0]: C-ADC Regulator Current Threshold Level**

  The C-ADC Regular Current Comparator Threshold registers, CADRCLH and CADRCLL determine the threshold level for the Regular Current Comparator detection. The value is in unsigned format and bit 15 will always read zero.

### 26.6.11 VADICH and VADICL - V-ADC Instantaneous Conversion Result

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xF2) | VADICH[15:8] | | | | | | | | VADICH |
| (0xF1) | VADICL[7:0] | | | | | | | | VADICL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

When a V-ADC Instantaneous Conversion is complete, the result is found in these two registers. The VADICH and VADICL contain the Instantaneous Voltage measurements in unsigned format.

When VADICL is read, the V-ADC Instantaneous Conversion register is not updated until VADICH is read. During a read of the data values the VADIC Read Out Busy Flag, VADICRB in Section 26.6.2 “ADSCSRB - ADC Synchronization Control and Status Register B” on page 150 will be high. Reading the registers in the sequence VADICL, VADICH will ensure that consistent values are read. When a conversion is completed, both registers must be read before the next conversion is completed, otherwise data will be lost.

### 26.6.12 VADAC3, VADAC2, VADAC1 and VADAC0 - V-ADC Accumulated Conversion Result

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
|---|---|---|---|---|---|---|---|---|---|
| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| (0xF6) | VADAC3[31:24] | | | | | | | | VADAC3 |
| (0xF5) | VADAC2[23:16] | | | | | | | | VADAC2 |
| (0xF4) | VADAC1[15:8] | | | | | | | | VADAC1 |
| (0xF3) | VADAC0[7:0] | | | | | | | | VADAC0 |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| | 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 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

When a V-ADC Accumulated Conversion is complete, the result is found in these four registers. The VADAC3, VADAC2, VADAC1 and VADAC0 registers contain the Accumulate Voltage measurements in unsigned format. Bits 31:15 are the 17-bit ADC result, while bits 14:0 will read all zeros.

When VADAC0, VADAC1 or VADAC2 is read, the V-ADC Accumulated Conversion register is not updated until VADAC3 is read. During a read of the data values the VADAC Read Out Busy Flag, VADACRB in ADSCSRB - ADC Synchronization Control and Status Register B will be high. Reading the registers in the sequence VADAC0, VADAC1, VADAC2, VADAC3 will ensure that consistent values are read. When a conversion is completed, all registers must be read before the next conversion is completed, otherwise data will be lost.

### 26.6.13 CADICH and CADICL - C-ADC Instantaneous Conversion Result

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xEC) | CADICH[15:8] | | | | | | | | CADICH |
| (0xEB) | CADICL[7:0] | | | | | | | | CADICL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

When a C-ADC Instantaneous Conversion is complete, the result is found in these two registers. The CADICH and CADICL contain the Instantaneous Current measurements in 2's complement format. When CADICL is read, the C-ADC Instantaneous Conversion register is not updated until CADICH is read. During a read of the data values the CADIC Read Out Busy Flag, CADICRB in Section 26.6.2 “ADSCSRB - ADC Synchronization Control and Status Register B” on page 150 will be high. Reading the registers in the sequence CADICL, CADICH will ensure that consistent values are read. When a conversion is completed, both registers must be read before the next conversion is completed, otherwise data will be lost.

### 26.6.14 CADAC3, CADAC2, CADAC1 and CADAC0 - C-ADC Accumulated Conversion Result

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
|---|---|---|---|---|---|---|---|---|---|
| | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| (0xF0) | CADAC3[31:24] | | | | | | | | CADAC3 |
| (0xEF) | CADAC2[23:16] | | | | | | | | CADAC2 |
| (0xEE) | CADAC1[15:8] | | | | | | | | CADAC1 |
| (0xED) | CADAC0[7:0] | | | | | | | | CADAC0 |
| Read/Write | R | R | R | R | R | R | R | R | |
| | R | R | R | R | R | R | R | R | |
| | 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 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

When a C-ADC Accumulated Conversion is complete, the result is found in these four registers. The CADAC3, CADAC2, CADAC1 and CADAC0 Registers contain the Accumulate Current measurements in 2's complement format. Bits 31:14 are the 18-bit ADC result (including sign), while bit 13:0 will read all zeros for positive results and all ones for negative results.

When CADAC0, CADAC1 or CADAC2 is read, the C-ADC Accumulated Conversion register is not updated until CADAC3 is read. During a read of the data values the CADAC Read Out Busy Flag, CADACRB in Section 26.6.2 “ADSCSRB - ADC Synchronization Control and Status Register B” on page 150 will be high. Reading the registers in the sequence CADAC0, CADAC1, CADAC2, CADAC3 will ensure that consistent values are read. When a conversion is completed, all registers must be read before the next conversion is completed, otherwise data will be lost.

### 26.6.15 DIDR0 – Digital Input Disable Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x7E) | – | – | – | – | – | – | PA1DID | PA0DID | DIDR0 |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:2 - Reserved**

  These bits are reserved and is always read as zero.

- **Bit 1:0 - PA1DID:PA0DID**

  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 PA1DID:PA0DID 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.

# 27. Band Gap Reference and Temperature Sensor

## 27.1 Features

- Accurate voltage reference of 1.100V
- Internal temperature sensor
- Curvature compensation to cancel higher order temperature dependence
- Lock register for locking parameter and control registers
- External decoupling for optimum noise performance
- Low Power Consumption Mode in Power-down Sleep Mode

## 27.2 Overview

A low power band-gap reference provides Atmel® AVR MCU with a highly accurate on-chip voltage reference VREF of 1.100V. This reference voltage is used as reference for the V-ADC, C-ADC and Brown-out Detector. The reference to the ADCs uses a buffer with external decoupling capacitor to enable excellent noise performance with minimum power consumption.

The reference voltage $V_{REF_P}/V_{REF_N}$ to the C-ADC is scaled to match the full-scale requirement at the current sense input pins. This configuration also enables concurrent operation of both V-ADC and C-ADC.

To guarantee ultra low temperature drift after factory calibration, the Atmel AVR MCU features a two-step calibration algorithm. The calibration steps are performed at 25°C and 125°C. The result is stored in the signature row. Temperature drift after this calibration is guaranteed by design and characterization to be less than 20 ppm/°C from -40°C to 125°C.

The Atmel AVR MCU has an on-chip temperature sensor for monitoring the die temperature. A voltage Complementary-To-Absolute-Temperature, $V_{CTAT}$, is generated in the voltage reference circuit and connected to the multiplexer at the V-ADC input. 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 $V_{CTAT}$ voltage must be scaled with the $V_{CTAT}$ factory calibration values, $VTEMP_{SLOPE}$ and $VTEMP_{BASE}$, stored in the signature row.

## 27.3 $VTEMP_{BASE}$ and $VTEMP_{SLOPE}$

$VTEMP_{BASE}$ is an unsigned 16-bit value. It gives the VADC reading for Vtemp(T0) for the ADC-conversion used when finding actual temperature.

$VTEMP_{SLOPE}$ is a signed 8 bit value. It is 4-times the deviation from the typical slope of –96LSB/K. $VTEMP_{SLOPE_hot}$ should be used for temperatures above 25°C, while $VTEMP_{SLOPE_cold}$ should be used for temperatures below 25°C. If the ADC reading is below $VTEMP_{BASE}$ use $VTEMP_{SLOPE_hot}$, and if the ADC reading is above $VTEMP_{BASE}$ use $VTEMP_{SLOPE_cold}$.

When using these values to find actual temperature, use the following formula:

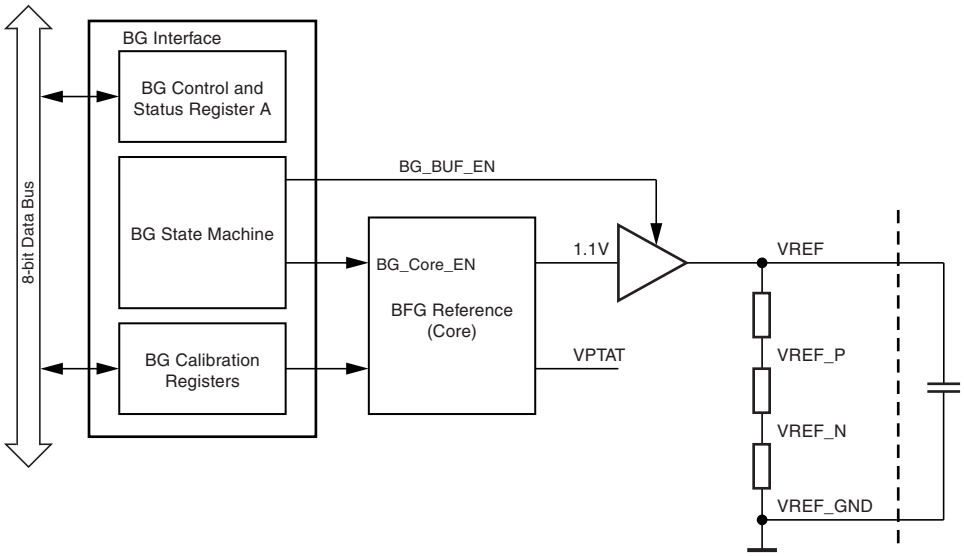
$$\mathrm{Temp} = \frac{1}{\frac{\mathrm{VTEMP_{SLOPE}}}{4} - 96}(\mathrm{ADC_{VTEMP}} - \mathrm{VTEMP_{BASE}}) + \mathrm{TEMP_{BASE}}$$

$ADCV_{TEMP}$ is the raw ADC reading then measuring the VTEMP channel. $TEMP_{BASE}$ is assumed to be 25°C giving:

$$\mathrm{Temp[K]} = \frac{1}{\frac{\mathrm{VTEMP_{SLOPE}}}{4}\left[\frac{\mathrm{LSB}}{\mathrm{K}}\right] - 96\left[\frac{\mathrm{LSB}}{\mathrm{K}}\right]}(\mathrm{ADC_{VTEMP}[LSB]} - \mathrm{VTEMP_{BASE}[LSB]}) + 298.15[\mathrm{K}]$$

See Section 29.8.9 “Reading the Signature Row from Software” on page 173 for details.

**Figure 27-1. Reference Circuitry**

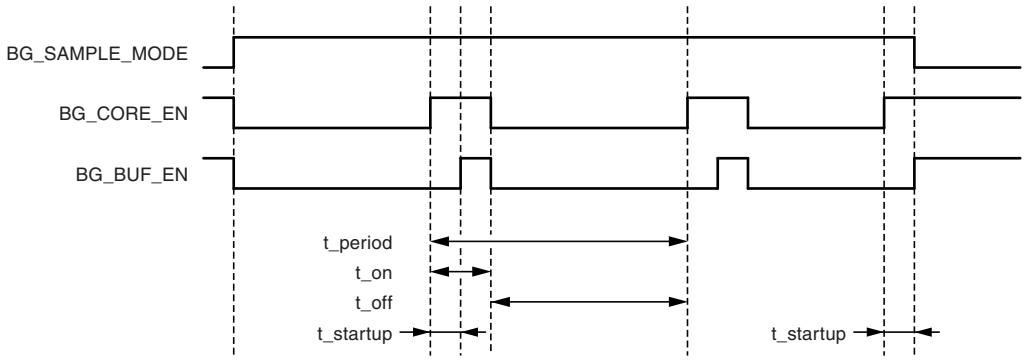



## 27.4 Band Gap Sample Mode

The Band Gap module has two operation modes, continuous mode and sampled mode. When the Band Gap module is run in sampled mode, the Band Gap core and output buffer are switched on and off at regular time intervals. When the Band Gap core and output buffer are off, the reference voltage is stored on the external capacitor. In continuous mode, the core and output buffer are switched on all the time. The sampled mode is enabled automatically in Power Down sleep mode. In all other operating modes, the continuous mode is used.

The timing of the signals in sampled mode is shown in Figure 27-2. The BGSC bits in the BGCSRA register control the timing of t_off. For the timing of BG_CORE_EN and BG_BUF_EN also see Section 31. “Electrical Characteristics AVR MCU” on page 192ff.

**Figure 27-2. Band Gap Timing**




The timing of the sampled mode is dependent on the leakage current from the external capacitor storing the VREF voltage during the off period. For instance a ceramic capacitor of 1 µF may have an insulation resistance of ~500M , while a tantalum capacitor of 1 µF may have an insulation resistance of ~1M . The maximum off period is given by:

$$t_off < \frac{CREF \times \Delta VREF \times R_{ins}}{VREF}$$

where CREF and $R_{ins}$ are the capacitance and the insulation resistance of the external decoupling capacitor, VREF is the bandgap output voltage and $\Delta VREF$ is the maximum allowed variation in VREF. As the Brown-out Detector is the only module using the Voltage Reference in Power Down the maximum acceptable degradation of the BOD level in Power-down is the deciding factor when setting VREF. Setting ΔVREF to 15.5mV gives t_off < 7s for a ceramic capacitor and t_off < 14 ms for a tantalum capacitor. Timeout settings longer than t_off will violate the VREF requirement used in this example.

## 27.5 Register Description

### 27.5.1 BGCRA - Band Gap Calibration Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xD3) | BGCN[7:0] | | | | | | | | BGCRA |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | Device specific calibration value | | | | | | | | |

Notes: 1. The register is only reset by POR.
2. The register can be locked by a timed sequence described in the BGLR register.

- **Bit 7:0 - BGCN[7:0]: Band Gap Calibration Nominal**

  The calibration of the nominal value for the Band Gap module is done by Atmel factory calibration. The factory calibrated value is automatically written to this register during chip reset, and should not be changed by the SW.

### 27.5.2 BGCRB - Band Gap Calibration Register B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xD2) | BGCL[7:0] | | | | | | | | BGCRB |
| Read/Write | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | Device specific calibration value | | | | | | | | |

Notes: 1. The register is only reset by POR.
2. The register can be locked by a timed sequence described in the BGLR register.

- **Bit 7:0 - BGCL[7:0]: Band Gap Calibration Linear**

  The calibration of the linear value for the Band Gap module is done by Atmel factory calibration. The factory calibrated value is automatically written to this register during chip reset, and should not be changed by the SW.

### 27.5.3 BGCSRA - Band Gap Control and Status Register A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xD1) | – | – | – | – | – | BGCS[2:0] | | | BGCSRA |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Notes: 1. Due to synchronization of parameters between clock domains, a guard time of 3 ULP oscillator cycles + 3 CPU clock cycles is required between each time the BGCSRA register is written. Any writing to the BGCSRA register during this period will be ignored.
2. The register can be locked by a timed sequence described in the BGLR register.
3. After this register has been written, a guard time of 3 ULP oscillator cycles has to be added before entering Power-down sleep mode.

- **Bit 7:3 - Reserved**

  These bits are reserved and will always read as zero.

- **Bit 2:0 - BGSC: Band Gap Sample Configuration**

  These bits control the sample mode functionality. When BGSC is set to 000, Band Gap sample mode is disabled and Band Gap module will remain constant on in all sleep modes. When set to 111, the Band Gap module will remain constant off in Power-down sleep mode. When set to other values, the Band Gap module will enter sample mode in Power-down sleep mode. Timeout settings for the Band Gap Sample Mode are shown in Table 27-1.

**Table 27-1. Timeout Settings for the Bandgap Sample Mode**

| BGSC | t_off |
|---|---|
| 000 | Continuous mode |
| 001 | 1ms |
| 010 | 2ms |
| 011 | 4ms |
| 100 | 8ms |
| 101 | 16ms |
| 110 | 32ms |
| 111[1] | Infinite |

Note: 1. This setting is not recommended unless BOD is disabled.

### 27.5.4 BGLR - Band Gap Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xD4) | – | – | – | – | – | – | BGPLE | BGPL | BGLR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7:2 - Reserved**

  These bits are reserved and will always read as zero.

- **Bit 1 - BGPLE: Band Gap Lock**

  The BGCRA, BGCRB and BGCSRA registers can be locked from any further software updates. Once locked, these registers cannot be accessed until the next hardware reset.

  To lock these registers, the following algorithm must be followed:

  1. In the same operation, write a logic one to BGPLE and BGPL.
  2. Within the next four clock cycles, in the same operation, write a logic zero to BGPLE and a logic one to BGPL.

# 28. debugWIRE On-chip Debug System

## 28.1 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

## 28.2 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.

## 28.3 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 28-1. The debugWIRE Setup**

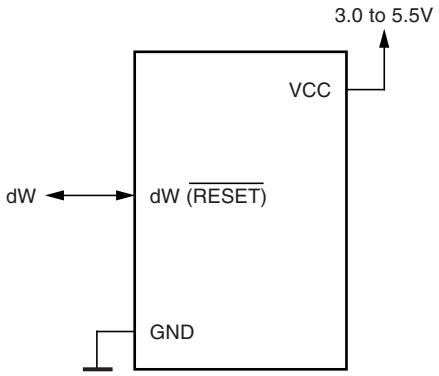



Figure 28-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the OSCSEL Fuse.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to $V_{CC}$ will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

## 28.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio® will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

## 28.5 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.

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.

When using debugWIRE to access the flash (reading/writing), one must make sure the SPMCSR register is not locked from before. If SPMCSR is locked the result of the operation may not be as expected.

## 28.6 Register Description

The following section describes the registers used with the debugWire.

### 28.6.1 DWDR – debugWire Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x31 (0x51) | 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.

# 29. Boot Loader Support – Read-While-Write Self-Programming

## 29.1 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 Section 30.5 “Page Size” on page 183) used during programming. The page organization does not affect normal operation.

## 29.2 Overview

The Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

## 29.3 Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section. The size of the different sections is configured by the BOOTSZ Fuses as shown in Section 29.8.13 “Atmel ATmega32HVE Boot Loader Parameters” on page 176 and Figure 29-2. These two sections can have different level of protection since they have different sets of Lock bits.

### 29.3.1 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 30-2 on page 180. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

### 29.3.2 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 30-2 on page 180.

## 29.4 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Section 29.8.13 “Atmel ATmega32HVE Boot Loader Parameters” on page 176 and Figure 29-2 on page 169. 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.

### 29.4.1 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 on-going programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See Section 29.9.1 “SPMCSR – Store Program Memory Control and Status Register” on page 178 for details on how to clear RWWSB.

### 29.4.2 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 29-1. Read-While-Write Features**

| Which Section does the Z-pointer Address During the Programming? | Which Section Can be Read During Programming? | CPU Halted? | Read-While-Write Supported? |
|---|---|---|---|
| RWW Section | NRWW Section | No | Yes |
| NRWW Section | None | Yes | No |

**Figure 29-1. Read-While-Write versus No Read-While-Write**

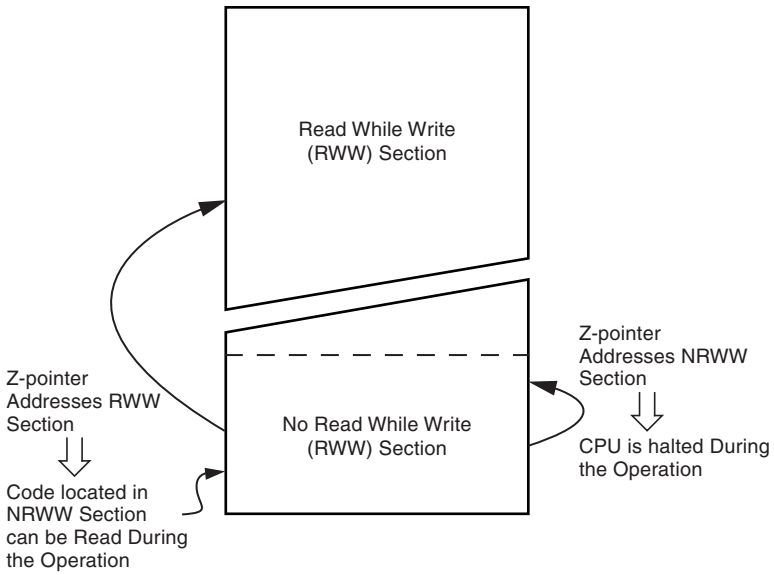
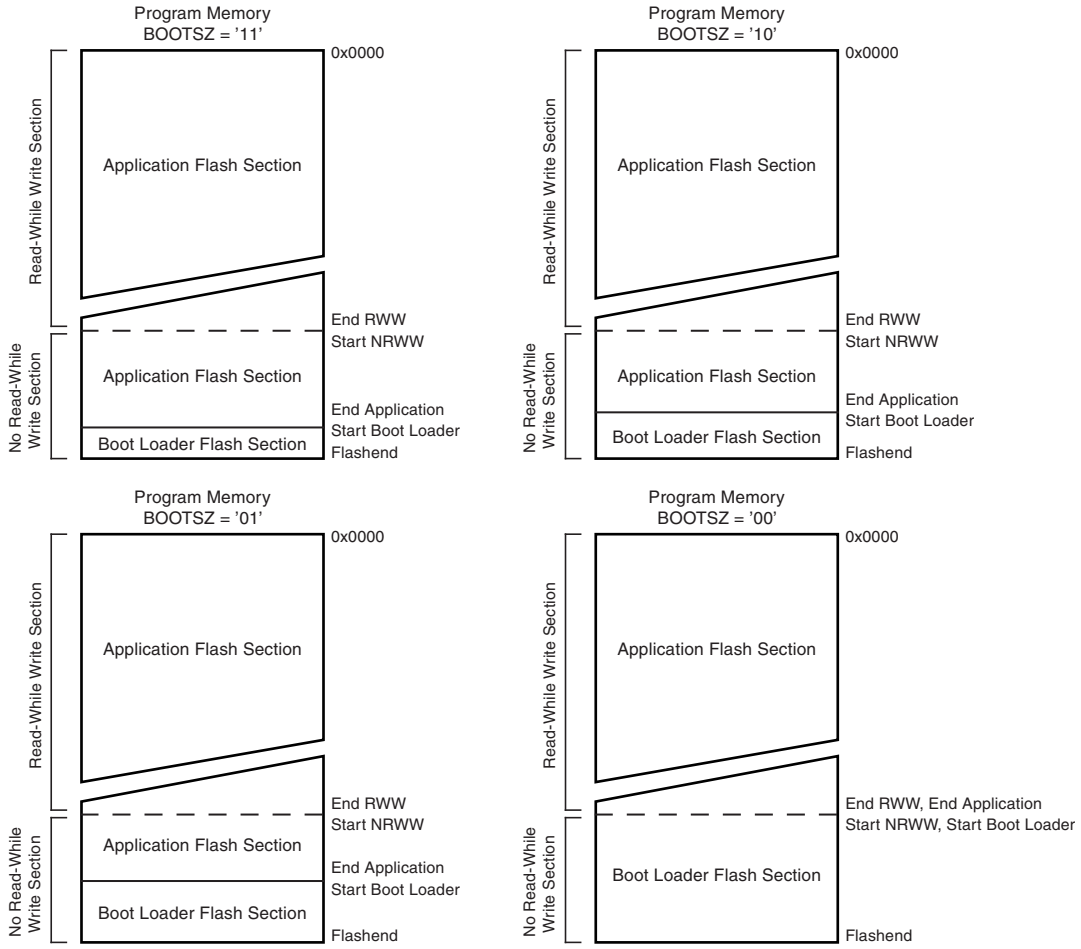


**Figure 29-2. Memory Sections**




Note: The parameters in the figure above are given in Section 29.8.13 “Atmel ATmega32HVE Boot Loader Parameters” on page 176.

## 29.5 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 30-2 on page 180 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.

## 29.6 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 the SPI or LIN. 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 29-2. Boot Reset Fuse(1)**

| BOOTRST | Reset Address |
|---|---|
| 1 | Reset Vector = Application Reset (address 0x0000) |
| 0 | Reset Vector = Boot Loader Reset (see Table 29-5 on page 176) |

Note: 1. “1” means unprogrammed, “0” means programmed

## 29.7 Addressing the Flash During Self-Programming

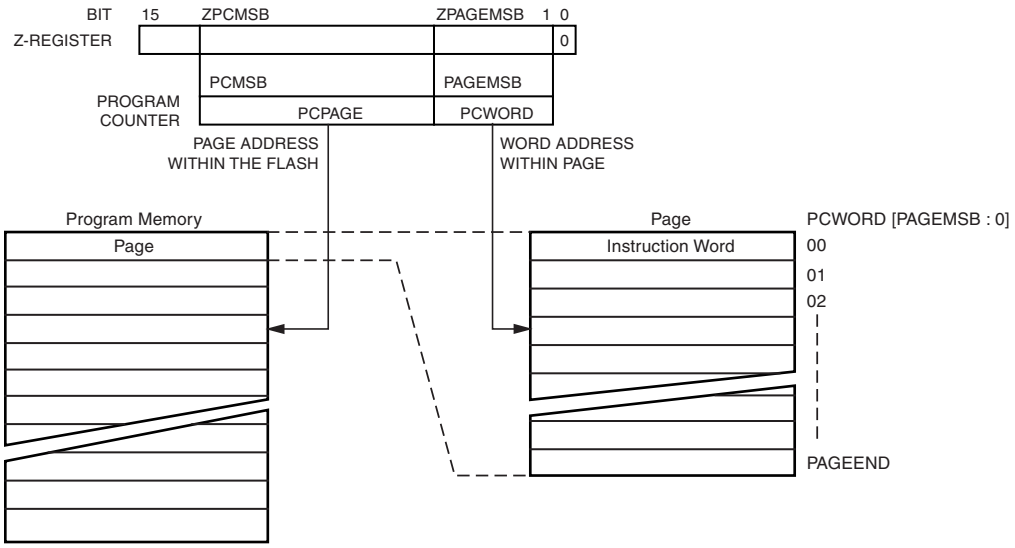
The Z-pointer is used to address the SPM commands.

| 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 Section 30.2 “Fuse Bits” on page 181), 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 29-3. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

**Figure 29-3. Addressing the Flash During SPM(1)**




Note: 1. The different variables used in Figure 29-3 are listed in Section 29.8.13 “Atmel ATmega32HVE Boot Loader Parameters” on page 176 and Section 29.8.14 “Atmel ATmega64HVE Boot Loader Parameters” on page 177.

## 29.8 Self-Programming the Flash

The Self-Programming routine should always ensure that the PLL is in lock before continuing. 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 Section 29.8.12 “Simple Assembly Code Example for a Boot Loader” on page 174 for an assembly code example.

### 29.8.1 Performing Page Erase by SPM

To execute Page Erase, wait until the PLL enters LOCK[1], 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.

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

### 29.8.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, wait until the PLL enters LOCK[1], set up the address in the Z-pointer and data in R1:R0, write “X0000001” 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. Any EEPROM read or write during an SPM operation is not recommended.

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

### 29.8.3 Performing a Page Write

To execute Page Write, wait until the PLL enters LOCK[1], 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 will be ignored 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.

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

### 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** |

Similarly, when reading the Fuse High byte, load 0x0003 in the Z-pointer. 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 High byte (FHB) will be loaded in the destination register as shown below. Refer to Table 30-3 on page 181 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 |

Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

### 29.8.9 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Section 30.3 “Signature Bytes” on page 182 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 6 cycles after writing to SPMCSR, which is locked for further writing during these cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction set Manual

**Table 29-3. Signature Bytes**

| Word-Address | Byte Description |
|---|---|
| 0x0010 | $VTEMP_{BASE}$ (16 bit value) |
| 0x0011 | $VTEMP_{SLOPE}$ (16 bit value); high byte: slope hot, low byte: slope cold |

All other addresses are reserved for future use.

### 29.8.10 SPMCSR Writing Restrictions

Writing any other combination than “100001”, “010001”, “001001”, “000101”, “000011” or “000001” in the lower six bits will have no effect.

SPMCSR is locked for writing under the following conditions:

- One or more of the bits 5:0 in SPMCSR is set to 1
- During EEPROM write (status bit EEWE in EECR is set)

SPMCSR will be cleared at the following events:

- on completion of successful execution the following instructions:
  - LPM with LBSET and SPMEN set
  - LPM with SIGRD and SPMEN set
  - SPM with LBSET and SPMEN set
  - SPM with PGERS and SPMEN set
  - SPM with PGWRT and SPMEN set
  - SPM with SPMEN set
- six cycles after writing SPMCSR if any other or no LPM/SPM is executed

### 29.8.11 Programming Time for Flash when Using SPM

The PLL Oscillator is used to time Flash accesses. The PLL Oscillator should be in lock before writing to the flash. Table 29-4 shows the typical programming time for Flash accesses from the CPU.

**Table 29-4. SPM Programming Time(1)**

| Symbol | Min Programming Time | Max Programming Time |
|---|---|---|
| Flash write (Page Erase, Page Write, and write Lock bits by SPM) | 3.7ms | 4.5ms |

Note: 1. Minimum and maximum programming time is per individual operation.

### 29.8.12 Simple Assembly Code Example for a Boot Loader

```
        ;-the routine writes one page of data from RAM to Flash
        ; the first data location in RAM is pointed to by the Y pointer
        ; the first data location in Flash is pointed to by the Z-pointer
        ;-error handling is not included
        ;-checking that the PLL is in lock 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, EEWE
        rjmp                Wait_ee
        ;       SPM timed sequence
        out                 SPMCSR, spmcrval
        spm
        ;       restore SREG (to enable interrupts if originally enabled)
        out                 SREG, temp2
        ret
```

### 29.8.13 Atmel ATmega32HVE Boot Loader Parameters

In Table 29-5 through Table 29-7, the parameters used in the description of the Self-Programming are given.

**Table 29-5. Boot Size Configuration(1)**

| 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 | 0x0000 - 0x3EFF | 0x3F00 - 0x3FFF | 0x3EFF | 0x3F00 |
| 1 | 0 | 512 words | 8 | 0x0000 - 0x3DFF | 0x3E00 - 0x3FFF | 0x3DFF | 0x3E00 |
| 0 | 1 | 1024 words | 16 | 0x0000 - 0x3BFF | 0x3C00 - 0x3FFF | 0x3BFF | 0x3C00 |
| 0 | 0 | 2048 words | 32 | 0x0000 - 0x37FF | 0x3800 - 0x3FFF | 0x37FF | 0x3800 |

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 29-2

**Table 29-6. Read-While-Write Limit(1)**

| Section | Pages | Address |
|---|---|---|
| Read-While-Write section (RWW) | 224 | 0x0000 - 0x37FF |
| No Read-While-Write section (NRWW) | 32 | 0x3800 - 0x3FFF |

Note: 1. For details about these two section, see Section 29.4.2 “NRWW – No Read-While-Write Section” on page 168 and Section 29.4.1 “RWW – Read-While-Write Section” on page 168.

**Table 29-7. Explanation of Different Variables Used in Figure 29-3 on page 170 and the Mapping to the Z-pointer(1)**

| Variable | | Corresponding Z-value | Description |
|---|---|---|---|
| PCMSB | 13 | | Most significant bit in the Program Counter. (The Program Counter is 14 bits PC[13:0]) |
| PAGEMSB | 5 | | Most significant bit which is used to address the words within one page (64 words in a page requires six bits PC [5:0]). |
| ZPCMSB | | Z14 | Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1. |
| ZPAGEMSB | | Z6 | Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1. |
| PCPAGE | PC[13:6] | Z13:Z7 | Program Counter page address: Page select, for Page Erase and Page Write |
| PCWORD | PC[5:0] | Z6:Z1 | Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation) |

Note: 1. Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See Section 29.7 “Addressing the Flash During Self-Programming” on page 170 for details about the use of Z-pointer during Self-Programming.

### 29.8.14 Atmel ATmega64HVE Boot Loader Parameters

In Table 29-5 through Table 29-7, the parameters used in the description of the Self-Programming are given.

**Table 29-8. Boot Size Configuration(1)**

| 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 | 0x0000 - 0x7EFF | 0x7F00 - 0x7FFF | 0x7EFF | 0x7F00 |
| 1 | 0 | 512 words | 8 | 0x0000 - 0x7DFF | 0x7E00 - 0x7FFF | 0x7DFF | 0x7E00 |
| 0 | 1 | 1024 words | 16 | 0x0000 - 0x7BFF | 0x7C00 - 0x7FFF | 0x7BFF | 0x7C00 |
| 0 | 0 | 2048 words | 32 | 0x0000 - 0x77FF | 0x7800 - 0x7FFF | 0x77FF | 0x7800 |

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 29-2

**Table 29-9. Read-While-Write Limit(1)**

| Section | Pages | Address |
|---|---|---|
| Read-While-Write section (RWW) | 480 | 0x0000 - 0x77FF |
| No Read-While-Write section (NRWW) | 32 | 0x7800 - 0x7FFF |

Note: 1. For details about these two section, see Section 29.4.2 “NRWW – No Read-While-Write Section” on page 168 and Section 29.4.1 “RWW – Read-While-Write Section” on page 168.

**Table 29-10. Explanation of Different Variables used in Figure 29-3 on page 170 and the Mapping to the Z-pointer(1)**

| Variable | | Corresponding Z-value | Description |
|---|---|---|---|
| PCMSB | 14 | | Most significant bit in the Program Counter. (The Program Counter is 15 bits PC[14:0]) |
| PAGEMSB | 5 | | Most significant bit which is used to address the words within one page (64 words in a page requires six bits PC [5:0]). |
| ZPCMSB | | Z15 | Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1. |
| ZPAGEMSB | | Z6 | Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1. |
| PCPAGE | PC[14:6] | Z14:Z7 | Program Counter page address: Page select, for Page Erase and Page Write |
| PCWORD | PC[5:0] | Z6:Z1 | Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation) |

Note: 1. Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See Section 29.7 “Addressing the Flash During Self-Programming” on page 170 for details about the use of Z-pointer during Self-Programming.

## 29.9 Register Description

### 29.9.1 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 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x37 (0x57) | SPMIE | RWWSB | SIGRD | RWWSRE | LBSET | 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 Section 29.8.9 “Reading the Signature Row from Software” on page 173 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 – LBSET: 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 Lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The LBSET bit will automatically be cleared upon completion of the Lock bit set, or after six cycles if no SPM instruction is executed within four clock cycles.

  An LPM instruction within three cycles after LBSET 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 Section 29.8.8 “Reading the Fuse and Lock Bits from Software” on page 172 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 after six cycles 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 after six cycles 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: Store Program Memory Enable**

  This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, LBSET, 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 after six cycles 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.

# 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 30-2. Lock Bit Protection Modes[1][2] (Continued)**

| Memory Lock Bits | | | Protection Type |
|---|---|---|---|
| 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

## 30.2 Fuse Bits

The Atmel® AVR MCU has two Fuse bytes. Table 30-4 and Table 30-3 describe briefly the functionality of all the fuses and how they are mapped into the Fuse byte. Note that the fuses are read as logical zero, “0”, if they are programmed.

### 30.2.1 High Byte

**Table 30-3. Fuse High Byte**

| Bit No | Fuse High Byte | Description | Default Value |
|---|---|---|---|
| 3 | DWEN | Enable debugWire | 1 (unprogrammed) |
| 2 | BOOTSZ1 | Select Boot Size | 0 (programmed)[1] |
| 1 | BOOTSZ0 | Selecrt Boot Size | 0 (programmed)[1] |
| 0 | BOOTRST | Select Reset Vector | 1 (unprogrammed) |

Note: 1. The default value of BOOTSZ1:0 results in maximum Boot Size.

### 30.2.2 Low Byte

**Table 30-4.** Fuse Low Byte

| Bit No | Fuse Low Byte | Description | Default Value |
|---|---|---|---|
| 7 | WDTON | Watchdog Timer always on | 1 (unprogrammed)(1) |
| 6 | EESAVE | EEPROM memory is preserved through the Chip Erase | 1 (unprogrammed, EEPROM not preserved) |
| 5 | SPIEN | Enable Serial Programmable Data Downloading | 0 (programmed, SPI programming enabled) |
| 4 | BODEN | BOD Enable | 1 (unprogrammed, BOD disabled)(5) |
| 3 | CKDIV8(3) | Divide clock by 8 | 0 (programmed) |
| 2:1 | SUT1:0 | Select start-up time | 11 (unprogrammed)(2) |
| 0 | OSCSEL0 | Oscillator Select | 1 (unprogrammed)(4) |

Notes:
1. The Watchdog is enabled/disabling by writing to the Watchdog Timer Control and Status Register (WDTCSR). But as a fail-safe, the WDTON fuse can be used to force the Watchdog to run in System Reset mode.
2. The SUTx fuse bits are used to configure the startup time from sleep or reset. By default the longest startup time is selected.
3. See Section 15.4 “System Clock Prescaler” on page 50 for details.
4. When unprogrammed, PLL is used as system clock source. Programming this fuse is for test purpose only, and should not be used in application.
5. Disabling BOD assumed that safe VCC operation is guaranteed by other parts of the application.

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.

### 30.2.3 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.

## 30.3 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both Programming mode, also when the device is locked. The three bytes reside in a separate address space. The signature bytes of the Atmel® AVR MCU are given in Table 30-5.

**Table 30-5.** Device ID

| Part | Signature Bytes Address | | |
|---|---|---|---|
| | 0x000 | 0x001 | 0x002 |
| Atmel ATmega32HVE | 0x1E | 0x95 | 0x13 |
| Atmel ATmega64HVE | 0x1E | 0x96 | 0x10 |

## 30.4 Calibration Bytes

The Atmel® AVR MCU has calibration bytes for the RC Oscillators, internal voltage reference, internal temperature reference and TBD. These bytes reside in the signature address space. See Section 29.8.9 “Reading the Signature Row from Software” on page 173 for details.

## 30.5 Page Size

**Table 30-6. No. of Words in a Page and No. of Pages in the Flash, Atmel AVR MCU**

| Flash Size | Page Size | PCWORD | No. of Pages | PCPAGE | PCMSB |
|---|---|---|---|---|---|
| 16K words ( 32Kbytes) | 64 words | PC[5:0] | 256 | PC[13:6] | 13 |
| 32K words ( 64Kbytes) | 64 words | PC[5:0] | 512 | PC[14:6] | 14 |

**Table 30-7. No. of Words in a Page and No. of Pages in the EEPROM, Atmel AVR MCU**

| EEPROM Size | Page Size | PCWORD | No. of Pages | PCPAGE | EEAMSB |
|---|---|---|---|---|---|
| 1Kbytes | 4 bytes | EEA[1:0] | 256 | EEA[9:2] | 9 |

## 30.6 Serial Programming

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 30-8 on page 183, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

**Figure 30-1. Serial Programming and Verify**

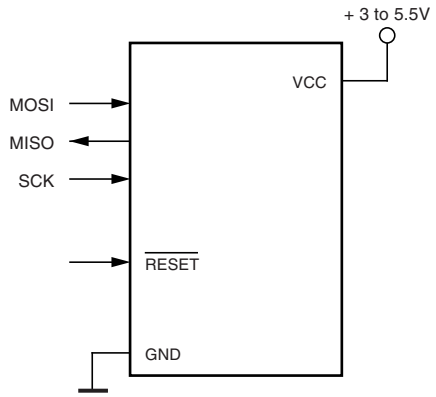



**Table 30-8. Pin Mapping Serial Programming**

| Symbol | Pins | I/O | Description |
|---|---|---|---|
| SCK | PB5 | I | Serial Clock |
| MOSI | PB6 | I | Serial Data in |
| MISO | PB7 | O | Serial Data out |

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 the OSCSEL Fuse, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2.2 CPU clock cycles for $f_{ck}$ < 12 MHz, 3 CPU clock cycles for $f_{ck}$ >= 12 MHz

High: > 2.2 CPU clock cycles for $f_{ck}$ < 12 MHz, 3 CPU clock cycles for $f_{ck}$ >= 12 MHz

### 30.6.1 Serial Programming Algorithm

When writing serial data to the Atmel® AVR MCU, data is clocked on the rising edge of SCK.

When reading data from the Atmel AVR MCU, data is clocked on the falling edge of SCK.

To program and verify the Atmel AVR MCU in the Serial Programming mode, the following sequence is recommended. See four byte instruction formats in following Table 30-10 on page 185

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. For this instruction, minimum low and high periods for the serial clock (SCK) must be doubled.
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. Wait for at least 1.3 ms after successful Programming Enable before issuing any programming commands.
5. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 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 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least $t_{WD_FLASH}$ before issuing the next page. (See Table 30-9.) Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
6. **A:** 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 (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least $t_{WD_EEPROM}$ before issuing the next byte. (See Table 30-9.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   **B:** The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM Memory Page is stored by loading the Write EEPROM Memory Page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least $t_{WD_EEPROM}$ before issuing the next page (See Table 30-9 on page 184). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
7. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
8. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
9. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn $V_{CC}$ power off.

**Table 30-9. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location**

| Symbol | Minimum Wait Delay |
|---|---|
| $t_{WD_FLASH}$ | 4.5ms |
| $t_{WD_EEPROM}$ | 4.0ms |
| $t_{WD_ERASE}$ | 4.0ms |
| $t_{WD_FUSE}$ | 4.5ms |

### 30.6.2 Serial Programming Instruction set

Table 30-10 on page 185 and Figure 30-2 on page 186 describes the Instruction set.

**Table 30-10.** Serial Programming Instruction Set

| Instruction/Operation | Instruction Format | | | |
|---|---|---|---|---|
| | Byte 1 | Byte 2 | Byte 3 | Byte4 |
| Programming Enable | $AC | $53 | $00 | $00 |
| Chip Erase (Program Memory/EEPROM) | $AC | $80 | $00 | $00 |
| Poll RDY/$\overline{\text{BSY}}$ | $F0 | $00 | $00 | data byte out |
| Load Instructions | | | | |
| Load Extended Address byte(1) | $4D | $00 | Extended adr | $00 |
| Load Program Memory Page, High byte | $48 | adr MSB | adr LSB | high data byte in |
| Load Program Memory Page, Low byte | $40 | adr MSB | adr LSB | low data byte in |
| Load EEPROM Memory Page (page access) | $C1 | adr MSB | adr LSB | data byte in |
| Read Instructions | | | | |
| Read Program Memory, High byte | $28 | adr MSB | adr LSB | high data byte out |
| Read Program Memory, Low byte | $20 | adr MSB | adr LSB | low data byte out |
| Read EEPROM Memory | $A0 | adr MSB | adr LSB | data byte out |
| Read Lock bits | $58 | $00 | $00 | data byte out |
| Read Signature Byte | $30 | $00 | adr LSB | data byte out |
| Read Fuse bits | $50 | $00 | $00 | data byte out |
| Read Fuse High bits | $58 | $08 | $00 | data byte out |
| Read Extended Fuse Bits | $50 | $08 | $00 | data byte out |
| Read Calibration Byte | $38 | $00 | $00 | data byte out |
| Write Instructions(6) | | | | |
| Write Program Memory Page | $4C | adr MSB | adr LSB | $00 |
| Write EEPROM Memory | $C0 | adr MSB | adr LSB | data byte in |
| Write EEPROM Memory Page (page access) | $C2 | adr MSB | adr LSB | $00 |
| Write Lock bits | $AC | $E0 | $00 | data byte in |
| Write Fuse bits | $AC | $A0 | $00 | data byte in |
| Write Fuse High bits | $AC | $A8 | $00 | data byte in |
| Write Extended Fuse Bits | $AC | $A4 | $00 | data byte in |

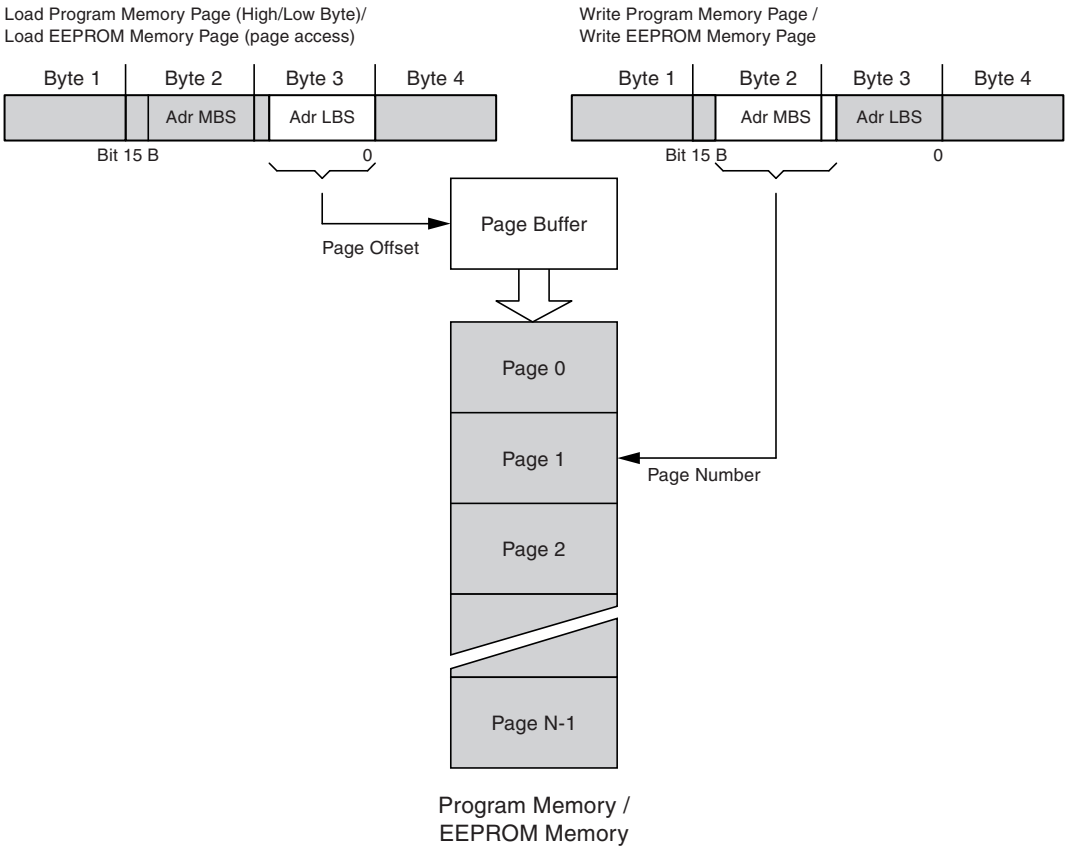
Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. Instructions accessing program memory use word address. This address may be random within the page range.
7. See htt://www.atmel.com/avr for Application Notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 30-2 on page 186.

**Figure 30-2. Serial Programming Instruction Example**




## 30.7 High-Voltage Serial Programming

This section describes how to program and verify Flash Program memory, EEPROM Data memory, Lock bits and Fuse bits in the Atmel® AVR MCU.

**Figure 30-3. High-voltage Serial Programming**

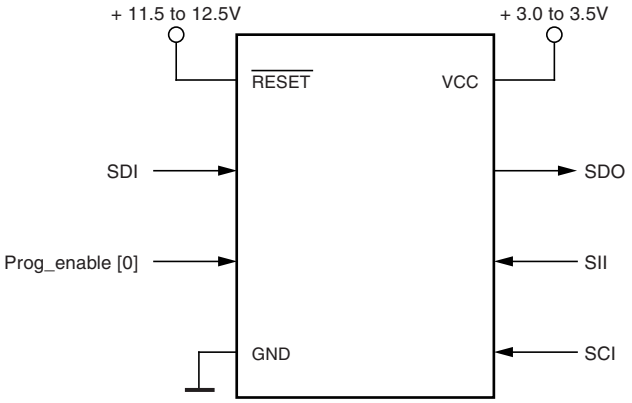


**Table 30-11. Pin Name Mapping**

| Signal Name in High-voltage Serial Programming Mode | Pin Name | I/O | Function |
|---|---|---|---|
| SDO | PB5 | O | Serial Data Output |
| SDI | PB6 | I | Serial Data Input |
| SII | PB7 | I | Serial Instruction Input |
| SCI | PB2 | I | Serial Clock Input (min. $2/f_{ck}$ period) |

**Table 30-12. Pin Values Used to Enter Programming Mode**

| Pin Name | Symbol | Value |
|---|---|---|
| PB4 | Prog_enable[0] | 0 |
| PB5 | Prog_enable[1] | 0 |
| PB6 | Prog_enable[2] | 0 |
| PB7 | Prog_enable[3] | 0 |

## 30.8 High-voltage Serial Programming Algorithm

To program and verify the Atmel® AVR MCU in the High-voltage Serial Programming mode, the following sequence is recommended (See instruction formats in Table 30-14):

### 30.8.1 Enter High-voltage Serial Programming Mode

The following algorithm puts the device in Serial (High-voltage) Programming mode:

1. Set Prog_enable pins listed in Table 30-12 on page 187 to “0000”, RESET pin to 0V and $V_{CC}$ to 0V.
2. Apply 3.0 - 3.5V between $V_{CC}$ and GND. Ensure that $V_{CC}$ reaches at least 1.8V within the next 20 µs.
3. Wait 20 - 60 µs, and apply $V_{HRST}$ - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least $t_{HVRST}$ after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Release Prog_enable[1] pin to avoid drive contention on the Prog_enable[1]/SDO pin.
6. Wait at least 1.3 ms before giving any serial instructions on SDI/SII.
7. If the rise time of the $V_{CC}$ is unable to fulfill the requirements listed above, the following alternative algorithm can be used.
8. Set Prog_enable pins listed in Table 30-12 on page 187 to “0000”, RESET pin to 0V and $V_{CC}$ to 0V.
9. Apply 3.0 - 3.5V between $V_{CC}$ and GND.
10. Monitor $V_{CC}$, and as soon as $V_{CC}$ reaches 0.9 - 1.1V, apply $V_{HRST}$ - 12.5V to RESET.
11. Keep the Prog_enable pins unchanged for at least $t_{HVRST}$ after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
12. Release Prog_enable[1] pin to avoid drive contention on the Prog_enable[1]/SDO pin.
13. Wait until $V_{CC}$ actually reaches 3.0 - 3.5V.
14. Wait at least 1.3ms before giving any serial instructions on SDI/SII.

**Table 30-13. High-voltage Reset Characteristics**

| Supply Voltage | RESET Pin High-voltage Threshold | Minimum High-voltage Period for Latching Prog_enable |
|---|---|---|
| $V_{CC}$ | $V_{HVRST}$ | $t_{HVRST}$ |
| 3.0V | 11.5V | 10 µs |
| 3.5V | 11.5V | 10 µs |

### 30.8.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address High byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

### 30.8.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM[1] memories plus Lock bits. The Lock bits are not reset until the Program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are re-programmed.

Note: 1. The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

1. Load command “Chip Erase” (see Table 30-14).
2. Wait after Instr.3 until SDO goes high for the “Chip Erase” cycle to finish.
3. Load Command “No Operation”.

### 30.8.4 Programming the Flash

The Flash is organized in pages, see Table 30-10 on page 185. 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:

1. Load Command “Write Flash” (see Table 30-14).
2. Load Flash Page Buffer.
3. Load Flash High Address and Program Page. Wait after Instr. 3 until SDO goes high for the “Page Programming” cycle to finish.
4. Repeat 2 through 3 until the entire Flash is programmed or until all data has been programmed.
5. End Page Programming by Loading Command “No Operation”.

When writing or reading serial data to the Atmel® AVR MCU, data is clocked on the rising edge of the serial clock, see Figure 30-5, Figure 31-5 and Section 31.8.2 “High-voltage Serial Programming” on page 202 for details.

**Figure 30-4. Addressing the Flash which is Organized in Pages**

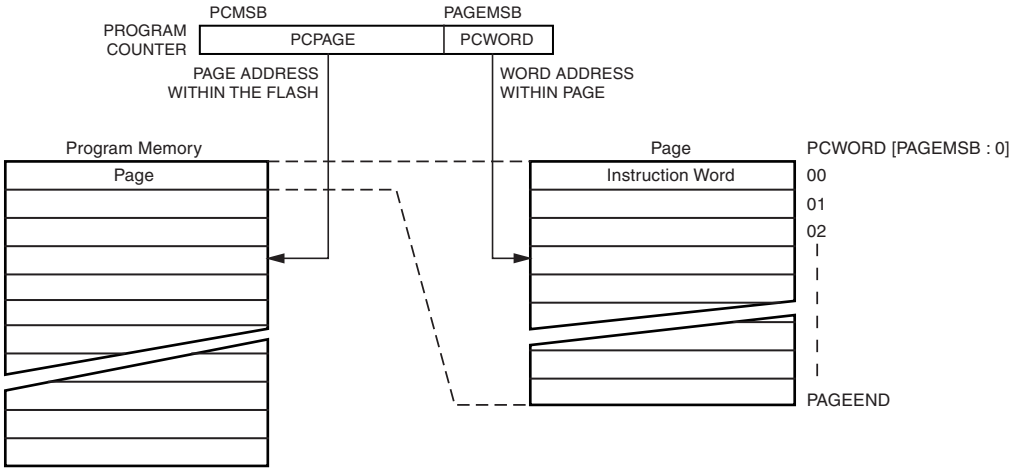
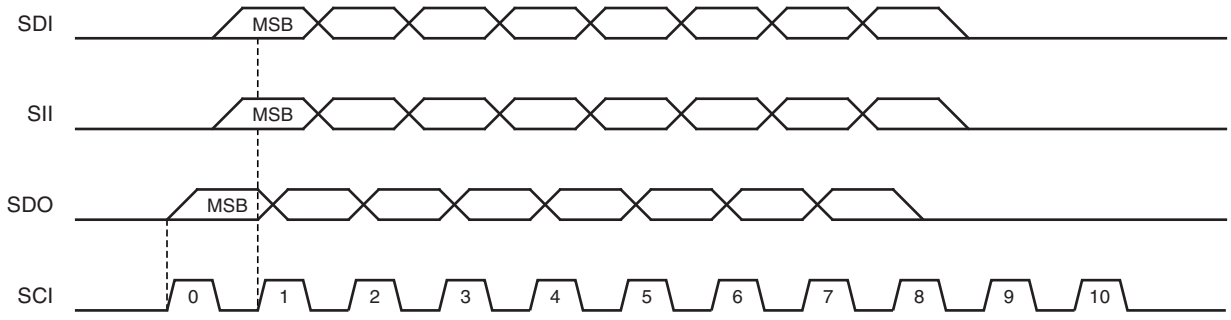


**Figure 30-5. High-voltage Serial Programming Waveforms**




### 30.8.5 Programming the EEPROM

The EEPROM is organized in pages, see Section 31.8.2 “High-voltage Serial Programming” on page 202. When programming the EEPROM, the 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 Table 30-14 on page 190):

1. Load Command “Write EEPROM”.
2. Load EEPROM Page Buffer.
3. Program EEPROM Page. Wait after Instr. 2 until SDO goes high for the “Page Programming” cycle to finish.
4. Repeat 2 through 3 until the entire EEPROM is programmed or until all data has been programmed.
5. End Page Programming by Loading Command “No Operation”.

### 30.8.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to Table 30-14 on page 190):

1. Load Command "Read Flash".
2. Read Flash Low and High Bytes. The contents at the selected address are available at serial output SDO.

### 30.8.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Table 30-14 on page 190):

1. Load Command “Read EEPROM”.
2. Read EEPROM Byte. The contents at the selected address are available at serial output SDO.

### 30.8.8 Programming and Reading the Fuse and Lock Bits

The algorithms for programming and reading the Fuse Low/High bits and Lock bits are shown in Table 30-14 on page 190.

### 30.8.9 Reading the Signature Bytes and Calibration Byte

The algorithms for reading the Signature bytes and Calibration byte are shown in Table 30-14 on page 190.

### 30.8.10 Power-off sequence

Exit Programming mode by powering the device down, or by bringing RESET pin to 0V.

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

| Instruction | | Instruction Format | | | | Operation Remarks |
|---|---|---|---|---|---|---|
| | | Instr.1/5 | Instr.2/6 | Instr.3 | Instr.4 | |
| Chip Erase | SDI | 0_1000_0000_00 | 0_0000_0000_00 | 0_0000_0000_00 | | |
| | SII | 0_0100_1100_00 | 0_0110_0100_00 | 0_0110_1100_00 | | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | | |
| Load “Write Flash” Command | SDI | 0_0001_0000_00 | | | | |
| | SII | 0_0100_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |
| Load Flash Page Buffer | SDI | 0_ **bbbb_bbbb** _00 | 0_**eeee_eeee**_00 | 0_**dddd_dddd**_00 | 0_0000_0000_00 | |
| | SII | 0_0000_1100_00 | 0_0010_1100_00 | 0_0011_1100_00 | 0_0111_1101_00 | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | |
| | SDI | 0_0000_0000_00 | | | | |
| | SII | 0_0111_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |
| Load Flash High Address and Program Page | SDI | 0_**aaaa_aaaa**_00 | 0_0000_0000_00 | 0_0000_0000_00 | | |
| | SII | 0_0001_1100_00 | 0_0110_0100_00 | 0_0110_1100_00 | | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | | |
| Load “Read Flash” Command | SDI | 0_0000_0010_00 | | | | |
| | SII | 0_0100_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |
| Read Flash Low and High Bytes | 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**x_xx | |
| | SDI | 0_0000_0000_00 | 0_0000_0000_00 | | | |
| | SII | 0_0111_1000_00 | 0_0111_1100_00 | | | |
| | SDO | x_xxxx_xxxx_xx | **p_pppp_ppp**x_xx | | | |
| Load “Write EEPROM” Command | SDI | 0_0001_0001_00 | | | | |
| | SII | 0_0100_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |
| Load EEPROM Page Buffer | SDI | 0_**bbbb_bbbb**_00 | 0_**eeee_eeee**_00 | 0_0000_0000_00 | 0_0000_0000_00 | |
| | SII | 0_0000_1100_00 | 0_0010_1100_00 | 0_0110_1101_00 | 0_0110_1100_00 | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | |
| Program EEPROM Page | SDI | 0_0000_0000_00 | 0_0000_0000_00 | | | |
| | SII | 0_0110_0100_00 | 0_0110_1100_00 | | | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | | | |
| Write EEPROM Byte | SDI | 0_**bbbb_bbbb**_00 | 0_**eeee_eeee**_00 | 0_0000_0000_00 | 0_0000_0000_00 | |
| | SII | 0_0000_1100_00 | 0_0010_1100_00 | 0_0110_1101_00 | 0_0110_0100_00 | |
| | SDO | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | x_xxxx_xxxx_xx | |
| | SDI | 0_0000_0000_00 | | | | |
| | SII | 0_0110_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |
| Load “Read EEPROM” Command | SDI | 0_0000_0011_00 | | | | |
| | SII | 0_0100_1100_00 | | | | |
| | SDO | x_xxxx_xxxx_xx | | | | |

**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.

# 31. Electrical Characteristics AVR MCU

Unless otherwise noted all parameters in this section are valid for a supply voltage of 3.0V to 3.6V and a junction temperature range from –40°C to +125°C. All voltages refer to pin GND = 0 if not otherwise specified.

## 31.1 Power Consumption Characteristics

$T_A$= –40°C to 125°C unless otherwise noted

| No. | Parameters | Test Conditions | | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.1 | Active current | All I/O disabled. WUT, WDT, Bandgap enabled | 14.33MHz | VCC, AVCC | | 4.5 | 5.5 | 6.5 | mA | A |
| 1.2 | | All I/O disabled. WUT, WDT, Bandgap enabled | 1.79MHz | | | | 1.7 | | mA | C |
| 1.3 | | All I/O enabled, VADC (Batt channel) and CADC 256x gain. 512kHz ADC clock, WUT, WDT, Bandgap enabled | 14.33MHz | | | 6.0 | 7.2 | 9.0 | mA | A |
| 1.4 | Idle current | All I/O disabled. WUT, WDT, Bandgap enabled, CLKDIV=1 | 14.33MHz | | | | 800 | | µA | C |
| 1.5 | | All I/O disabled. WUT, WDT, Bandgap enabled | 1.79MHz | | | 500 | 800 | 1000 | µA | A |
| 1.6 | | All I/O enabled, VADC (Batt channel) and CADC 256x gain. 512kHz ADC clock, WUT, WDT, Bandgap enabled | 14.33MHz | | | 1800 | 2600 | 3500 | µA | A |
| 1.7 | Reset current | Device in Reset | | VCC, AVCC | | 350 | 500 | 1000 | µA | A |
| 1.8 | Power Save current | WUT, WDT, Bandgap enabled | | VCC, AVCC | | 90 | 105 | 150 | µA | A |
| 1.9 | | CADC with 128kHz clock - 256x gain, WUT, WDT, Bandgap enabled | | | | 200 | 450 | 650 | | A |
| 1.10 | | VADC (Batt channel) and CADC 256x gain. 512kHz ADC clock, WUT, WDT, Bandgap enabled | | | | | 1400 | | | C |
| 1.11 | Power Down current | WUT, WDT, Bandgap (BGSC = 011) enabled | | | | 8 | 15 | 50 | | A |
| | | WUT, WDT, Bandgap (BGSC = 111) enabled | | | | | 0.3 | 40 | | A |

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

## 31.2 Bandgap Curvature Compensated (BGCC)

### 31.2.1 DC Characteristics

| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 2.1 | Initial accuracy | $T_A$ = 25°C | VREF | | 1.097 | 1.1 | 1.103 | V | A |
| 2.2 | Temperature drift | After 2 temp factory calibration | VREF | | –20 | ±5 | +20 | ppm/°C | A |

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

### 31.2.2 AC Characteristics

| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 2.4 | VREF startup time from power-off state | VREF = 0 - 1.1V, CREF = 1.0µF | | | | 2 | 5 | ms | C |
| 2.5 | VREF ripple, toff = 32ms | Measured at 25°C with 10MΩ load | | | | 5 | 10 | mV | C |

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

### 31.2.3 Die Temperature Measurement

| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 2.6 | Measurement of internal Temperature; accuracy | Temperature measurement according to Section 27.3 on page 161 | | | | ±2 | ±5 | °C | A |

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

## 31.3 ADC Characteristics

### 31.3.1 Voltage ADC Characteristics - Operating Conditions

$T_A$= –40°C to +125°C, $V_{CC}$ = 3V to 3.6V, $V_{REF}$ = 1.1V, unless otherwise noted.

| No. | Parameters | Test Conditions | | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 3.1 | FSR, Resolution and Conversion rates | Full Scale Range | | | | 0 | | 1.1 | V | D |
| 3.2 | | Number of bits (unsigned) | Instantaneous output | | | 16 | | | Bits | D |
| 3.3 | | | Accumulate output | | | 16 | 17 | | | D |
| 3.4 | | 1 LSB | Instantaneous output | | | | 16.8 | | µV | D |
| 3.5 | | | Accumulate output | | | | 16.8 | | | D |
| 3.6 | | Inst. output conversion rate | No chopper normal mode | | | 1000 | | 8000 | Hz | D |
| 3.7 | | | No chopper low-power mode | | | 250 | | 2000 | | D |
| 3.8 | | | Fast chopper normal mode | | | 332 | | 2667 | | D |
| 3.9 | | | Fast chopper low-power mode | | | 83 | | 667 | | D |
| 3.10 | | | Slow chopper normal mode | | | 1000 | | 8000 | | D |
| 3.11 | | | Slow chopper low-power mode | | | 250 | | 2000 | | D |
| 3.12 | | Acc. output conversion rate | No chopper normal mode | | | 1.96 | | 2000 | | D |
| 3.13 | | | No chopper low-power mode | | | 0.49 | | 500 | | D |
| 3.14 | | | Fast chopper normal mode | | | 0.64 | | 667 | | D |
| 3.15 | | | Fast chopper low-power mode | | | 0.16 | | 167 | | D |
| 3.16 | | | Slow chopper normal mode | | | 1.96 | | 1333 | | D |
| 3.17 | | | Slow chopper low-power mode | | | 0.49 | | 333 | | D |
| 3.18 | PV2/NV2 | INL without divider | Range 0V to 1V | | | | 0.008 | 0.03 | %FSR | C |
| | | | Range 0.4V to 0.6V | | | | 0.008 | 0.01 | %FSR | C |
| 3.19 | | INL incl divider | Range 9.6V to 14.4V | | | | 0.003 | 0.006 | %FSR | C |
| 3.20 | | Offset[1] | No chopper | | | | ±101 | ±672 | µV | C |
| 3.21 | | | Fast chopper, IC output | | | | ±0 | ±17 | µV | C |
| 3.22 | | | Slow chopper, AC ouput | | | | ±0 | ±17 | µV | C |
| 3.23 | | Offset drift[1] | No chopper | | | | ±0.2 | ±0.5 | µV/°C | C |
| 3.24 | | | Fast chopper, IC output | | | | ±0 | ±17 | µV | C |
| 3.25 | | | Slow chopper, AC ouput | | | | ±0 | ±17 | µV | C |
| 3.26 | | Gain drift | Including VREF drift, range 0V to 1V | | | | ±5 | ±20 | ppm/°C | C |
| 3.27 | | | Including VREF drift, range 0.4V to 0.6V | | | | ±5 | ±20 | ppm/°C | C |
| 3.28 | | Equivalent input impedance | Normal mode | | | | 100 | | MΩ | D |
| 3.29 | | | LP mode | | | | 60 | | | D |

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

Note: 1. Measured in test mode with inputs shorted

### 31.3.1 Voltage ADC Characteristics - Operating Conditions (Continued)

$T_A$= –40°C to +125°C, $V_{CC}$ = 3V to 3.6V, $V_{REF}$ = 1.1V, unless otherwise noted.

| No. | Parameters | Test Conditions | | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 3.30 | ADC0/SGND; ADC1/SGND | INL | Range 0V to 1V | | | | 0.02 | 0.06 | %FSR | C |
| 3.31 | | Offset | No chopper | | | | ±34 | ±336 | µV | C |
| 3.32 | | | Fast chopper | | | | ±0 | ±17 | µV | C |
| 3.33 | | | Slow chopper | | | | ±0 | ±17 | µV | C |
| 3.34 | ADC0/SGND; ADC1/SGND | Offset drift | No chopper | | | | ±0.3 | ±0.8 | µV/°C | C |
| 3.35 | | | Fast chopper | | | | ±0 | ±17 | µV | C |
| 3.36 | | | Slow chopper | | | | ±0 | ±17 | µV | C |
| 3.37 | | Gain drift | Including VREF drift, NOM mode, range 0V to 1V | | | | ±5 | ±25 | ppm/°C | C |
| 3.38 | | | Including VREF drift, LP mode, range 0V to 1V | | | | ±5 | ±25 | ppm/°C | C |
| 3.39 | VREF measurement according to VDAC diagnosis mode, see Section 26.5 on page 149 | VADC reading | | | | –4% | 0.55 | +4% | V | A |
| 3.40 | Pull-up PV2/NV2 open diagnosis | Register ADCRE VADPDM[1:0]=11 | | PV2, NV | | | 22 | | kΩ | A |

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

Note: 1. Measured in test mode with inputs shorted

### 31.3.2 Current ADC Characteristics - Operating Conditions

$T_A$= -40 to 125°C unless otherwise noted.

| No. | Parameters | Test Conditions | | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 3.42 | FSR, Resolution and Conversion rates | Full Scale Range | After gain amplifier at ADC input | | | -660 | | 660 | mV | D |
| 3.43 | | Number of bits | Instantaneous output | | | 16 | | | Bits | D |
| 3.44 | | | Accumulate output | | | 16 | 18 | | | D |
| 3.45 | | 1 LSB referred to 16-bit result | PGA gain = 4 | | | | 5.035 | | µV | D |
| 3.46 | | | PGA gain = 8 | | | | 2.5175 | | | D |
| 3.47 | | | PGA gain = 16 | | | | 1.2588 | | | D |
| 3.48 | | | PGA gain = 32 | | | | 0.6294 | | | D |
| 3.49 | | | PGA gain = 64 | | | | 0.3147 | | | D |
| 3.50 | | | PGA gain = 128 | | | | 0.1573 | | | D |
| 3.51 | | | PGA gain = 256 | | | | 0.0787 | | | D |
| 3.52 | | Inst. output conversion rate(1) | No chopper normal mode | | | 1000 | | 8000 | Hz | D |
| 3.53 | | | No chopper low-power mode | | | 250 | | 2000 | | D |
| 3.54 | | | Fast chopper normal mode | | | 332 | | 2667 | | D |
| 3.55 | | | Fast chopper low-power mode | | | 83 | | 667 | | D |
| 3.56 | | | Slow chopper normal mode | | | 1000 | | 8000 | | D |
| 3.57 | | | Slow chopper low-power mode | | | 250 | | 2000 | | D |
| 3.58 | | Acc. output conversion rate(1) | No chopper normal mode | | | 1.96 | | 2000 | | D |
| 3.59 | | | No chopper low-power mode | | | 0.49 | | 500 | | D |
| 3.60 | | | Fast chopper normal mode | | | 0.64 | | 667 | | D |
| 3.61 | | | Fast chopper low-power mode | | | 0.16 | | 167 | | D |
| 3.62 | | | Slow chopper normal mode | | | 1.96 | | 1333 | | D |
| 3.63 | | | Slow chopper low-power mode | | | 0.49 | | 333 | | D |

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

Notes:
1. Conversion rate scales proportionally to frequency in SlowRC oscillator
2. Measured in Power Save, 256x gain
3. Measured with PI-NI shorted

### 31.3.2 Current ADC Characteristics - Operating Conditions (Continued)

$T_A$= -40 to 125°C unless otherwise noted.

| No. | Parameters | Test Conditions | | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 3.64 | Accuracy | INL | 90% FSR, 256x gain | | | | 0.15 | 0.3 | %FSR | C |
| 3.65 | | | 90% FSR, 4x gain | | | | 0.02 | 0.06 | | C |
| 3.66 | | Offset (referred to input)(2)(3) | No chopper | | | | ±35 | ±150 | µV | C |
| 3.67 | | | Fast chopper | | | | ±2.5 | ±5 | | C |
| 3.68 | | | Slow chopper | | | | ±2.5 | ±5 | | C |
| 3.69 | | Offset drift (referred to input)(2)(3) | No chopper | | | | ±80 | ±300 | nV/°C | C |
| 3.70 | | | Fast chopper | | | | ±6 | ±20 | | C |
| 3.71 | | | Slow chopper | | | | ±6 | ±20 | | C |
| 3.72 | | Uncompensated gain error | Including VREF drift, PGA gain = 4 | | | | 0.05 | 0.1 | %FSR | C |
| 3.73 | | | Including VREF drift, PGA gain = 256 | | | | 1.5 | 2 | | C |
| 3.74 | | Gain drift | Including VREF drift, PGA gain = 4 | | | | ±5 | ±25 | ppm/°C | C |
| 3.75 | | | Including VREF drift, PGA gain = 256 | | | | ±10 | ±50 | | C |
| 3.76 | | Input Impedance | Normal mode | | | | 100 | | kΩ | D |
| 3.77 | | | LP mode | | | | 60 | | | D |
| 3.78 | vrefp_test measurement according to Section 26.5 on page 149 | CADC reading | | | | –4% | 0.33 | +4% | V | A |
| 3.79 | Pull-up PI/NI open diagnosis | Register ADCRD CADPDM[1:0]=11 | | PI, NI | | | 22 | | kΩ | A |

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

Notes:
1. Conversion rate scales proportionally to frequency in SlowRC oscillator
2. Measured in Power Save, 256x gain
3. Measured with PI-NI shorted

## 31.4 Oscillator Characteristics

$T_A$= -40 to 125°C, $V_{CC}$ = 3.3V unless otherwise noted.

| No. | Parameters | | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|---|
| 4.1 | Slow RC Oscillator | Frequency | Reference frequency: Slow RCOSC | | | –4% | 128 | +4% | kHz | A |
| 4.2 | | Temperature drift | Centered at 25°C | | | –1 | | +1 | % | A |
| 4.3 | PLL | Frequency multiplication factor | | | | | 112 | | | D |
| 4.4 | | Startup time from Pdown/Psave | Latency from module enabled until first clock edge (ref = 128kHz) | | | | 25 | 40 | µs | C |
| 4.5 | | Frequency settling time | Setting to 99% of target | | | | 180 | 250 | µs | C |
| 4.6 | Ultra Low Power RC Oscillator | Frequency, initial accuracy | | | | –20% | 131 | +20% | kHz | A |
| 4.7 | | Temperature drift | Centered at 25°C. VCC = 3.3V | | | –5 | | +5 | % | A |

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

## 31.5 External Interrupt Characteristics

Asynchronous External Interrupt Characteristics

| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 5.1 | Pulse width for asynchronous external interrupt | | | $t_{INT}$ | | 500 | 2000 | ns | C |

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

## 31.6 General I/O Lines Characteristics

### 31.6.1 Port A and B Characteristics

$T_A$ = -40 to 125°C, $V_{CC}$ = 3.3V (unless otherwise noted)

| No. | Parameters | Test Conditions | Pin | Symbol | Min. | Typ. | Max. | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 6.1 | Input Low Voltage, Except RESET pin | | PA[1:0], PB[7:0] | $V_{IL}$ | –0.5 | | $0.3V_{CC}$(1) | V | C |
| 6.2 | Input Low Voltage, RESET pin | | Reset | $V_{IL1}$ | | | $0.3V_{CC}$(1) | V | C |
| 6.3 | Input High Voltage, Except RESET pin | | PA[1:0], PB[7:0] | $V_{IH}$ | $0.6V_{CC}$(2) | | $V_{CC}$ + 0.5 | V | C |
| 6.4 | Input High Voltage, RESET pin | | Reset | $V_{IH1}$ | $0.9V_{CC}$(2) | | $V_{CC}$ + 0.5 | V | C |
| 6.5 | Output Low Voltage(3) | $I_{OL}$ = 5mA | PA[1:0], PB[7:0] | $V_{OL}$ | | | 0.5 | V | A |
| 6.6 | Output High Voltage(4) | $I_{OH}$ = 2 mA | PA[1:0], PB[7:0] | $V_{OH}$ | 2.3 | | | V | A |
| 6.7 | Input Leakage Current I/O Pin | Pin low (absolute value) | PA[1:0], PB[7:0] | $I_{IL}$ | | | ±1 | µA | A |
| 6.8 | Input Leakage Current I/O Pin | Pin high (absolute value) | PA[1:0], PB[7:0] | $I_{IH}$ | | | ±1 | µA | A |
| 6.9 | Reset Pull-up Resistor | | Reset | $R_{RST}$ | 30 | | 60 | kΩ | A |
| 6.10 | I/O Pin Pull-up Resistor | | PA[1:0], PB[7:0] | $R_{PU}$ | 20 | | 50 | kΩ | A |

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

Notes:
1. “Max” means the highest value where the pin is guaranteed to be read as low
2. “Min” means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (5 mA at $V_{CC}$ = 3.3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all IOL should not exceed 20 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (2 mA at $V_{CC}$ = 3.3V) under steady state conditions (non-transient), the following must be observed:
   - The sum of all IOH should not exceed 2 mA.

## 31.7 SPI Timing Characteristics

See Figure 31-1 on page 200 and Figure 31-2 on page 201 for details.

### 31.7.1 SPI Timing Parameters

| No. | Parameters | Test Conditions | Pin | Symbol | Min. | Typ. | Max. | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 7.1 | SCK period | Master | | | | See | | | D |
| 7.2 | SCK high/low | Master | | | | 50% duty | | | D |
| 7.3 | Rise/Fall time | Master | | | | 3.6 | | | D |
| 7.4 | Setup | Master | | | | 10 | | | D |
| 7.5 | Hold | Master | | | | 10 | | | D |
| 7.6 | Out to SCK | Master | | | | $0.5 \times t_{sck}$ | | ns | D |
| 7.7 | SCK to out | Master | | | | 10 | | | D |
| 7.8 | SCK to out high | Master | | | | 10 | | | D |
| 7.9 | $\overline{SS}$ low to out | Slave | | | | 15 | | | D |
| 7.10 | SCK period | Slave | | | $4 \times t_{ck}$ + | | | | D |
| 7.11 | SCK high/low(1) | Slave | | | $2 \times t_{ck}$ + | | | | D |
| 7.12 | Rise/Fall time | Slave | | | | 1.6 | | µs | D |
| 7.13 | Setup | Slave | | | 10 | | | | D |
| 7.14 | Hold | Slave | | | $t_{ck}$ | | | | D |
| 7.15 | SCK to out | Slave | | | | 15 | | ns | D |
| 7.16 | SCK to $\overline{SS}$ high | Slave | | | 20 | | | | D |
| 7.17 | $\overline{SS}$ high to tri-state | Slave | | | | 10 | | | D |
| 7.18 | $\overline{SS}$ low to SCK | Slave | | | 20 | | | | D |

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

Note: 1. Refer to Section 30.6 “Serial Programming” on page 183 for serial programming requirements.

**Figure 31-1. SPI Interface Timing Requirements (Master Mode)**

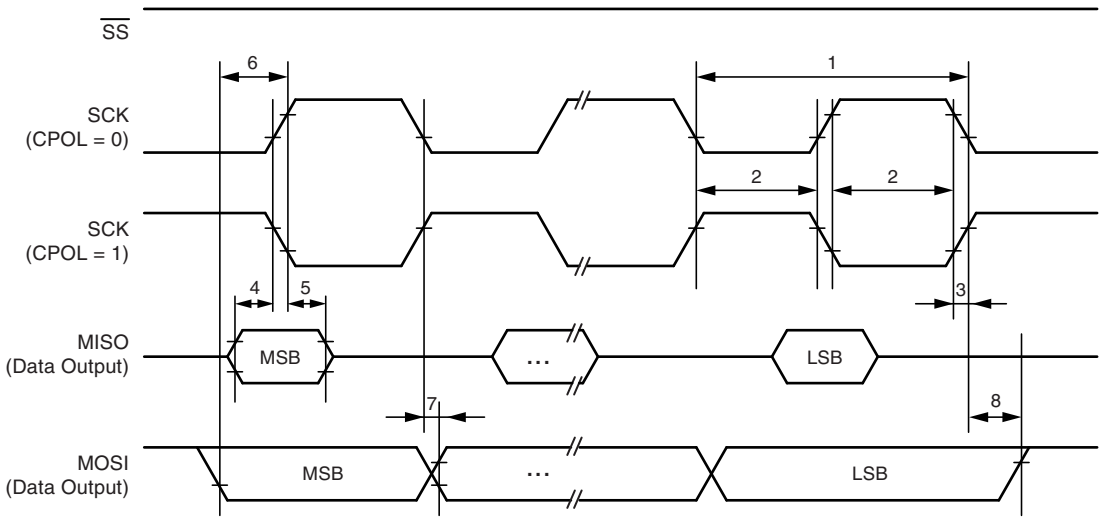
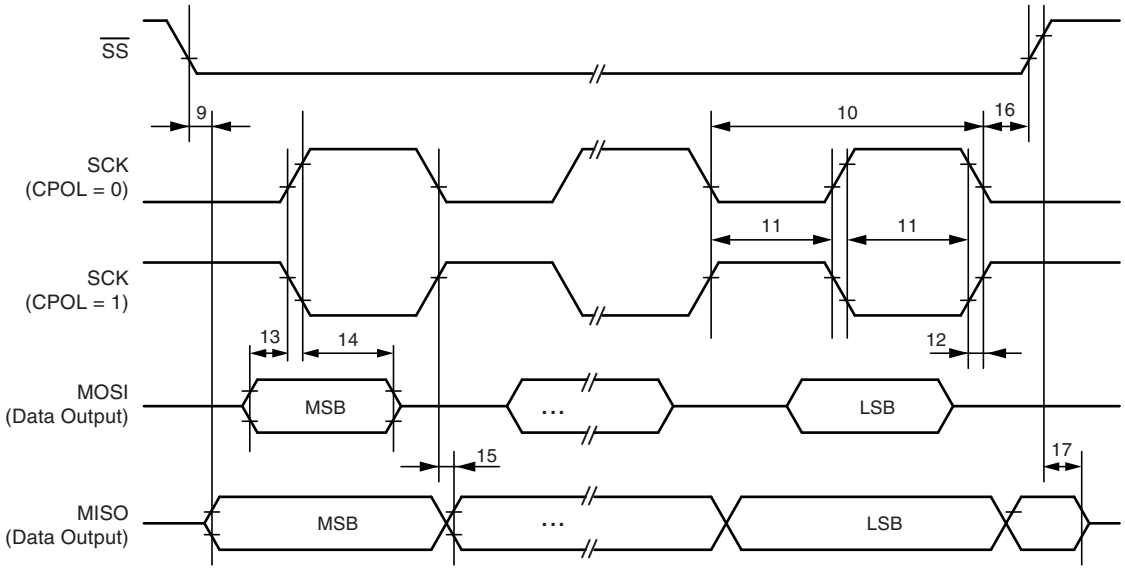


**Figure 31-2. SPI Interface Timing Requirements (Slave Mode)**




## 31.8 Programming Characteristics

### 31.8.1 Serial Programming

**Figure 31-3. Serial Programming Timing**

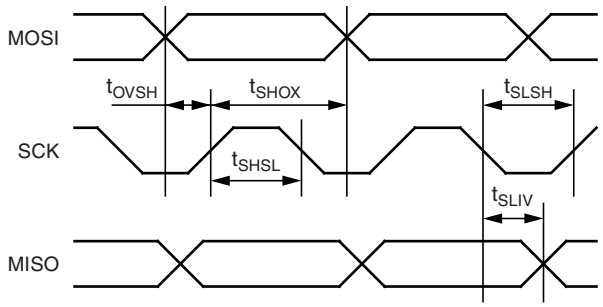
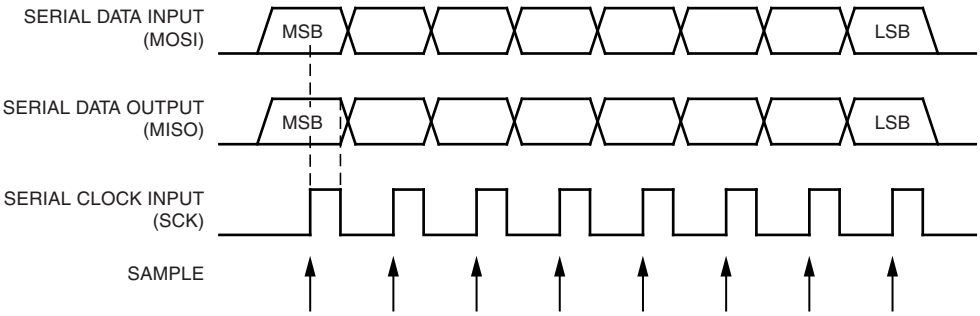



**Figure 31-4. Serial Programming Waveforms**



### 31.8.1.1 Serial Programming Characteristics

$T_A$ = -10°C to 70°C, $V_{CC}$ = 3.0 - 5.5V (Unless Otherwise Noted)

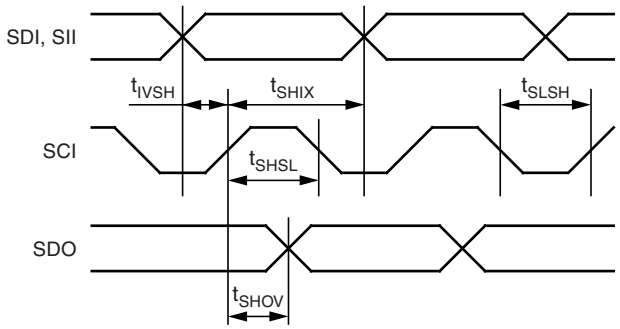
| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 8.1 | Oscillator Frequency (Atmel AVR MCU) | | | $1/t_{CLCL}$ | 0 | | 4 | MHz | D |
| 8.2 | Oscillator Period (Atmel AVR MCU) | | | $t_{CLCL}$ | 250 | | | ns | D |
| 8.3 | SCK Pulse Width High | | | $t_{SHSL}$ | 2.2 $t_{CLCL}$ (1) | | | | D |
| 8.4 | SCK Pulse Width Low | | | $t_{SLSH}$ | 2.2 $t_{CLCL}$ (1) | | | | D |
| 8.5 | MOSI Setup to SCK High | | | $t_{OVSH}$ | $t_{CLCL}$ | | | | D |
| 8.6 | MOSI Hold after SCK High | | | $t_{SHOX}$ | 2 $t_{CLCL}$ | | | | D |
| 8.7 | SCK Low to MISO Valid | | | $t_{SLIV}$ | | 15 | | ns | D |

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

Note: 1. 2.2 $t_{CLCL}$ for $f_{ck}$ < 12MHz, 3.3 $t_{CLCL}$ for $f_{ck}$ ≥ 12MHz

### 31.8.2 High-voltage Serial Programming

**Figure 31-5. High-voltage Serial Programming Timing**




### 31.8.2.1 High-voltage Serial Programming Characteristics $T_A$ = 25°C ± 10%, $V_{CC}$ = 3.3V ± 10% (Unless otherwise noted)

| No. | Parameters | Test Conditions | Pin | Symbol | Min | Typ | Max | Unit | Type* |
|---|---|---|---|---|---|---|---|---|---|
| 8.8 | SCI (PC0) Pulse Width High | | | $t_{SHSL}$ | $1/f_{ck}$ | | | ns | D |
| 8.9 | SCI (PC0) Pulse Width Low | | | $t_{SLSH}$ | $1/f_{ck}$ | | | ns | D |
| 8.10 | SDI (PB2), SII (PB3) Valid to SCI (PC0) High | | | $t_{IVSH}$ | 50 | | | ns | D |
| 8.11 | SDI (PB2), SII (PB3) Hold after SCI (PC0) High | | | $t_{SHIX}$ | 50 | | | ns | D |
| 8.12 | SCI (PC0) High to SDO (PB1) Valid | | | $t_{SHOV}$ | | 16 | | ns | D |
| 8.13 | Wait after Instr. 3 for Write Fuse Bits | | | $t_{WLWH_PFB}$ | | 2.5 | | ms | D |

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

# 32. 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) | VADAC3 | VADAC3[31:24] | | | | | | | | 158 |
| (0xF5) | VADAC2 | VADAC2[23:16] | | | | | | | | 158 |
| (0xF4) | VADAC1 | VADAC1[15:8] | | | | | | | | 158 |
| (0xF3) | VADAC0 | VADAC0[7:0] | | | | | | | | 158 |
| (0xF2) | VADICH | VADICH[15:8] | | | | | | | | 158 |
| (0xF1) | VADICL | VADICL[7:0] | | | | | | | | 158 |
| (0xF0) | CADAC3 | CADAC3[31:24] | | | | | | | | 159 |
| (0xEF) | CADAC2 | CADAC2[23:16] | | | | | | | | 159 |
| (0xEE) | CADAC1 | CADAC1[15:8] | | | | | | | | 159 |
| (0xED) | CADAC0 | CADAC0[7:0] | | | | | | | | 159 |
| (0xEC) | CADICH | CADICH[15:8] | | | | | | | | 159 |
| (0xEB) | CADICL | CADICL[7:0] | | | | | | | | 159 |
| (0xEA) | CADRCLH | CADRCLH[15:8] | | | | | | | | 157 |
| (0xE9) | CADRCLL | CADRCLL[7:0] | | | | | | | | 157 |
| (0xE8) | ADIMR | – | – | VADACIE | VADICIE | – | CADRCIE | CADACIE | CADICIE | 157 |
| (0xE7) | ADIFR | – | – | VADACIF | VADICIF | – | CADRCIF | CADACIF | CADICIF | 156 |
| (0xE6) | ADCRE | VADEN | – | VADREFS | VADPDM1 | VADPDM0 | VAMUX2 | VAMUX1 | VAMUX0 | 155 |
| (0xE5) | ADCRD | – | – | CADG2 | CADG1 | CADG0 | CADPDM1 | CADPDM0 | CADDSEL | 154 |
| (0xE4) | ADCRC | CADEN | – | CADRCM1 | CADRCM0 | CADRCT3 | CADRCT2 | CADRCT1 | CADRCT0 | 153 |
| (0xE3) | ADCRB | – | – | – | ADIDES1 | ADIDES0 | ADADES2 | ADADES1 | ADADES0 | 152 |
| (0xE2) | ADCRA | – | – | – | – | ADPSEL | ADCMS1 | ADCMS0 | CKSEL | 151 |
| (0xE1) | ADSCSRB | – | VADICPS | VADACRB | VADICRB | – | CADICPS | CADACRB | CADICRB | 150 |
| (0xE0) | ADSCSRA | – | – | – | – | – | SBSY | SCMD1 | SCMD0 | 149 |
| (0xDF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDC) | PBOV | PBOVCE | – | – | – | PBOE3 | – | – | PBOE0 | 89 |
| (0xDB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD8) | PLLCSR | – | – | SWEN | LOCK | – | – | PLLCIF | PLLCIE | 51 |
| (0xD7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD4) | BGLR | – | – | – | – | – | – | BGPLE | BPGL | 164 |
| (0xD3) | BGCRA | BGCN[7:0] | | | | | | | | 163 |
| (0xD2) | BGCRB | BGCL[7:0] | | | | | | | | 163 |
| (0xD1) | BGCSRA | – | – | – | – | – | | BGSC[2:0] | | 163 |
| (0xD0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCD) | Reserved | – | – | – | – | – | – | – | – | |

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $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.

# 32. Register Summary (Continued)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| (0xCC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCA) | LINDAT | LDATA[7:0] | | | | | | | | 137 |
| (0xC9) | LINSEL | – | – | – | – | $\overline{\text{LAINC}}$ | | LINDX[2:0] | | 137 |
| (0xC8) | LINIDR | LP1 | LP0 | LID5/LIDL1 | LID4/LIDL0 | | LID[3:0] | | | 136 |
| (0xC7) | LINLDR | LTXDL[7:0] | | | | | | | | 136 |
| (0xC6) | LINBRH | – | – | – | – | | LDIV[11:8] | | | 135 |
| (0xC5) | LINBRL | LDIV[7:0] | | | | | | | | 135 |
| (0xC4) | LINBTR | LDISR | – | LBT[5:0] | | | | | | 135 |
| (0xC3) | LINERR | LABORT | LTOERR | LOVERR | LFERR | LSERR | LPERR | LCERR | LBERR | 134 |
| (0xC2) | LINENIR | – | – | – | – | LENERR | LENIDOK | LENTXOK | LENRXOK | 134 |
| (0xC1) | LINSIR | | LIDST[2:0] | | LBUSY | LERR | LIDOK | LTXOK | LRXOK | 133 |
| (0xC0) | LINCR | LSWRES | LIN13 | LCONF[1:0] | | LENA | | LCMD[2:0] | | 132 |
| (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 | – | – | – | – | – | – | – | – | |

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.

## 32. Register Summary (Continued)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| (0x99) | Reserved | – | – | – | – | – | – | – | – | |
| (0x98) | Reserved | – | – | – | – | – | – | – | – | |
| (0x97) | Reserved | – | – | – | – | – | – | – | – | |
| (0x96) | Reserved | – | – | – | – | – | – | – | – | |
| (0x95) | Reserved | – | – | – | – | – | – | – | – | |
| (0x94) | Reserved | – | – | – | – | – | – | – | – | |
| (0x93) | Reserved | – | – | – | – | – | – | – | – | |
| (0x92) | Reserved | – | – | – | – | – | – | – | – | |
| (0x91) | Reserved | – | – | – | – | – | – | – | – | |
| (0x90) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8F) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8E) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8D) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8C) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8B) | Reserved | – | – | – | – | – | – | – | – | |
| (0x8A) | Reserved | – | – | – | – | – | – | – | – | |
| (0x89) | OCR1B | Timer/Counter1 – Output Compare Register B | | | | | | | | 106 |
| (0x88) | OCR1A | Timer/Counter1 – Output Compare Register A | | | | | | | | 106 |
| (0x87) | Reserved | – | – | – | – | – | – | – | – | |
| (0x86) | Reserved | – | – | – | – | – | – | – | – | |
| (0x85) | TCNT1H | Timer/Counter1 (8 Bit) High Byte | | | | | | | | 106 |
| (0x84) | TCNT1L | Timer/Counter1 (8 Bit) Low Byte | | | | | | | | 105 |
| (0x83) | Reserved | – | – | – | – | – | – | – | – | |
| (0x82) | TCCR1C | – | – | – | – | – | – | ICS11 | ICS10 | 105 |
| (0x81) | TCCR1B | – | – | – | – | – | CS12 | CS11 | CS10 | 92 |
| (0x80) | TCCR1A | TCW1 | ICEN1 | ICNC1 | ICES1 | – | – | – | WGM10 | 104 |
| (0x7F) | Reserved | – | – | – | – | – | – | – | – | |
| (0x7E) | DIDR0 | – | – | – | – | – | – | PA1DID | PA0DID | 160 |
| (0x7D) | Reserved | – | – | – | – | – | – | – | – | |
| (0x7C) | Reserved | – | – | – | – | – | – | – | – | |
| (0x7B) | Reserved | – | – | – | – | – | – | – | – | |
| (0x7A) | Reserved | – | – | – | – | – | – | – | – | |
| (0x79) | Reserved | – | – | – | – | – | – | – | – | |
| (0x78) | Reserved | – | – | – | – | – | – | – | – | |
| (0x77) | Reserved | – | – | – | – | – | – | – | – | |
| (0x76) | Reserved | – | – | – | – | – | – | – | – | |
| (0x75) | Reserved | – | – | – | – | – | – | – | – | |
| (0x74) | Reserved | – | – | – | – | – | – | – | – | |
| (0x73) | Reserved | – | – | – | – | – | – | – | – | |
| (0x72) | Reserved | – | – | – | – | – | – | – | – | |
| (0x71) | Reserved | – | – | – | – | – | – | – | – | |
| (0x70) | Reserved | – | – | – | – | – | – | – | – | |
| (0x6F) | TIMSK1 | – | – | – | – | ICIE1 | OCIE1B | OCIE1A | TOIE1 | 106 |
| (0x6E) | TIMSK0 | – | – | – | – | ICIE0 | OCIE0B | OCIE0A | TOIE0 | 106 |
| (0x6D) | Reserved | – | – | – | – | – | – | – | – | |
| (0x6C) | PCMSK1 | PCINT[9:2] | | | | | | | | 77 |
| (0x6B) | PCMSK0 | – | – | – | – | – | – | PCINT[1:0] | | 77 |
| (0x6A) | Reserved | – | – | – | – | – | – | – | – | |
| (0x69) | EICRA | – | – | – | – | – | – | ISC01 | ISC00 | 75 |
| (0x68) | PCICR | – | – | – | – | – | – | PCIE1 | PCIE0 | 76 |
| (0x67) | SOSCCALB | Slow RC Oscillator Calibration Register B | | | | | | | | 51 |

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.

## 32. Register Summary (Continued)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| (0x66) | SOSCCALA | Slow RC Oscillator Calibration Register A | | | | | | | | 51 |
| (0x65) | Reserved | – | – | – | – | – | – | – | – | |
| (0x64) | PRR0 | – | | | – | PRLIN | PRSPI | PRTIM1 | PRTIM0 | 57 |
| (0x63) | WDTCLR | – | – | – | – | – | WDCL1 | WDCL0 | WDCLE | |
| (0x62) | WUTCSR | WUTIF | WUTIE | – | WUTR | WUTE | WUTP2 | WUTP1 | WUTP0 | |
| (0x61) | CLKPR | CLKPCE | – | – | – | – | – | CLKPS1 | CLKPS0 | 52 |
| (0x60) | WDTCSR | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | 65 |
| 0x3F (0x5F) | SREG | I | T | H | S | V | N | Z | C | 36 |
| 0x3E (0x5E) | SPH | SP15 | SP14 | SP13 | SP12 | SP11 | SP10 | SP9 | SP8 | 38 |
| 0x3D (0x5D) | SPL | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | 38 |
| 0x3C (0x5C) | Reserved | – | – | – | – | – | – | – | – | |
| 0x3B (0x5B) | Reserved | – | – | – | – | – | – | – | – | |
| 0x3A (0x5A) | Reserved | – | – | – | – | – | – | – | – | |
| 0x39 (0x59) | Reserved | – | – | – | – | – | – | – | – | |
| 0x38 (0x58) | Reserved | – | – | – | – | – | – | – | – | |
| 0x37 (0x57) | SPMCSR | – | – | SIGRD | CTPB | RFLB | PGWRT | PGERS | SPMEN | 178 |
| 0x36 (0x56) | Reserved | – | – | – | – | – | – | – | – | |
| 0x35 (0x55) | MCUCR | – | – | CKOE | PUD | – | – | IVSEL | IVCE | 88/52 |
| 0x34 (0x54) | MCUSR | – | – | – | OCDRF | WDRF | BODRF | EXTRF | PORF | 65 |
| 0x33 (0x53) | SMCR | – | – | – | – | | SM[2:0] | | SE | 56 |
| 0x32 (0x52) | Reserved | – | – | – | – | – | – | – | – | |
| 0x31 (0x51) | DWDR | debugWIRE Data Register | | | | | | | | 166 |
| 0x30 (0x50) | Reserved | – | – | – | – | – | – | – | – | |
| 0x2F (0x4F) | TCCR0C | – | – | – | – | – | – | ICS01 | ICS00 | 105 |
| 0x2E (0x4E) | SPDR | SPI Data Register | | | | | | | | 115 |
| 0x2D (0x4D) | SPSR | SPIF | WCOL | – | – | – | – | – | SPI2X | 115 |
| 0x2C (0x4C) | SPCR | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 | 114 |
| 0x2B (0x4B) | GPIOR2 | General Purpose I/O Register 2 | | | | | | | | 47 |
| 0x2A (0x4A) | GPIOR1 | General Purpose I/O Register 1 | | | | | | | | 47 |
| 0x29 (0x49) | OCR0B | Timer/Counter0 Output Compare Register B | | | | | | | | 106 |
| 0x28 (0x48) | OCR0A | Timer/Counter0 Output Compare Register A | | | | | | | | 106 |
| 0x27 (0x47) | TCNT0H | Timer/Counter0 (8 Bit) High Byte | | | | | | | | 106 |
| 0x26 (0x46) | TCNT0L | Timer/Counter0 (8 Bit) Low Byte | | | | | | | | 105 |
| 0x25 (0x45) | TCCR0B | – | – | – | – | – | CS02 | CS01 | CS00 | 92 |
| 0x24 (0x44) | TCCR0A | TCW0 | ICEN0 | ICNC0 | ICES0 | – | – | – | WGM00 | 104 |
| 0x23 (0x43) | GTCCR | TSM | – | – | – | – | – | – | PSRSYNC | |
| 0x22 (0x42) | EEARH | – | – | – | – | – | – | EEPROM High byte | | 44 |
| 0x21 (0x41) | EEARL | High Byte only 1 bit | | | | | | | | 44 |
| 0x20 (0x40) | EEDR | EEPROM Data Register | | | | | | | | 44 |
| 0x1F (0x3F) | EECR | – | – | EEPM1 | EEPM0 | EERIE | EEMPE | EEPE | EERE | 44 |
| 0x1E (0x3E) | GPIOR0 | General Purpose I/O Register 0 | | | | | | | | 47 |
| 0x1D (0x3D) | EIMSK | – | – | – | – | – | – | – | INT0 | 75 |
| 0x1C (0x3C) | EIFR | – | – | – | – | – | – | – | INTF0 | 76 |
| 0x1B (0x3B) | PCIFR | – | – | – | – | – | – | PCIF1 | PCIF0 | 76 |
| 0x1A (0x3A) | Reserved | – | – | – | – | – | – | – | – | |
| 0x19 (0x39) | Reserved | – | – | – | – | – | – | – | – | |
| 0x18 (0x38) | Reserved | – | – | – | – | – | – | – | – | |
| 0x17 (0x37) | Reserved | – | – | – | – | – | – | – | – | |
| 0x16 (0x36) | TIFR1 | – | – | – | – | ICF1 | OCF1B | OCF1A | TOV1 | 107 |
| 0x15 (0x35) | TIFR0 | – | – | – | – | ICF0 | OCF0B | OCF0A | TOV0 | 107 |
| 0x14 (0x34) | Reserved | – | – | – | – | – | – | – | – | |

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $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.

## 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.

# 33. Instruction Set Summary

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| ARITHMETIC AND LOGIC INSTRUCTIONS | | | | | |
| ADD | Rd, Rr | Add two Registers | Rd ←Rd + Rr | Z,C,N,V,H | 1 |
| ADC | Rd, Rr | Add with Carry two Registers | Rd ←Rd + Rr + C | Z,C,N,V,H | 1 |
| ADIW | Rdl,K | Add Immediate to Word | Rdh:Rdl ←Rdh:Rdl + K | Z,C,N,V,S | 2 |
| SUB | Rd, Rr | Subtract two Registers | Rd ←Rd - Rr | Z,C,N,V,H | 1 |
| SUBI | Rd, K | Subtract Constant from Register | Rd ←Rd - K | Z,C,N,V,H | 1 |
| SBC | Rd, Rr | Subtract with Carry two Registers | Rd ←Rd - Rr - C | Z,C,N,V,H | 1 |
| SBCI | Rd, K | Subtract with Carry Constant from Reg. | Rd ←Rd - K - C | Z,C,N,V,H | 1 |
| SBIW | Rdl,K | Subtract Immediate from Word | Rdh:Rdl ←Rdh:Rdl - K | Z,C,N,V,S | 2 |
| AND | Rd, Rr | Logical AND Registers | Rd ←Rd • Rr | Z,N,V | 1 |
| ANDI | Rd, K | Logical AND Register and Constant | Rd ←Rd • K | Z,N,V | 1 |
| OR | Rd, Rr | Logical OR Registers | Rd ←Rd v Rr | Z,N,V | 1 |
| ORI | Rd, K | Logical OR Register and Constant | Rd ←Rd v K | Z,N,V | 1 |
| EOR | Rd, Rr | Exclusive OR Registers | Rd ←Rd ⊕ Rr | Z,N,V | 1 |
| COM | Rd | One's Complement | Rd ←0xFF –Rd | Z,C,N,V | 1 |
| NEG | Rd | Two's Complement | Rd ←0x00 –Rd | Z,C,N,V,H | 1 |
| SBR | Rd,K | Set Bit(s) in Register | Rd ←Rd v K | Z,N,V | 1 |
| CBR | Rd,K | Clear Bit(s) in Register | Rd ←Rd • (0xFF - K) | Z,N,V | 1 |
| INC | Rd | Increment | Rd ←Rd + 1 | Z,N,V | 1 |
| DEC | Rd | Decrement | Rd ←Rd –1 | Z,N,V | 1 |
| TST | Rd | Test for Zero or Minus | Rd ←Rd • Rd | Z,N,V | 1 |
| CLR | Rd | Clear Register | Rd ←Rd ⊕ Rd | Z,N,V | 1 |
| SER | Rd | Set Register | Rd ←0xFF | None | 1 |
| MUL | Rd, Rr | Multiply Unsigned | R1:R0 ←Rd x Rr | Z,C | 2 |
| MULS | Rd, Rr | Multiply Signed | R1:R0 ←Rd x Rr | Z,C | 2 |
| MULSU | Rd, Rr | Multiply Signed with Unsigned | R1:R0 ←Rd x Rr | Z,C | 2 |
| FMUL | Rd, Rr | Fractional Multiply Unsigned | R1:R0 ←(Rd x Rr) << 1 | Z,C | 2 |
| FMULS | Rd, Rr | Fractional Multiply Signed | R1:R0 ←(Rd x Rr) << 1 | Z,C | 2 |
| FMULSU | Rd, Rr | Fractional Multiply Signed with Unsigned | R1:R0 ←(Rd x Rr) << 1 | Z,C | 2 |
| BRANCH INSTRUCTIONS | | | | | |
| RJMP | k | Relative Jump | PC ←PC + k + 1 | None | 2 |
| IJMP | | Indirect Jump to (Z) | PC ←Z | None | 2 |
| JMP | k | Direct Jump | PC ←k | None | 3 |
| RCALL | k | Relative Subroutine Call | PC ←PC + k + 1 | None | 3 |
| ICALL | | Indirect Call to (Z) | PC ←Z | None | 3 |
| CALL | k | Direct Subroutine Call | PC ←k | None | 4 |
| 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 |

## 33. Instruction Set Summary (Continued)

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| BRBS | s, k | Branch if Status Flag Set | if (SREG(s) = 1) then PC←PC+k + 1 | None | 1/2 |
| BRBC | s, k | Branch if Status Flag Cleared | if (SREG(s) = 0) then PC←PC+k + 1 | None | 1/2 |
| BREQ | k | Branch if Equal | if (Z = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRNE | k | Branch if Not Equal | if (Z = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRCS | k | Branch if Carry Set | if (C = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRCC | k | Branch if Carry Cleared | if (C = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRSH | k | Branch if Same or Higher | if (C = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRLO | k | Branch if Lower | if (C = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRMI | k | Branch if Minus | if (N = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRPL | k | Branch if Plus | if (N = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRGE | k | Branch if Greater or Equal, Signed | if (N ⊕ V= 0) then PC ←PC + k + 1 | None | 1/2 |
| BRLT | k | Branch if Less Than Zero, Signed | if (N ⊕ V= 1) then PC ←PC + k + 1 | None | 1/2 |
| BRHS | k | Branch if Half Carry Flag Set | if (H = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRHC | k | Branch if Half Carry Flag Cleared | if (H = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRTS | k | Branch if T Flag Set | if (T = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRTC | k | Branch if T Flag Cleared | if (T = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRVS | k | Branch if Overflow Flag is Set | if (V = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRVC | k | Branch if Overflow Flag is Cleared | if (V = 0) then PC ←PC + k + 1 | None | 1/2 |
| BRIE | k | Branch if Interrupt Enabled | if ( I = 1) then PC ←PC + k + 1 | None | 1/2 |
| BRID | k | Branch if Interrupt Disabled | if ( I = 0) then PC ←PC + k + 1 | None | 1/2 |
| 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 |

## 33. Instruction Set Summary (Continued)

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| SEH | | Set Half Carry Flag in SREG | H ←1 | H | 1 |
| CLH | | Clear Half Carry Flag in SREG | H ←0 | H | 1 |
| DATA TRANSFER INSTRUCTIONS | | | | | |
| MOV | Rd, Rr | Move Between Registers | Rd ←Rr | None | 1 |
| MOVW | Rd, Rr | Copy Register Word | Rd+1:Rd ←Rr+1:Rr | None | 1 |
| LDI | Rd, K | Load Immediate | Rd ←K | None | 1 |
| LD | Rd, X | Load Indirect | Rd ←(X) | None | 2 |
| LD | Rd, X+ | Load Indirect and Post-Inc. | Rd ←(X), X ←X + 1 | None | 2 |
| LD | Rd, - X | Load Indirect and Pre-Dec. | X ←X - 1, Rd ←(X) | None | 2 |
| LD | Rd, Y | Load Indirect | Rd ←(Y) | None | 2 |
| LD | Rd, Y+ | Load Indirect and Post-Inc. | Rd ←(Y), Y ←Y + 1 | None | 2 |
| LD | Rd, - Y | Load Indirect and Pre-Dec. | Y ←Y - 1, Rd ←(Y) | None | 2 |
| LDD | Rd,Y+q | Load Indirect with Displacement | Rd ←(Y + q) | None | 2 |
| LD | Rd, Z | Load Indirect | Rd ←(Z) | None | 2 |
| LD | Rd, Z+ | Load Indirect and Post-Inc. | Rd ←(Z), Z ←Z+1 | None | 2 |
| LD | Rd, -Z | Load Indirect and Pre-Dec. | Z ←Z - 1, Rd ←(Z) | None | 2 |
| LDD | Rd, Z+q | Load Indirect with Displacement | Rd ←(Z + q) | None | 2 |
| LDS | Rd, k | Load Direct from SRAM | Rd ←(k) | None | 2 |
| ST | X, Rr | Store Indirect | (X) ←Rr | None | 2 |
| ST | X+, Rr | Store Indirect and Post-Inc. | (X) ←Rr, X ←X + 1 | None | 2 |
| ST | - X, Rr | Store Indirect and Pre-Dec. | X ←X - 1, (X) ←Rr | None | 2 |
| ST | Y, Rr | Store Indirect | (Y) ←Rr | None | 2 |
| ST | Y+, Rr | Store Indirect and Post-Inc. | (Y) ←Rr, Y ←Y + 1 | None | 2 |
| ST | - Y, Rr | Store Indirect and Pre-Dec. | Y ←Y - 1, (Y) ←Rr | None | 2 |
| STD | Y+q,Rr | Store Indirect with Displacement | (Y + q) ←Rr | None | 2 |
| ST | Z, Rr | Store Indirect | (Z) ←Rr | None | 2 |
| ST | Z+, Rr | Store Indirect and Post-Inc. | (Z) ←Rr, Z ←Z + 1 | None | 2 |
| ST | -Z, Rr | Store Indirect and Pre-Dec. | Z ←Z - 1, (Z) ←Rr | None | 2 |
| STD | Z+q,Rr | Store Indirect with Displacement | (Z + q) ←Rr | None | 2 |
| STS | k, Rr | Store Direct to SRAM | (k) ←Rr | None | 2 |
| LPM | | Load Program Memory | R0 ←(Z) | None | 3 |
| LPM | Rd, Z | Load Program Memory | Rd ←(Z) | None | 3 |
| LPM | Rd, Z+ | Load Program Memory and Post-Inc | Rd ←(Z), Z ←Z+1 | None | 3 |
| SPM | | Store Program Memory | (Z) ←R1:R0 | None | - |
| IN | Rd, P | In Port | Rd ←P | None | 1 |
| OUT | P, Rr | Out Port | P ←Rr | None | 1 |
| PUSH | Rr | Push Register on Stack | STACK ←Rr | None | 2 |
| POP | Rd | Pop Register from Stack | Rd ←STACK | None | 2 |
| MCU CONTROL INSTRUCTIONS | | | | | |
| 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 |
| MATHS EXTENSION INSTRUCTIONS | | | | | |
| ADD.L | Rd.l, Rs.l | 32-bit Add | Rd.l ←Rd.l + Rs.l | 1 | 1 |
| ADC.L | Rd.l, Rs.l | 32-bit Add with Carry | Rd.l ←Rd.l + Rs.l + C | 1 | 1 |

## 33. Instruction Set Summary (Continued)

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| SUB.L | Rd.l, Rs.l | 32-bit Substract | Rd.l ←Rd.l –Rs.l | 1 | 1 |
| SBC.L | Rd.l, Rs.l | 32-bit Substract with Carry | Rd.l ←Rd.l –Rs.l –C | 1 | 1 |
| CP.L | Rd.l, Rs.l | 32-bit Substract | Rd.l –Rd.l | 1 | 1 |
| CPC.L | Rd.l, Rs.l | 32-bit Substract with Carry | Rd.l –Rd.l –Rs.l | 1 | 1 |
| RSUB.L | Rd.l, Rs.l | 32-bit Reverse Substract | Rs.l ←Rd.l –Rs.l | 1 | 1 |
| RSBC.L | Rd.l, Rs.l | 32-bit Reverse Substract with Carry | Rs.l ←Rd.l –Rs.l –C | 1 | 1 |
| DIVINIT.L | Rd.l (Divider), *type (optional)* | 32-bit Div initialization, type selects shadow reg | Dividend ←Rd.l, Rd.l ←Dividend | 1 | 1 |
| DIV.L | Rd.l (Remainder), Rs.l (Divisor) | 32-bit Division (Iterative) | Rd.l ←Divider / Divisor (Iterative) | 1 | 1 |
| DIV.LL | Rd.ll (Remainder), Rs.ll (Divisor) | 64-bit Division (Iterative) | Rd.ll ←Divider / Divisor (Iterative) | 1 | 2 |
| MOV.L | Rd.l, Rs.l | 32-bit Moy | Rd.l ←Rd.l | 1 | 1 |
| MOV.LL | Rd.ll, Rs.ll | 64-bit Moy | Rd.ll ←Rd.ll | 1 | 2 |
| TST.L | Rd.l | 32-bit Test for zero or neg. | Rd.l ←Rd.l • Rd.l | 1 | 1 |
| TSL:LL | Rd.ll | 64-bit Test for zero or neg. | Rd.ll ←Rd.ll • Rd.ll | 1 | 2 |
| LSR.L | Rd.l | 32-bit Logical shift right | Rd.l(n) ←Rd.l(n+1), Rd.l(31) ←0, C ← Rd.l(0) | 1 | 1 |
| LSR.LL | Rd.ll | 64-bit Logical shift right | Rd.ll(n) ←Rd.ll(n+1), Rd.ll(31) ←0, C ←Rd.ll(0) | 1 | 2 |
| ASR.L | Rd.l | 32-bit Arithmetic shift right | Rd.l(n) ←Rd.l(n+1), n = 0...30, C ← Rd.l(0) | 1 | 1 |
| ASR.LL | Rd.ll | 64-bit Arithmetic shift right | Rd.ll(n) ←Rd.ll(n+1), n = 0...30, C ← Rd.ll(0) | 1 | 2 |
| ROR.L | Rd.l | 32-bit Rotate right | Rd.l(31) ←C, Rd.l(n) ←Rd.l(n+1), C ← Rd.l(0) | 1 | 1 |
| ROR.LL | Rd.ll | 64-bit Rotate right | Rd.ll(31) ←C, Rd.ll(n) ←Rd.ll(n+1), C ←Rd.ll(0) | 1 | 2 |
| NEG.L | Rd.l | 32-bit Two's Complement | Rd.l ←$0 –Rd.l | 1 | 1 |
| NEG.LL | Rd.ll | 64-bit Two's Complement | Rd.ll ←$0 –Rd.ll | 1 | 2 |
| NEGTS.L | Rd.l | 32-bit Two's Complement if T set | Rd.l ←$0 –Rd.l | 1 | 1 |
| NEGTS.LL | Rd.ll | 64-bit Two's Complement if T set | Rd.ll ←$0 –Rd.ll | 1 | 2 |
| ABS.L | Rd.l | 32-bit Absolute value | T ←Rd.l[31], Rd.l ←ABS(Rd.l) | 1 | 1 |
| ABS.LL | Rd.ll | 64-bit Absolute value | T ←Rd.ll[63], Rd.ll ←ABS(Rd.ll) | 1 | 2 |
| ABSXT.L | Rd.l | 32-bit Absolute value, sign xor T | T ←T ∧ Rd.l[31], Rd.l ←ABS(Rd.l) | 1 | 1 |
| ABSXT.LL | Rd.ll | 64-bit Absolute value, sign xor T | T ←T ∧ Rd.ll[63], Rd.ll ←ABS(Rd.ll) | 1 | 2 |
| MUL.W | Rd.w, Rx.w, Ry.w | 16-bit Multiplication with 16-bit result | Rd.w ←Rx.w * Ry.w (UU) | 2 | 3 |
| MULL.W | Rd.l, Rx.w, Ry.w | 16-bit Multiplication with 32-bit result | Rd.l ←Rx.w * Ry.w (UU) | 2 | 3 |
| MULLS.W | Rd.l, Rx.w, Ry.w | 16-bit Signed Multiplication with 32-bit result | Rd.l ←Rx.w * Ry.w (SS) | 2 | 4 |
| MULLSU.W | Rd.l, Rx.w, Ry.w | 16-bit Signed Unsigned Multiplication with 32-bit result | Rd.l ←Rx.w * Ry.w (SU) | 2 | 4 |
| MUL.L | Rd.l, Rx.l, Ry.l | 32-bit Multiplication with 32-bit result | Rd.l ←Rx.l * Ry.l (UU) | 2 | 5 |
| MULL.L | RdH.l, RdL.l, Rx.l, Ry.l | 32-bit Multiplication with 64-bit result | RdH.l, RdL.l ←Rx.l * Ry.l (UU) | 2 | 6 |
| MULLS.L | RdH.l, RdL.l, Rx.l, Ry.l | 32-bit Signed Multiplication with 64-bit result | RdH.l, RdL.l ←Rx.l * Ry.l (SS) | 2 | 6 |
| MULSU.L | RdH.l, RdL.l, Rx.l, Ry.l | 32-bit Signed Unsigned Multiplication with 64-bit result | RdH.l, RdL.l ←Rx.l * Ry.l (SU) | 2 | 6 |
| MUL.LL | RdH.l, RdL.l, Rx.ll, Ry.ll, RdH.l, RdL.l, != Rxy.ll | 64-bit Multiplication with 64-bit result | RdH.l, RdL.l ←Rx.ll * Ry.ll (UU) | 2 | 12 |

# 34. Operating Circuit

**Figure 34-1. Operating Circuit without Watchdog**

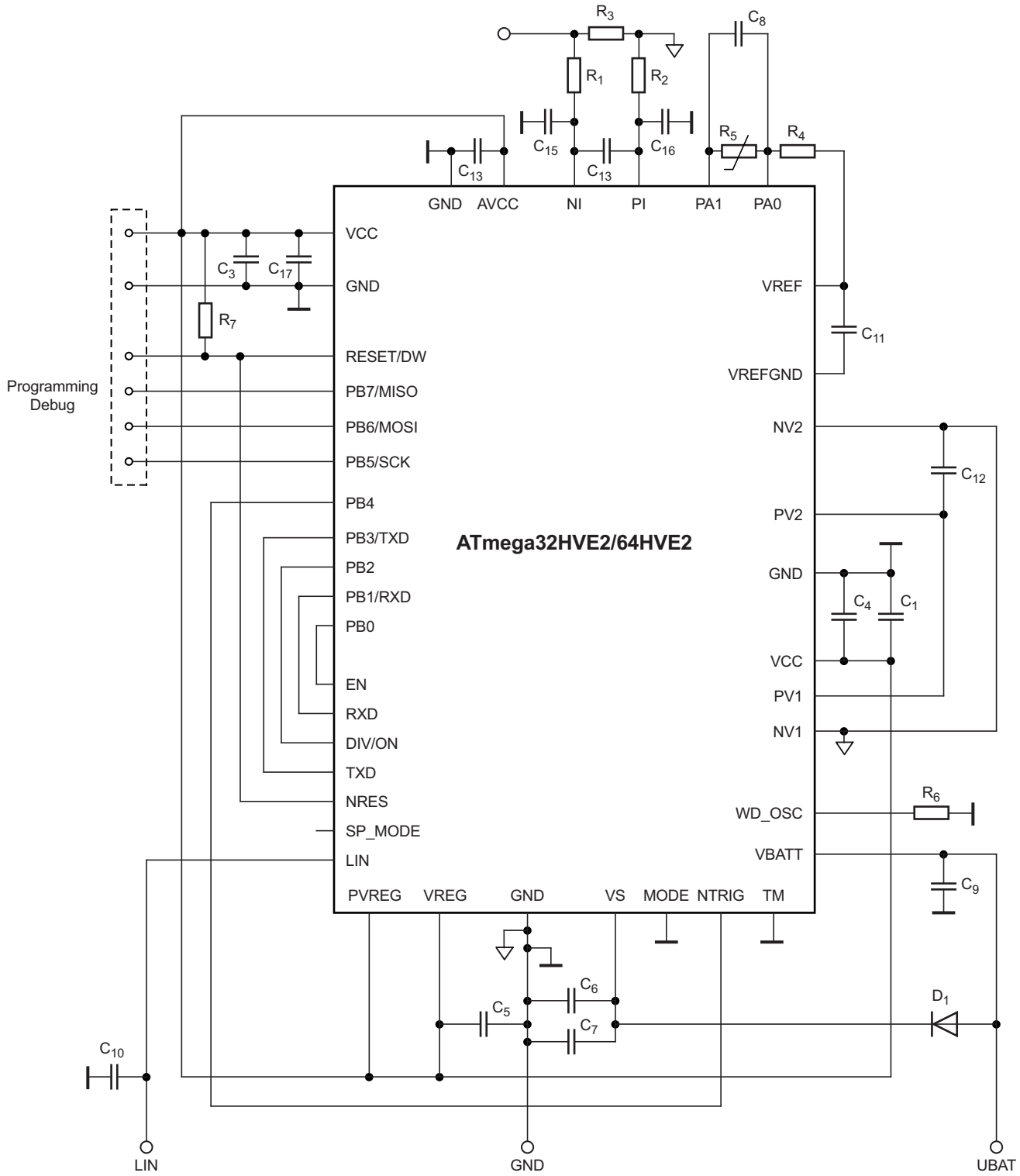
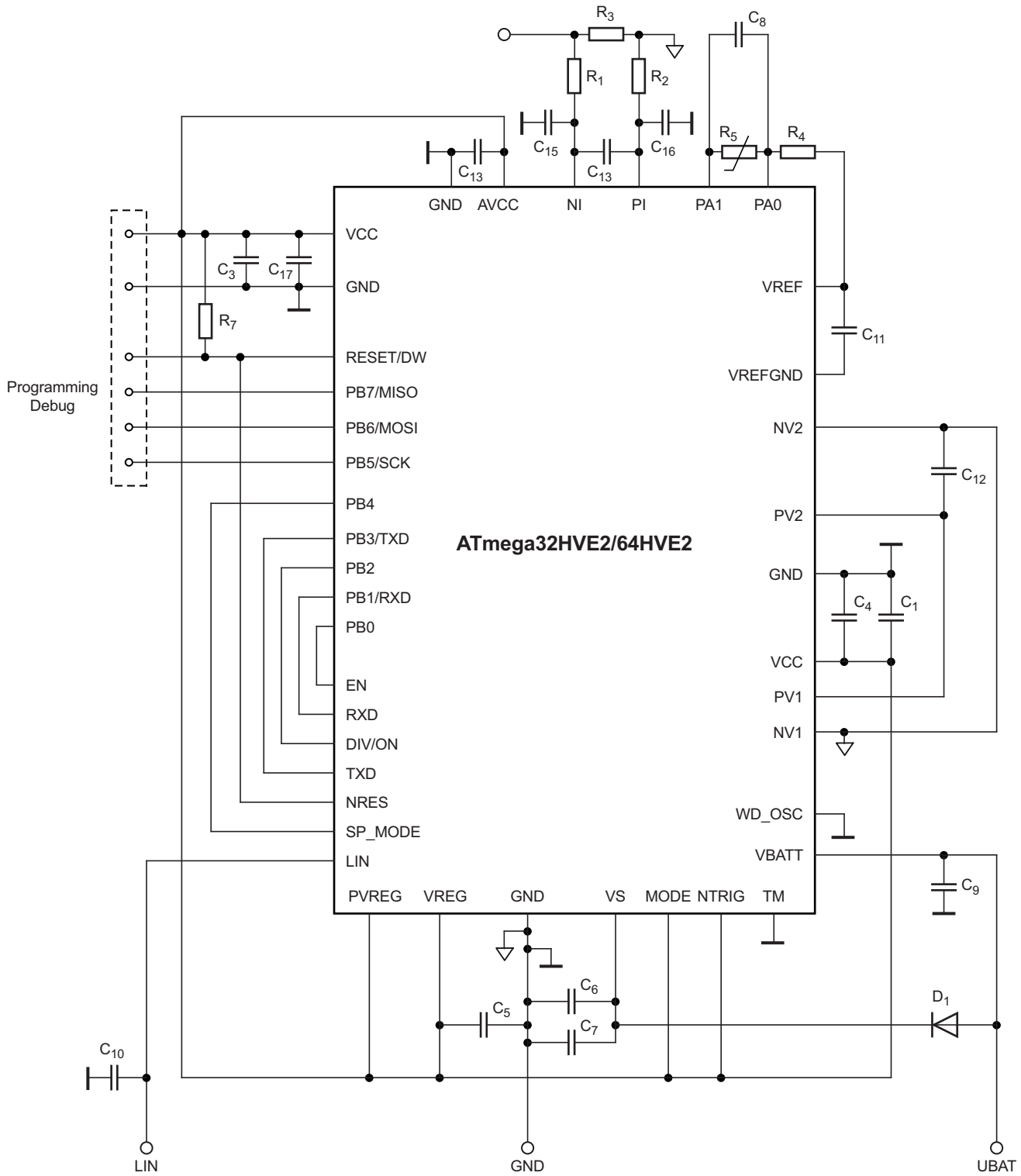


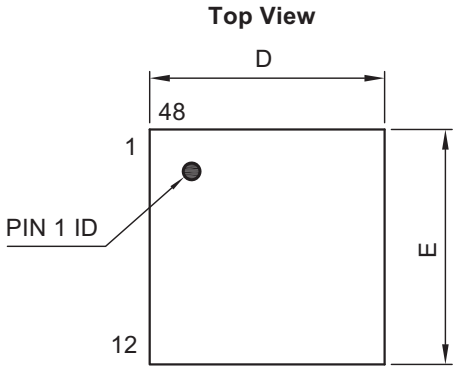
**Figure 34-2. Operating Circuit with Watchdog**

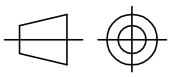


# 35. Ordering Information

| Extended Type Number | Package | MOQ |
|---|---|---|
| ATmega32HVE2-PLPW | QFN48, 7 × 7 | 1,000 pieces |
| ATmega32HVE2-PLQW | QFN48, 7 × 7 | 4,000 pieces |
| ATmega64HVE2-PLPW | QFN48, 7 × 7 | 1,000 pieces |
| ATmega64HVE2-PLQW | QFN48, 7 × 7 | 4,000 pieces |

# 36. Packaging Information

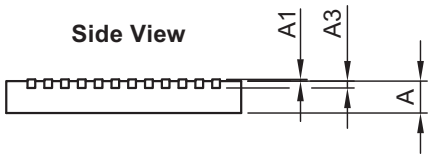


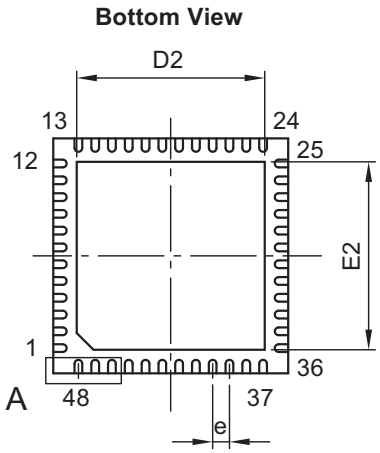



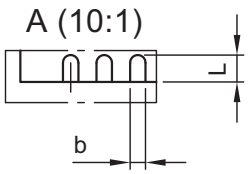
technical drawings
according to DIN
specifications

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

| | TITLE | GPC | DRAWING NO. | REV. |
|---|---|---|---|---|
| Atmel **Package Drawing Contact:** packagedrawings@atmel.com | **Package: VQFN_7x7_48L**<br>Exposed pad 5.6x5.6 | | 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

# 37. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

| Revision No. | History |
|---|---|
| 8096C-AVR-01/13 | • Section 26.5 “Diagnosis Mode” on page 149 updated |
| 8096B-AVR-11/12 | • Section 27.3 “$VTEMP_{BASE}$ and $VTEMP_{SLOPE}$” on page 161 updated<br>• Section 29.8.9 “Reading the Signature Row from Software” on page 173 updated |

Atmel | Enabling Unlimited Possibilities®

**Atmel Corporation**
1600 Technology Drive
San Jose, CA 95110
USA
**Tel:** (+1) (408) 441-0311
**Fax:** (+1) (408) 487-2600
www.atmel.com

**Atmel Asia Limited**
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Roa
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 G.K.**
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
**Tel:** (+81) (3) 6417-0300
**Fax:** (+81) (3) 6417-0370

© 2013 Atmel Corporation. All rights reserved. / Rev.: 8096C–AVR–01/13

Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.