Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	4MHz
Connectivity	SPI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	7
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 9V
Data Converters	A/D 5x12b
Oscillator Type	Internal
Operating Temperature	-20°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-TSSOP (0.465", 11.80mm Width)
Supplier Device Package	28-TSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega16hva-4tur

1.3.7 GND

Ground

1.3.8 Port A (PA1..PA0)

Port A serves as a low-voltage 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 also serves the functions of various special features of the ATmega8HVA/16HVA as listed in ”Alternate Functions of Port A” on page 70.

1.3.9 Port B (PB3..PB0)

Port B is a low-voltage 4-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 ATmega8HVA/16HVA as listed in ”Alternate Functions of Port B” on page 71.

1.3.10 PC0

Port C serves the functions of various special features of the ATmega8HVA/16HVA as listed in ”Alternate Functions of Port C” on page 61.

1.3.11 OC

High voltage output to drive Charge FET.

1.3.12 OD

High voltage output to drive Discharge FET.

1.3.13 NI

NI is the filtered negative input from the current sense resistor.

1.3.14 PI

PI is the filtered positive input from the current sense resistor.

1.3.15 NV/PV1/PV2

NV, PV1, and PV2 are the inputs for battery cells 1 and 2.

1.3.16 BATT

Input for detecting when a charger is connected.

1.3.17 $\overline{\text{RESET}}$/dw

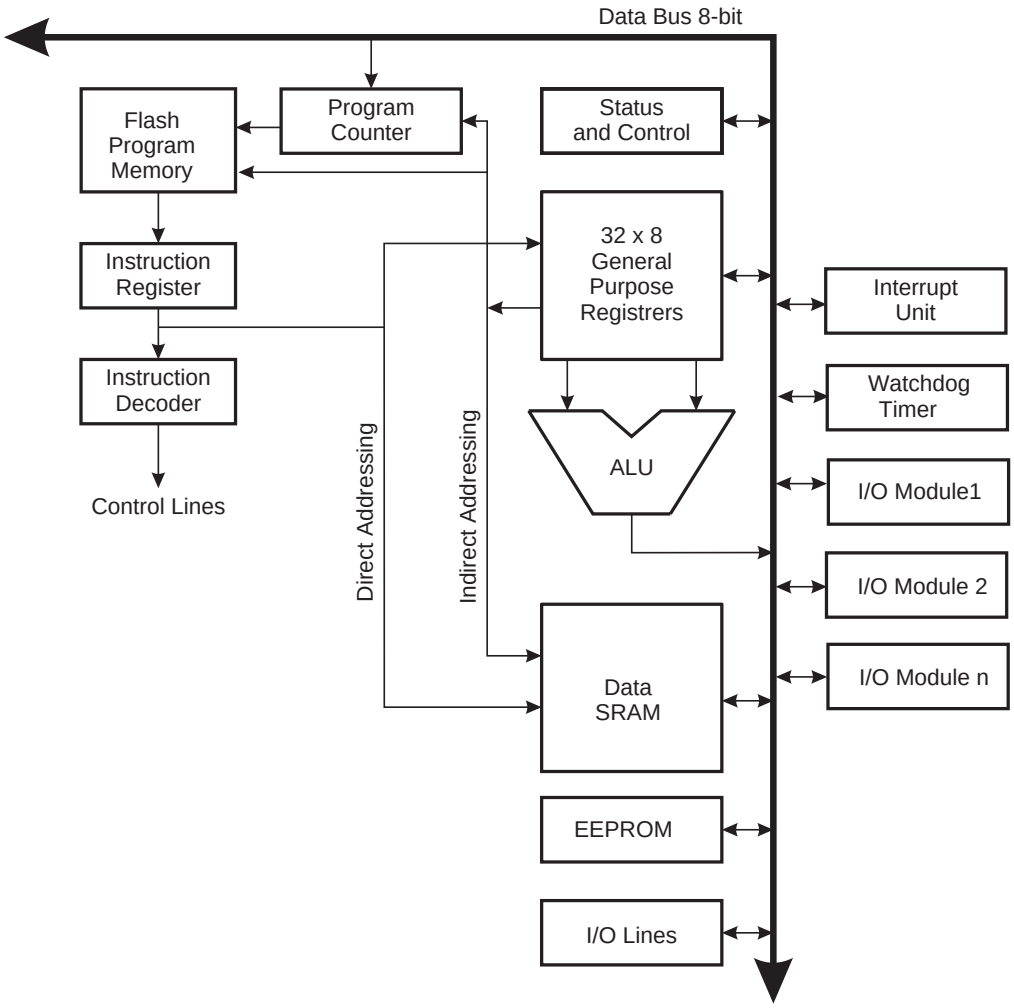
Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 11 on page 38. Shorter pulses are not guaranteed to generate a reset. This pin is also used as debugWIRE communication pin.

7. AVR CPU Core

7.1 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 7-1. 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 typ-

8. AVR Memories

8.1 Overview

This section describes the different memories in the ATmega8HVA/16HVA. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega8HVA/16HVA features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

8.2 In-System Reprogrammable Flash Program Memory

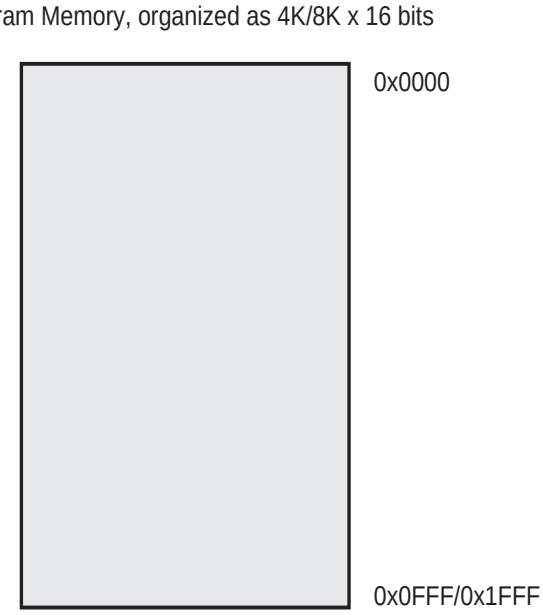
The ATmega8HVA/16HVA contains 8K/16K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K/8K x 16.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8HVA/16HVA Program Counter (PC) is 12/13 bits wide, thus addressing the 4K/8K program memory locations. ”Memory Programming” on page 149 contains a detailed description on Flash data serial 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 ”Instruction Execution Timing” on page 13.

Figure 8-1. Program Memory Map



8.3 SRAM Data Memory

Figure 8-2 on page 17 shows how the ATmega8HVA/16HVA SRAM Memory is organized.

The ATmega8HVA/16HVA 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

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8HVA/16HVA 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 8-1. 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 8-1. EEPROM Mode Bits

EEPM1	EEPM0	Typ Programming Time, f_{osc} = 4.0 MHz	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

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

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when 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

9.7 Watchdog Timer, Battery Protection and Coulomb Counter ADC Clock

The clock source for the Watchdog Timer, Battery Protection and Coulomb Counter ADC (CC-ADC) is the Ultra Low Power RC Oscillator. The Oscillator is automatically enabled in all operational modes. It is also enabled during reset.

9.8 Clock Startup Sequence

When the CPU wakes up from Power-save, the CPU clock source is used to time the start-up, ensuring a stable clock before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the voltage regulator 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 9-1 on page 25. The number of Ultra Low Power RC Oscillator cycles used for each time-out is shown in Table 9-2.

Table 9-2. Number of Ultra Low Power RC Oscillator Cycles

Typ Time-out[1]	Number of Cycles
4 ms	512
8 ms	1K
16 ms	2K
32 ms	4K
64 ms	8K
128 ms	16K
256 ms	32K
512 ms	64K

Note: 1. The actual value depends on the actual clock period of the Ultra Low Power RC Oscillator, refer to "Ultra Low Power RC Oscillator" on page 26 for details.

9.9 Clock Output

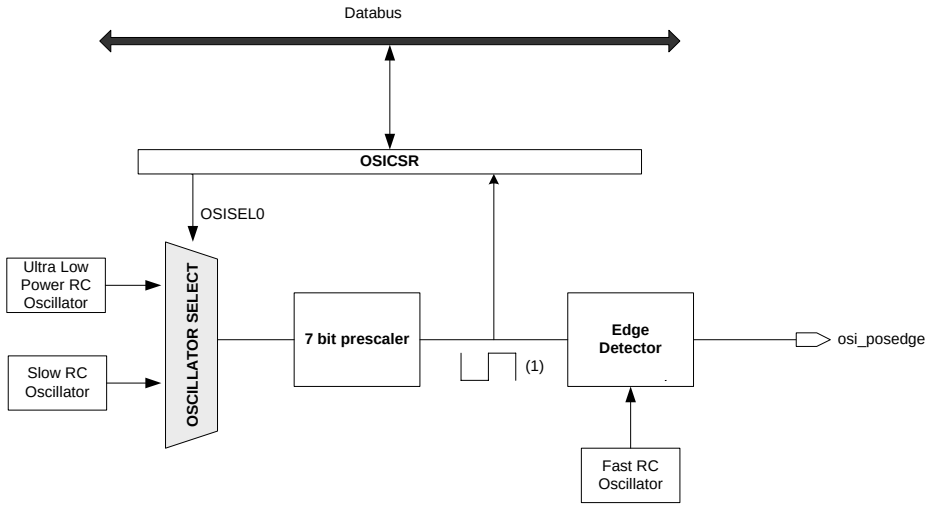
The CPU clock divided by 2 can be output to the PB0 pin. The CPU can enable the clock output function by setting the CKOE bit in the MCU Control Register. The clock will not run in any sleep modes.

9.10 System Clock Prescaler

The ATmega8HVA/16HVA has a System Clock Prescaler, used to prescale the Calibrated Fast RC Oscillator. The system clock can be divided by setting the "CLKPR – Clock Prescale Register" on page 31, 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 9-3 on page 32.

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.

Figure 9-2. Oscillator Sampling Interface Block Diagram



Note: 1. One prescaled Slow RC/ULP oscillator period corresponds to 128 times the actual Slow RC/ULP oscillator period.

The osi_posedge signal pulses on each rising edge of the prescaled Slow RC/ ULP oscillator clock. This signal is not directly accessible by the CPU, but can be used to trigger the input capture function of Timer/Counter0. Using OSI in combination with the input capture function of Timer/Counter0 facilitates accurate measurement of the oscillator frequencies with a minimum of CPU calculation. Refer to "Timer/Counter(T/C0,T/C1)" on page 77 for details on how to enable the Input Capture function.

9.12.3 Usage

The Slow RC oscillator represents a highly predictable and accurate clock source over the entire temperature range and provides an excellent reference for calibrating the Fast RC oscillator runtime. Typically, runtime calibration is needed to provide an accurate Fast RC frequency for asynchronous serial communication in the complete temperature range. An accurate time reference is also needed to calculate accumulated charge during a CC-ADC measurement.

The Slow RC frequency at T_{HOT} (calibration temperature) and the Slow RC temperature coefficient are stored in the signature row. The value of T_{HOT} is also stored in the signature row. These characteristics can be used to calculate the actual Slow RC clock period at a given temperature with high precision. Refer to "Slow RC Oscillator" on page 26 for details.

By measuring the number of CPU cycles of one or more prescaled Slow RC clock periods, the actual Fast RC oscillator clock period can be determined. The Fast RC clock period can then be adjusted by writing to the FOSCCAL register. The new Fast RC clock period after calibration should be verified by repeating the measurement and repeating the calibration if necessary. The Fast RC clock period as a function of the Slow RC clock period is given by:

$$T_{FastRC} = T_{SlowRC} \cdot \frac{128 \cdot n}{\text{number of CPU cycles in n prescaled Slow RC periods}}$$

where n is the number of prescaled Slow RC periods that is used in the measurement. Using more prescaled Slow RC periods decreases the measurement error, but increases the time consumed for calibration. Note that the Slow RC Oscillator needs very short time to stabilize after

Note that if a level triggered interrupt is used for wake-up from Power-save mode, the changed level must be held for some time to wake up the MCU. Refer to ”External Interrupts” on page 56 for details.

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 ”Clock Sources” on page 25.

10.5 Power-off Mode

When the SM2..0 bits are written to 100 and the SE bit is set, the SLEEP instruction makes the CPU shut down the Voltage Regulator, leaving only the Charger Detect Circuitry operational. To ensure that the MCU enters Power-off mode only when intended, the SLEEP instruction must be executed within 4 clock cycles after the SM2..0 bits are written. The MCU will reset when returning from Power-off mode.

Note: Before entering Power-off sleep mode, interrupts should be disabled by software. Otherwise interrupts may prevent the SLEEP instruction from being executed within the time limit.

10.6 Power Reduction Register

The Power Reduction Register (PRR), see ”PRR0 – Power Reduction Register 0” on page 39, 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.

10.7 Minimizing Power Consumption

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

10.7.1 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 except Power-off. The Watchdog Timer current consumption is significant only in Power-save mode. Refer to ”Watchdog Timer” on page 46 for details on how to configure the Watchdog Timer.

10.7.2 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section ”Digital Input Enable and Sleep Modes” on page 67 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 $V_{REG}/2$, the input buffer will use excessive power.

11.3 Watchdog Timer

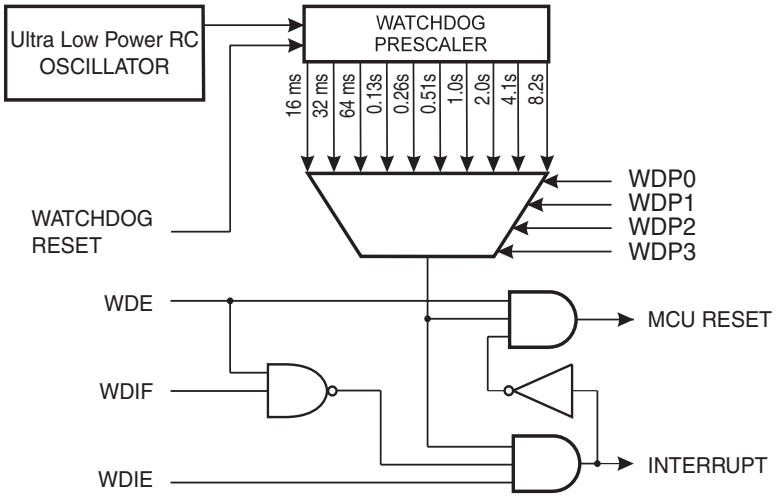
11.3.1 Features

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

11.3.2 Overview

ATmega8HVA/16HVA 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 11-6. 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. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

12.3 Interrupt Vectors in ATmega16HVA

.

Table 12-2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and debugWIRE Reset
2	0x0002	BPINT	Battery Protection Interrupt
3	0x0004	VREGMON	Voltage Regulator Monitor Interrupt
4	0x0006	INT0	External Interrupt Request 0
5	0x0008	INT1	External Interrupt Request 1
6	0x000A	INT2	External Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER1 IC	Timer 1 input Capture
9	0x0010	TIMER1 COMPA	Timer 1 Compare Match A
10	0x0012	TIMER1 COMPB	Timer 1 Compare Match B
11	0x0014	TIMER1 OVF	Timer 1 Overflow
12	0x0016	TIMER0 IC	Timer 0 input Capture
13	0x0018	TIMER0 COMPA	Timer 0 Compare Match A
14	0x001A	TIMER0 COMPB	Timer 0 Compare Match B
15	0x001C	TIMER0 OVF	Timer 0 Overflow
16	0x001E	SPI, STC	SPI, Serial Transfer Complete
17	0x0020	VADC	Voltage ADC Conversion Complete
18	0x0022	CCADC CONV	CC-ADC Instantaneous Current Conversion Complete
19	0x0024	CCADC REG CUR	CC-ADC Regular Current
20	0x0026	CCADC ACC	CC-ADC Accumulate Current Conversion Complete
21	0x0028	EE READY	EEPROM Ready

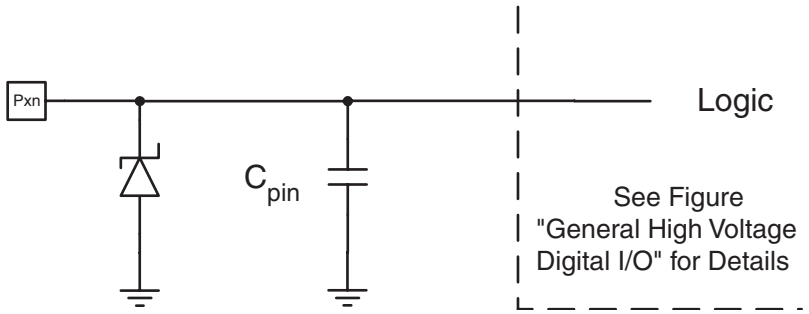
If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations.

14. High Voltage I/O Ports

14.1 Overview

All high voltage AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the state of one port pin can be changed without unintentionally changing the state of any other pin with the SBI and CBI instructions. All high voltage I/O pins have protection Zener diodes to Ground as indicated in Figure 14-1. See ”Electrical Characteristics” on page 165 for a complete list of parameters.

Figure 14-1. High Voltage 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, PORTC3 for bit number three in Port C, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in ”Register Description” on page 62.

One I/O Memory address location is allocated for each high voltage port, the Data Register – PORTx. The Data Register is read/write.

Using the I/O port as General Digital Output is described in ”High Voltage Ports as General Digital I/O” on page 59.

17.5.5 8-bit Input Capture Mode

The Timer/Counter can be used in a 8-bit Input Capture mode, see Table 17-2 on page 80 for bit settings. For full description, see "Input Capture Unit" on page 82.

17.5.6 16-bit Input Capture Mode

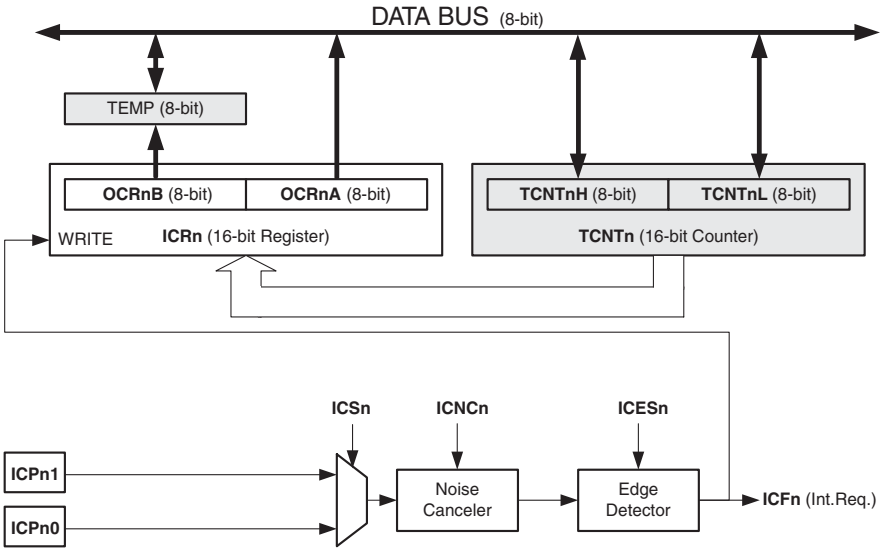
The Timer/Counter can also be used in a 16-bit Input Capture mode, see Table 17-2 on page 80 for bit settings. For full description, see "Input Capture Unit" on page 82.

17.6 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 events can be applied via the PC0 pin (ICP01), or alternatively via the osi_posedge pin on the Oscillator Sampling Interface (ICP00). For Timer/Counter1, the events can be applied by the Battery Protection Interrupt (ICP10) or alternatively by the Voltage Regulator Interrupt (ICP11). 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 17-4 on page 82. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 17-4. 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 "Accessing Registers in 16-bit Mode" on page 86.

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 17-3. Timer/Counter0 Input Capture Source (ICS)

ICS0	Source
0	ICP00: osi_posedge pin from OSI module(1)
1	ICP01: Port PC0

Note: 1. See "OSI – Oscillator Sampling Interface" on page 28 for details.

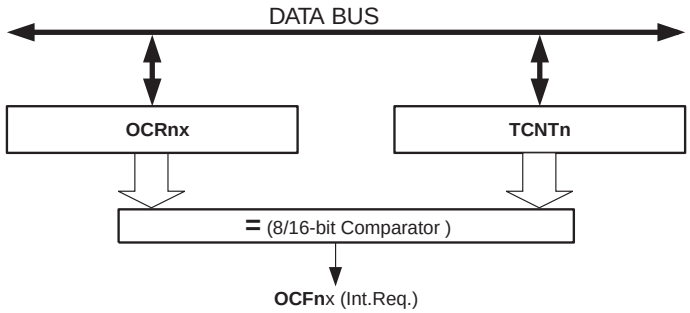
Table 17-4. Timer/Counter1 Input Capture Source (ICS)

ICS1	Source
0	ICP10: Battery Protection Interrupt
1	ICP11: Voltage Regulator Interrupt

17.7 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 Registers, 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 17-5 on page 84 shows a block diagram of the Output Compare unit.

Figure 17-5. Output Compare Unit, Block Diagram



18.5.3 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.

of lower resolution. The Accumulate Current Output provides a highly accurate current measurement for Coulomb Counting.

The CC-ADC also provides a special Regular Current detection mode. This allows ultra-low power operation in Power-save mode when small charge or discharge currents are flowing.

For offset cancellation the polarity of the input signal could be switched run time. Using this feature correctly will remove the internal CC-ADC offset. See application note AVR352.

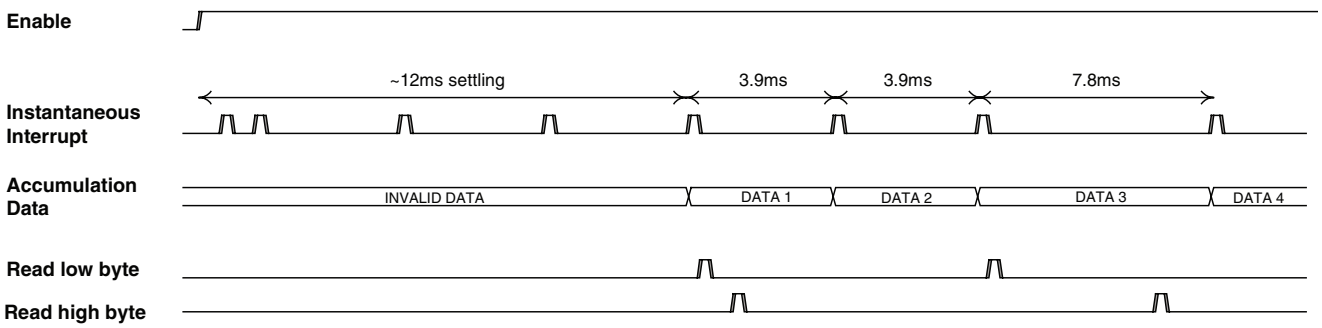
19.3 Normal Operation

When enabled the CC-ADC continuously measures the voltage over the external sense resistor R_{SENSE}. Running in normal conversion mode, two data conversion output is provided.

- Instantaneous Conversion Result
- Accumulation Conversion Result

The Instantaneous Current conversion time is fixed to 3.9 ms (typical value) allowing the output value to closely follow the input. After each Instantaneous Current conversion an interrupt is generated if the interrupt is enabled. Data from conversion will be updated in the Instantaneous Current registers CADICL and CADICH at the same time as the interrupt is given. To avoid losing conversion data, both the low and high byte must be read within a 3.9 ms timing window after the corresponding interrupt is given. When the low byte register is read, updating of the Instantaneous Current registers and interrupts will be stopped until the high byte is read. Figure 19-2 shows an Instantaneous Current conversion diagram, where DATA4 will be lost because DATA3 reading is not completed within the limited period.

Figure 19-2. Instantaneous Current Conversions



The Accumulate Current output is a high-resolution, high accuracy output with programmable conversion time selected by the CADAS bits in CADCSRA. The converted value is an accurate measurement of the average current flow during one conversion period. The CC-ADC generates an interrupt each time a new Accumulate Current conversion has finished if the interrupt is enabled. Data from conversion will be updated in the Accumulation Current registers CADAC0, CADAC1, CADAC2 and CADAC3 at the same time as the interrupt is given. To avoid losing conversion data, all bytes must be read within the selected conversion period. When the lower byte registers are read, updating of the Accumulation Current registers and interrupts will be stopped until the highest byte is read. Table 19-3 shows an Accumulation Current conversion diagram, where DATA4 will be lost because DATA3 reading is not completed within the limited period.

When the Discharge High-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the D-FET is re-enabled while the loading of the battery still is too large, the Discharge High-current Protection will be activated again.

23.7 Charge High-current Protection

If the voltage at the PI/NI pins is above the Charge High-current Detection level for a time longer than High-current Protection Reaction Time, the chip activates Charge High-current Protection.

When the Charge High-current Protection is activated, the external D-FET and C-FET are disabled and a Current Protection Timer is started. This timer ensures that the FETs are disabled for at least one second. The application software must then set the DFE and CFE bits in the FET Control and Status Register to re-enable normal operation. If the C-FET is re-enabled and the charger continues to supply too high currents, the Charge High-current Protection will be activated again.

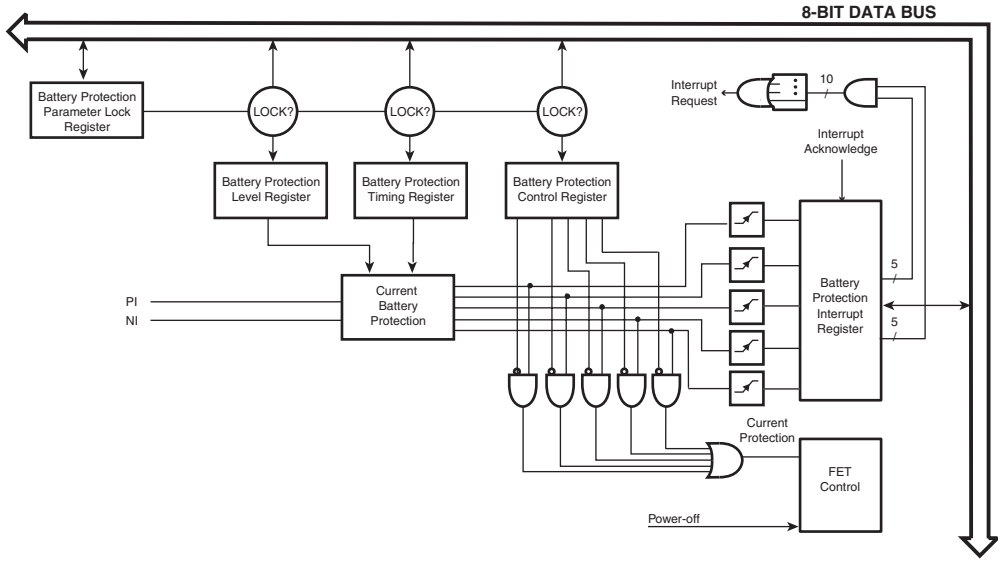
The Short-circuit, Over-current and High-current Protection parameters are programmable to adapt to different types of batteries. The parameters are set by writing to I/O Registers. The Parameter Registers can be locked after the initial configuration, prohibiting any further updates until the next Hardware Reset.

Refer to ”Register Description for Battery Protection” on page 125 for register descriptions.

23.8 Battery Protection CPU Interface

The Battery Protection CPU Interface is illustrated in Figure 22-1.

Figure 23-1. Battery Protection CPU Interface



Each protection has an Interrupt Flag. Each Flag can be read and cleared by the CPU, and each flag has an individual interrupt enable. All enabled flags are combined into a single battery pro-

tection interrupt request to the CPU. This interrupt can wake up the CPU from any operation mode, except Power-off. The interrupt flags are cleared by writing a logic ‘1’ to their bit locations from the CPU.

Note that there are neither flags nor status bits indicating that the chip has entered the Power Off mode. This is because the CPU is powered down in this mode. The CPU will, however be able to detect that it came from a Power-off situation by monitoring CPU reset flags when it resumes operation.

23.9 Register Description

23.9.1 BPPLR – Battery Protection Parameter Lock Register

Bit	7	6	5	4	3	2	1	0	
(0xFE)	–	–	–	–	–	–	BPPLE	BPPL	BPPLR
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 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 1 – BPPLE: Battery Protection Parameter Lock Enable**

- **Bit 0 – BPPL: Battery Protection Parameter Lock**

The BPCR, BPHCTR, BPOCTR, BPSCTR, BPDHCD, BPCHCD, BPDOCD, BPCOCD and BPSCD Battery Protection registers can be locked from any further software updates. Once locked, these registers cannot be accessed until the next hardware reset. This provides a safe method for protecting the registers from unintentional modification by software runaway. It is recommended that software sets these registers shortly after reset, and then protect the registers from further updates.

To lock these registers, the following algorithm must be followed:

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

23.9.2 BPCR – Battery Protection Control Register

Bit	7	6	5	4	3	2	1	0	
(0xFD)	–	–	–	SCD	DOCD	COCD	DHCD	CHCD	BPCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bits 7:6 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 5 – Res: Reserved Bits**

This bit are reserved and will always read as one.

25. debugWIRE On-chip Debug System

25.1 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**

25.2 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.

25.3 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 25-1. The debugWIRE Setup

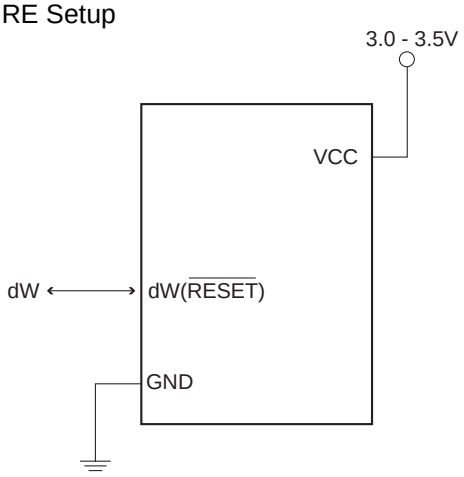


Figure 25-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the OSCSEL Fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

25.4 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.

25.5 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.

25.6 Register Description

The following section describes the registers used with the debugWire.

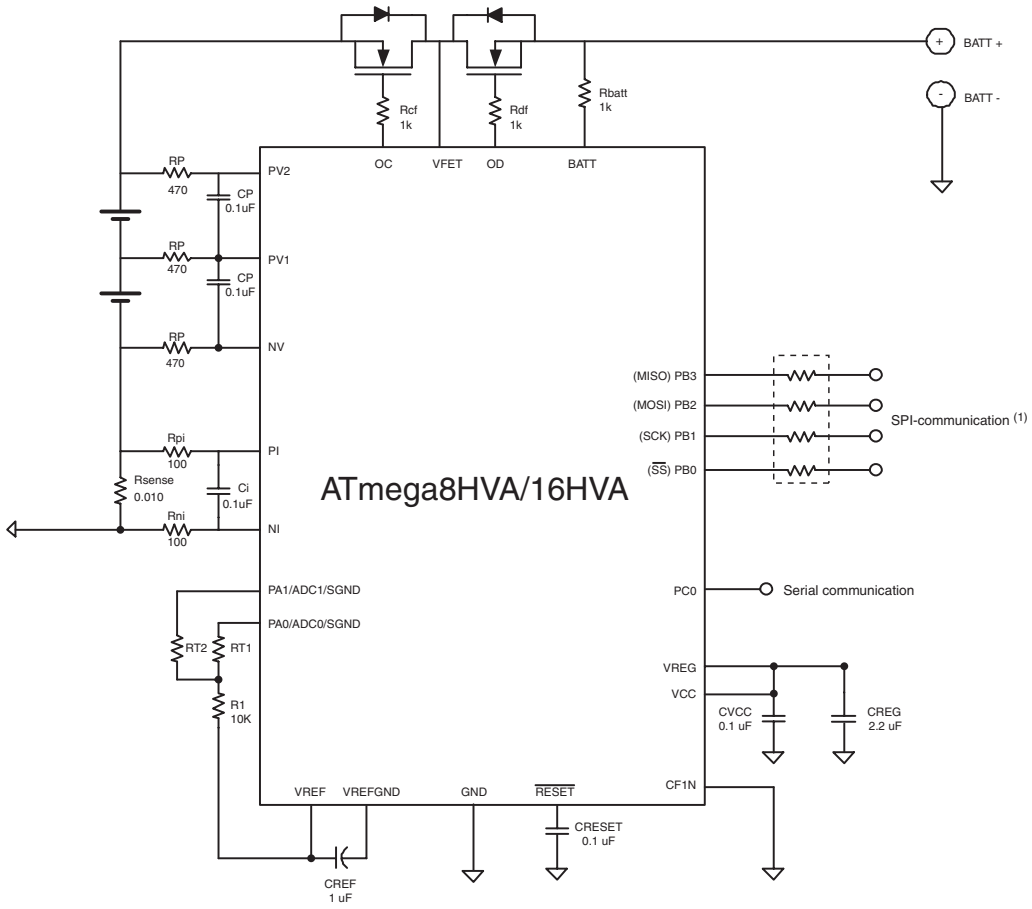
25.6.1 DWDR – debugWire Data Register

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.

28. Operating Circuit

Figure 28-1. Operating Circuit Diagram, 2-cell.



Notes:
1. The series resistors on the SPI lines are required for In-System Programming and On-chip Debug support. The value of the series resistor depends on the application. A value of 10k will ensure that programming and debugging operates correctly, but it must be determined by the end user that this does not affect the normal operation of the SPI interface.
2. PA1 should be connected to SGND when measuring $V(RT_2)$.
   PA0 should be connected to SGND when measuring $V(RT_1)$.
3. It is recommended to connect CF1P, CF2N, and CF2P to GND.

Table 29-1. Electrical Characteristics[1] (T_A = -10°C to 70°C unless otherwise specified) (Continued)

	Parameter	Condition	Min	Typ	Max	Unit
Coulomb Counter	Reference Voltage			± 110		mV
	Conversion Time and Resolution[5]	26.9 µV Resolution		3.9		ms
		0.84 µV Resolution		1000		ms
	INL[5]				4	LSB
	CC-ADC Offset[5][6]			2.5	± 15	LSB
	Gain Error[3]	-100 mV < $V_{PI - NI}$ < 100 mV		± 0.1	± 1	%
Temperature Sensor	V_{PTAT}, Voltage Proportional to Absolute Temperature			0.67		mV/K
	Absolute Accuracy[4]	Measured in Active mode		± 2	± 5	K
Slow RC Oscillator[10]	Frequency		91	131	171	kHz
	Frequency drift over temperature[5]			1.5		%
	Slow RC Frequency predicion error[5]				1	%
Ultra Low Power RC Oscillator[10]	Frequency		89	128	167	kHz
	Frequency drift over temperature[5]			6		%

Notes:
1. All DC Characteristics contained in this data sheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.
2. Voltage Regulator performance is based on 220 nF fly capacitors and 2.2 µF smooth capacitor.
3. After VREF calibration at a second temperature. By default the first calibration is performed at temperature T_{HOT} in Atmel factory test. The value of T_{HOT} is stored in the signature row. The second calibration step can easily be implemented in a standard test flow at room temperature.
4. The measured V_{PTAT} voltage must be scaled with the calibration value stored in the VPTAT Calibration Register to get the absolute temperature. The specified value represents target accuracy after Atmel factory calibration. Accuracy can be further improved by doing a system calibration measurement at a well-known temperature.
5. This value is not tested in production.
6. After software offset compensation, using the polarity switching (CADPOL) feature.
7. After scaling of VADC raw data using Gain and Offset Calibration values stored in Signature Row.
8. Actual offset for each channel stored in signature row can be used to remove this offset error.
9. If the cell input needs to be measured when PV1 is below 1.5V, Atmel can provide data that facilitates less accurate measurements in this range.
10. Actual frequency measured at Atmel factory stored in signature row.

29.3 External Interrupt Characteristics

Table 29-2. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns