Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32x8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz and 1.8V
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 128K Bytes of In-System Self-Programmable Flash
    - Endurance: 2000 Write/Erase Cycles @ 85°C
  - 4K Bytes EEPROM
    - Endurance: 2000 Write/Erase Cycles @ 85°C
  - 16K Bytes Internal SRAM
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash EEPROM, Fuses and Lock Bits through the JTAG interface
- Peripheral Features
  - Multiple Timer/Counter & PWM channels
  - Real Time Counter with Separate Oscillator
  - 10-bit, 330 ks/s A/D Converter; Analog Comparator; On-chip Temperature Sensor
  - Master/Slave SPI Serial Interface
  - Two Programmable Serial USART
  - Byte Oriented 2-wire Serial Interface
- Advanced Interrupt Handler
- Watchdog Timer with Separate On-Chip Oscillator
- Power-on Reset and Low Current Brown-Out Detector
- Advanced Power Save Modes
- Fully integrated Low Power Transceiver for 2.4 GHz ISM Band
  - Supported Data Rates: 250 kb/s and 500 kb/s, 1 Mb/s, 2 Mb/s
  - -100 dBm RX Sensitivity; TX Output Power up to 3.5 dBm
  - Hardware Assisted MAC (Auto-Acknowledge, Auto-Retry)
  - 32 Bit IEEE 802.15.4 Symbol Counter
  - Baseband Signal Processing
  - SFR-Detection, Spreading; De-Spreading; Framing ; CRC-16 Computation
  - Antenna Diversity and TX/RX control
  - TX/RX 128 Byte Frame Buffer
- Hardware Security (AES, True Random Generator)
- Integrated Crystal Oscillators (32.768 kHz & 16 MHz)
- I/O and Package
  - 38 Programmable I/O Lines
  - 64-pad QFN (RoHS/Fully Green)
- Temperature Range: -40°C to 85°C Industrial
- Supply voltage range 1.8V to 3.6V with integrated voltage regulators
- Ultra Low Power consumption (1.8 to 3.6V) for Rx/Tx & AVR: <18.6 mA
  - CPU Active Mode (16MHz): 4.1 mA
  - 2.4GHz Transceiver: RX_ON 12.5 mA / TX 14.5 mA (maximum TX output power)
  - Deep Sleep Mode: <250nA @ 25°C
- Speed Grade: 0 – 16 MHz @ 1.8 – 3.6V

Applications

- ZigBee® / IEEE 802.15.4-2006/2003™ – Full And Reduced Function Device (FFD/RFD)
- General Purpose 2.4GHz ISM Band Transceiver with Microcontroller
- RF4CE, SP100, WirelessHART™, ISM Applications and IPv6 / 6LoWPAN

8-bit AVR® Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4

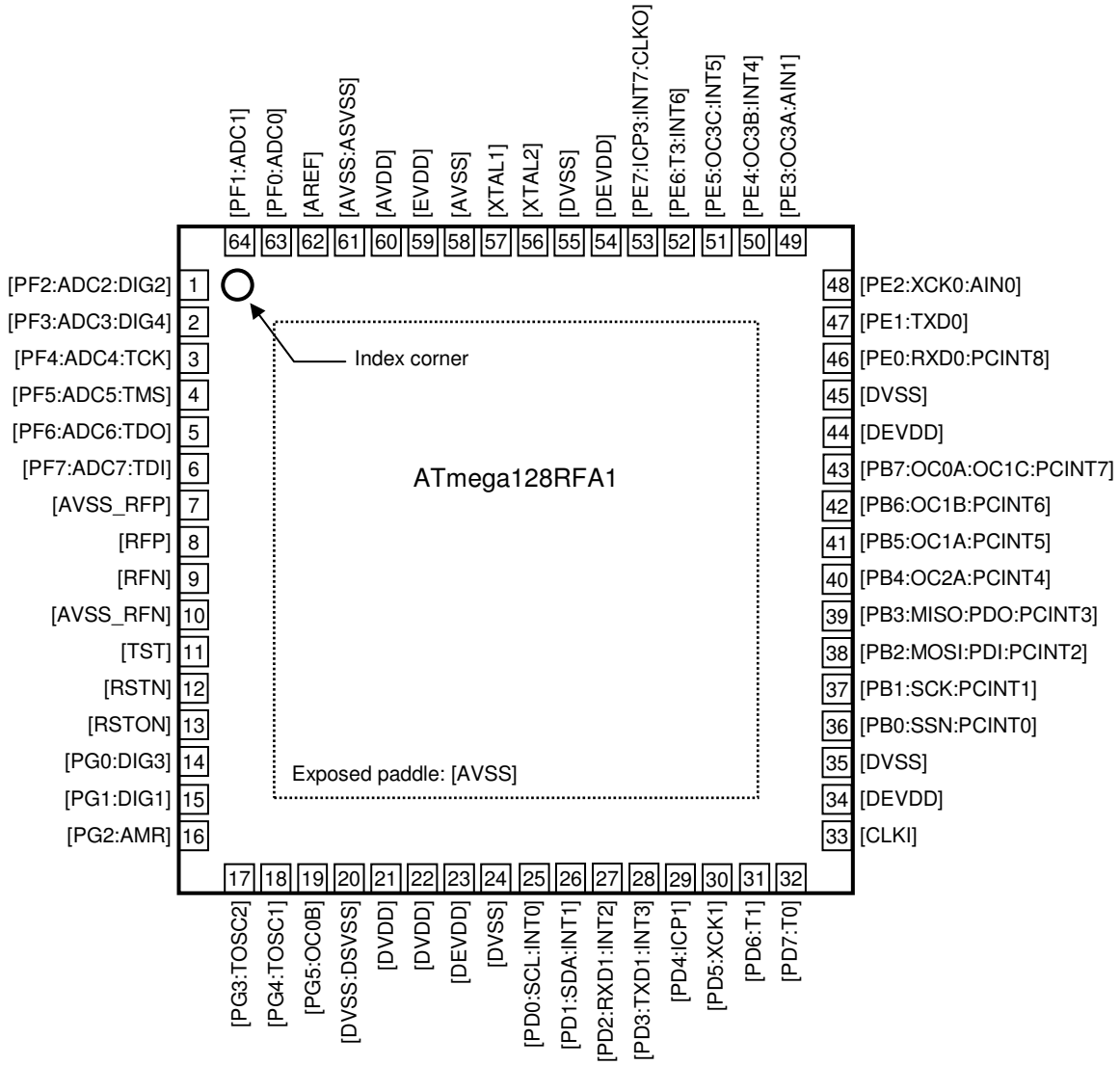
ATmega128RFA1

PRELIMINARY

8266A-MCU Wireless-12/09

1 Pin Configurations

Figure 1-1. Pinout ATmega128RFA1



Note: The large center pad underneath the QFN/MLF package is made of metal and internally connected to AVSS. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board

2 Disclaimer

Typical values contained in this datasheet are based on simulation and characterization results of other AVR microcontrollers and radio transceivers manufactured in a similar process technology. Minimum and Maximum values will be available after the device is characterized.

3 Overview

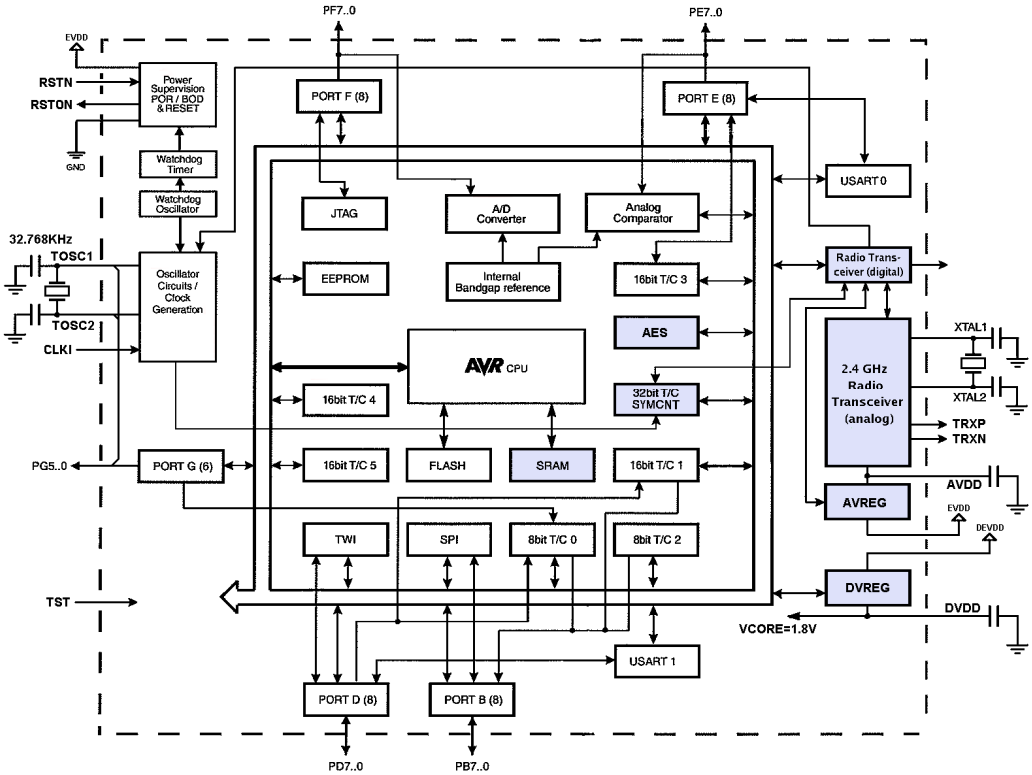
The ATmega128RFA1 is a low-power CMOS 8 bit microcontroller based on the AVR enhanced RISC architecture combined with a high data rate transceiver for the 2.4 GHz ISM band. It is derived from the ATmega1281 microcontroller and the AT86RF231 radio transceiver.

By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The radio transceiver provides high data rates from 250 kb/s up to 2 Mb/s, frame handling, outstanding receiver sensitivity and high transmit output power enabling a very robust wireless communication.

3.1 Block Diagram

Figure 3-1 Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU). Two independent registers can be accessed with one single instruction executed in one clock cycle. The resulting architecture is very code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The system includes internal voltage regulation and an advanced power management. Distinguished by the small leakage current it allows an extended operation time from battery.

The radio transceiver is a fully integrated ZigBee solution using a minimum number of external components. It combines excellent RF performance with low cost, small size and low current consumption. The radio transceiver includes a crystal stabilized fractional-N synthesizer, transmitter and receiver, and full Direct Sequence Spread

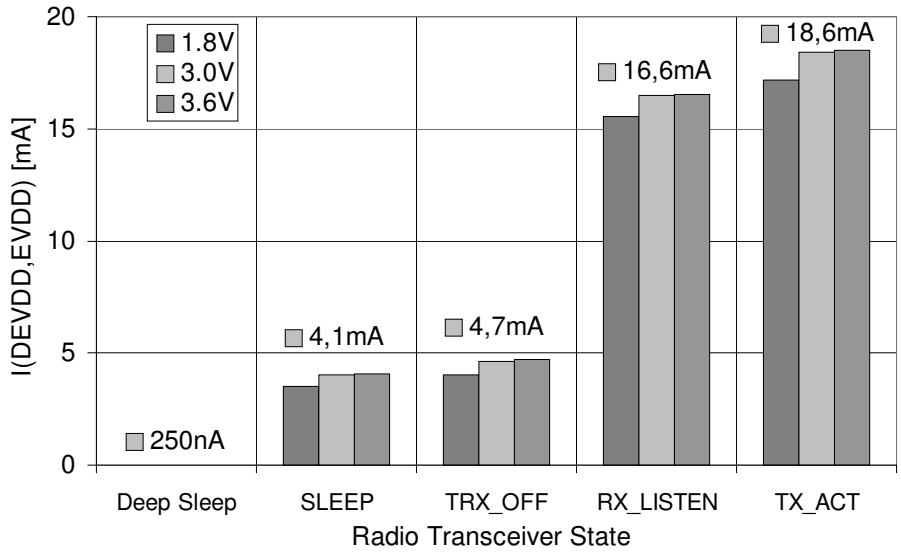
Spectrum Signal (DSSS) processing with spreading and despreading. The device is fully compatible with IEEE802.15.4-2006/2003 and ZigBee standards.

The ATmega128RFA1 provides the following features: 128 kbytes of In-System Programmable (ISP) Flash with read-while-write capabilities, 4 kbytes EEPROM, 16 kbytes SRAM, up to 35 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), 6 flexible Timer/Counters with compare modes and PWM, USART, a byte oriented 2-wire Serial Interface, a 8 channel, 10 bit analog to digital converter (ADC) with an optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, a SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and 6 software selectable power saving modes.

The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the RC oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main RC oscillator and the asynchronous timer continue to run.

Typical supply current of the microcontroller with CPU clock set to 16MHz and the radio transceiver for the most important states is shown in the Figure 3-2 below.

Figure 3-2 Radio transceiver and microcontroller (16MHz) supply current



The transmit output power is set to maximum. If the radio transceiver is in SLEEP mode the current is dissipated by the AVR microcontroller only.

In Deep Sleep mode all major digital blocks with no data retention requirements are disconnected from main supply providing a very small leakage current. Watchdog timer, MAC symbol counter and 32.768kHz oscillator can be configured to continue to run.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system

trough an SPI serial interface, by a conventional nonvolatile memory programmer, or by on 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 on a monolithic chip, the Atmel ATmega128RFA1 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega128RFA1 AVR is supported with a full suite of program and system development tools including: C compiler, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

3.2 Pin Descriptions

3.2.1 EVDD

External analog supply voltage;

3.2.2 DEVDD

External digital supply voltage;

3.2.3 AVDD

Regulated analog supply voltage (internally generated);

3.2.4 DVDD

Regulated digital supply voltage (internally generated);

3.2.5 DVSS

Digital ground;

3.2.6 AVSS

Analog ground;

3.2.7 Port B (PB7...PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also provides functions of various special features of the ATmega128RFA1.

3.2.8 Port D (PD7...PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also provides functions of various special features of the ATmega128RFA1.

3.2.9 Port E (PE7...PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also provides functions of various special features of the ATmega128RFA1.

3.2.10 Port F (PF7...PF0)

Port F is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port F also provides functions of various special features of the ATmega128RFA1.

3.2.11 Port G (PG5…PG0)

Port G is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. However the driver strength of PG3 and PG4 is reduced compared to the other port pins. The output voltage drop (V_{OH}, V_{OL}) is higher while the leakage current is smaller. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also provides functions of various special features of the ATmega128RFA1.

3.2.12 AVSS_RFP

AVSS_RFP is a dedicated ground pin for the bi-directional, differential RF I/O port.

3.2.13 AVSS_RFN

AVSS_RFN is a dedicated ground pin for the bi-directional, differential RF I/O port.

3.2.14 RFP

RFP is the positive terminal for the bi-directional, differential RF I/O port.

3.2.15 RFN

RFN is the negative terminal for the bi-directional, differential RF I/O port.

3.2.16 RSTN

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. Shorter pulses are not guaranteed to generate a reset.

3.2.17 RSTON

Reset output. A low level on this pin indicates a reset initiated by the internal reset sources or the pin RSTN.

3.2.18 XTAL1

Input to the inverting 16MHz crystal oscillator amplifier. In general a crystal between XTAL1 and XTAL2 provides the 16MHz reference clock of the radio transceiver.

3.2.19 XTAL2

Output of the inverting 16MHz crystal oscillator amplifier;

3.2.20 AREF

Reference voltage output of the A/D Converter. In general this pin is left open.

3.2.21 TST

Programming and test mode enable pin;

3.2.22 CLKI

Input to the clock system. If selected, it provides the operating clock of the microcontroller.

3.3 Compatibility to ATmega1281/2561

The basic AVR feature set of the ATmega128RFA1 is derived from the ATmega1281/2561. Address locations and names of the implemented modules and

registers are unchanged as long as it fits the target application of a very small and power efficient radio system. In addition, several new features were added.

Backward compatibility of the ATmega128RFA1 to the ATmega1281/2561 is provided in most cases. However some incompatibilities between the microcontrollers exist.

3.3.1 Port A and Port C

Port A and Port C are not implemented. The associated registers are available but will not provide any port control. Remaining ports are kept at their original address location to not require changes of existing software packages.

3.3.2 External Memory Interface

The alternate pin function “External Memory interface” using Port A and Port C is not implemented due to the missing ports.

The large internal data memory (SRAM) does not require an external memory and the associated parallel interface. It keeps the system radiation (EMC) at a very small level to provide very high sensitivity at the antenna input.

3.3.3 High Voltage Programming Mode

Alternate pin function BS2 (high voltage programming) of pin PA0 is mapped to a different pin. Entering the parallel programming mode is controlled by the TST pin.

3.3.4 AVR Oscillators and External Clock

The AVR microcontroller can utilize the high performance crystal oscillator of the 2.4GHz transceiver connected to the pins XTAL1 and XTAL2. An external clock can be applied to the microcontroller using the clock input CLKI.

3.3.5 Analog Frontend

The ATmega128RFA1 has a new A/D converter. Software compatibility is basically assured. Nevertheless to benefit from the higher conversion speeds and the better performance some changes are required.

4 Resources

A comprehensive set of development tools and application notes, and datasheets are available for download on http://www.atmel.com.

5 About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. 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.

These code examples assume that the part specific header file is included before compilation. 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".

6 Data Retention

Reliability Qualification results show that the projected data retention failure rate for the given ambient temperature is less than TBD PPM

- over 10 years at 85°C
- TBD years at 25°C.

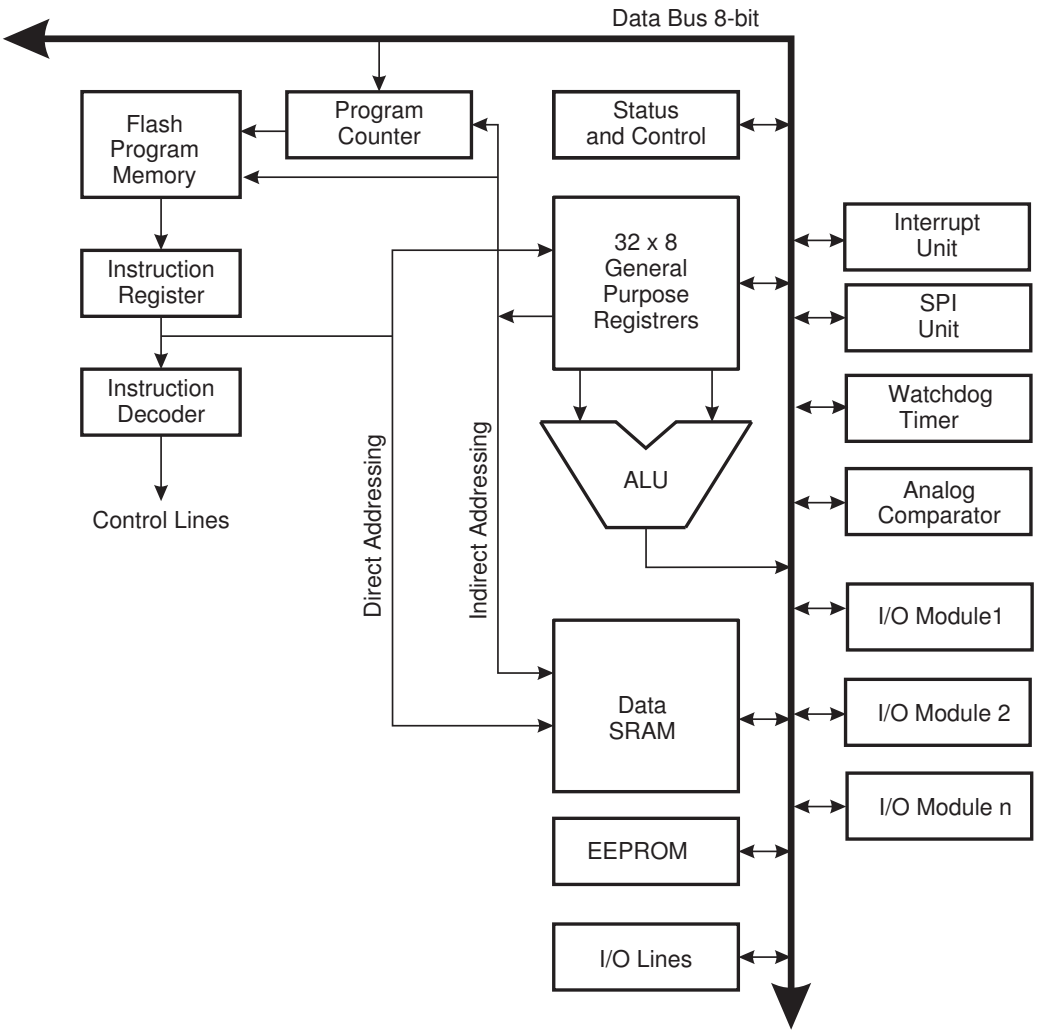
7 AVR CPU Core

7.1 Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculation, control peripherals, and handle interrupts.

7.2 Architectural Overview

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 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.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher 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 ATmega128RFA1 has Extended I/O space from 0x60 - 0x1FF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

7.3 ALU – Arithmetic Logic Unit

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

7.4 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set 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.

7.4.1 SREG – Status Register

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I - Global Interrupt Enable**

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable bit is cleared (zero), 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.

- **Bit 6 – T - Bit Copy Storage**

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and 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. See the Instruction Set Description for detailed information.

- **Bit 4 – S - Sign Bit**

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 after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

- **Bit 1 – Z - Zero Flag**

The zero flag Z indicates a zero result after the different arithmetic and logic operations. 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. Note that the status register is not automatically

stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

7.5 General Purpose Register File

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

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 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 7-1 below shows the structure of the 32 general purpose working registers in the CPU.

Figure 7-1. AVR CPU General Purpose Working Registers

	7 0	Addr.	
	R0	0x00	
	R1	0x01	
	R2	0x02	
	…		
	R13	0x0D	
General	R14	0x0E	
Purpose	R15	0x0F	
Working	R16	0x10	
Registers	R17	0x11	
	…		
	R26	0x1A	X-register Low Byte
	R27	0x1B	X-register High Byte
	R28	0x1C	Y-register Low Byte
	R29	0x1D	Y-register High Byte
	R30	0x1E	Z-register Low Byte
	R31	0x1F	Z-register High Byte

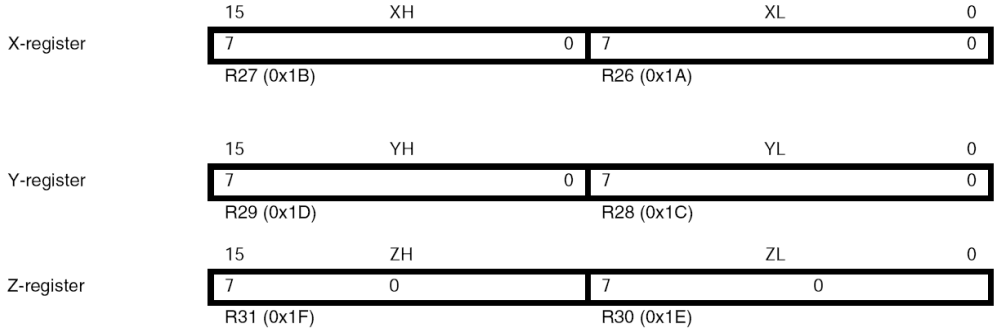
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 7-1 above on page 12, 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.

7.5.1 The X-register, Y-register, and Z-register

The registers R26...R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 7-2 on page 13.

Figure 7-2. The X-, Y-, 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).

7.6 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x0200. The initial value of the stack pointer is the last address of the internal SRAM.

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.

When the FLASH memory exceeds 128Kbyte one additional cycle is required. In this case the Stack Pointer is decremented by three when the return address is pushed onto the Stack with subroutine call or interrupt and is incremented by three when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

7.6.1 SPH – Stack Pointer High

Bit	7	6	5	4	3	2	1	0	
$3E ($5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	1	0	0	0	0	1	

The AVR Stack Pointer is implemented as two 8-bit registers SPL and SPH in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

- **Bit 7:0 – SP15:8 - Stack Pointer High Byte**

7.6.2 SPL – Stack Pointer Low

Bit	7	6	5	4	3	2	1	0	
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

The AVR Stack Pointer is implemented as two 8-bit registers SPL and SPH in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

- **Bit 7:0 – SP7:0 - Stack Pointer Low Byte**

7.6.3 RAMPZ – Extended Z-pointer Register for ELPM/SPM

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	Res5	Res4	Res3	Res2	Res1	Res0	RAMPZ1	RAMPZ0	RAMPZ
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

For ELPM/SPM instructions, the Z-pointer is a concatenation of RAMPZ, ZH, and ZL. Note that LPM is not affected by the RAMPZ setting.

- **Bit 7:2 – Res5:0 - Reserved**

For compatibility with future devices, be sure to write these bits to zero.

- **Bit 1:0 – RAMPZ1:0 - Extended Z-Pointer Value**

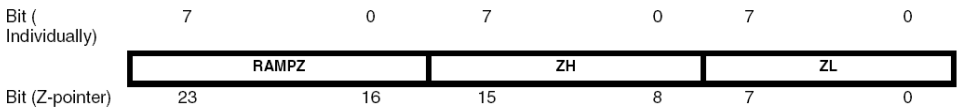
These two bits represent the MSB's of the Z-Pointer.

Table 7-2 RAMPZ Register Bits

Register Bits	Value	Description
RAMPZ1:0	0	Default value of Z-pointer MSB's.

For ELPM/SPM instructions, the Z-pointer is a concatenation of RAMPZ, ZH, and ZL, as shown in Figure 7-3 below. Note that LPM is not affected by the RAMPZ setting.

Figure 7-3. The Z-pointer used by ELPM and SPM



The actual number of bits is implementation dependent. Unused bits in an implementation will always read as zero. For compatibility with future devices, be sure to write these bits to zero.

7.7 Instruction Execution Timing

Figure 7-4. The Parallel Instruction Fetches and Instruction Executions

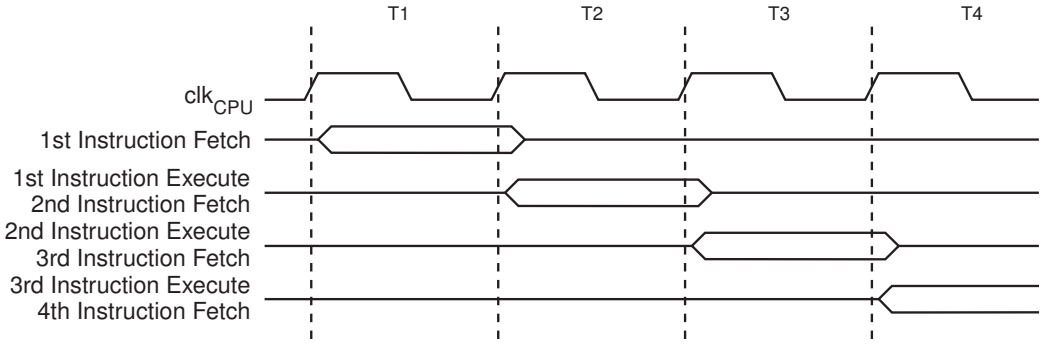
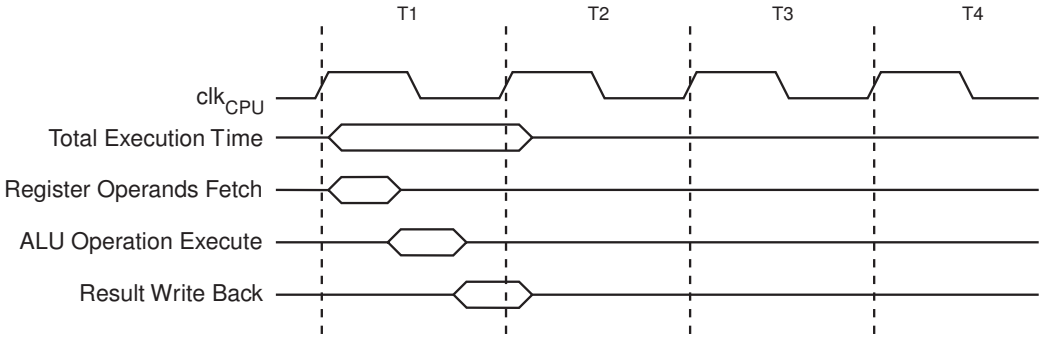


Figure 7-5 below shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7-5. Single Cycle ALU operation



7.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 464 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 211. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to "Interrupts" on page 211 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see "Memory Programming" on page 464.

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 */
__disable_interrupt();
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
```

Assembly Code Example
`; note: will enter sleep before any pending` `; interrupt(s)`
C Code Example

`__enable_interrupt(); /* set Global Interrupt Enable */`
`__sleep(); /* enter sleep, waiting for interrupt */`
`/* note: will enter sleep before any pending interrupt(s) */`

7.8.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is five clock cycles minimum. After five clock cycles the program vector address for the actual interrupt handling routine is executed. During these five 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 five 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 five clock cycles. During these five clock cycles, the Program Counter (three bytes) is popped back from the Stack, the Stack Pointer is incremented by three, and the I-bit in SREG is set.

8 AVR Memories

This section describes the different memories in the ATmega128RFA1. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega128RFA1 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

8.1 In-System Reprogrammable Flash Program Memory

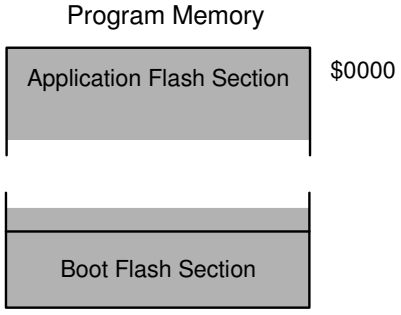
The ATmega128RFA1 contains 128K bytes On-chip In-System Reprogrammable Flash memory for program storage, see Figure 8-6 below. Since all AVR instructions are 16 or 32 bits wide, the Flash is 16 bit wide. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 2000 write/erase cycles. The ATmega128RFA1 Program Counter (PC) is 16 bits wide, thus addressing the required program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 450. "Memory Programming" on page 464 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

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

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 15.

Figure 8-6. Program Flash Memory Map



8.2 SRAM Data Memory

Figure 8-7 on page 19 shows how the ATmega128RFA1 SRAM Memory is organized. The ATmega128RFA1 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from $060 – $1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The first 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 416 locations of Extended I/O memory and the following 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.

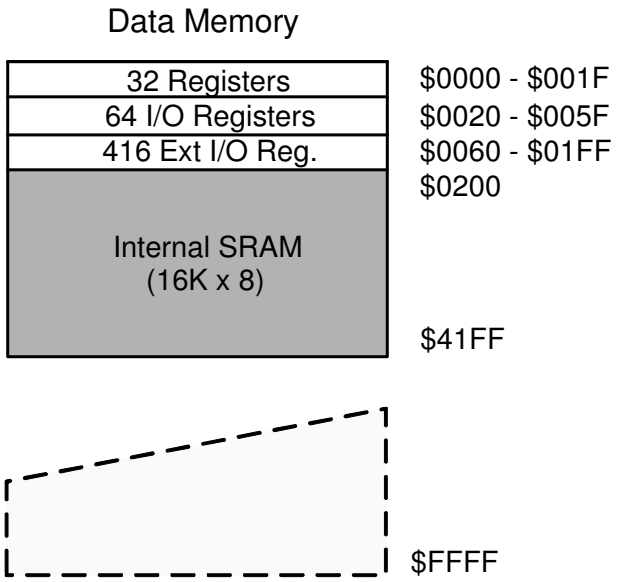
The direct addressing reaches the entire data space.

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, and the internal data SRAM in the ATmega128RFA1 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 12.

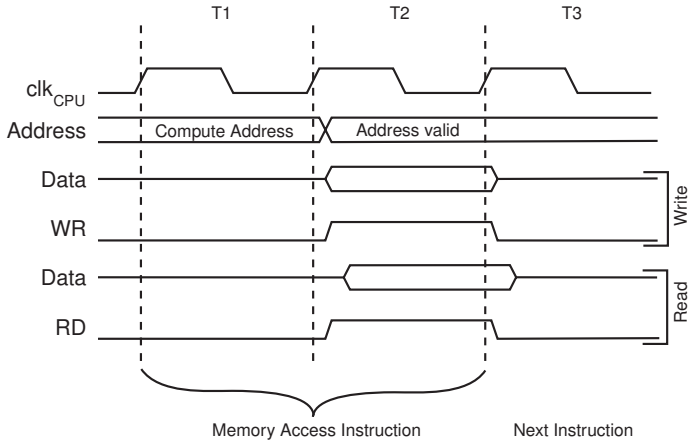
Figure 8-7. Data Memory Map



8.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. Access to the internal data SRAM is performed in two clk_{CPU} cycles as described in Figure 8-8 on page 20.

Figure 8-8. On-Chip Data SRAM Access Cycles



8.3 EEPROM Data Memory

The ATmega128RFA1 contains 4Kbyte 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 2000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG and Parallel data downloading to the EEPROM, see "Serial Downloading" on page 477, "Programming via the JTAG Interface" on page 481, and "Programming the EEPROM" on page 491 respectively.

8.3.1 EEPROM Read Write Access

The EEPROM Access Registers are accessible in the I/O space, see "EEPROM Register Description" on page 23.

The write access time for the EEPROM is given in Table 8-3 below. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, DVDD is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 22 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. See the description of the EEPROM Control Register for details on this, "EEPROM Register Description" on page 23.

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.

The calibrated Oscillator is used to time the EEPROM accesses. The following table lists the typical programming time for EEPROM access from the CPU.

Table 8-3. EEPROM Programming Time

Symbol	Typical Programming time
EEPROM write (from CPU)	**4ms**
EEPROM erase (from CPU)	**8ms**

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
```

Assembly Code Example

```
    sbic EECR,EEPE
    rjcmp 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;
}
```

8.3.2 Preventing EEPROM Corruption

During periods of low DEVDD, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low VCC reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

8.4 EEPROM Register Description

8.4.1 EEARH – EEPROM Address Register High Byte

Bit	7	6	5	4	3	2	1	0	
$22 ($42)	Res3	Res2	Res1	Res0	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	X	X	X	X	

The EEPROM Address Registers EEARH and EEARL specify the EEPROM address in the 4K bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 4096. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

- **Bit 7:4 – Res3:0 - Reserved**
- **Bit 3:0 – EEAR11:8 - EEPROM Address**

8.4.2 EEARL – EEPROM Address Register Low Byte

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	X	X	X	X	X	X	X	X	

The EEPROM Address Registers EEARH and EEARL specify the EEPROM address in the 4K bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 4096. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

- **Bit 7:0 – EEAR7:0 - EEPROM Address**

8.4.3 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
$20 ($40)	EEDR7:0								EEDR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

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.

- **Bit 7:0 – EEDR7:0 - EEPROM Data**

8.4.4 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
$1F ($3F)	Res1	Res0	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	X	X	0	0	X	0	

- **Bit 7:6 – Res1:0 - Reserved**
- **Bit 5:4 – EEPM1:0 - EEPROM Programming Mode**

The EEPROM Programming mode bit setting defines which programming action 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 the following table. While EEPE is set, any write to EEPM1:0 will be ignored. During reset, the EEPM1:0 bits will be reset to 0 unless the EEPROM is busy programming.

Table 8-4 EEPM Register Bits

Register Bits	Value	Description
EEPM1:0	0x00	Erase and Write in one operation (Atomic Operation)
	0x01	Erase only
	0x02	Write only
	0x03	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 Programming 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 adopted when writing the EEPROM (the order of steps 3 and 4 is not essential):

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

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted.

Caution: an interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all 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.

- **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.

8.5 I/O Memory

The Input/Output (I/O) space definition of the ATmega128RFA1 is shown in "Register Summary" on page 496.

All ATmega128RFA1 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 AVR instruction set 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 ATmega128RFA1 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 – 0x1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits may not be modified. Reserved registers and I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The control registers of I/O and peripherals are explained in later sections.

8.6 General Purpose I/O Registers

The ATmega128RFA1 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.

8.6.1 GPIOR0 – General Purpose IO Register 0

Bit	7	6	5	4	3	2	1	0	
$1E ($3E)	GPIOR07:00								GPIOR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The three General Purpose I/O Registers can be used for storing any information.

- **Bit 7:0 – GPIOR07:00 - General Purpose I/O Register 0 Value**

8.6.2 GPIOR1 – General Purpose IO Register 1

Bit	7	6	5	4	3	2	1	0	
$2A ($4A)	GPIOR17:10								GPIOR1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The three General Purpose I/O Registers can be used for storing any information.

- **Bit 7:0 – GPIOR17:10 - General Purpose I/O Register 1 Value**

8.6.3 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
$2B ($4B)	GPIOR27:20								GPIOR2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The three General Purpose I/O Registers can be used for storing any information.

- **Bit 7:0 – GPIOR27:20 - General Purpose I/O Register 2 Value**

8.7 Other Port Registers

The inherited control registers of missing ports located in the I/O space are kept in the ATmega128RFA1. They can be used as general purpose I/O registers for storing any information. Registers placed in the address range 0x00 – 0x1F are directly bit-accessible using the SBI, CBI, SBIS and SBIC instructions.

8.7.1 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
$02 ($22)	PORTA7:0								PORTA
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The PORTA register can be used as a General Purpose I/O Register for storing any information.

- **Bit 7:0 – PORTA7:0 - Port A Data Register Value**

8.7.2 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$01 ($21)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDRA register can be used as a General Purpose I/O Register for storing any information.

- **Bit 7:0 – DDA7:0 - Port A Data Direction Register Value**

8.7.3 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$00 ($20)	PINA7:0								PINA
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The PINA register is reserved for interal use and cannot be used as a General Purpose I/O Register.

- **Bit 7:0 – PINA7:0 - Port A Input Pins**

8.7.4 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	PORTC7:0								PORTC
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The PORTC register can be used as a General Purpose I/O Register for storing any information.

- **Bit 7:0 – PORTC7:0 - Port C Data Register Value**

8.7.5 DDRC – Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$07 ($27)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDRC register can be used as a General Purpose I/O Register for storing any information.

- **Bit 7:0 – DDC7:0 - Port C Data Direction Register Value**

8.7.6 PINC – Port C Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$06 ($26)	PINC7:0								PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The PINC register is reserved for interal use and cannot be used as a General Purpose I/O Register.

- **Bit 7:0 – PINC7:0 - Port C Input Pins**

9 Low-Power 2.4 GHz Transceiver

9.1 Features

- High performance RF-CMOS 2.4 GHz radio transceiver targeted for IEEE 802.15.4™, ZigBee™, IPv6 / 6LoWPAN, RF4CE, SP100, WirelessHART™ and ISM applications
- Outstanding link budget (103.5 dB):
  - Receiver sensitivity -100 dBm
  - Programmable output power from -17 dBm up to +3.5 dBm
- Ultra-low current consumption:
  - TRX_OFF = 0.4 mA
  - RX_ON = 12.5 mA
  - BUSY_TX = 14.5 mA (at max. transmit power of +3.5 dBm)
- Optimized for low BoM cost and ease of production:
  - Few external components necessary (crystal, capacitors and antenna)
  - Excellent ESD robustness
- Easy to use interface:
  - Registers and frame buffer access from software
  - Dedicated radio transceiver interrupts
- Radio transceiver features:
  - 128 byte FIFO (SRAM) for data buffering
  - Integrated RX/TX switch
  - Fully integrated, fast settling PLL to support frequency hopping
  - Battery monitor
  - Fast wake-up time < 0.25 ms
- Special IEEE 802.15.4 2006 hardware support:
  - FCS computation and clear channel assessment (CCA)
  - RSSI measurement, energy detection and link quality indication
- MAC hardware accelerator:
  - Automated acknowledgement, CSMA-CA and frame retransmission
  - Automatic address filtering
  - Automated FCS check
- Extended Feature Set Hardware Support:
  - AES 128 bit hardware accelerator
  - RX/TX indication (external RF front-end control)
  - RX antenna diversity
  - Supported PSDU data rates: 250 kb/s, 500 kb/s, 1 Mb/s and 2 Mb/s
  - True random number generation for security applications
- Compliant to IEEE 802.15.4-2006, IEEE 802.15.4-2003 and RF4CE
- Compliant to EN 300 328/440, FCC-CFR-47 Part 15, ARIB STD-66, RSS-210

The ATmega128RFA1 features a low-power 2.4 GHz radio transceiver designed for industrial and consumer ZigBee/IEEE 802.15.4, 6LoWPAN, RF4CE and high data rate 2.4 GHz ISM band applications. The radio transceiver is a true peripheral block of the AVR microcontroller. All RF-critical components except the antenna, crystal and de-coupling capacitors are integrated on-chip. Therefore, the ATmega128RFA1 is particularly suitable for applications like:

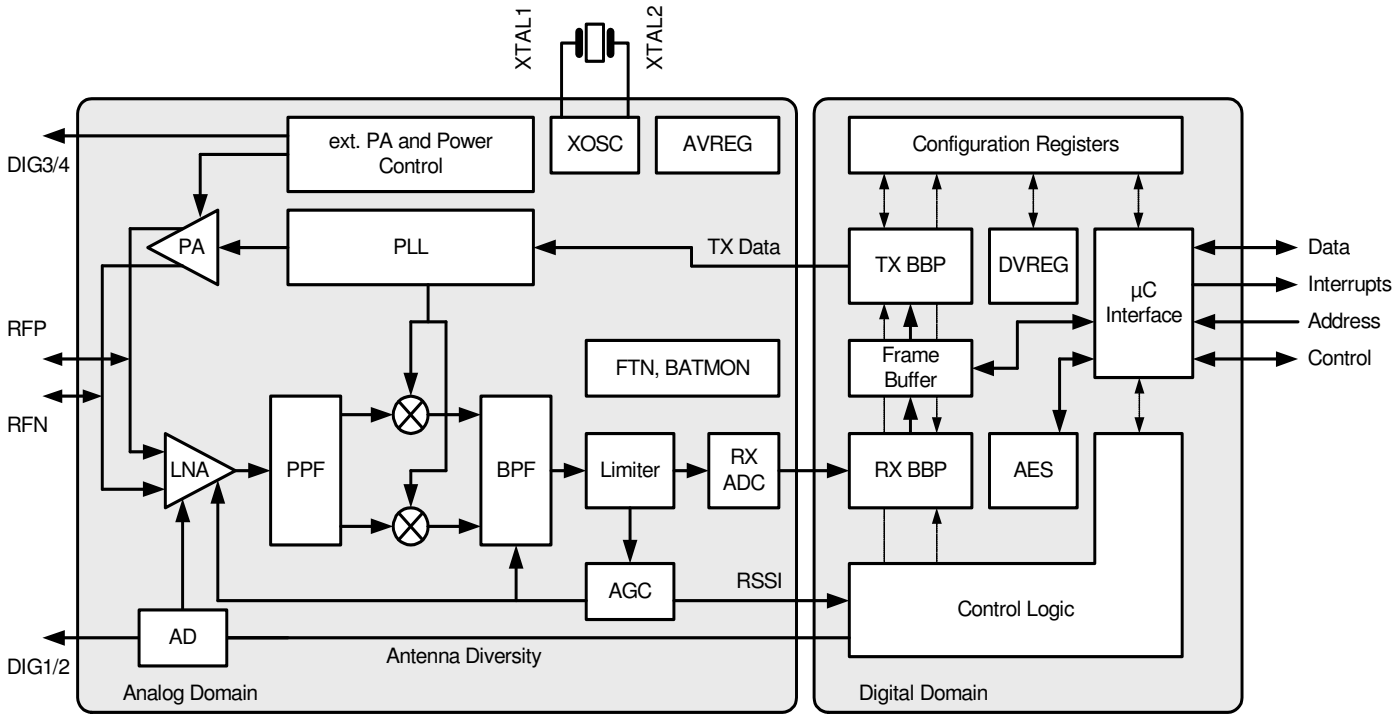
- 2.4 GHz IEEE 802.15.4 and ZigBee systems
- 6LoWPAN and RF4CE systems
- Wireless sensor networks
- Industrial control, sensing and automation (SP100, WirelessHART)
- Residential and commercial automation
- Health care
- Consumer electronics
- PC peripherals

9.2 General Circuit Description

This radio transceiver is part of a system-on-chip solution with an AVR® microcontroller. It comprises a complex peripheral component containing the analog radio, digital modulation and demodulation including time and frequency synchronization and data buffering. The number of external components for the transceiver operation is minimized such that only the antenna, the crystal and decoupling capacitors are required. The bidirectional differential antenna pins (RFP, RFN) are used for transmission and reception, thus no external antenna switch is needed.

The ATmega128RFA1 block diagram is shown in Figure 9-9 below.

Figure 9-9. Transceiver Block Diagram



The received RF signal at pins RFN and RFP is differentially fed through the low-noise amplifier (LNA) to the RF filter (PPF) to generate a complex signal, driving the integrated channel filter (BPF). The limiting amplifier provides sufficient gain to drive the succeeding analog-to-digital converter (RX ADC) and generates a digital RSSI signal. The RX ADC output signal is sampled by the digital base band receiver (RX BBP).

The transmit modulation scheme is offset-QPSK (O-QPSK) with half-sine pulse shaping and 32-length block coding (spreading) according to [1] on page 100 and [2] on page 100. The modulation signal is generated in the digital transmitter (TX BBP) and applied to the fractional-N frequency synthesis (PLL), to ensure the coherent phase modulation required for demodulation of O-QPSK signals. The frequency-modulated signal is fed to the power amplifier (PA).

A differential pin pair DIG3/DIG4 can be enabled to control an external RF front-end.

The two on-chip low-dropout voltage regulators (A|DVREG) provide the analog and digital 1.8V supply.

An internal 128-byte RAM for RX and TX (Frame Buffer) buffers the data to be transmitted or received.

The configuration of the reading and writing of the Frame Buffer is controlled via the microcontroller interface.

The transceiver further contains comprehensive hardware-MAC support (Extended Operating Mode) and a security engine (AES) to improve the overall system power efficiency and timing. The 128-bit AES engine can be accessed in parallel to all PHY operational transactions and states using the microcontroller interface, except during transceiver power down state.

For applications not necessarily targeting IEEE 802.15.4 compliant networks, the radio transceiver also supports alternative data rates up to 2 Mb/s.

For long-range applications or to improve the reliability of an RF connection the RF performance can further be improved by using an external RF front-end or Antenna Diversity. Both operation modes are supported by the radio transceiver with dedicated control pins without the interaction of the microcontroller.

Additional features of the Extended Feature Set, see section "Radio Transceiver Extended Feature Set" on page 85, are provided to simplify the interaction between radio transceiver and microcontroller.

9.3 Transceiver to Microcontroller Interface

This section describes the internal Interface between the transceiver module and the microcontroller. Unlike all other AVR I/O modules, the transceiver module can operate asynchronously to the controller. The transceiver requires an accurate 16MHz crystal clock for operation, but the controller can run at any frequency within its operating limits.

9.3.1 Transceiver Configuration and Data Access

9.3.1.1 Register Access

All transceiver registers are mapped into I/O space of the controller. Due to the asynchronous interface a register access can take up to three transceiver clock cycles. Depending on the controller clock speed, program execution wait cycles are generated. That means if the controller runs with about 16MHz or faster, at least three wait cycles are generated, but if the controller runs with about 4MHz, no wait cycles are inserted. A

register access is only possible, if the transceiver clock is available. Therefore the transceiver must be enabled (PRR1 Register) and not in SLEEP state.

9.3.1.2 Frame Buffer Access

The 128-byte Frame Buffer can hold the PHY service data unit (PSDU) data of one IEEE 802.15.4 compliant RX or one TX frame of maximum length at a time. A detailed description of the Frame Buffer can be found in section "Frame Buffer" on page 77. An introduction to the IEEE 802.15.4 frame format can be found in section "Introduction – IEEE 802.15.4-2006 Frame Format" on page 61.

The Frame Buffer is located within the controller I/O address space above of the transceiver register set. The first byte of the Frame Buffer can be accessed with the symbolical address TRXFBST and the last byte can be accessed with the symbolical address TRXFBEND. Random access to single frame bytes is possible with “TRXFBST + byte index” or “TRXFBEND – byte index”. In contrast to the transceiver register access, the Frame Buffer allows single cycle read/write operations for all controller clock speeds.

The content of the Frame Buffer is only overwritten by a new received frame or a Frame Buffer write access.

The Frame Buffer usage is different between received and transmitted frames. Therefore it is not possible to retransmit a received frame without modifying the frame buffer.

On received frames, the frame length byte is not stored in the Frame Buffer, but can be accessed over the TST_FRAME_LENGTH register. During frame receive, the Link Quality Indication (LQI) value (refer to "Link Quality Indication (LQI)" on page 72) is appended to the frame data in the Frame Buffer.

For frame transmission, the first byte of the Frame Buffer must contain the frame length information followed by the frame data. The TST_FRAME_LENGTH register does not need to be written in this case.

A detailed description of the Frame Buffer usage for receive and transmit frames can be found in Figure 9-31 on page 78.

Notes:

1. The Frame Buffer is shared between RX and TX; therefore, the frame data are overwritten by new incoming frames. If the TX frame data are to be retransmitted, it must be ensured that no frame was received in the meanwhile.
2. To avoid overwriting during receive, Dynamic Frame Buffer Protection can be enabled. For details about this feature refer to section "Dynamic Frame Buffer Protection" on page 91.
3. It is not possible to retransmit received frames without inserting the frame length information at the beginning of the Frame Buffer. That requires a complete read out of the received frame and rewriting the modified frame to the Frame Buffer.
4. For exceptions, e.g. receiving acknowledgement frames in Extended Operating Mode (TX_ARET) refer to section "TX_ARET_ON – Transmit with Automatic Retry and CSMA-CA Retry" on page 57.

9.3.1.3 Transceiver Pin Register TRXPR

The Transceiver Pin Register TRXPR is located in the Controller clock domain and is accessible even if the transceiver is in sleep state. This register provides access to the pin functionality, known from the RF231 devices (two chip solution).

The register (TRXRST) can be used to reset the transceiver without resetting the controller. After the reset bit was set, it is cleared immediately.

A second configuration bit (SLPTR) is used to control frame transmission or sleep and wakeup of the transceiver. This bit is not cleared automatically.

The function of the SLPTR bit relates to the current state of the transceiver module and is summarized in Table 9-1 below. The radio transceiver states are explained in detail in section "Operating Modes" on page 35.

Table 9-1. SLPTR Multi-functional Configuration bit

Transceiver Status	Function	SLPTR Bit	Description
PLL_ON	TX start	“0” ⇨ “1”	Starts frame transmission
TX_ARET_ON	TX start	“0” ⇨ “1”	Starts TX_ARET transaction
TRX_OFF	Sleep	“0” ⇨ “1”	Takes the radio transceiver into SLEEP state
SLEEP	Wakeup	“1” ⇨ “0”	Takes the radio transceiver back into TRX_OFF state;

In states PLL_ON and TX_ARET_ON, bit SLPTR is used to initiate a TX transaction. Here bit SLPTR is sensitive on the transition from “0” to “1” only. The bit should be cleared before the frame transmission is finished.

After initiating a state change by a “0” to “1” transition at bit SLPTR in radio transceiver states TRX_OFF, RX_ON or RX_AACK_ON, the radio transceiver remains in the new state as long as the bit is logical “1” and returns to the preceding state if the bit is set to “0”.

SLEEP state

The SLEEP state is used when radio transceiver functionality is not required, and thus the receiver module can be powered down to reduce the overall power consumption.

When the radio transceiver is in TRX_OFF state the microcontroller forces the transceiver to SLEEP by setting SLPTR = “1”. The transceiver awakes when the microcontroller releases bit SLPTR.

9.3.2 Interrupt Logic

9.3.2.1 Overview

The transceiver module differentiates between eight interrupt events. Internally all pending interrupt are stored in a separate bit of the interrupt status register (IRQ_STATUS). Each interrupt is enabled by setting the corresponding bit in the interrupt mask register (IRQ_MASK). If an IRQ is enabled an interrupt service routine must be defined to handle the IRQ. A pending IRQ is cleared automatically if an Interrupt service routine is called. It is also possible to handle IRQs manually by polling the IRQ_STATUS register. If an IRQ occurred, the appropriate IRQ_STATUS register bit is set. The IRQ can be cleared by writing ‘1’ to the register bit. It is recommended to clear the corresponding status bit before enabling an interrupt.

Interrupts are not cleared automatically when the event that caused them vanishes. More information about interrupt handling by the controller can be found in section "Interrupts" on page 211.

The supported interrupts for the Basic Operating Mode are summarized in Table 9-2 on **page 34**.

Table 9-2. Interrupt Description in Basic Operating Mode

IRQ Vector Number/ Priority [(1)]	IRQ Name	Description	Section
64	TRX24_AWAKE	Indicates radio transceiver reached TRX_OFF state RESET, or SLEEP states	"TRX_OFF – Clock State" on page 36
63	TRX24_TX_END	Indicates the completion of a frame transmission	"Frame Transmit Procedure" on page 84
62	TRX24_XAH_AMI	Indicates address matching	"Frame Filtering" on page 54
61	TRX24_CCA_ED_DONE	Indicates the end of a CCA or ED measurement	"Energy Detection (ED)" on page 68
60	TRX24_RX_END	Indicates the completion of a frame reception	"Frame Transmit Procedure" on page 84
59	TRX24_RX_START	Indicates the start of a PSDU reception. The TRX_STATE changes to BUSY_RX, the PHR is ready to be read from Frame Buffer	"Frame Receive Procedure" on page 84
58	TRX24_PLL_UNLOCK	Indicates PLL unlock. If the radio transceiver is in BUSY_TX / BUSY_TX_ARET state, the PA is turned off immediately	"Interrupt Handling" on page 83
57	TRX24_PLL_LOCK	Indicates PLL lock	"Interrupt Handling" on page 83

Note: 1. The lowest IRQ Number has the highest priority.

During startup from SLEEP or RESET, the radio transceiver issues an TRX24_AWAKE interrupt when it enters state TRX_OFF.

If the microcontroller initiates an energy-detect (ED) or clear-channel-assessment (CCA) measurement, the completion of the measurement is indicated by interrupt TRX24_CCA_ED_DONE, refer to sections "Energy Detection (ED)" on page 68 and "Clear Channel Assessment (CCA)" on page 70 for details.

After RESET all interrupts are disabled. During radio transceiver initialization it is recommended to enable AWAKE to be notified once the TRX_OFF state is entered. Note that the TRX24_AWAKE interrupt can usually not be seen when the transceiver enters TRX_OFF state after RESET, because register IRQ_MASK is reset to mask all interrupts. In this case, state TRX_OFF is normally entered before the microcontroller could modify the register.

The interrupt handling in Extended Operating Mode is described in section "Interrupt Handling" on page 59.

9.3.3 Radio Transceiver Identification

The ATmega128RFA1 Transceiver module can be identified by four registers (PART_NUM, VERSION_NUM, MAN_ID_0, MAN_ID_1). One register contains a unique part number and one register the corresponding version number. Two additional registers contain the JTAG manufacture ID. The transceiver identification registers are provided for compatibility to the transceiver only device.

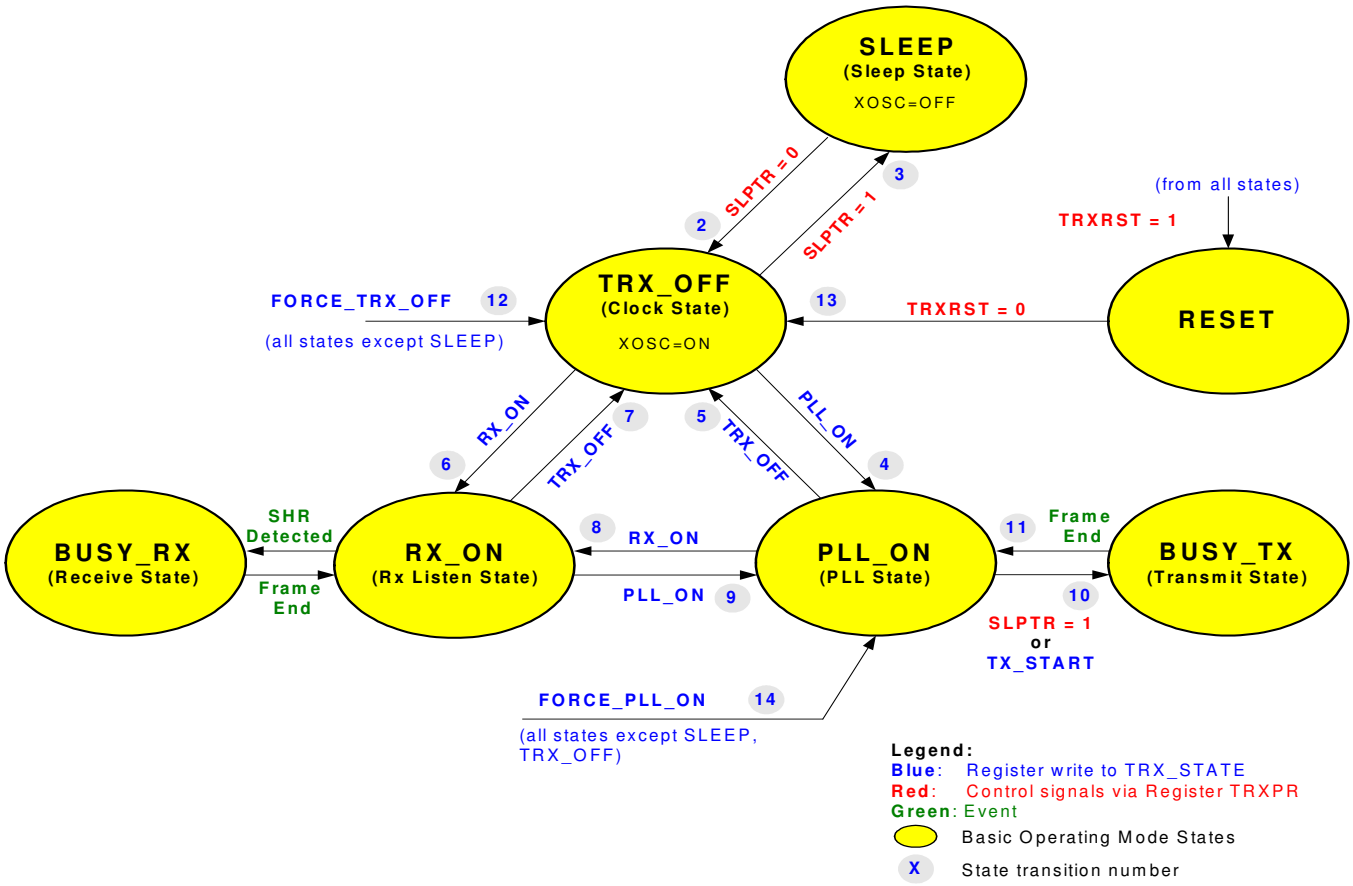
A unique device identification is also possible with the three AVR signature bytes. For details about accessing this information refer to "Signature Bytes" on page 466.

9.4 Operating Modes

9.4.1 Basic Operating Mode

This section summarizes all states to provide the basic functionality of the 2.4GHz radio transceiver, such as receiving and transmitting frames, the power up sequence and radio transceiver sleep. The Basic Operating Mode is designed for IEEE 802.15.4 and ISM applications; the corresponding radio transceiver states are shown in Figure 9-12 below.

Figure 9-12. Basic Operating Mode State Diagram (for timing refer to **Table 9-3 on page 42**)



Note: 1. State transition numbers correspond to Table 9-3 on page 42.

9.4.1.1 State Control

The radio transceiver states are controlled either by writing commands to bits TRX_CMD of register TRX_STATE, or directly by the two control bits SLPTR and TRXRST of the TRXPR register. A successful state change can be verified by reading the radio transceiver status from register TRX_STATUS.

If TRX_STATUS = 0x1F (STATE_TRANSITION_IN_PROGRESS) the radio transceiver is on a state transition. Do not try to initiate a further state change while the radio transceiver is in STATE_TRANSITION_IN_PROGRESS.

Bit SLPTR is a multifunctional bit (refer to section "Transceiver Pin Register TRXPR" on page 32 for more details). Dependent on the radio transceiver state, a “0” to “1” transition on SLPTR causes the following state transitions:

- TRX_OFF ➔ SLEEP
- PLL_ON ➔ BUSY_TX

Whereas resetting bit SLPTR to “0” causes the following state transitions:

- SLEEP ➔ TRX_OFF

Bit TRXRST causes a reset of all radio transceiver registers and forces the radio transceiver into TRX_OFF state.

For all states except SLEEP, the state change commands FORCE_TRX_OFF or TRX_OFF lead to a transition into TRX_OFF state. If the radio transceiver is in active receive or transmit states (BUSY_*), the command FORCE_TRX_OFF interrupts these active processes, and forces an immediate transition to TRX_OFF. In contrast a TRX_OFF command is stored until an active state (receiving or transmitting) has been finished. After that the transition to TRX_OFF is performed.

For a fast transition from receive or active transmit states to PLL_ON state the command FORCE_PLL_ON is provided. In contrast to FORCE_TRX_OFF this command does not disable the PLL and the analog voltage regulator AVREG. It is not available in states SLEEP, and RESET.

The completion of each requested state-change shall always be confirmed by reading the bits TRX_STATUS of register TRX_STATUS.

9.4.1.2 Basic Operating Mode Description

9.4.1.2.1 SLEEP – Sleep State

In radio transceiver SLEEP state, the entire radio transceiver is disabled. No circuitry is operating. The radio transceiver’s current consumption is reduced to leakage current only. This state can only be entered from state TRX_OFF, by setting the bit SLPTR = “1”.

Setting SLPTR = “0” returns the radio transceiver to the TRX_OFF state. During radio transceiver SLEEP the register contents remains valid while the content of the Frame Buffer and the security engine (AES) are cleared.

TRXRST = “1” in SLEEP state returns the radio transceiver to TRX_OFF state and thereby sets all registers to their reset values.

9.4.1.2.2 TRX_OFF – Clock State

This state is reached immediately after Power On or Reset. In TRX_OFF the crystal oscillator is running. The digital voltage regulator is enabled, thus the radio transceiver registers, the Frame Buffer and security engine (AES) are accessible (see section "Frame Buffer" on page 77 and "Security Module (AES)" on page 92).

SLPTR and TRXRST in register TRXPR can be used for state control (see "State Control" on page 35 for details). The analog front-end is disabled during TRX_OFF.

Entering the TRX_OFF state from radio transceiver SLEEP, or RESET state is indicated by the TRX24_AWAKE interrupt.

9.4.1.2.3 PLL_ON – PLL State

Entering the PLL_ON state from TRX_OFF state first enables the analog voltage regulator (AVREG). After the voltage regulator has been settled the PLL frequency synthesizer is enabled. When the PLL has been settled at the receive frequency to a channel defined by bits CHANNEL of register PHY_CC_CCA a successful PLL lock is indicated by issuing a TRX24_PLL_LOCK interrupt.

If an RX_ON command is issued in PLL_ON state, the receiver is immediately enabled. If the PLL has not been settled before the state change nevertheless takes place. Even if the register bits TRX_STATUS of register TRX_STATUS indicates RX_ON, actual frame reception can only start once the PLL has locked.

The PLL_ON state corresponds to the TX_ON state in IEEE 802.15.4.

9.4.1.2.4 RX_ON and BUSY_RX – RX Listen and Receive State

In RX_ON state the receiver blocks and the PLL frequency synthesizer are enabled.

The receive mode is internally separated into the RX_ON and BUSY_RX states. There is no difference between these states with respect to the analog radio transceiver circuitry, which are always turned on. In both states the receiver and the PLL frequency synthesizer are enabled.

During RX_ON state the receiver listens for incoming frames. After detecting a valid synchronization header (SHR), the receiver automatically enters the BUSY_RX state. The reception of a valid PHY header (PHR) generates an TRX24_RX_START interrupt and receives and demodulates the PSDU data.

During PSDU reception the frame data are stored continuously in the Frame Buffer until the last byte was received. The completion of the frame reception is indicated by an TRX24_RX_END interrupt and the radio transceiver reenters the state RX_ON. At the same time the bits RX_CRC_VALID of register PHY_RSSI are updated with the result of the FCS check (see "Frame Check Sequence (FCS)" on page 66).

Received frames are passed to the frame filtering unit, refer to section "Frame Filtering" on page 54. If the content of the MAC addressing fields of a frame (refer to IEEE 802.15.4 section 7.2.1) matches to the expected addresses, which is further dependent on the addressing mode, an address match interrupt (TRX24_XAH_AMI) is issued, refer to "Interrupt Logic" on page 33. The expected address values are to be stored in the registers Short-Address, PAN-ID and IEEE-address. Frame filtering is available in Basic and Extended Operating Mode, refer to section "Frame Filtering" on page 54.

Leaving state RX_ON is only possible by writing a state change command to bits TRX_CMD of register TRX_STATE.

9.4.1.2.5 BUSY_TX – Transmit State

A transmission can only be initiated in state PLL_ON. There are two ways to start a transmission:

- Setting Bit SLPTR of register TRXPR to ‘1’. The bit should be cleared before the frame has been transmitted. This mode is for legacy operation and should be replaced by the TX_START command below.
- TX_START command to bits TRX_CMD of register TRX_STATE.

Either of these causes the radio transceiver into the BUSY_TX state.

During the transition to BUSY_TX state, the PLL frequency shifts to the transmit frequency. The actual transmission of the first data chip of the SHR starts after 16 µs to allow PLL settling and PA ramp-up, see Figure 9-16 on page 40. After transmission of the SHR, the Frame Buffer content is transmitted. In case the PHR indicates a frame length of zero, the transmission is aborted.

After the frame transmission has completed, the radio transceiver automatically turns off the power amplifier, generates a TRX24_TX_END interrupt and returns into PLL_ON state.

9.4.1.2.6 RESET State

The RESET state is used to set back the state machine and to reset all registers of the radio transceiver to their default values.

A reset forces the radio transceiver into the TRX_OFF state.

A reset is initiated by a ATmega128RFA1 main reset (see "Resetting the AVR" on page 176) or a radio transceiver reset (see "Transceiver Pin Register TRXPR" on page 32).

During radio transceiver reset the TRXPR register is not cleared and therefore the application software has to set the SLPTR bit to “0”.

9.4.1.3 Interrupt Handling

All interrupts provided by the radio transceiver are supported in Basic Operating Mode (see Table 9-2 on page 34).

Required interrupts must be enabled by writing to register IRQ_MASK and the global interrupt enable flag must be set. For a general explanation of the interrupt handling refer to "Reset and Interrupt Handling" on page 15 and "Interrupt Logic" on page 33.

For example, interrupts are provided to observe the status of the RX and TX operations.

On receive the TRX24_RX_START interrupt indicates the detection of a valid PHR, the TRX24_XAH_AMI interrupt an address match and the TRX24_RX_END interrupt the completion of the frame reception.

On transmit the TRX24_TX_END interrupt indicates the completion of the frame transmission.

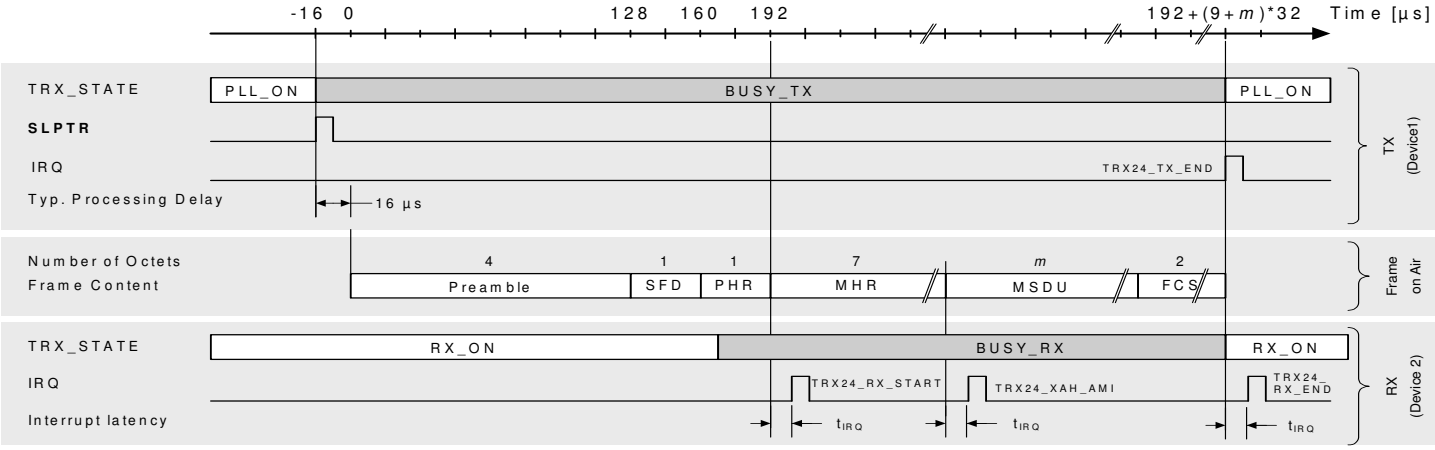
Figure 9-13 on page 39 shows an example for a transmit/receive transaction between two devices and the related interrupt events in Basic Operating Mode. Device 1 transmits a frame containing a MAC header (in this example of length 7), payload and valid FCS. The frame is received by Device 2 which generates the interrupts during the processing of the incoming frame. The received frame is stored in the Frame Buffer.

If the received frame passes the address filter (refer to section "Frame Filtering" on page 54) an address match TRX24_XAH_AMI interrupt is issued after the reception of the MAC header (MHR).

In Basic Operating Mode the TRX24_RX_END interrupt is issued at the end of the received frame. In Extended Operating Mode (refer to "Extended Operating Mode" on page 43) the interrupt is only issued if the received frame passes the address filter and the FCS is valid. Further exceptions are explained in "Extended Operating Mode" on page 43.

Processing delay t_{IRQ} is a typical value (see chapter "Digital Interface Timing Characteristics" on page 507).

Figure 9-13. Timing of TRX24_RX_START, TRX24_XAH_AMI, TRX24_TX_END and TRX24_RX_END Interrupts in Basic Operating Mode



9.4.1.4 Basic Operating Mode Timing

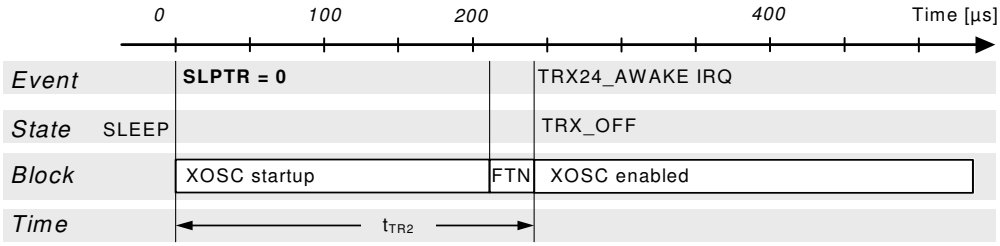
The following paragraphs depict state transitions and their timing properties. Timing figures are explained in Table 9-3 on page 42 and section "Digital Interface Timing Characteristics" on page 507.

9.4.1.4.1 Wake-up Procedure

The wake-up procedure from radio transceiver SLEEP state is shown in Figure 9-14 below. This figure implies, that the microcontroller is already running and hence, the digital voltage regulator is enabled. If the microcontroller clock source is set to Transceiver Clock, the crystal oscillator is also running, which reduces the radio transceiver wake-up time further. For information about the wake-up timing of the microcontroller, depending on the different clock source options, refer to "System Clock and Clock Options" on page 147.

In order to calculate the total wake-up delay from microcontroller sleep mode (see "Power Management and Sleep Modes" on page 156), the microcontroller wake-up time, including the voltage regulator ramp-up and the radio transceiver wake-up time has to be added.

Figure 9-14. Wake-up Procedure from Transceiver SLEEP State



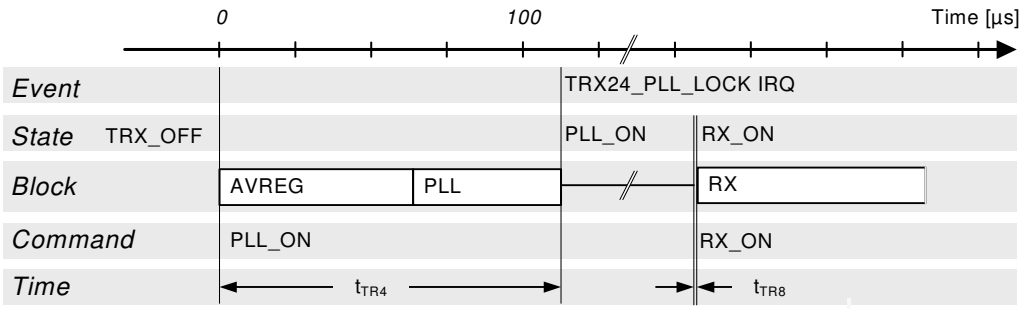
The radio transceiver SLEEP state is left by releasing bit SLPTR to “0”. This restarts the XOSC if it is not already running. After t_{TR2} = 215 µs + 25 µs = 240 µs (see Table 9-3 on page 42) the radio transceiver enters TRX_OFF state. If the XOSC is already running, the radio transceiver enters TRX_OFF state after 25 µs.

During this wake-up procedure the calibration of the filter-tuning network (FTN) is performed. Entering TRX_OFF state is signaled by the TRX24_AWAKE interrupt, if enabled.

9.4.1.4.2 PLL_ON and RX_ON States

The transition from TRX_OFF to PLL_ON and RX_ON mode is shown in Figure 9-15 below.

Figure 9-15. Transition from TRX_OFF to PLL_ON and RX_ON State



Note: 1. If TRX_CMD = RX_ON in TRX_OFF state RX_ON state is entered immediately, even if the PLL has not settled.

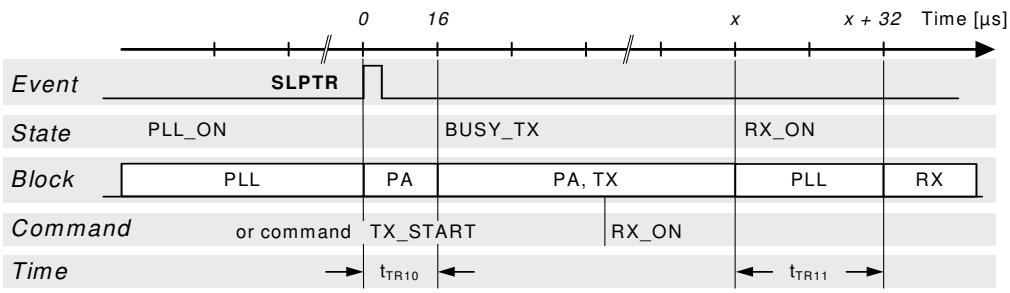
2. If the AVR ADC module is enabled, the AVREG is already started and thus the state transition time t_{TR4} is reduced.

Entering the commands PLL_ON or RX_ON in TRX_OFF state initiates a ramp-up sequence of the internal 1.8V voltage regulator for the analog domain (AVREG), if AVREG is not already enabled by the AVR ADC module. RX_ON state can be entered any time from PLL_ON state regardless whether the PLL has already locked as indicated by the TRX24_PLL_LOCK interrupt.

9.4.1.4.3 BUSY_TX and RX_ON States

The transition from PLL_ON to BUSY_TX state and subsequent to RX_ON state is shown in Figure 9-16 below.

Figure 9-16. PLL_ON to BUSY_TX to RX_ON Timing



Starting from PLL_ON state it is assumed that the PLL is already locked. A transmission is initiated either by writing “1” to bit SLPTR or by command TX_START. The PLL settles to the transmit frequency and the PA is enabled.

t_{TR10} = 16 µs after initiating the transmission, the radio transceiver changes into BUSY_TX state and the internally generated SHR is transmitted. After that the PSDU data are transmitted from the Frame Buffer.

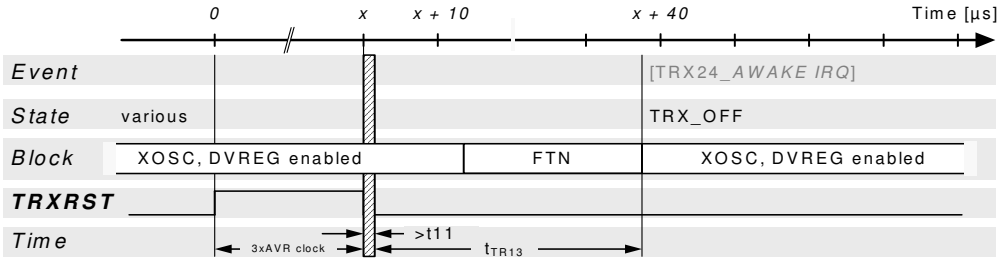
After completing the frame transmission, indicated by the TRX24_TX_END interrupt, the PLL settles back to the receive frequency within t_{TR11} = 32 µs in state PLL_ON.

If during TX_BUSY the radio transmitter is programmed to change to a receive state it automatically proceeds the state change to RX_ON state after finishing the transmission.

9.4.1.4.4 Reset Procedure

The radio transceiver reset procedure is shown in Figure 9-17 below.

Figure 9-17. Reset Procedure



Note: 1. Timing parameter t_{TR13} = 37 µs refers to Table 9-3 on page 42; t_{11} refers to "Digital Interface Timing Characteristics" on page 507.

2. If TRXRST is set during radio transceiver SLEEP state, the XOSC startup delay is extended by the XOSC startup time.

TRXRST = “1” resets all radio transceiver registers to their default values.

The radio transceiver reset is released automatically after 3 AVR clock cycles and the wake-up sequence without restarting XOSC and DVREG, nevertheless an FTN calibration cycle is performed, refer to "Automatic Filter Tuning (FTN)" on page 83. After that the TRX_OFF state is entered.

Figure 9-17 above illustrates the radio transceiver reset procedure if the radio transceiver is in any state but not in SLEEP state.

If the radio transceiver was in SLEEP state, the SLPTR bit in the TRXPR register must be cleared prior to clearing the TRXRST bit in order to enter the TRX_OFF state. Otherwise the radio transceiver enters the SLEEP state immediately.

If the radio transceiver was in SLEEP state and the Transceiver Clock is not selected as the microcontroller clock source, the XOSC is enabled before entering TRX_OFF state.

If register TRX_STATUS indicates STATE_TRANSITION_IN_PROGRESS during system initialization until the radio transceiver reaches TRX_OFF, do not try to initiate a further state change while the radio transceiver is in this state.

Note that before accessing the radio transceiver module the TRX24_AWAKE event should be checked.

9.4.1.4.5 State Transition Timing Summary

The transition numbers correspond to Table 9-3 below. See measurement setup in "Basic Application Schematic" on page 493.

Table 9-3. Radio Transceiver State Transition Timing

No	Symbol	Transition			Time [µs], (typ)	Comments
1	t_{TR2}	SLEEP	⇨	TRX_OFF	240	Depends on crystal oscillator setup (CL = 10 pf) TRX_OFF state indicated by TRX24_AWAKE interrupt
2	t_{TR3}	TRX_OFF	⇨	SLEEP	$35 \cdot 1 / f_{CLKM}$	For f_{CLKM} > 250 kHz
3	t_{TR4}	TRX_OFF	⇨	PLL_ON	110	Depends on external capacitor at AVDD (1 µF nom)
4	t_{TR5}	PLL_ON	⇨	TRX_OFF	1	
5	t_{TR6}	TRX_OFF	⇨	RX_ON	110	Depends on external capacitor at AVDD (1 µF nom)
6	t_{TR7}	RX_ON	⇨	TRX_OFF	1	
7	t_{TR8}	PLL_ON	⇨	RX_ON	1	
8	t_{TR9}	RX_ON	⇨	PLL_ON	1	Transition time is also valid for TX_ARET_ON, RX_AACK_ON
9	t_{TR10}	PLL_ON	⇨	BUSY_TX	16	When setting bit SLPTR or TRX_CMD = TX_START, the first symbol transmission is delayed by 16 µs (PLL settling and PA ramp up).
10	t_{TR11}	BUSY_TX	⇨	PLL_ON	32	PLL settling time from TX_BUSY to PLL_ON state
11	t_{TR12}	All modes	⇨	TRX_OFF	1	Using TRX_CMD = FORCE_TRX_OFF (see register TRX_STATE), Not valid for SLEEP state
12	t_{TR13}	RESET	⇨	TRX_OFF	37	Not valid for SLEEP state
13	t_{TR14}	Various states	⇨	PLL_ON	1	Using TRX_CMD = FORCE_PLL_ON (see register TRX_STATE), Not valid for SLEEP, RESET and TRX_OFF

The state transition timing is calculated based on the timing of the individual blocks shown in Table 9-8 on page 51. The worst case values include maximum operating temperature, minimum supply voltage, and device parameter variations.

Table 9-8. Analog Block Initialization and Settling Time

No	Symbol	Block	Time [µs], (typ)	Time [µs], (max)	Comments
15	t_{TR15}	XOSC	215	1000	Leaving SLEEP state, depends on crystal Q factor and load capacitor
16	t_{TR16}	FTN		25	FTN tuning time, fixed
17	t_{TR17}	DVREG	60	1000	Depends on external bypass capacitor at DVDD (CB3 = 1 µF nom., 10 µF worst case), depends on V_{DEVDD}
18	t_{TR18}	AVREG	60	1000	Depends on external bypass capacitor at AVDD (CB1 = 1 µF nom., 10 µF worst case) , depends on V_{EVDD}
19	t_{TR19}	PLL, initial	110	155	PLL settling time TRX_OFF ⇨ PLL_ON, including 60 µs AVREG settling time
20	t_{TR20}	PLL, settling	11	24	Settling time between channel switch
21	t_{TR21}	PLL, CF cal	35		PLL center frequency calibration, refer to "Calibration Loops" on page 83
22	t_{TR22}	PLL, DCU cal	6		PLL DCU calibration, refer to "Calibration Loops" on page 83
23	t_{TR23}	PLL, RX ⇨ TX	16		Maximum PLL settling time RX ⇨ TX

No	Symbol	Block	Time [µs], (typ)	Time [µs], (max)	Comments
24	t_{TR24}	PLL, TX ⇨ RX	32		Maximum PLL settling time TX ⇨ RX
25	t_{TR25}	RSSI, update	2		RSSI update period in receive states, refer to "Reading RSSI" on page 68
26	t_{TR26}	ED	140		ED measurement period, refer to "Measurement Description" on page 69
27	t_{TR27}	SHR, sync	96		Typical SHR synchronization period, refer to "Measurement Description" on page 69
28	t_{TR28}	CCA	140		CCA measurement period, refer to "Configuration and CCA Request" on page 71
29	t_{TR29}	Random value	1		Random value update period, refer to "Random Number Generator" on page 85

9.4.2 Extended Operating Mode

The Extended Operating Mode is a hardware MAC accelerator and goes beyond the basic radio transceiver functionality provided by the Basic Operating Mode. It handles time critical MAC tasks requested by the IEEE 802.15.4 standard or by hardware such as automatic acknowledgement, automatic CSMA-CA and retransmission. This results in a more efficient IEEE 802.15.4 software MAC implementation including reduced code size and may allow operating at lower microcontroller clock rates.

The Extended Operating Mode is designed to support IEEE 802.15.4-2006 compliant frames; the mode is backward compatible to IEEE 802.15.4-2003 and supports non IEEE 802.15.4 compliant frames. This mode comprises the following procedures:

Automatic acknowledgement (RX_AACK) divides into the tasks:

- Frame reception and automatic FCS check;
- Configurable addressing fields check;
- Interrupt indicating address match;
- Interrupt indicating frame reception, if it passes address filtering and FCS check;
- Automatic ACK frame transmission (if the received frame passed the address filter and FCS check and if an ACK is required by the frame type and ACK request);
- Support of slotted acknowledgment using SLPTR bit for frame start.

Automatic CSMA-CA and Retransmission (TX_ARET) divides into the tasks:

- CSMA-CA including automatic CCA retry and random back-off;
- Frame transmission and automatic FCS field generation;
- Reception of ACK frame (if an ACK was requested);
- Automatic frame retry if ACK was expected but not received;
- Interrupt signaling with transaction status.

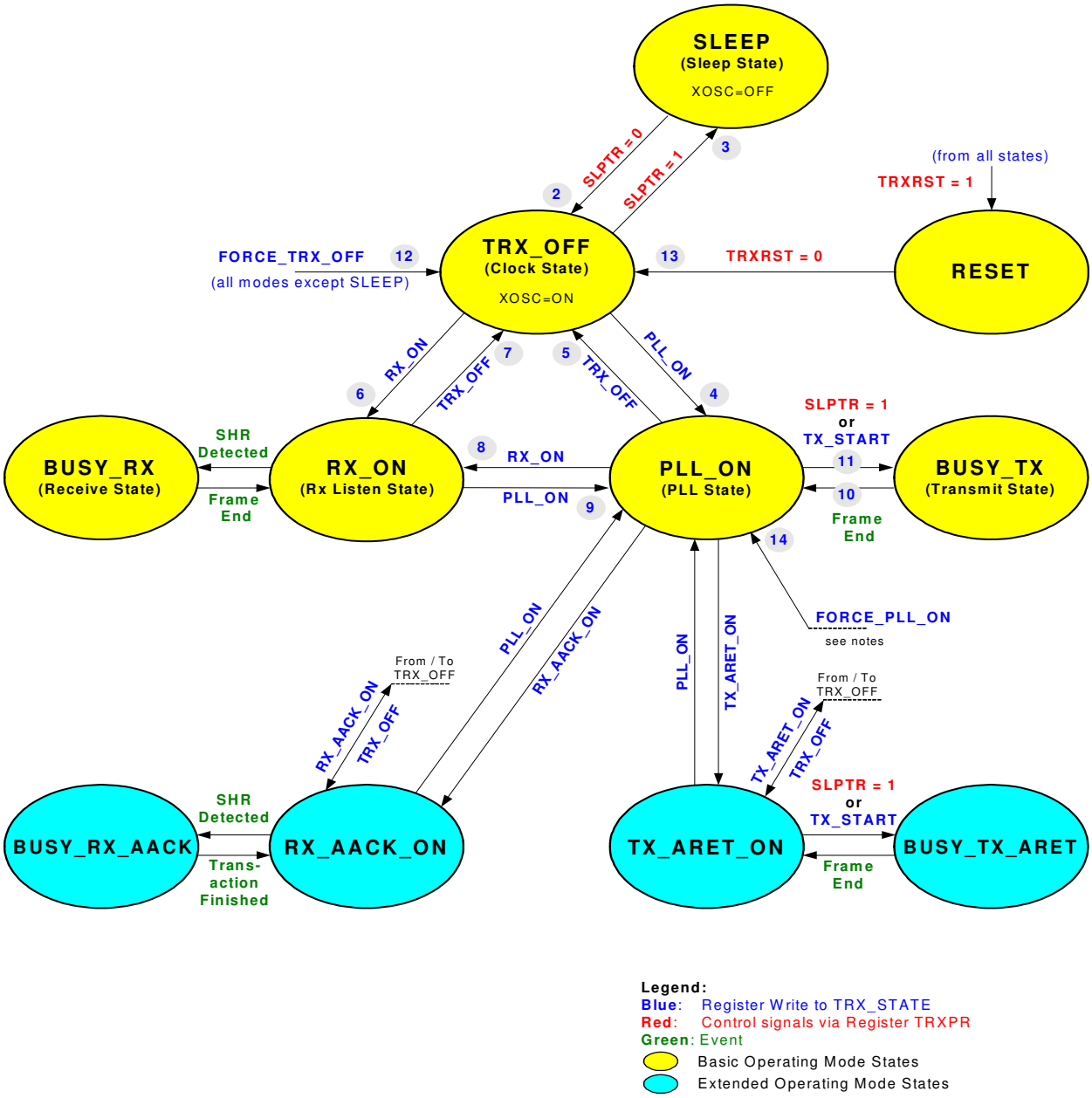
Automatic FCS check and generation (refer to "Frame Check Sequence (FCS)" on page 66) is used by the RX_AACK and TX_ARET modes. In RX_AACK mode an automatic FCS check is always performed for incoming frames.

An ACK received in TX_ARET mode within the time required by IEEE 802.15.4 is accepted if the FCS is valid and if the sequence number of the ACK matches the sequence number of the previously transmitted frame. Dependent on the value of the

frame pending subfield in the received acknowledgement frame the transaction status is set according to Table 9-16 on page 58.

The state diagram including the Extended Operating Mode states is shown in Figure 9-18 below. Yellow marked states represent the Basic Operating Mode; blue marked states represent the Extended Operating Mode.

Figure 9-18. Extended Operating Mode State Diagram



Note: 1. State transition numbers correspond to Table 9-3 on page 42.

9.4.2.1 State Control

The Extended Operating Mode states RX_AACK and TX_ARET are controlled via the bits TRX_CMD of register TRX_STATE, which receives the state transition commands. The states are entered from TRX_OFF or PLL_ON state as illustrated in Figure 9-18 on page 44. The completion of each state change command shall always be confirmed by reading the TRX_STATUS register.

RX_AACK - Receive with Automatic ACK

A state transition to RX_AACK_ON from PLL_ON or TRX_OFF is initiated by writing the command RX_AACK_ON to the register bits TRX_CMD. The state change can be confirmed by reading register TRX_STATUS, those changes to RX_AACK_ON or BUSY_RX_AACK on success. BUSY_RX_AACK is returned if a frame is currently being received.

The RX_AACK state is left by writing command TRX_OFF or PLL_ON to the register bits TRX_CMD. If the radio transceiver is within a frame receive or acknowledgment procedure (BUSY_RX_AACK) the state change is executed after finish. Alternatively, the commands FORCE_TRX_OFF or FORCE_PLL_ON can be used to cancel the RX_AACK transaction and change into radio transceiver state TRX_OFF or PLL_ON respectively.

TX_ARET - Transmit with Automatic Retry and CSMA-CA Retry

Similarly, a state transition to TX_ARET_ON from PLL_ON or TRX_OFF is initiated by writing command TX_ARET_ON to register bits TRX_CMD. The radio transceiver is in the TX_ARET_ON state after TRX_STATUS register changes to TX_ARET_ON. The TX_ARET transaction is started with writing ‘1’ to the SLPTR bit of the TRXPR register or writing the command TX_START to register bits TRX_CMD.

TX_ARET state is left by writing the command TRX_OFF or PLL_ON to the register bits TRX_CMD. If the radio transceiver is within a CSMA-CA, a frame-transmit or an acknowledgment procedure (BUSY_TX_ARET) the state change is executed after finish. Alternatively, the command FORCE_TRX_OFF or FORCE_PLL_ON can be used to instantly terminate the TX_ARET transaction and change into radio transceiver states TRX_OFF or PLL_ON, respectively.

Note that a state change request from TRX_OFF to RX_AACK_ON or TX_ARET_ON internally passes the state PLL_ON to initiate the radio transceiver. Thus the readiness to receive or transmit data is delayed accordingly. It is recommended to use interrupt TRX24_PLL_LOCK as an indicator.

9.4.2.2 Configuration

The use of the Extended Operating Mode is based on Basic Operating Mode functionality. Only features beyond the basic radio transceiver functionality are described in the following sections. For details on the Basic Operating Mode refer to section "Basic Operating Mode" on page 35.

When using the RX_AACK or TX_ARET modes, the following registers needs to be configured.

RX_AACK configuration steps:

- Short address, PAN-ID and IEEE address (register SHORT_AADR_0, SHORT_ADDR_1, PAN_ID_0, PAN_ID_1, IEEE_ADDR_0 … IEEE_ADDR_7)
- Configure RX_AACK properties (register XAH_CTRL_0, CSMA_SEED_1)
    - Handling of Frame Version Subfield

- Handling of Pending Data Indicator
- Characterize as PAN coordinator
- Handling of Slotted Acknowledgement

- Additional Frame Filtering Properties (register XAH_CTRL_1, CSMA_SEED_1)
  - Promiscuous Mode
  - Enable or disable automatic ACK generation
  - Handling of reserved frame types

The addresses for the address match algorithm are to be stored in the appropriate address registers. Additional control of the RX_AACK mode is done with registers XAH_CTRL_1 and CSMA_SEED_1.

As long as a short address has not been set, only broadcast frames and frames matching the IEEE address can be received.

Configuration examples for different device operating modes and handling of various frame types can be found in section "Description of RX_AACK Configuration Bits" on page 49.

TX_ARET configuration steps:

- Leave register bit TX_AUTO_CRC_ON = 1 register TRX_CTRL_1
- Configure CSMA-CA
  - MAX_FRAME_RETRIES register XAH_CTRL_0
  - MAX_CSMA_RETRIES register XAH_CTRL_0
  - CSMA_SEED registersCSMA_SEED_0, CSMA_SEED_1
  - MAX_BE, MIN_BE register CSMA_BE
- Configure CCA (see section "Configuration and CCA Request" on page 71)

MAX_FRAME_RETRIES (register XAH_CTRL_0) defines the maximum number of frame retransmissions.

The register bits MAX_CSMA_RETRIES (register XAH_CTRL_0) configure the number of CSMA-CA retries after a busy channel is detected.

The CSMA_SEED_0 and CSMA_SEED_1 registers define a random seed for the back-off-time random-number generator of the radio transceiver.

The MAX_BE and MIN_BE register bits (register CSMA_BE) set the maximum and minimum CSMA back-off exponent (according to [1] on page 100).

9.4.2.3 RX_AACK_ON – Receive with Automatic ACK

The general functionality of the RX_AACK procedure is shown in Figure 9-19 on page 48.

The gray shaded area is the standard flow of a RX_AACK transaction for IEEE 802.15.4 compliant frames (refer to section "Configuration of IEEE Scenarios" on page 50). All other procedures are exceptions for specific operating modes or frame formats (refer to section "Configuration of non IEEE 802.15.4 Compliant Scenarios" on page 52).

The frame filtering operation is described in detail in section "Frame Filtering" on page 54.

In RX_AACK_ON state, the radio transceiver listens for incoming frames. After detecting SHR and a valid PHR, the radio transceiver parses the frame content of the MAC header (MHR) as described in section "PHY Header (PHR)" on page 61.

Generally, at nodes, configured as a normal device or PAN coordinator, a frame is not indicated if the frame filter does not match and the FCS is invalid. Otherwise, the TRX_24_RX_END interrupt is issued after the completion of the frame reception. The microcontroller can then read the frame. An exception applies if promiscuous mode is enabled (see section "Configuration of IEEE Scenarios" on page 50). In that case a TRX_24_RX_END interrupt is issued even if the FCS fails.

If the content of the MAC addressing fields of the received frame (refer to IEEE 802.15.4 section 7.2.1) matches one of the configured addresses, dependent on the addressing mode, an address match interrupt (TRX24_XAH_AMI) is issued (refer to section "Frame Filtering" on page 54). The expected address values are to be stored in registers Short-address, PAN-ID and IEEE-address. Frame filtering as described in section "Frame Filtering" on page 54 is also valid for Basic Operating Mode.

During reception the radio transceiver parses bit[5] (ACK Request) of the frame control field of the received data or the MAC command frame to check if an ACK reply is expected. In that case and if the frame passes the third level of filtering (see IEEE 802.15.4-2006, section 7.5.6.2), the radio transceiver automatically generates and transmits an ACK frame. After the ACK transmission is finished, a TRX24_TX_END interrupt is generated.

The content of the frame pending subfield of the ACK response is set by bit AACK_SET_PD of register CSMA_SEED_1 when the ACK frame is sent in response to a data request MAC command frame, otherwise this subfield is set to “0”. The sequence number is copied from the received frame.

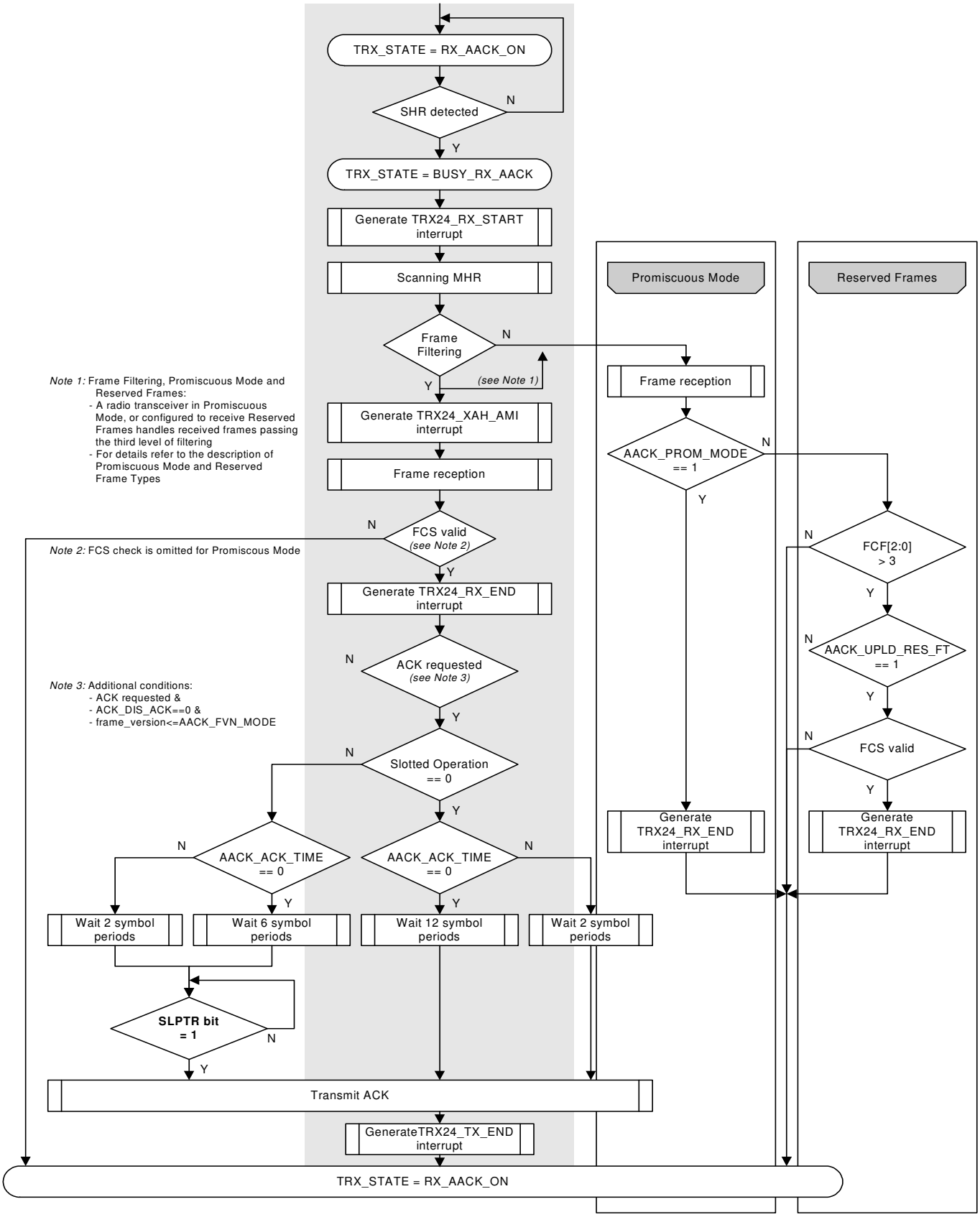
Optionally, the start of the transmission of the acknowledgement frame can be influenced by register bit AACK_ACK_TIME. Default value (according to standard IEEE 802.15.4, page 54) is 12 symbol times after the reception of the last symbol of a data or MAC command frame.

If the bit AACK_DIS_ACK of register CSMA_SEED_1 is set, no acknowledgement frame is sent even if an acknowledgment frame was requested. This is useful for operating the MAC hardware accelerator in promiscuous mode (see section "Configuration of non IEEE 802.15.4 Compliant Scenarios" on page 52).

The status of the RX_AACK operation is indicated by the bits TRAC_STATUS of register TRAC_STATUS.

During the operations described above the radio transceiver remains in BUSY_RX_AACK state.

Figure 9-19. Flow Diagram of RX_AACK



9.4.2.3.1 Description of RX_AACK Configuration Bits

Overview

The following table summarizes all register bits which affect the behavior of a RX_AACK transaction. For address filtering it is further required to setup address registers to match to the expected address.

Configuration and address bits are to be set in TRX_OFF or PLL_ON state prior to switching to RX_AACK mode.

A graphical representation of various operating modes is illustrated in Figure 9-19 on page 48.

Table 9-5. Overview of RX_AACK Configuration Bits

Register Name	Register Bits	Description
SHORT_ADDR_0/1 PAN_ADDR_0/1 IEEE_ADDR_0 … IEEE_ADDR_7		Set node addresses
RX_SAFE_MODE	7	Protect buffer after frame receive
AACK_PROM_MODE	1	Support promiscuous mode
AACK_ACK_TIME	2	Change auto acknowledge start time
AACK_UPLD_RES_FT	4	Enable reserved frame type reception, needed to receive non-standard compliant frames
AACK_FLTR_RES_FT	5	Filter reserved frame types like data frame type, needed for filtering of non-standard compliant frames
SLOTTED_OPERATION	0	If set, acknowledgment transmission has to be triggered by register bit SLPTR
AACK_I_AM_COORD	3	If set, the device is a PAN coordinator
AACK_DIS_ACK	4	Disable generation of acknowledgment
AACK_SET_PD	5	Set frame pending subfield in Frame Control Field (FCF), refer to section "Overview" on page 66
AACK_FVN_MODE	7:6	Controls the ACK behavior, depending on FCF frame version number

The usage of the RX_AACK configuration bits for various operating modes of a node is explained in the following sections. Configuration bits not mentioned in the following two sections should be set to their reset values.

All registers mentioned in Table 9-5 above are described in section "Register Summary" on page 60.

Note, that the general behavior of the Extended Feature Set settings:

- OQPSK_DATA_RATE (PSDU data rate)
- SFD_VALUE (alternative SFD value)
- ANT_DIV (Antenna Diversity)
- RX_PDT_LEVEL (blocking frame reception of lower power signals)

are completely independent from RX_AACK mode (see "Radio Transceiver Extended Feature Set" on page 85). Each of these operating modes can be combined with the RX_AACK mode.

9.4.2.3.2 Configuration of IEEE Scenarios

Normal Device

The Table 9-6 below shows a typical RX_AACK configuration of an IEEE 802.15.4 device operated as a normal device rather than a PAN coordinator or router.

Table 9-6. Configuration of IEEE 802.15.4 Devices

Register Name	Register Bits	Description
SHORT_ADDR_0/1 PAN_ADDR_0/1 IEEE_ADDR_0 … IEEE_ADDR_7		Set node addresses
RX_SAFE_MODE	7	*0*: disable frame protection *1*: enable frame protection
SLOTTED_OPERATION	0	*0*: if transceiver works in unslotted mode *1*: if transceiver works in slotted mode
AACK_FVN_MODE	7:6	Controls the ACK behavior, depending on FCF frame version number *0x00* : acknowledges only frames with version number 0, i.e. according to IEEE 802.15.4-2003 frames *0x01* : acknowledges only frames with version number 0 or 1, i.e. frames according to IEEE 802.15.4-2006 *0x10* : acknowledges only frames with version number 0 or 1 or 2 *0x11* : acknowledges all frames, independent of the FCF frame version number

Notes:
1. If no short address has been configured before the device has been assigned one by the PAN-coordinator, only frames directed to either the broadcast address or the IEEE address are received.
2. In IEEE 802.15.4-2003 standard the frame version subfield did not yet exist but was marked as reserved. According to this standard, reserved fields have to be set to zero. On the other hand, IEEE 802.15.4-2003 standard requires ignoring reserved bits upon reception. Thus, there is a contradiction in the standard which can be interpreted in two ways:
   a) If a network should only allow access to nodes which use the IEEE 802.15.4-2003, then AACK_FVN_MODE should be set to 0.
   b) If a device should acknowledge all frames independent of its frame version, AACK_FVN_MODE should be set to 3. However, this can result in conflicts with co-existing IEEE 802.15.4-2006 standard compliant networks.

   The same holds for PAN coordinators as described below.

PAN-Coordinator

Table 9-7 on page 51 shows the RX_AACK configuration for a PAN coordinator.

Table 9-7. Configuration of a PAN Coordinator

Register Name	Register Bits	Description
SHORT_ADDR_0/1 PAN_ADDR_0/1 IEEE_ADDR_0 … IEEE_ADDR_7		Set node addresses
RX_SAFE_MODE	7	*0*: disable frame protection *1*: enable frame protection
SLOTTED_OPERATION	0	*0*: if transceiver works in unslotted mode *1*: if transceiver works in slotted mode
AACK_I_AM_COORD	3	*1*: device is PAN coordinator
AACK_SET_PD	5	*0*: frame pending subfield is not set in FCF *1*: frame pending subfield is set in FCF
AACK_FVN_MODE	7:6	Controls the ACK behavior, depends on FCF frame version number *0x00* : acknowledges only frames with version number 0, i.e. according to IEEE 802.15.4-2003 frames *0x01* : acknowledges only frames with version number 0 or 1, i.e. frames according to IEEE 802.15.4-2006 *0x10* : acknowledges only frames with version number 0 or 1 or 2 *0x11* : acknowledges all frames, independent of the FCF frame version number

Promiscuous Mode

The promiscuous mode is described in IEEE 802.15.4-2006, section 7.5.6.5. This mode is further illustrated in Radio Transceiver Extended Feature Set on page 85. According to IEEE 802.15.4-2006 when in promiscuous mode, the MAC sub layer shall pass received frames with correct FCS to the next higher layer without further processing. That implies that frames should never be acknowledged.

Only second level filter rules as defined by IEEE 802.15.4-2006, section 7.5.6.2, are applied to the received frame.

Table 9-8 below shows the typical configuration of a device operating in promiscuous mode.

Table 9-8. Configuration of Promiscuous Mode

Register Name	Register Bits	Description
SHORT_ADDR_0/1 PAN_ADDR_0/1 IEEE_ADDR_0 … IEEE_ADDR_7		Each address shall be set: 0x00
AACK_PROM_MODE	1	*1*: Enable promiscuous mode
AACK_DIS_ACK	4	*1*: Disable generation of acknowledgment

Register Name	Register Bits	Description
AACK_FVN_MODE	7:6	Controls the ACK behavior, depends on FCF frame version number *0x00* : acknowledges only frames with version number 0, i.e. according to IEEE 802.15.4-2003 frames *0x01* : acknowledges only frames with version number 0 or 1, i.e. frames according to IEEE 802.15.4-2006 *0x10* : acknowledges only frames with version number 0 or 1 or 2 *0x11* : acknowledges all frames, independent of the FCF frame version number

Second level of filtering according to IEEE 802.15.4-2006, section 7.5.6.2, is applied to a received frame if the radio transceiver is in promiscuous mode. However, a TRX24_RX_END interrupt is issued even if the FCS is invalid. Thus it is necessary to read bit RX_CRC_VALID of register PHY_RSSI after the TRX24_RX_END interrupt in order to verify the reception of a frame with a valid FCS.

If a device, operating in promiscuous mode, receives a frame with a valid FCS which in addition passed the third level filtering according to IEEE 802.15.4-2006, section 7.5.6.2, an acknowledgement frame would be transmitted. According to the definition of the promiscuous mode a received frame shall not be acknowledged even if it is requested. Thus bit AACK_DIS_ACK of register CSMA_SEED_1 has to be set to 1.

In all receive modes a TRX24_AMI interrupt is issued, when the received frame matches the node's address according to the filter rules described in section "Frame Filtering" on page 54.

Alternatively, in RX_ON state of the Basic Operating Mode when a valid PHR is detected a TRX24_RX_START interrupt is generated and the frame is received. The end of the frame reception is signalized with a TRX24_RX_END interrupt. At the same time the bit RX_CRC_VALID of register PHY_RSSI is updated with the result of the FCS check (see "Overview" on page 66). The RX_CRC_VALID bit must be checked in order to dismiss corrupted frames according to the definition of the promiscuous mode.

9.4.2.3.3 Configuration of non IEEE 802.15.4 Compliant Scenarios

Sniffer

Table 9-9 below shows a RX_AACK configuration to setup a sniffer device. Other RX_AACK configuration bits should be set to their reset values (see Table 9-5 on page 49). All frames received are indicated by a TRX24_RX_START and TRX24_RX_END interrupt. Bit RX_CRC_VALID of register PHY_RSSI is updated after frame reception with the result of the FCS check (see "Overview" on page 66). The RX_CRC_VALID bit needs to be checked in order to dismiss corrupted frames.

Table 9-9. Configuration of a Sniffer Device

Register Name	Register Bits	Description
AACK_PROM_MODE	1	*1*: Enable promiscuous mode
AACK_DIS_ACK	4	*1*: Disable generation of acknowledgment

This operating mode is similar to the promiscuous mode.

Reception of Reserved Frames

Frames with reserved frame types (see section Table 9-16 on page 63) can also be handled in RX_AACK mode. This might be required when implementing proprietary, non-standard compliant protocols. It is an extension of the address filtering in RX_AACK mode. Received frames are either handled similar to data frames or may be allowed to completely bypass the address filter.

Table 9-10 below shows the required configuration for a node to receive reserved frames and Figure 9-19 on page 48 shows the corresponding flow chart.

Table 9-10. RX_AACK Configuration to Receive Reserved Frame Types

Register Name	Register Bits	Description
SHORT_ADDR_0/1 PAN_ADDR_0/1 IEEE_ADDR_0 … IEEE_ADDR_7		Set node addresses
RX_SAFE_MODE	7	*0*: disable frame protection *1*: enable frame protection
AACK_UPLD_RES_FT	4	*1* : Enable reserved frame type reception
AACK_FLTR_RES_FT	5	Filter reserved frame types like data frame type, see note below *0* : disable *1* : enable
SLOTTED_OPERATION	0	*0*: if transceiver works in un-slotted mode *1*: if transceiver works in slotted mode
AACK_I_AM_COORD	3	*0*: device is not PAN coordinator *1*: device is PAN coordinator
AACK_DIS_ACK	4	*0*: Enable generation of acknowledgment *1*: Disable generation of acknowledgment
AACK_FVN_MODE	7:6	Controls the ACK behavior, depends on FCF frame version number *0x00* : acknowledges only frames with version number 0, i.e. according to IEEE 802.15.4-2003 frames *0x01* : acknowledges only frames with version number 0 or 1, i.e. frames according to IEEE 802.15.4-2006 *0x10* : acknowledges only frames with version number 0 or 1 or 2 *0x11* : acknowledges all frames, independent of the FCF frame version number

There are two different options for handling reserved frame types.

1. AACK_UPLD_RES_FT = 1, AACK_FLT_RES_FT = 0:

   Any non-corrupted frame with a reserved frame type is indicated by a TRX24_RX_END interrupt. No further address filtering is applied on those frames. A TRX24_AMI interrupt is never generated and the acknowledgment subfield is ignored.

2. AACK_UPLD_RES_FT = 1, AACK_FLT_RES_FT = 1:

If AACK_FLT_RES_FT = 1 any frame with a reserved frame type is filtered by the address filter similar to a data frame as described in the standard. Consequently, a TRX24_AMI interrupt is generated upon address match. A TRX24_RX_END interrupt is only generated if the address matched and the frame was not corrupted. An acknowledgment is only send, when the ACK request subfield was set in the received frame and a TRX24_RX_END interrupt occurred.

Note that It is not allowed to set AACK_FLTR_RES_FT = 1 and have register bit AACK_FLTR_RES_FT set to 0.

Short Acknowledgment Frame (ACK) Start Timing

The bit AACK_ACK_TIME of register XAH_CTRL_1 defines the symbol time between frame reception and transmission of an acknowledgment frame.

Table 9-11. Overview of RX_AACK Configuration Bits

Register Name	Register Bit	Description
AACK_ACK_TIME	2	*0*: Standard compliant acknowledgement timing of 12 symbol periods. In slotted acknowledgement operation mode, the acknowledgment frame transmission can be triggered 6 symbol periods after reception of the frame earliest. *1*: Reduced acknowledgment timing of 2 symbol periods (32 µs).

Note that this feature can be used in all scenarios, independent of other configurations. However, shorter acknowledgment timing is especially useful when using High Data Rate Modes to increase battery lifetime and to improve the overall data throughput; see "High Data Rate Modes" on page 86 for details.

9.4.2.4 Frame Filtering

Frame Filtering is an evaluation whether or not a received frame is dedicated for this node. To accept a received frame and to generate an address match interrupt (TRX24_AMI) a filtering procedure as described in IEEE 802.15.4-2006 chapter 7.5.6.2. (Third level of filtering) is applied to the frame. The radio transceiver's RX_AACK mode accepts only frames that satisfy all of the following requirements (quote from IEEE 802.15.4-2006, 7.5.6.2):

1. The Frame Type subfield shall not contain a reserved frame type.
2. The Frame Version subfield shall not contain a reserved value.
3. If a destination PAN identifier is included in the frame, it shall match macPANId or shall be the broadcast PAN identifier (0xFFFF).
4. If a short destination address is included in the frame, it shall match either macShortAddress or the broadcast address (0xFFFF). Otherwise, if an extended destination address is included in the frame, it shall match aExtendedAddress.
5. If the frame type indicates that the frame is a beacon frame, the source PAN identifier shall match macPANId unless macPANId is equal to 0xFFFF, in which case the beacon frame shall be accepted regardless of the source PAN identifier.
6. If only source addressing fields are included in a data or MAC command frame, the frame shall be accepted only if the device is the PAN coordinator and the source PAN identifier matches macPANId.

The radio transceiver requires two additional rules:

1. The frame type indicates that the frame is not an ACK frame (refer toTable 9-6 on page 50).

2. At least one address field must be configured.

Address match, indicated by the TRX24_AMI interrupt is further controlled by the content of subfields of the frame control field of a received frame according to the following rule:

If (Destination Addressing Mode = 0 OR 1) AND (Source Addressing Mode = 0) no TRX24_AMI interrupt is generated, refer to Figure 9-26 on page 63. This effectively causes all acknowledgement frames not to be announced which otherwise always pass the filter regardless of whether they are intended for this device or not.

For backward compatibility to IEEE 802.15.4-2003 third level filter rule 2 (Frame Version) can be disabled by the bits AACK_FVN_MODE of register CSMA_SEED_1.

Frame filtering is available in Extended and Basic Operating Mode (see section "Basic Operating Mode" on page 35); a frame passing the frame filtering generates an TRX24_AMI interrupt, if enabled.

Note:
1. Filter rule 1 is affected by register bits AACK_FLTR_RES_FT and AACK_UPLD_RES_FT (see register "XAH_CTRL_1 – Transceiver Acknowledgment Frame Control Register 1" on page 119).
2. Filter rule 2 is affected by register bits AACK_FVN_MODE (see register "CSMA_SEED_1 – Transceiver Acknowledgment Frame Control Register 2" on page 129).

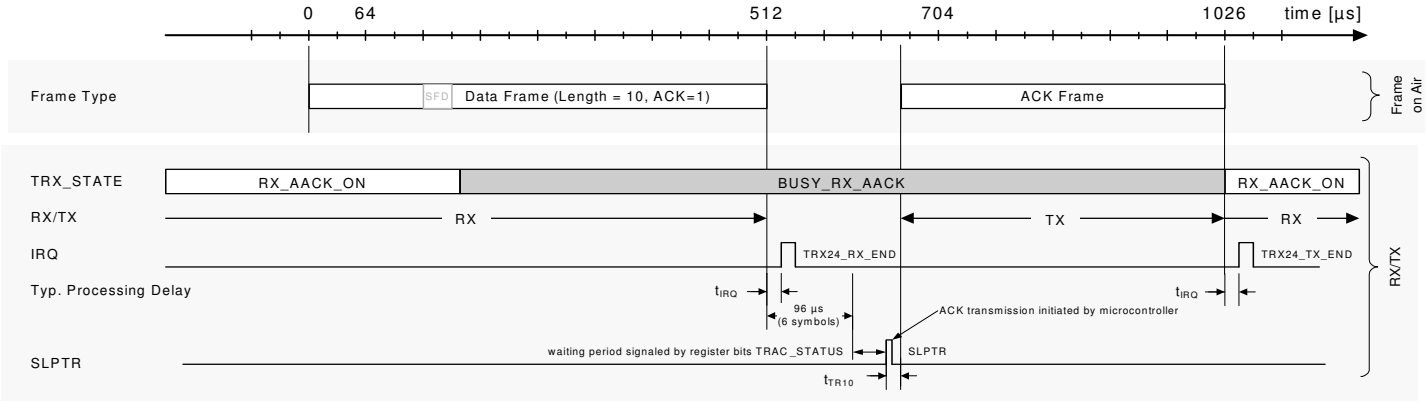
9.4.2.4.1 RX_AACK Slotted Operation – Slotted Acknowledgement

The radio transceiver supports slotted acknowledgement operation according to IEEE 802.15.4-2006, section 5.5.4.1.

In RX_AACK mode with bit SLOTTED_OPERATION of register XAH_CTRL_0 set, the transmission of an acknowledgement frame has to be controlled by the microcontroller. If an ACK frame has to be transmitted the radio transceiver expects writing SLPTR=1 to actually start the transmission. This waiting state is signaled 6 symbol periods after the reception of the last symbol of a data or MAC command frame by bits TRAC_STATUS of register XAH_CTRL_0, which are set to SUCCESS_WAIT_FOR_ACK in that case. In networks using slotted operation the start of the acknowledgment frame and thus the exact timing must be provided by the microcontroller.

A timing example of an RX_AACK transaction with bit SLOTTED_OPERATION of register XAH_CTRL_0 set is shown in the next figure. The acknowledgement frame is ready to transmit 6 symbol times after the reception of the last symbol of a data or MAC command frame. The transmission of the acknowledgement frame is initiated by the microcontroller by writing SLPTR=1 and starts 16µs (t_{TR10}) later. The interrupt latency t_{IRQ} is specified in section "Digital Interface Timing Characteristics" on page 507.

Figure 9-10. Example Timing of an RX_AACK Transaction for Slotted Operation

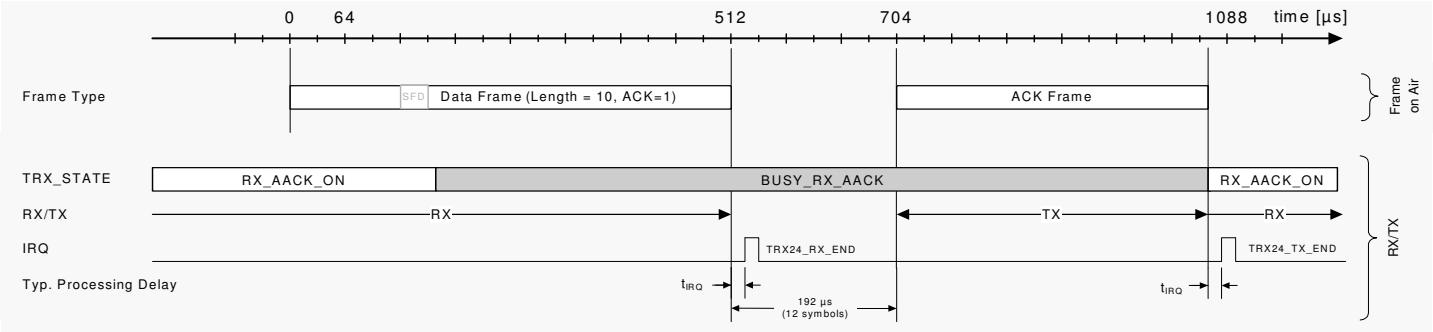


If bit AACK_ACK_TIME of register XAH_CTRL_1 is set, an acknowledgment frame can be sent already 2 symbol times after the reception of the last symbol of a data or MAC command frame.

9.4.2.4.2 RX_AACK Mode Timing

A timing example of an RX_AACK transaction is shown in the next figure. In this example a data frame of length 10 with an ACK request is received. The radio transceiver changes to state BUSY_RX_AACK after SFD detection. The completion of the frame reception is indicated by a TRX24_RX_END interrupt. Interrupts TRX24_RX_START and TRX24_AMI are disabled in this example. The ACK frame is automatically transmitted after a default wait period of 12 symbols (192 µs), bit AACK_ACK_TIME = 0 (reset value). The interrupt latency t_{IRQ} is specified in section "Digital Interface Timing Characteristics" on page 507.

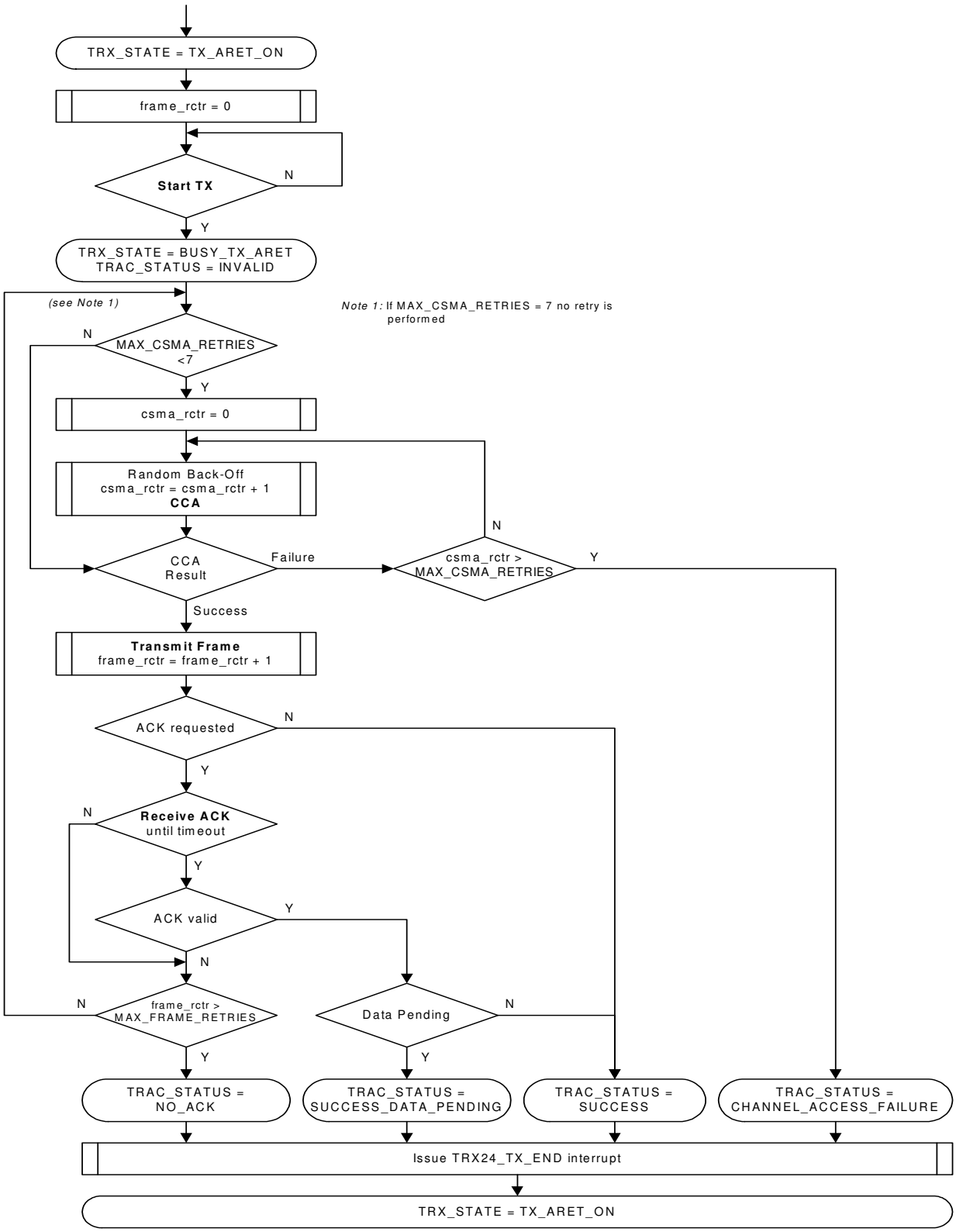
Figure 9-11. Example Timing of an RX_AACK Transaction



If bit AACK_ACK_TIME of register XAH_CTRL_1 is set, an acknowledgment frame is sent already 2 symbol times after the reception of the last symbol of a data or MAC command frame.

9.4.2.5 TX_ARET_ON – Transmit with Automatic Retry and CSMA-CA Retry

Figure 9-12. Flow Diagram of TX_ARET



Overview

The implemented TX_ARET algorithm is shown in Figure 9-12 on page 57.

In TX_ARET mode, the radio transceiver first executes the CSMA-CA algorithm, as defined by IEEE 802.15.4–2006, section 7.5.1.4, initiated by a transmit start event. If the channel is IDLE a frame is transmitted from the Frame Buffer. If the acknowledgement frame is requested the radio transceiver additionally checks for an ACK reply.

A TRX24_TX_END interrupt indicates the completion of the TX_ARET transmit transaction.

Description

Configuration and address bits are to be set in TRX_OFF or PLL_ON state prior to switching to TX_ARET mode. It is further recommended to transfer the PSDU data to the Frame Buffer in advance. The transaction is started by either writing SLPTR=1 as described in section "Transceiver Pin Register TRXPR" on page 32 or writing a TX_START command to register TRX_STATE.

If the CSMA-CA detects a busy channel, it is retried as specified by bits MAX_CSMA_RETRIES of register XAH_CTRL_0. In case that CSMA-CA does not detect a clear channel after MAX_CSMA_RETRIES it aborts the TX_ARET transaction, issues a TRX24_TX_END interrupt and sets the value of the TRAC_STATUS register bits to CHANNEL_ACCESS_FAILURE.

During transmission of a frame the radio transceiver parses bit 5 (ACK Request) of the MAC header (MHR) frame control field of the PSDU data (PSDU octet #1) to be transmitted to check if an ACK reply is expected.

If an ACK is expected the radio transceiver automatically switches into receive mode to wait for a valid ACK reply. After receiving an ACK frame the Frame Pending subfield of that frame is parsed and the status register bits TRAC_STATUS are updated accordingly (see Table 9-16 below). This receive procedure does not overwrite the Frame Buffer content. Transmit data in the Frame Buffer is not changed during the entire TX_ARET transaction. Received frames other than the expected ACK frame are discarded.

If no valid ACK is received or after timeout of 54 symbol periods (864 µs), the radio transceiver retries the entire transaction (including CSMA-CA) until the maximum number of retransmissions as set by the bits MAX_FRAME_RETRIES in register XAH_CTRL_0 is exceeded.

After that, the microcontroller may read the value of the bits TRAC_STATUS of register TRX_STATE to verify whether the transaction was successful or not. The register bits are set according to the following cases:

Table 9-16. Interpretation of the TRAC_STATUS register bits

Value	Name	Description
0	SUCCESS	The transaction was responded by a valid ACK, or, if no ACK is requested, after a successful frame transmission
1	SUCCESS_DATA_PENDING	Equivalent to SUCCESS; indicates pending frame data according to the MHR frame control field of the received ACK response
3	CHANNEL_ACCESS_FAILURE	Channel is still busy after MAX_CSMA_RETRIES of CSMA-CA

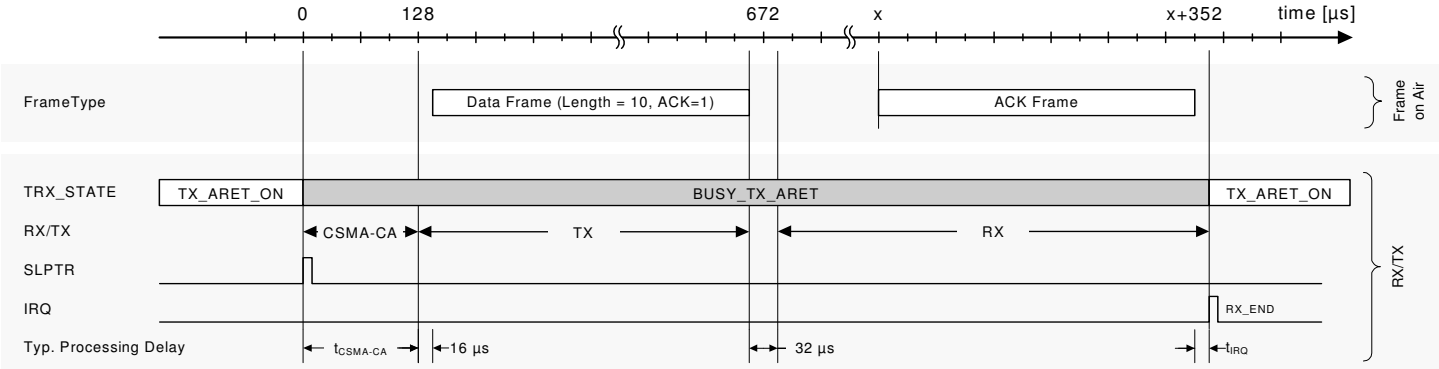
Value	Name	Description
5	NO_ACK	No acknowledgement frames were received during all retry attempts
7	INVALID	Entering TX_ARET mode sets TRAC_STATUS = 7

Note that if no ACK is expected (according to the content of the received frame in the Frame Buffer), the radio transceiver issues a TRX24_TX_END interrupt directly after the frame transmission has been completed. The value of the bits TRAC_STATUS of register TRX_STATE is set to SUCCESS.

A value of MAX_CSMA_RETRIES = 7 initiates an immediate TX_ARET transaction without performing CSMA-CA. This is required to support slotted acknowledgement operation. Further the value MAX_FRAME_RETRIES is ignored and the TX_ARET transaction is performed only once.

A timing example of a TX_ARET transaction is shown in Figure 9-13 below .

Figure 9-13. Example Timing of a TX_ARET Transaction



Note: 1. $t_{CSMA-CA}$ defines the random CSMA-CA processing time.

Here an example data frame of length 10 with an ACK request is transmitted, see Table 9-13 on page 60. After the transmission the radio transceiver switches to receive mode and expects an acknowledgement response. During the whole transaction including frame transmit, wait for ACK and ACK receive the radio transceiver status register TRX_STATUS signals BUSY_TX_ARET.

A successful reception of the acknowledgment frame is indicated by the TRX24_TX_END interrupt. The status register TRX_STATUS changes back to TX_ARET_ON. The TX_ARET status register TRAC_STATUS changes as well to TRAC_STATUS = SUCCESS or TRAC_STATUS = SUCCESS_DATA_PENDING if the frame pending subfield of the received ACK frame was set to 1.

9.4.2.6 Interrupt Handling

The interrupt handling in the Extended Operating Mode is similar to the Basic Operating Mode (see section "Interrupt Handling" on page 38). The microcontroller enables interrupts by setting the appropriate bit in register IRQ_MASK.

For RX_AACK and TX_ARET the following interrupts (Table 9-13 on page 60) inform about the status of a frame reception and transmission:

Table 9-13. Interrupt Handling in Extended Operating Mode

Mode	Interrupt	Description
RX_AACK	TRX24_RX_START	Indicates a PHR reception
	TRX24_AMI	Issued at address match
	TRX24_RX_END	Signals completion of RX_AACK transaction if successful – A received frame must pass the address filter; – The FCS is valid
TX_ARET	TRX24_TX_END	Signals completion of TX_ARET transaction
Both	TRX24_PLL_LOCK	Entering RX_AACK_ON or TX_ARET_ON state from TRX_OFF state, the TRX24_PLL_LOCK interrupt signals that the transaction can be started

RX_AACK

For RX_AACK it is recommended to enable the TRX24_RX_END interrupt. This interrupt is issued only if a frame passes the frame filtering (see section "Frame Filtering" on page 54) and has a valid FCS. This is different to Basic Operating Mode (see section "Basic Operating Mode" on page 35). The use of the other interrupts is optional.

On reception of a valid PHR a TRX24_RX_START interrupt is issued. The TRX24_AMI interrupt indicates an address match (see filter rules in section "Frame Filtering" on page 54). The completion of a frame reception with a valid FCS is indicated by the TRX24_RX_END interrupt.

Thus it can happen that a TRX24_RX_START and/or a TRX24_AMI interrupt are issued, but no TRX24_RX_END interrupt.

The end of an acknowledgment transmission is confirmed by a TRX24_TX_END interrupt.

TX_ARET

In TX_ARET interrupt TRX24_TX_END is only issued after completing the entire TX_ARET transaction.

Acknowledgement frames do not issue a TRX24_RX_START, TRX24_AMI or a TRX24_RX_END interrupt.

All other interrupts as described in section Table 9-2 on page 34 are also available in Extended Operating Mode.

9.4.2.7 Register Summary

The following registers (Table 9-14 below) are to be configured to control the Extended Operating Mode:

Table 9-14. Register Summary

Register Name	Description
TRX_STATUS	Radio transceiver status, CCA result
TRX_STATE	Radio transceiver state control, TX_ARET status
TRX_CTRL_1	TX_AUTO_CRC_ON
PHY_CC_CCA	CCA mode control, Table 9-21 on page 70
CCA_THRES	CCA threshold settings, see "Overview" on page 70
XAH_CTRL_1	RX_AACK control

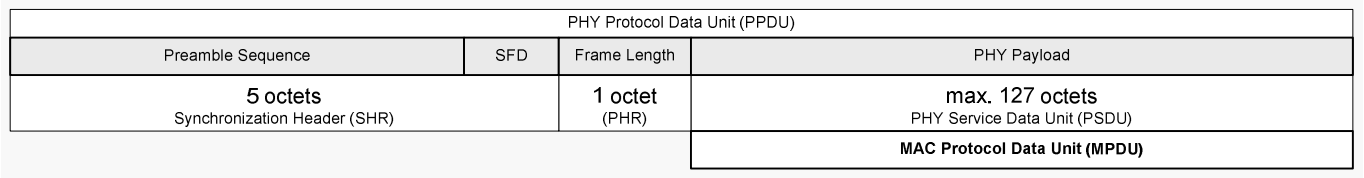
Register Name	Description
IEEE_ADDR7 .... IEEE_ADDR0 PAN_ID1 PAN_ID0 SHORT_ADDR1 SHORT_ADDR0	Address filter configuration Short address, PAN-ID and IEEE address
XAH_CTRL_0	TX_ARET control, retries value control
CSMA_SEED_0	CSMA-CA seed value
CSMA_SEED_1	CSMA-CA seed value, RX_AACK control
CSMA_BE	CSMA-CA back-off exponent control

9.5 Functional Description

9.5.1 Introduction – IEEE 802.15.4-2006 Frame Format

Figure 9-14 below provides an overview of the physical layer (PHY) frame structure as defined by IEEE 802.15.4. Figure 9-15 on page 62 shows the frame structure of the medium access control (MAC) layer.

Figure 9-14. IEEE 802.15.4 Frame Format - PHY-Layer Frame Structure (PPDU)



9.5.1.1 PHY Protocol Layer Data Unit (PPDU)

9.5.1.1.1 Synchronization Header (SHR)

The SHR consists of a four-octet preamble field (all zero), followed by a single byte start-of-frame delimiter (SFD) which has the predefined value 0xA7. During transmit, the SHR is automatically generated by the radio transceiver, thus the Frame Buffer shall contain PHR and PSDU only.

The transmission of the SHR requires 160 µs (10 symbols). As the frame buffer access is normally faster than the over-air data rate, this allows the application software to initiate a transmission without having transferred the full frame data already. Instead it is possible to subsequently write the frame content.

During frame reception, the SHR is used for synchronization purposes. The matching SFD determines the beginning of the PHR and the following PSDU payload data.

9.5.1.1.2 PHY Header (PHR)

The PHY header is a single octet following the SHR. The least significant 7 bits denote the frame length of the following PSDU, while the most significant bit of that octet is reserved, and shall be set to 0 for IEEE 802.15.4 compliant frames.

On receive the PHR is returned as the first octet during Frame Buffer read access. Even though the standard only defines frame lengths ≤127 bytes, the radio transceiver is able to transmit and receive frame length values >127. For IEEE 802.15.4 compliant operation bit 8 has to be masked by software. The reception of a valid PHR is signaled by a TRX24_RX_START interrupt.

On transmit the PHR has to be written first to the Frame Buffer.

9.5.1.1.3 PHY Payload (PHY Service Data Unit, PSDU)

The PSDU has a variable length between 0 and *aMaxPHYPacketSize* (127, maximum PSDU size in octets) whereas the last two octets are used for the Frame Check Sequence (FCS). The length of the PSDU is signaled by the frame length field (PHR) as described in Table 9-15 below. The PSDU contains the MAC Protocol Layer Data Unit (MPDU).

Received frames with a frame length field set to 0x00 (invalid PHR) are not by an interrupt.

Table 9-15 below summarizes the type of payload versus the frame length value.

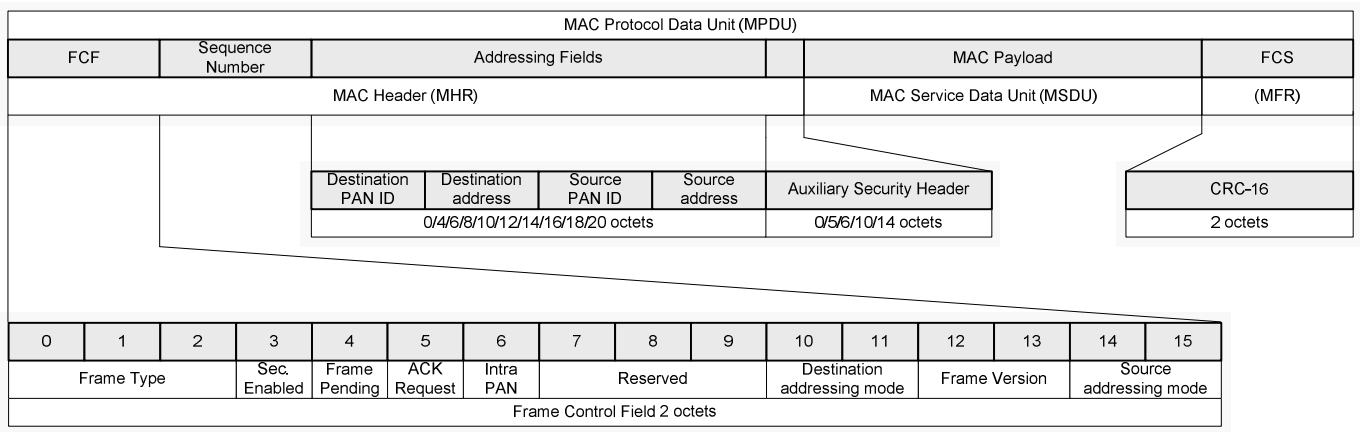
Table 9-15. Frame Length Field - PHR

Frame Length Value	Payload
0 - 4	Reserved
5	MPDU (Acknowledgement)
6 – 8	Reserved
9 - *aMaxPHYPacketSize*	MPDU

9.5.1.2 MAC Protocol Layer Data Unit (MPDU)

Figure 9-15 below shows the frame structure of the MAC layer.

Figure 9-15. IEEE 802.15.4 Frame Format - MAC-Layer Frame Structure (MPDU)



9.5.1.2.1 MAC Header (MHR) Fields

The MAC header consists of the Frame Control Field (FCF), a sequence number, and the addressing fields (which are of variable length and can even be empty in certain situations).

9.5.1.2.2 Frame Control Field (FCF)

The FCF consists of 16 bits, and occupies the first two octets of either the MPDU or the PSDU, respectively.

Figure 9-26. IEEE 802.15.4-2006 Frame Control Field (FCF)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Frame Type			Sec. Enabled	Frame Pending	ACK Request	Intra PAN	Reserved			Destination addressing mode		Frame Version		Source addressing mode	
Frame Control Field 2 octets															

Bit [2:0]: describe the frame type. Table 9-16 below summarizes frame types defined by IEEE 802.15.4, section 7.2.1.1.1.

Table 9-16. Frame Control Field – Frame Type Subfield

Frame Control Field Bit Assignments		**Description**
Frame Type Value $b_2\ b_1\ b_0$	**Value**	
000	0	Beacon
001	1	Data
010	2	Acknowledge
011	3	MAC command
100 – 111	4 – 7	Reserved

This subfield is used for address filtering by the third level filter rules. Only frame types 0 – 3 pass the third level filter rules (refer to section "Frame Filtering" on page 54). Automatic address filtering of the radio transceiver is enabled when using the RX_AACK mode (refer to "RX_AACK_ON – Receive with Automatic ACK" on page 46).

A reserved frame (frame type value > 3) can be received if bit AACK_UPLD_RES_FT of register XAH_CTRL_1 is set. For details refer to chapter "Configuration of non IEEE 802.15.4 Compliant Scenarios" on page 52. Address filtering is also provided in Basic Operating Mode as explained in "Basic Operating Mode" on page 35.

Bit 3: indicates whether security processing applies to this frame.

Bit 4: is the “Frame Pending” subfield. This field can be set in an acknowledgment frame (ACK) in response to a data request MAC command frame. This bit indicates that the node, which transmitted the ACK, has more data to send to the node receiving the ACK.

For acknowledgment frames automatically generated by the radio transceiver, this bit is set according to the content of bit AACK_SET_PD of register CSMA_SEED_1 if the received frame was a data request MAC command frame.

Bit 5: forms the “Acknowledgment Request” subfield. If this bit is set within a data or MAC command frame that is not broadcast, the recipient shall acknowledge the reception of the frame within the time specified by IEEE 802.15.4 (i.e. within 192 µs for non beacon-enabled networks).

The radio transceiver parses this bit during RX_AACK mode and transmits an acknowledgment frame if necessary.

In TX_ARET mode this bit indicates if an acknowledgement frame is expected after transmitting a frame. If this is the case, the receiver waits for the acknowledgment frame, otherwise the TX_ARET transaction is finished.

Bit 6: the “Intra-PAN” subfield indicates that in a frame, where both, the destination and source addresses are present, the PAN-ID of the source address filed is omitted. In RX_AACK mode this bit is evaluated by the address filter logic of the radio transceiver.

Bit [11:10]: the “Destination Addressing Mode” subfield describes the format of the destination address of the frame. The values of the address modes are summarized in Table 9-17 below according to IEEE 802.15.4:

Table 9-17. Frame Control Field – Destination and Source Addressing Mode

Frame Control Field Bit Assignments		Description
Addressing Mode b_{11} b_{10} b_{15} b_{14}	**Value**	
00	0	PAN identifier and address fields are not present
01	1	Reserved
10	2	Address field contains a 16-bit short address
11	3	Address field contains a 64-bit extended address

If the destination address mode is either 2 or 3 (i.e. if the destination address is present), it always consists of a 16-bit PAN-ID first followed by either the 16-bit or 64-bit address as defined by the mode.

Bit [13:12]: the “Frame Version” subfield specifies the version number corresponding to the frame. These register bits are reserved in IEEE-802.15.4-2003.

This subfield shall be set to 0 to indicate a frame compatible with IEEE 802.15.4-2003 and 1 to indicate an IEEE 802.15.4-2006 frame. All other subfield values shall be reserved for future use.

The bit AACK_FVN_MODE of register CSMA_SEED_1 controls the RX_AACK behavior of frame acknowledgements. This register determines if, depending on the Frame Version Number, a frame is acknowledged or not. This is necessary for backward compatibility to IEEE 802.15.4-2003 and for future use. Even if frame version numbers 2 and 3 are reserved, it can be handled by the radio transceiver. For details refer to "CSMA_SEED_1 – Transceiver Acknowledgment Frame Control Register 2" on page 129.

See IEEE 802.15.4-2006, section 7.2.3 for details on frame compatibility.

Table 9-18. Frame Control Field – Frame Version Subfield

Frame Control Field Bit Assignments		Description
Frame Version b_{13} b_{12}	**Value**	
00	0	Frames are compatible with IEEE 802.15.4-2003
01	1	Frames are compatible with IEEE 802.15.4-2006
10	2	Reserved
11	3	Reserved

Bit [15:14]: the “Source Addressing Mode” subfield, with similar meaning as “Destination Addressing Mode” (refer to Table 9-17 above).

The subfields of the FCF (Bits 0–2, 3, 6, 10–15) affect the address filter logic of the radio transceiver while executing a RX_AACK operation (see "RX_AACK_ON – Receive with Automatic ACK" on page 46).

9.5.1.2.3 Frame Compatibility between IEEE 802.15.4-2003 and IEEE 802.15.4-2006

All unsecured frames according to IEEE 802.15.4-2006 are compatible with unsecured frames compliant with IEEE 802.15.4-2003 with two exceptions: a coordinator realignment command frame with the "Channel Page" field present (see IEEE 802.15.4-2006 7.3.8) and any frame with a MAC Payload field larger than *aMaxMACSafePayloadSize* octets.

Compatibility for secured frames is shown in the following table, which identifies the security operating modes for IEEE 802.15.4-2006.

Table 9-19. Frame Control Field – Security and Frame Version

Frame Control Field Bit Assignments		Description
Security Enabled b_3	**Frame Version** b_{13} b_{12}	
0	00	No security. Frames are compatible between IEEE 802.15.4-2003 and IEEE 802.15.4-2006.
0	01	No security. Frames are not compatible between IEEE 802.15.4-2003 and IEEE 802.15.4-2006.
1	00	Secured frame formatted according to IEEE 802.15.4-2003. This frame type is not supported in IEEE 802.15.4-2006.
1	01	Secured frame formatted according to IEEE 802.15.4-2006

9.5.1.2.4 Sequence Number

The one-octet sequence number following the FCF identifies a particular frame, so that duplicated frame transmissions can be detected. While operating in RX_AACK mode, the content of this field is copied from the frame to be acknowledged into the acknowledgment frame.

9.5.1.2.5 Addressing Fields

The addressing fields of the MPDU are used by the radio transceiver for address matching indication. The destination address (if present) is always first, followed by the source address (if present). Each address field consists of the Intra PAN-ID and a device address. If both addresses are present and the "Intra PAN-ID compression" subfield in the FCF is set to one, the source Intra PAN-ID is omitted.

Note that in addition to these general rules IEEE 802.15.4 further restricts the valid address combinations for the individual possible MAC frame types. For example the situation where both addresses are omitted (source addressing mode = 0 and destination addressing mode = 0) is only allowed for acknowledgment frames. The address filter in the radio transceiver has been designed to apply to IEEE 802.15.4 compliant frames. It can be configured to handle other frame formats and exceptions.

9.5.1.2.6 Auxiliary Security Header Field

The Auxiliary Security Header specifies information required for security processing and has a variable length. This field determines how the frame is actually protected (security level) and which keying material from the MAC security PIB is used (see IEEE 802.15.4-2006, 7.6.1). This field shall be present only if the Security Enabled

subfield b3 is set to one (see section "Frame Compatibility between IEEE 802.15.4-2003 and IEEE 802.15.4-2006" on page 65). For details of its structure see IEEE 802.15.4-2006, 7.6.2 Auxiliary security header.

9.5.1.2.7 MAC Service Data Unit (MSDU)

This is the actual MAC payload. It is usually structured according to the individual frame type. A description can be found in IEEE 802.15.4-2006, chapter 5.5.3.2.

9.5.1.2.8 MAC Footer (MFR) Fields

The MAC footer consists of a two-octet Frame Checksum (FCS). For details refer to the following section "Frame Check Sequence (FCS)" below.

9.5.2 Frame Check Sequence (FCS)

The Frame Check Sequence (FCS) is characterized by:

- Indicate bit errors based on a cyclic redundancy check (CRC) of 16 bit length;
- Uses International Telecommunication Union (ITU) CRC polynomial;
- Automatically evaluated during reception;
- Can be automatically generated during transmission.

9.5.2.1 Overview

The FCS is intended for use at the MAC layer to detect corrupted frames at a first level of filtering. It is computed by applying an ITU CRC polynomial to all transferred bytes following the length field (MHR and MSDU fields). The frame check sequence has a length of 16 bit and is located in the last two bytes of a frame (MAC footer, see Figure 9-15 on page 62).

The radio transceiver applies an FCS check on each received frame. The result of the FCS check is stored in bit RX_CRC_VALID of register PHY_RSSI.

On transmit the radio transceiver generates and appends the FCS bytes during the frame transmission. This behavior can be disabled by setting the bit TX_AUTO_CRC_ON = 0 in register TRX_CTRL_1.

9.5.2.2 CRC calculation

The CRC polynomial used in IEEE 802.15.4 networks is defined by

$$G_{16}(x) = x^{16} + x^{12} + x^{5} + 1$$

The FCS shall be calculated for transmission using the following algorithm:

Let

$$M(x) = b_0 x^{k-1} + b_1 x^{k-2} + \ldots + b_{k-2} x + b_{k-1}$$

be the polynomial representing the sequence of bits for which the checksum is to be computed. Multiply M(x) by x^{16} giving the polynomial

$$N(x) = M(x) \cdot x^{16}$$

Divide $N(x)$ modulo 2 by the generator polynomial $G_{16}(x)$ to obtain the remainder polynomial

$$R(x) = r_0 x^{15} + r_1 x^{14} + \ldots + r_{14} x + r_{15}$$

The FCS field is given by the coefficients of the remainder polynomial, $R(x)$.

Example:

Consider a 5 octet ACK frame. The MHR field consists of

0100 0000 0000 0000 0101 0110.

The leftmost bit (b_0) is transmitted first in time. The FCS is in this case

0010 0111 1001 1110.

The leftmost bit (r_0) is transmitted first in time.

9.5.2.3 Automatic FCS generation

The automatic FCS generation is performed with register bit TX_AUTO_CRC_ON = 1 (reset value). This allows the radio transceiver to autonomously compute the FCS. For a frame with a frame length specified as N ($3 \leq N \leq 127$), the FCS is calculated on the first N-2 octets in the Frame Buffer and the resulting FCS field is transmitted in place of the last two octets from the Frame Buffer.

If the automatic FCS generation of the radio transceivers is enabled, the Frame Buffer write access can be stopped right after MAC payload. There is no need to write FCS dummy bytes.

In RX_AACK mode, when a received frame needs to be acknowledged, the FCS of the ACK frame is always automatically generated by the radio transceiver, independent of the TX_AUTO_CRC_ON setting.

Example:

A frame transmission of length five with TX_AUTO_CRC_ON set, is started with a Frame Buffer write access of five bytes (the last two bytes can be omitted). The first three bytes are used for FCS generation; the last two bytes are replaced by the internally calculated FCS.

9.5.2.4 Automatic FCS check

An automatic FCS check is applied on each received frame with a frame length $N \geq 2$. The bit RX_CRC_VALID of register PHY_RSSI is set if the FCS of a received frame is valid. The register bit is updated when issuing a TRX24_RX_END interrupt and remains valid until a new frame reception causes the next TRX24_RX_END interrupt.

In RX_AACK mode, the radio transceiver rejects the frame and the TRX24_RX_END interrupt is not issued if the FCS of the received frame is not valid.

In TX_ARET mode, the FCS and the sequence number of an ACK are automatically checked. The ACK is not accepted if one of those is not correct.

9.5.3 Received Signal Strength Indicator (RSSI)

The Received Signal Strength Indicator is characterized by:

- Minimum RSSI level is -90 dBm (RSSI_BASE_VAL);
- Dynamic range is 81 dB;
- Minimum RSSI value is 0;
- Maximum RSSI value is 28.

9.5.3.1 Overview

The RSSI is a 5-bit value indicating the receive power in the selected channel in steps of 3 dB. No attempt is made to distinguish IEEE 802.15.4 signals from others. Only the

received signal strength is evaluated. The RSSI provides the basis for an ED measurement. See section "Energy Detection (ED)" below for details.

9.5.3.2 Reading RSSI

In Basic Operating Mode the RSSI value is valid in any receive state, and is updated every t_{TR25} = 2 µs to register PHY_RSSI.

It is not recommended to read the RSSI value when using the Extended Operating Mode. The automatically generated ED value should then be used (see section "Energy Detection (ED)" below).

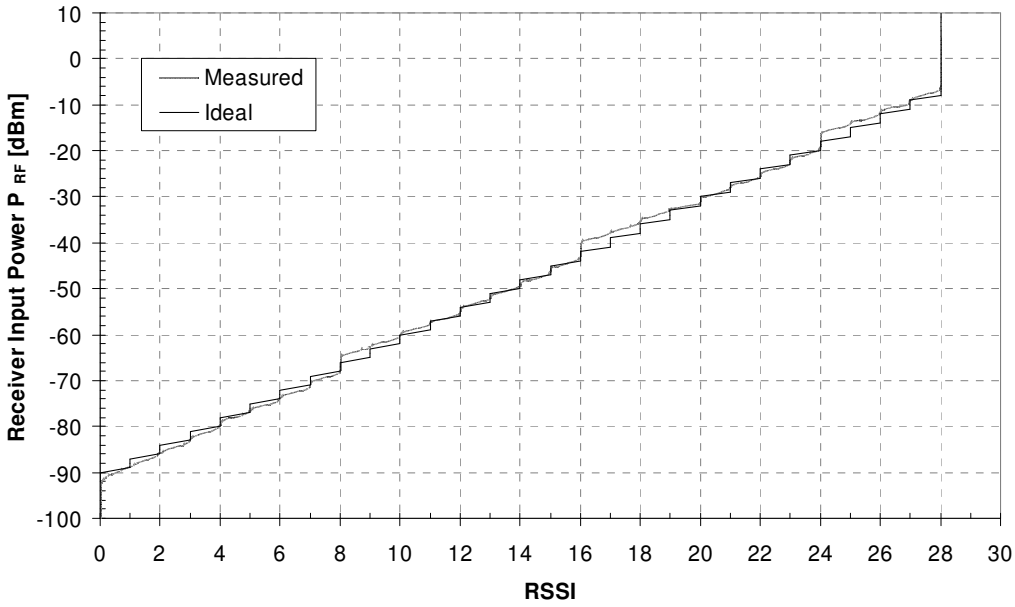
9.5.3.3 Data Interpretation

The RSSI value is a 5-bit value indicating the receive power in steps of 3 dB and with a range of 0- 28.

An RSSI value of 0 indicates a receiver RF input power of P_{RF} < -90 dBm. For an RSSI value in the range of 1 to 28, the RF input power can be calculated as follows:

$$P_{RF} = RSSI_BASE_VAL + 3 \bullet (RSSI - 1)\ [dBm]$$

Figure 9-17. Mapping between RSSI Value and Received Input Power



9.5.4 Energy Detection (ED)

The Energy Detection (ED) module is characterized by:

- 85 unique energy levels defined;
- 1 dB resolution.

9.5.4.1 Overview

The receiver ED measurement is used by the network layer as part of a channel selection algorithm. It is an estimation of the received signal power within the bandwidth of an IEEE 802.15.4 channel. No attempt is made to identify or decode signals on the channel. The ED value is calculated by averaging RSSI values over eight symbols (128 µs).

For High Data Rate Modes the automated ED measurement duration is reduced to 32 µs as described in "High Data Rate Modes" on page 86. The measurement period in these modes is still 128 µs for manually initiated ED measurements as long as the receiver is in RX_ON state.

9.5.4.2 Measurement Description

There are two ways to initiate an ED measurement:

- Manually, by writing an arbitrary value to register PHY_ED_LEVEL, or
- Automatically, after detection of a valid SHR of an incoming frame.

For manually initiated ED measurements the radio transceiver needs to be in one of the states RX_ON or BUSY_RX. The end of the ED measurement is indicated by a TRX24_CCA_ED_DONE interrupt.

The automatic ED measurement is started if a SHR is detected. The end of the automatic measurement is not signaled by an interrupt.

The measurement result is stored after t_{TR26} = 140 µs (128 µs measurement duration and processing delay) in register PHY_ED_LEVEL.

Thus by using Basic Operating Mode a valid ED value from the currently received frame is accessible 108 µs after the TRX24_RX_START interrupt and remains valid until the next incoming frame generates a new TRX24_RX_START interrupt or until another ED measurement is initiated.

When using the Extended Operating Mode it is recommended to mask the TRX24_RX_START interrupt. Hence the interrupt cannot be used as timing reference. A successful frame reception is signalized by the TRX24_RX_END interrupt. The minimum time span between a TRX24_RX_END interrupt and a following SFD detection is t_{TR27} = 96 µs due to the length of the SHR. The ED value needs to be read within 224 µs including the ED measurement time after the TRX24_RX_END interrupt. Otherwise it could be overwritten by the result of the next measurement cycle. This is important for time critical applications or if the TRX24_RX_START interrupt is not used to indicate the reception of a frame.

The values of the register PHY_ED_LEVEL are:

Table 9-20. Register Bit PHY_ED_LEVEL Interpretation

PHY_ED_LEVEL	Description
0xFF	Reset value
0x00 … 0x54	ED measurement result of the last ED measurement

Note: 1. It is not recommended to manually initiate an ED measurement when using the Extended Operating Mode.

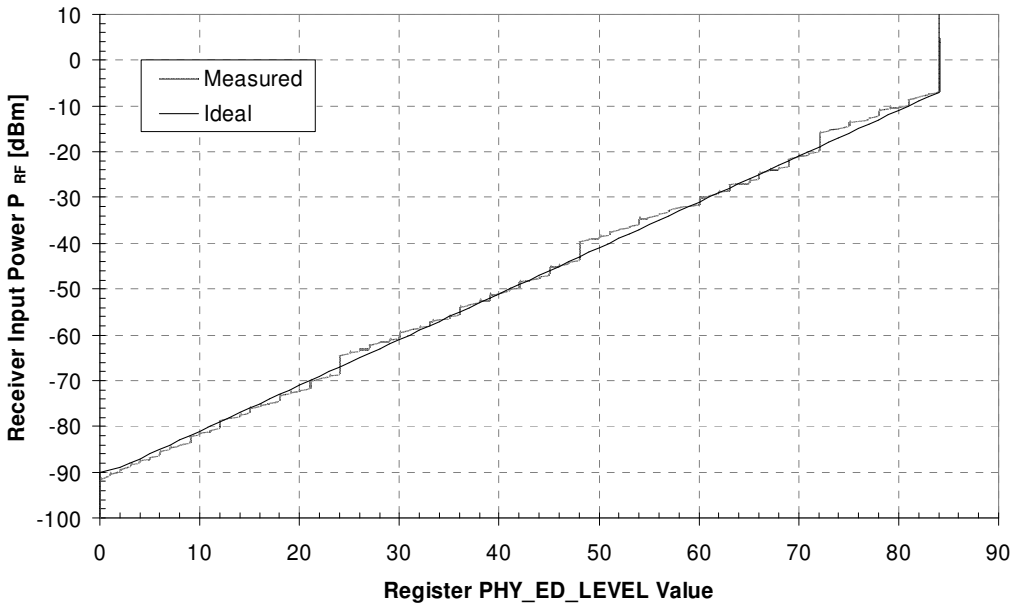
9.5.4.3 Data Interpretation

The PHY_ED_LEVEL is an 8-bit register. The ED value of the radio transceiver has a valid range from 0x00 to 0x54 with a resolution of 1 dB. All other values do not occur. A value of 0xFF indicates the reset value. A value of PHY_ED_LEVEL = 0 indicates that the measured energy is less than -90 dBm (see parameter RSSI_BASE_VAL in section "Receiver Characteristics" on page 508). Due to environmental conditions (temperature, voltage, semiconductor parameters etc.) the calculated ED value has a maximum tolerance of ±5 dB, this is to be considered as constant offset over the measurement range.

An ED value of 0 indicates an RF input power of $P_{RF} \leq$ -90 dBm. For an ED value in the range of 0 to 84, the RF input power can be calculated as follows:

P_{RF} = -90 + ED [dBm]

Figure 9-18. Mapping between values in PHY_ED_LEVEL and Received Input Power



9.5.4.4 Interrupt Handling

The TRX24_CCA_ED_DONE interrupt is issued at the end of a manually initiated ED measurement.

Note that an ED request should only be initiated in one of the receive states. Otherwise the radio transceiver generates a TRX24_CCA_ED_DONE interrupt but no ED measurement was performed.

9.5.5 Clear Channel Assessment (CCA)

The main features of the Clear Channel Assessment (CCA) module are:

- All 4 modes are available as defined by IEEE 802.15.4-2006 in section 6.9.9;
- Adjustable threshold for energy detection algorithm.

9.5.5.1 Overview

A CCA measurement is used to detect a clear channel. Four modes are specified by IEEE 802.15.4-2006:

Table 9-21. CCA Mode Overview

CCA Mode	Description
1	*Energy above threshold.* CCA shall report a busy medium upon detecting any energy above the ED threshold.
2	*Carrier sense only.* CCA shall report a busy medium only upon the detection of a signal with the modulation and spreading characteristics of an IEEE 802.15.4 compliant signal. The signal strength may be above or below the ED threshold.

CCA Mode	Description
0, 3	*Carrier sense with energy above threshold.* CCA shall report a busy medium using a logical combination of – Detection of a signal with the modulation and spreading characteristics of this standard and – Energy above the ED threshold. Where the logical operator may be configured as either OR (mode 0) or AND (mode 3).

9.5.5.2 Configuration and CCA Request

The CCA modes are configurable via register PHY_CC_CCA.

Usimg the Basic Operating Mode, a CCA request can be initiated manually by setting CCA_REQUEST = 1 of register PHY_CC_CCA, if the radio transceiver is in any RX state. The current channel status (CCA_STATUS) and the CCA completion status (CCA_DONE) are accessible in register TRX_STATUS.

The CCA evaluation is done over eight symbol periods and the result is accessible t_{TR28} = 140 µs (128 µs measurement duration and processing delay) after the request. The end of a manually initiated CCA measurement is indicated by a TRX24_CCA_ED_DONE interrupt.

The sub-register CCA_ED_THRES of register CCA_THRES defines the received power threshold of the “energy above threshold” algorithm. The threshold is calculated by RSSI_BASE_VAL + 2 • CCA_ED_THRES [dBm]. Any received power above this level is interpreted as a busy channel.

Note that it is not recommended to manually initiate a CCA measurement when using the Extended Operating Mode.

9.5.5.3 Data Interpretation

The current channel status (CCA_STATUS) and the CCA completion status (CCA_DONE) are accessible in register TRX_STATUS. Note, register bits CCA_DONE and CCA_STATUS are cleared in response to a CCA_REQUEST.

The completion of a measurement cycle is indicated by CCA_DONE = 1. If the radio transceiver detected no signal (idle channel) during the measurement cycle, the CCA_STATUS bit is set to 1.

When using the “energy above threshold” algorithm, any received power above CCA_ED_THRES level is interpreted as a busy channel. The “carrier sense” algorithm reports a busy channel when detecting an IEEE 802.15.4 signal above the RSSI_BASE_VAL (see parameter RSSI_BASE_VAL in "Transceiver Electrical Characteristics" on page 507). The radio transceiver is also able to detect signals below this value, but the detection probability decreases with the signal power.

9.5.5.4 Interrupt Handling

The TRX24_CCA_ED_DONE interrupt is issued at the end of a manually initiated CCA measurement.

Note: A CCA request should only be initiated in the receive states of Basic Operating Mode. Otherwise the radio transceiver generates a TRX24_CCA_ED_DONE interrupt and sets the register bit CCA_DONE = 1 even if no CCA measurement was performed.

9.5.5.5 Measurement Time

The response time for a manually initiated CCA measurement depends on the receiver state.

In RX_ON state the CCA measurement is done over eight symbol periods and the result is accessible 140 µs after the request (see section "Configuration and CCA Request" on page 71).

In BUSY_RX state the CCA measurement duration depends on the CCA Mode and the CCA request relative to the reception of an SHR. The end of the CCA measurement is indicated by a TRX24_CCA_ED_DONE interrupt. The variation of a CCA measurement period in BUSY_RX state is described in Table 9-22 below.

Table 9-22. CCA Measurement Period and Access in BUSY_RX state

CCA Mode	Request within ED measurement[1]	Request after ED measurement
1	*Energy above threshold.*	
	CCA result is available after finishing automated ED measurement period.	CCA result is immediately available after request.
2	*Carrier sense only.*	
	CCA result is immediately available after request.	
3	*Carrier sense with Energy above threshold (AND).*	
	CCA result is available after finishing automated ED measurement period.	CCA result is immediately available after request.
0	*Carrier sense with Energy above threshold (OR).*	
	CCA result is available after finishing automated ED measurement period.	CCA result is immediately available after request.

Note: 1. After receiving the SHR an automated ED measurement is started with a length of 8 symbol periods (PSDU rate 250 kb/s), refer to section "Energy Detection (ED)" on page 68. This automated ED measurement must be finished to provide a result for the CCA measurement. Only one automated ED measurement per frame is performed.

It is recommended to perform CCA measurements in RX_ON state only. To avoid accidental switching to BUSY_RX state the SHR detection can be disabled by setting bit RX_PDT_DIS of register RX_SYN. Refer to section "Receiver (RX)" on page 74 for details. The receiver remains in RX_ON state to perform a CCA measurement until the register bit RX_PDT_DIS is set back to continue the frame reception. In this case the CCA measurement duration is 8 symbol periods.

9.5.6 Link Quality Indication (LQI)

According to IEEE 802.15.4 the LQI measurement is a characterization of the strength and/or quality of a received packet. The measurement may be implemented using receiver ED, a signal-to-noise ratio estimation or a combination of these methods. The use of the LQI result by the network or application layers is not specified in this standard. LQI values shall be an integer ranging from 0x00 to 0xFF. The minimum and maximum LQI values (0x00 and 0xFF) should be associated with the lowest and highest quality compliant signals, respectively, and LQI values in between should be uniformly distributed between these two limits.

9.5.6.1 Overview

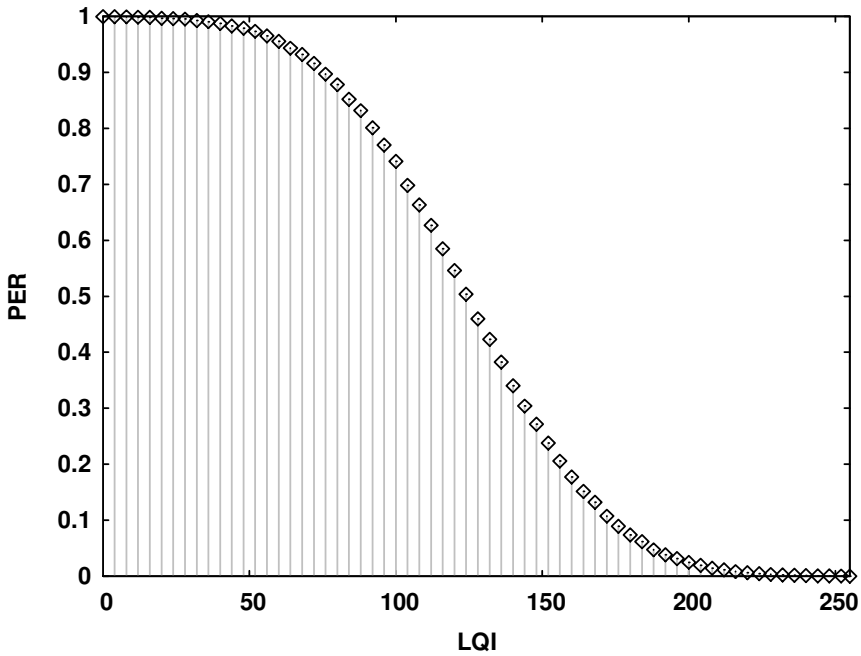
The LQI measurement of the radio transceiver is implemented as a measure of the link quality which can be described with the packet error rate (PER) of this link. A LQI value

can be associated with an expected packet error rate. The PER is the ratio of erroneous received frames to the total number of received frames. A PER of zero indicates no frame error whereas at a PER of one no frame was received correctly.

The radio transceiver uses correlation results of multiple symbols within a frame to determine the LQI value. This is done for each received frame. The minimum frame length for a valid LQI value is two octets PSDU. LQI values are integers ranging from 0 to 255.

The following figure shows an example of a conditional packet error rate when receiving a certain LQI value.

Figure 9-19. Conditional Packet Error Rate versus LQI



The values are taken from received frames of PSDU length of 20 octets on transmission channels with reasonable low multipath delay spreads. If the transmission channel characteristic has a higher multipath delay spread than assumed in the example, the PER is slightly higher for a certain LQI value. Since the packet error rate is a statistical value, the PER shown in Figure 9-19 above is based on a huge number of transactions. A reliable estimation of the packet error rate cannot be based on a single or a small number of LQI values.

9.5.6.2 Request a LQI Measurement

The LQI byte can be obtained after a frame has been received by the radio transceiver. One additional byte is automatically attached to the received frame containing the LQI value. This information can also be read via Frame Buffer read access, see "User accessible Frame Content" on page 78. The LQI byte can be read after the TRX24_RX_END interrupt.

9.5.6.3 Data Interpretation

According to IEEE 802.15.4 a low LQI value is associated with low signal strength and/or high signal distortions. Signal distortions are mainly caused by interference signals and/or multipath propagation. High LQI values indicate a sufficient high signal power and low signal distortions.

Note that the received signal power as indicated by the received signal strength indication (RSSI) value or energy detection (ED) value of the radio transceiver do not characterize the signal quality and the ability to decode a signal.

As an example, a received signal with an input power of about 6 dB above the receiver sensitivity likely results in a LQI value close to 255 for radio channels with very low signal distortions. For higher signal power the LQI value becomes independent of the actual signal strength. This is because the packet error rate for these scenarios tends towards zero and further increased signal strength i.e. increasing the transmission power does not decrease the error rate any further. In this case RSSI or ED can be used to evaluate the signal strength and the link margin.

ZigBee networks often require the identification of the “best” routing between two nodes. Both the LQI and the RSSI/ED can be used for this, dependent on the optimization criteria. If a low packet error rate (corresponding to high throughput) is the optimization criteria then the LQI value should be taken into consideration. If a low transmission power or the link margin is the optimization criteria then the RSSI/ED value is also helpful.

Combinations of LQI, RSSI and ED are possible for routing decisions. As a rule of thumb RSSI and ED values are useful to differentiate between links with high LQI values. Transmission links with low LQI values should be discarded for routing decisions even if the RSSI/ED values are high. This is because RSSI and ED do not say anything about the possibility to decode a signal. It is only an information about the received signal strength whereas the source can be an interferer.

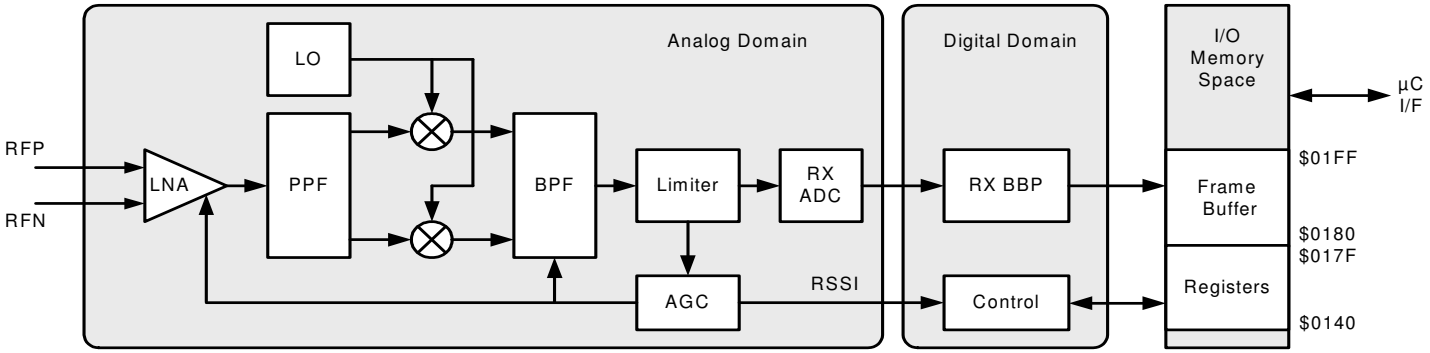
9.6 Module Description

9.6.1 Receiver (RX)

9.6.1.1 Overview

The receiver is split into an analog radio front-end and a digital base band processor (RX BBP) according to the following figure. The digital base band processor and the control engine are connected to the Frame Buffer and control registers which are located in the microcontroller I/O memory space (see "I/O Memory" on page 25 and "Transceiver to Microcontroller Interface" on page 31).

Figure 9-20. Receiver Block Diagram



The differential RF signal is amplified by a low noise amplifier (LNA), filtered (PPF) and down converted to an intermediate frequency by a mixer. Channel selectivity is performed using an integrated band pass filter (BPF). A limiting amplifier (Limiter) provides sufficient gain to overcome the DC offset of the succeeding analog-to-digital

converter (RX ADC) and generates a digital RSSI signal. The ADC output signal is sampled and processed further by the digital base band receiver (RX BBP).

The RX BBP performs additional signal filtering and signal synchronization. The frequency offset of each frame is calculated by the synchronization unit and is used during the remaining receive process to correct the offset. The receiver is designed to handle frequency and symbol rate deviations up to ±120 ppm caused by combined receiver and transmitter deviations. For details refer to chapter "General RF Specifications" on page 507. Finally the signal is demodulated and the data are stored in the Frame Buffer.

In Basic Operating Mode (see "Basic Operating Mode" on page 35), the reception of a frame is indicated by a TRX24_RX_START interrupt. Accordingly its end is signalized by a TRX24_RX_END interrupt. Based on the quality of the received signal a link quality indicator (LQI) is calculated and appended to the frame. For details refer to. Additional signal processing is applied to the frame data to provide further status information like ED value (register PHY_ED_LEVEL) and FCS correctness (register PHY_RSSI).

Beyond these features the Extended Operating Mode of the radio transceiver supports address filtering and pending data indication. For details refer to "Extended Operating Mode" on page 43.

9.6.1.2 Frame Receive Procedure

The frame receive procedure including the radio s setup for reception and reading PSDU data from the Frame Buffer is described in "Frame Receive Procedure" on page 84.

9.6.1.3 Configuration

In Basic Operating Mode the receiver is enabled by writing command RX_ON to the TRX_CMD bits of register TRX_STATE in the states TRX_OFF or PLL_ON. Similarly in Extended Operating Mode the receiver is enabled for RX_AACK operation from the states TRX_OFF or PLL_ON by writing the command RX_AACK_ON. There is no additional configuration required to receive IEEE 802.15.4 compliant frames when using the Basic Operating Mode. However, the frame reception in the Extended Operating Mode requires further register configurations. For details refer to "Extended Operating Mode" on page 43.

The receiver has an outstanding sensitivity performance of -100 dBm. At certain environmental conditions or for High Data Rate Modes (see "High Data Rate Modes" on page 86) it may be useful to manually decrease this sensitivity. This is achieved by adjusting the detector threshold of the synchronization header using the RX_PDT_LEVEL bits of register RX_SYN. Received signals with a RSSI value below the threshold do not activate the demodulation process.

Furthermore, it may be useful to protect a received frame against overwriting by subsequent received frames.

A Dynamic Frame Buffer Protection is enabled with register bit RX_SAFE_MODE (TRX_CTRL_2) set (refer to "Dynamic Frame Buffer Protection" on page 91). After a frame has been received, the buffer is protected for new incoming frames and the receiver remains in RX_ON or RX_AACK_ON state until the RX_SAFE_MODE bit is cleared by the controller. The Frame Buffer content is only protected if the FCS is valid.

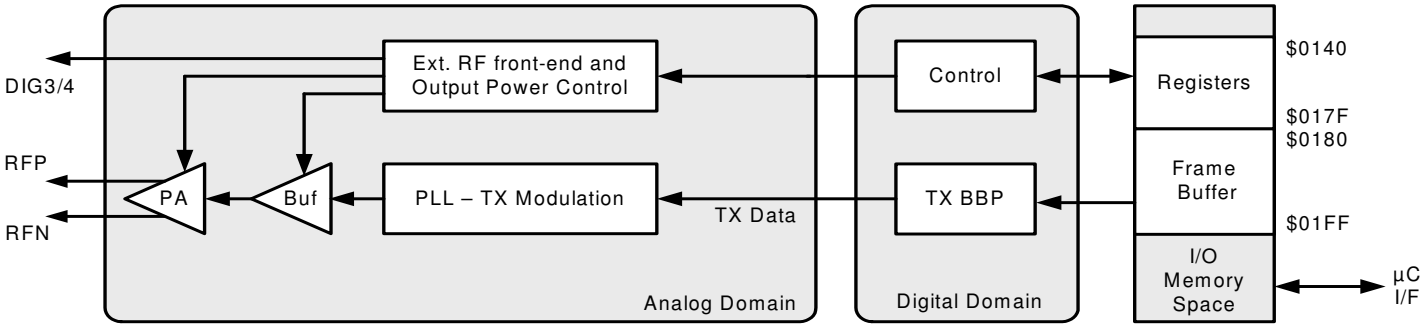
A Static Frame Buffer Protection is enabled with bit RX_PDT_DIS of register RX_SYN set. The receiver remains in RX_ON or RX_AACK_ON state and no further SHR is detected until the register bit RX_PDT_DIS is set back.

9.6.2 Transmitter (TX)

9.6.2.1 Overview

The transmitter consists of a digital base band processor (TX BBP) and an analog front end as shown in the following figure.

Figure 9-21. Transmitter Block Diagram



The TX BBP reads the frame data from the Frame Buffer and performs the bit-to-symbol and symbol-to-chip mapping as specified by IEEE 802.15.4 in section 6.5.2. The O-QPSK modulation signal is generated and fed into the analog radio front end.

The fractional-N frequency synthesizer (PLL) converts the baseband transmit signal to the RF signal which is amplified by the power amplifier (PA). The PA output is internally connected to bidirectional differential antenna pins (RFP, RFN) so that no external antenna switch is needed.

9.6.2.2 Frame Transmit Procedure

The frame transmit procedure including writing PSDU data in the Frame Buffer and initiating a transmission is described in section "Frame Transmit Procedure" on page 84. The controller must ensure to provide valid frame data before starting the frame transmission. For save operation, it is recommended to write the complete frame into the Frame Buffer before starting the frame transmission.

9.6.2.3 Configuration

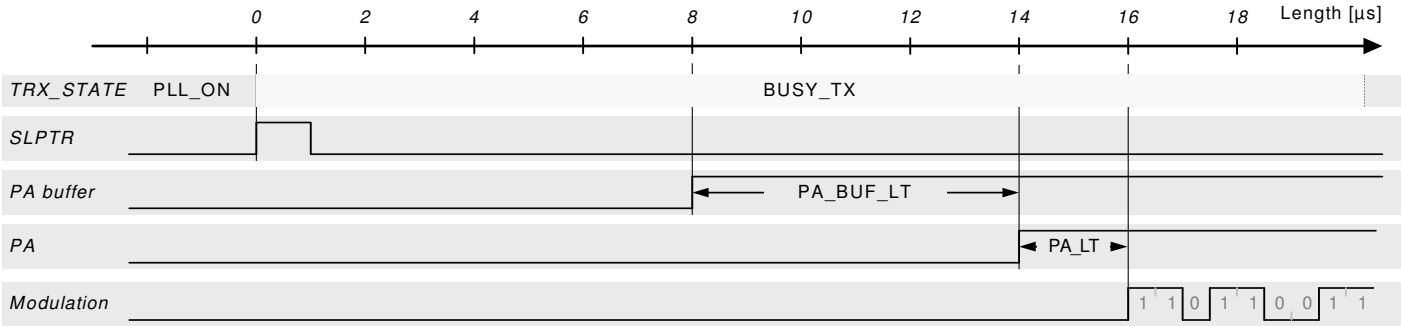
The maximum output power of the transmitter is typically +3.5 dBm. The output power can be configured via the TX_PWR bits of register PHY_TX_PWR. The output power of the transmitter can be controlled over a 20 dB range.

A transmission can be started from PLL_ON or TX_ARET_ON state by writing ‘1’ to bit SLPTR of the TRXPR register or by writing TX_START command to the TRX_CMD bits of register TRX_STATE.

9.6.2.4 TX Power Ramping

The PA buffer and PA are enabled sequentially to optimize the output power spectral density (PSD). A timing example using default settings illustrates the sequence in the next figure. In this example the transmission is initiated with the rising edge of the SLPTR bit. The radio transceiver state changes from PLL_ON to BUSY_TX. The modulation starts 16 µs after SLPTR.

Figure 9-22. TX Power Ramping



When using an external RF front-end (refer to "RX/TX Indicator" on page 90) it may be required to adjust the startup time of the external PA relative to the internal building blocks to optimize the overall PSD. This can be achieved using register bits PA_BUF_LT and PA_LT of register PHY_TX_PWR.

9.6.3 Frame Buffer

The radio transceiver contains a 128 byte dual port SRAM. One port of the frame buffer is directly connected to the controller I/O space. Therefore random access to single frame bytes is possible. The other port connects to the internal transmitter and receiver modules. Both ports are independent and simultaneously accessible for data communication.

The Frame Buffer uses the controller I/O address space 0x180 to 0x1FF for RX and TX operation of the radio transceiver and can keep one IEEE 802.15.4 RX or one TX frame of maximum length at a time.

Frame Buffer access is only possible if the radio transceiver is enabled (PRTRX24 bit in the Power Reduction Register PRR1 is not set) and not in SLEEP state.

9.6.3.1 Data Management

Data in the Frame Buffer (received data or data to be transmitted) remain valid as long as:

- No new frame or other data are written into the buffer;
- No new frame is received (in any BUSY_RX state);
- No state change into radio transceiver SLEEP state is made;
- No radio transceiver RESET (see bit TRXRST in "TRXPR – Transceiver Pin Register" on page 169) or system reset took place;
- Bit PRTRX24 in register "PRR1 – Power Reduction Register 1" on page 168 is not set;

By default there is no protection of the Frame Buffer against overwriting. If a frame is received during a Frame Buffer read access of a previously received frame, the stored data might be overwritten.

Finally the application software should check the transferred frame data integrity by a FCS check.

The state of the radio transceiver should be changed to PLL_ON state after reception to protect the Frame Buffer content against overwriting with new, incoming frames. This can be achieved by writing immediately the command PLL_ON to the TRX_CMD bits of register TRX_STATE after receiving the frame indicated by a TRX24_RX_END interrupt.

Alternatively Dynamic Frame Buffer Protection can be used to protect received frames against overwriting. For details refer to "Dynamic Frame Buffer Protection" on page 91.

Both procedures do not protect the Frame Buffer from overwriting by the application software.

In Extended Operating Mode during TX_ARET operation (see "TX_ARET_ON – Transmit with Automatic Retry and CSMA-CA Retry" on page 57) the radio transceiver switches to receive if an acknowledgement of a previously transmitted frame was requested. During this period received frames are evaluated but not stored in the Frame Buffer. This allows the radio transceiver to wait for an acknowledgement frame and retry the frame transmission without writing the frame data to the Frame Buffer again.

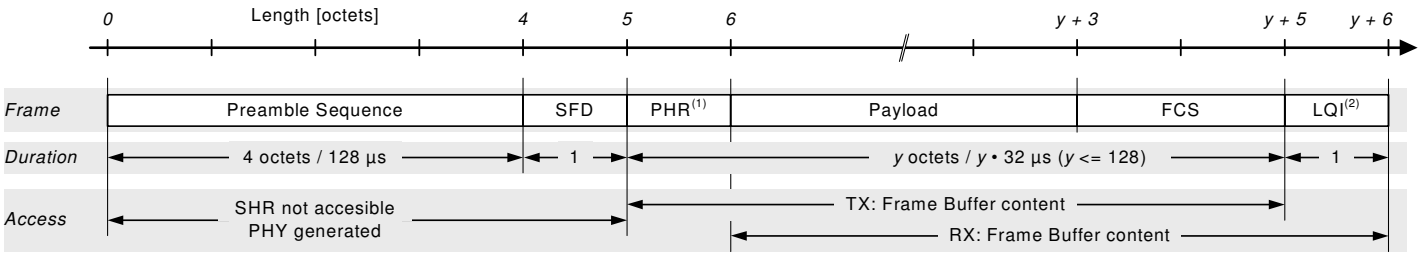
A radio transceiver state change except a transition to radio transceiver SLEEP state or a radio transceiver RESET does not affect the Frame Buffer content. The Frame Buffer is powered off and the stored data gets lost if the radio transceiver is forced into radio transceiver SLEEP state.

Access conflicts may occur when reading and writing data simultaneously at the two independent ports of the Frame Buffer TX/RX BBP and Controller interface.

9.6.3.2 User accessible Frame Content

The radio transceiver supports an IEEE 802.15.4 compliant frame format as shown in the following figure.

Figure 9-31. Transceiver Frame Structure



Notes: 1. Stored into Frame Buffer for TX operation
2. Stored into Frame Buffer during frame reception.

A frame comprises two sections. The radio transceiver internally generated SHR field and the user accessible part are stored in the Frame Buffer. The SHR contains the preamble and the SFD field. The variable frame section contains the PHR and the PSDU including the FCS (see "Overview" on page 66).

The Frame Buffer content differs depending on the direction of the communication (receive or transmit). To access the data follow the procedures described in "Radio Transceiver Usage" on page 83.

During frame reception, the payload and the link quality indicator (LQI) value of a successfully received frame are stored in the Frame Buffer. The radio transceiver appends the LQI value to the frame data after the last received octet. Information of the frame length is not stored in the Frame Buffer. The frame length information is located in register TST_RX_LENGTH.

The SHR (except the SFD used to generate the last octet of the SHR) can generally not be read by the application software.

The PHR and the PSDU need to be stored in the Frame Buffer for frame transmission. The PHR byte is the first byte in the Frame Buffer (address 0x180) and must be calculated based on the PHR and the PSDU. The maximum frame size supported by the radio transceiver is 128 bytes. If the TX_AUTO_CRC_ON bit is set in register PHY_TX_PWR, the FCS field of the PSDU is replaced by the automatically calculated FCS during frame transmission. There is no need to write the FCS field when using the automatic FCS generation.

Manipulating individual bytes of the Frame Buffer is simply possible by accessing the appropriate buffer address.

The minimum frame length supported by the radio transceiver for non IEEE 802.15.4 compliant frames is one byte (Frame Length Field + 1 byte of data).

9.6.4 Battery Monitor (BATMON)

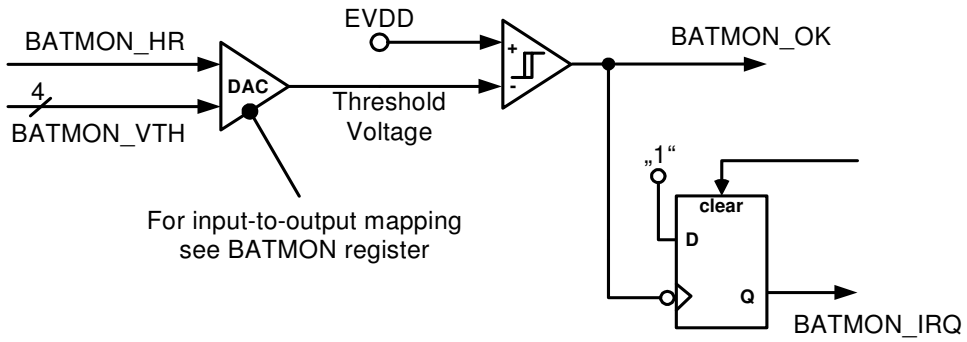
The main features of the battery monitor are:

- Configurable voltage threshold range from 1.7V to 3.675V
- Generates an interrupt when supply voltage drops below the threshold

9.6.4.1 Overview

The battery monitor (BATMON) detects and indicates a low supply voltage of EVDD. This is done by comparing the voltage of EVDD with a configurable, internal threshold voltage. A simplified schematic of the BATMON with the most important input and output signals is shown in the following figure.

Figure 9-24. Simplified Schematic of BATMON



9.6.4.2 Configuration

The Battery Monitor can be configured using the BATMON register. Register subfield BATMON_VTH sets the threshold voltage. It is configurable with a resolution of 75 mV in the upper voltage range (BATMON_HR = 1) and with a resolution of 50 mV in the lower voltage range (BATMON_HR = 0).

9.6.4.3 Data Interpretation

The bit BATMON_OK of register BATMON monitors the current value of the battery voltage:

- If BATMON_OK = 0 then the battery voltage is lower than the threshold voltage;
- If BATMON_OK = 1 then the battery voltage is higher than the threshold voltage;

The value BATMON_OK should be read out to verify the current supply voltage value after setting a new threshold.

Note: The battery monitor is inactive during SLEEP states. Refer to status register TRX_STATUS for details.

9.6.4.4 Interrupt Handling

A supply voltage drop below the configured threshold value is indicated by the BAT_LOW interrupt. The BAT_LOW status bit as well as the BATLOW_EN bit is located in the BATMON register. If BATLOW_EN =0, no IRQ is issued, but the status flag is set if the battery low event occurs.

The interrupt is only issued if BATMON_OK changes from 1 to 0 and the event is stored until the IRQ handler is called or the BAT_LOW IRQ is cleared manually by writing '1' to the BAT_LOW status flag.

No interrupt is generated when:

- The battery voltage is below the default 1.8V threshold at power up (BATMON_OK was never 1) or
- A new threshold is set which is still above the current supply voltage (BATMON_OK remains 0).

Noise or temporary voltage drops may generate unwanted interrupts when the battery voltage is close to the programmed threshold voltage. To avoid this:

- Disable the BAT_LOW interrupt with the BATLOW_EN Bit in the BATMON register and treat the battery as empty or
- Set a lower threshold value.

9.6.5 Crystal Oscillator (XOSC)

The main features of the crystal oscillator are:

- Amplitude controlled 16 MHz generation;
- 215 µs typical settling time after leaving SLEEP state;
- Configurable trimming with a capacitance array;

9.6.5.1 Overview

The crystal oscillator generates the reference frequency for the radio transceiver. All other internally generated frequencies of the radio transceiver are derived from this unique frequency. The overall system performance is therefore critically determined by the accuracy of the crystal reference frequency. The external components of the crystal oscillator should be selected carefully and the related board layout should be done with caution as described in section "Application Circuits" on page 493.

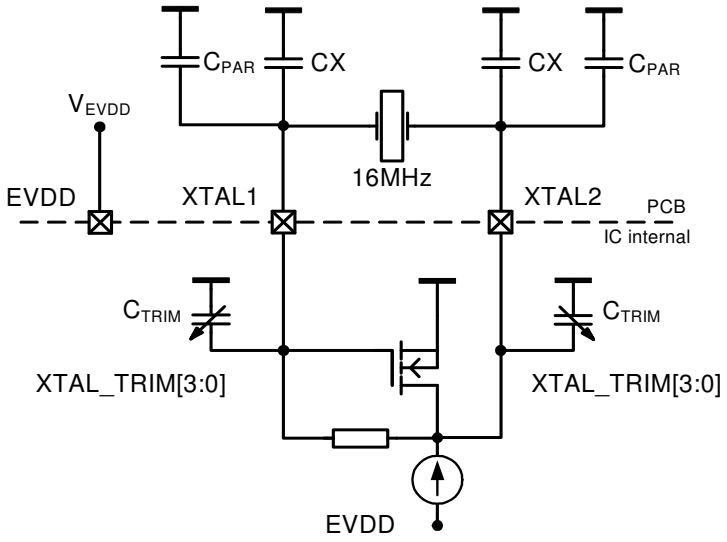
The register XOSC_CTRL provides access to the control signals of the oscillator. Two operating modes are supported. It is recommended to use the integrated oscillator setup as described in Figure 9-25 on page 81. Nevertheless a reference frequency can be fed to the internal circuitry by using an external clock reference as shown in Figure 9-26 on page 82.

9.6.5.2 Integrated Oscillator Setup

The output frequency of the internal oscillator depends on the load capacitance between the crystal pins XTAL1 and XTAL2. The total load capacitance C_L must be equal to the specified load capacitance of the crystal itself. It consists of the external capacitors CX and parasitic capacitances connected to the XTAL nodes.

The following figure shows all parasitic capacitances, such as PCB stray capacitances and the pin input capacitance summarized to C_{PAR}.

Figure 9-25. Simplified XOSC Schematic with External Components



Additional internal trimming capacitors C_{TRIM} are available. Any value in the range from 0 pF to 4.5 pF with a 0.3 pF resolution is selectable using XTAL_TRIM of register XOSC_CTRL. To calculate the total load capacitance, the following formula can be used

$$C_L = 0.5 \cdot (CX + C_{TRIM} + C_{PAR}).$$

The trimming capacitors provide the possibility to reduce frequency deviations caused by variations of the production process or by tolerances of external components. Note that the oscillation frequency can only be reduced by increasing the trimming capacitance. The frequency deviation caused by one step of C_{TRIM} decreases with increasing values of the crystal load capacitor.

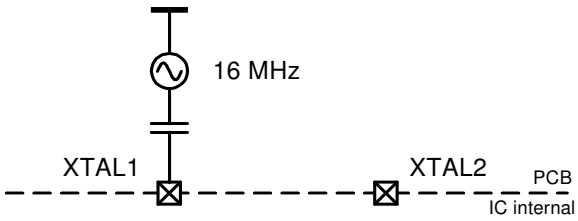
An amplitude control circuit is included to ensure stable operation under different operating conditions and for different crystal types. Enabling the crystal oscillator after leaving SLEEP state causes a slightly higher current during the amplitude build-up phase to guarantee a short start-up time. The current is reduced to the amount necessary for a robust oscillation during stable operation. This also keeps the drive level of the crystal low.

Crystals with a higher load capacitance are generally less sensitive to parasitic pulling effects caused by variations of external components or board and circuit parasitics. On the other hand a larger crystal load capacitance results in a longer start-up time and a higher steady state current consumption.

9.6.5.3 External Reference Frequency Setup

When using an external reference frequency, the signal must be connected to pin XTAL1 as indicated in Figure 9-26 on page 82 and the bits XTAL_MODE of register XOSC_CTRL need to be set to the external oscillator mode. The oscillation peak-to-peak amplitude shall between 100 mV and 500 mV, the optimum range is between 400 mV and 500 mV. Pin XTAL2 should not be wired

Figure 9-26. Setup for Using an External Frequency Reference



9.6.6 Frequency Synthesizer (PLL)

The main features of the phase-locked loop are:

- Generate RX/TX frequencies for all 2.4 GHz channels of IEEE 802.15.4;
- Autonomous calibration loops for stable operation within the operating range;
- Two PLL-interrupts for status indication;
- Fast PLL settling to support frequency hopping;

9.6.6.1 Overview

The PLL generates the RF frequencies for the radio transceiver. During receive operation the frequency synthesizer works as a local oscillator for the receive frequency of the radio transceiver. During transmit operation the voltage-controlled oscillator (VCO) is directly modulated to generate the RF transmit signal. The frequency synthesizer is implemented as a fractional-N PLL.

Two calibration loops ensure correct PLL functionality within the specified operating limits.

9.6.6.2 RF Channel Selection

The PLL is designed to support 16 channels in the 2.4 GHz ISM band with channel spacing of 5 MHz according to IEEE 802.15.4. The center frequency of these channels is defined as follows:

$F_c = 2405 + 5\ (k - 11)$ in [MHz], for $k = 11, 12 \ldots 26$

where k is the channel number.

The channel k is selected by the CHANNEL bits of register PHY_CC_CA.

9.6.6.3 Frequency Agility

When the PLL is enabled during state transition from TRX_OFF to PLL_ON the settling time is typically $t_{TR4} = 110$ µs including the settling time of the analog voltage regulator (AVREG) and the PLL self calibration (refer to Table 9-8 on page 42Table 9-8 and ? ? ? ? ? ? ? ? ? ? ? ?). A lock of the PLL is indicated with a TRX24_PLL_LOCK interrupt.

Switching between 2.4 GHz ISM band channels in PLL_ON or RX_ON states is typically done within $t_{TR20} = 11$ µs. This makes the radio transceiver highly suitable for frequency hopping applications.

The PLL frequency is changed to the transmit frequency within $t_{TR23} = 16$ µs after starting the transmit procedure and before starting the transmission. After the transmission the PLL settles back to the receive frequency within $t_{TR24} = 32$ µs. This frequency step does not generate a TRX24_PLL_LOCK or TRX24_PLL_UNLOCK interrupt within these time spans.

9.6.6.4 Calibration Loops

Due to temperature, supply voltage and part-to-part variations of the radio transceiver the VCO characteristics diverge. Two automated control loops are implemented to ensure a stable operation: center frequency (CF) tuning and delay cell (DCU) calibration. Both calibration loops are initiated automatically when the PLL is enabled during state transition from TRX_OFF to PLL_ON. The center frequency calibration is additionally initiated when the PLL changes to a center frequency of another channel.

It is recommended to initiate the calibration loops manually if the PLL operates for a long time on the same channel e.g. more than 5 min or the operating temperature changes significantly. Both calibration loops can be initiated manually by setting PLL_CF_START = 1 of register PLL_CF and PLL_DCU_START = 1 of register PLL_DCU. The device must be in PLL_ON or RX_ON state to start the calibration. The completion of the center frequency tuning is indicated by a TRX24_PLL_LOCK interrupt.

Both calibration loops may be run simultaneously.

9.6.6.5 Interrupt Handling

Two different interrupts indicate the PLL status. The TRX24_PLL_LOCK interrupt indicates that the PLL has locked. The TRX24_PLL_UNLOCK interrupt indicates an unexpected unlock condition.

A TRX24_PLL_LOCK interrupt is supposed to occur in the following situations:

- State change from TRX_OFF to PLL_ON / RX_ON/ RX_AACK_ON/ TX_ARET_ON;
- Channel change in states PLL_ON / RX_ON/ RX_AACK_ON/ TX_ARET_ON;

Any other occurrences of PLL interrupts indicate erroneous behavior and require checking of the actual device status.

The state transition from BUSY_TX to PLL_ON after successful transmission does not generate a TRX24_PLL_LOCK interrupt within the settling period.

9.6.7 Automatic Filter Tuning (FTN)

The FTN is incorporated to compensate device tolerances for temperature, supply voltage variations as well as part-to-part variations of the radio transceiver. The filter-tuning result is used to correct the transfer function of the analog baseband filter and the time constant of the PLL loop-filter (refer to "General Circuit Description" on page 30).

An FTN calibration cycle is initiated automatically when entering the radio transceiver TRX_OFF state from the SLEEP or RESET state.

Although receiver and transmitter are very robust against these variations, it is recommended to initiate the FTN manually if the radio transceiver does not use the SLEEP state. A calibration cycle is to be initiated in states TRX_OFF, PLL_ON or RX_ON if necessary. This applies in particular to the High Data Rate Modes with a much higher sensitivity to variations of the BPF transfer function. The recommended calibration interval is 5 min or less.

9.7 Radio Transceiver Usage

This section describes the basic procedures to receive and transmit frames with the radio transceiver.

9.7.1 Frame Receive Procedure

A frame reception comprises of two actions: The PHY listens for a frame, receives and demodulates the frame to the Frame Buffer and signalizes its reception to the application software. The application software reads the available frame data from the Frame Buffer after or during the progress of the frame reception.

While in state RX_ON or RX_AACK_ON the radio transceiver searches for incoming frames on the selected channel. First a TRX24_RX_START interrupt indicates the detection of an IEEE 802.15.4 compliant frame assuming the appropriate interrupts are enabled. The frame reception is completed when issuing the TRX24_RX_END interrupt.

Different Frame Buffer read access scenarios are recommended for:

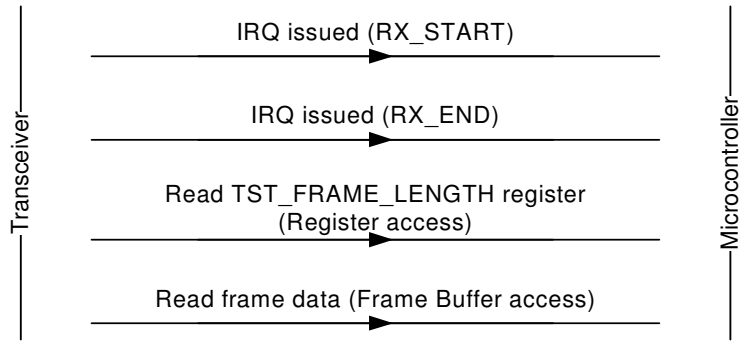
- Non-time critical applications: read access starts after the TRX24_RX_END interrupt;
- Time-critical applications: read access starts after the TRX24_RX_START interrupt;

The controller must ensure to read valid Frame Buffer contents. Reading frame data before frame reception is finished can lead to invalid data, if buffer regions are accessed which are not yet updated with the new frame.

While receiving a frame the data needs to be primarily stored in the Frame Buffer before reading it. This is ensured by accessing the first Frame Buffer byte at least 32 µs after the TRX24_RX_START interrupt.

It is recommended for operations considered to be not time-critical to wait for the TRX24_RX_END interrupt before starting a Frame Buffer read access. The following figure illustrates the frame receive procedure using the TRX24_RX_END interrupt.

Figure 9-27. Transactions between radio transceiver and microcontroller during receive



Critical protocol timing could require starting the Frame Buffer read access after the TRX24_RX_START interrupt. The first byte of the frame data can be read 32 µs after the TRX24_RX_START interrupt. The application software must ensure to read slower than the frame is received. Otherwise a Frame Buffer under-run occurs and the frame data may be not valid.

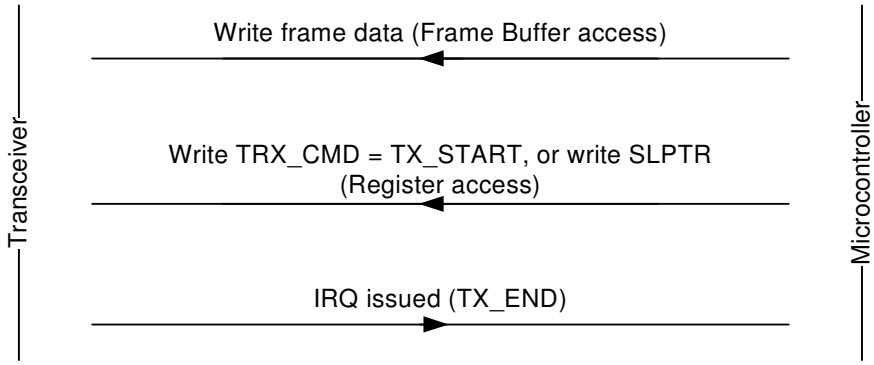
9.7.2 Frame Transmit Procedure

A frame transmission comprises of the two actions Frame Buffer write access and transmission of the Frame Buffer content. Both actions can be run in parallel if required by critical protocol timing.

Figure 9-28 on page 85 illustrates the frame transmit procedure by consecutively writing and transmitting the frame. The frame transmission is initiated writing SLPTR or writing

command TX_START to register TRX_STATE after a Frame Buffer write access and while the radio transceiver is in state PLL_ON or TX_ARET_ON. The TRX24_TX_END interrupt indicates the completion of the transaction.

Figure 9-28. Transaction between radio transceiver and microcontroller during transmit

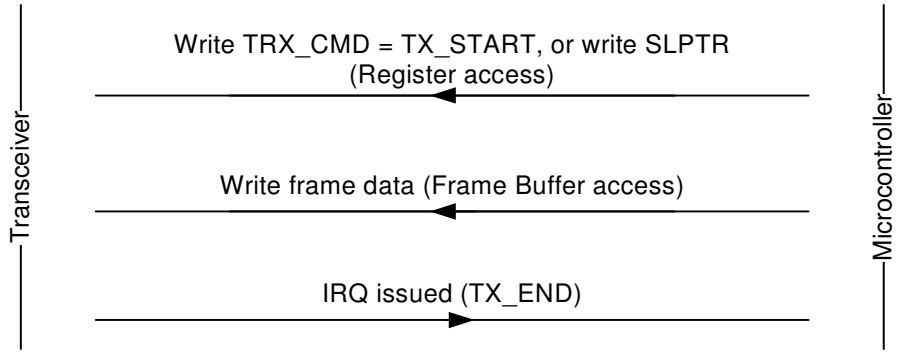


Alternatively a frame transmission can be started first, followed by the Frame Buffer write access (PSDU data) as shown in Figure 9-29 below. This is applicable for time critical applications.

A transmission is initiated either by writing SLPTR or by writing the TX_START command to the TRX_CMD bits of register TRX_STATE. The radio transceiver then starts transmitting the SHR which is internally generated.

This first phase requires 16 µs for PLL settling and 160 µs for SHR transmission. The PHR must be available in the Frame Buffer before this time elapses. Furthermore the Frame Buffer must be filled faster than the frame is transmitted to prevent a buffer under-run.

Figure 9-29. Time Optimized Frame Transmit Procedure



9.8 Radio Transceiver Extended Feature Set

9.8.1 Random Number Generator

The radio transceiver incorporates a 2-bit, noise observing, true random number generator to be used to:

- Generate random seeds for CSMA-CA algorithm (see"Extended Operating Mode" on page 43);

- Generate random values for AES key generation (see "Security Module (AES)" on page 92);

The random number is updated every t_{TR29} = 1 µs in Basic Operation Mode receive states. The values are stored in bits RND_VALUE of register PHY_RSSI.

9.8.2 High Data Rate Modes

The main features of the High Data Rate Modes are:

- High Data Rate Communication up to 2 Mb/s;
- Support of Basic and Extended Operating Mode;
- Support of other features of the Extended Feature Set;

9.8.2.1 Overview

The radio transceiver also supports alternative data rates higher than 250 kb/s for applications beyond IEEE 802.15.4 compliant networks.

The selection of a data rate does not affect the remaining functionality. Thus it is possible to run all features and operating modes of the radio transceiver in various combinations.

The data rate can be selected by writing bits OQPSK_DATA_RATE of register TRX_CTRL_2.

The High Data Rate Modes occupy the same RF channel bandwidth as the IEEE 802.15.4 – 2.4 GHz 250 kb/s standard mode. The sensitivity of the receiver is reduced due to the decreased spreading factor. The following table shows typical values of the sensitivity for different data rates.

Table 9-23. High Data Rate Sensitivity

High Data Rate	Sensitivity	Comment
250 kb/s	-100 dBm	PER ≤ 1%, PSDU length of 20 octets
500 kb/s	-9 dBm	PER ≤ 1%, PSDU length of 20 octets
1000 kb/s	-9 dBm	PER ≤ 1%, PSDU length of 20 octets
2000 kb/s	-8 dBm	PER ≤ 1%, PSDU length of 20 octets

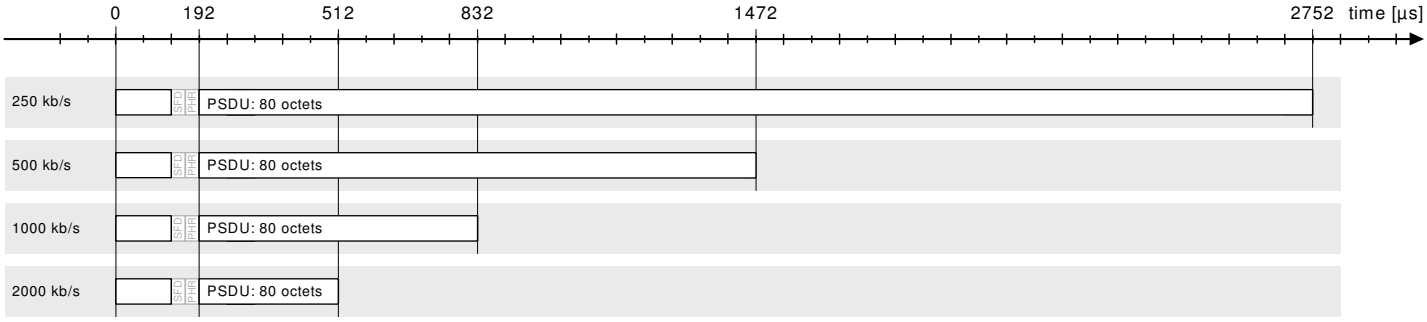
By default there is no header based signaling of the data rate within a transmitted frame. Thus nodes using a data rate other than the default IEEE 802.15.4 data rate of 250 kb/s are to be consistently configured in advance. The configurable start of frame delimiter (SFD) could be alternatively used as an indicator of the PHY data rate (see "Configurable Start-Of-Frame Delimiter (SFD)" on page 91).

9.8.2.2 High Data Rate Packet Structure

Higher data rate modulation is restricted to only the payload octets in order to allow appropriate frame synchronization. The SHR and the PHR field are transmitted with the IEEE 802.15.4 compliant data rate of 250 kb/s (refer to "Introduction – IEEE 802.15.4-2006 Frame Format" on page 61).

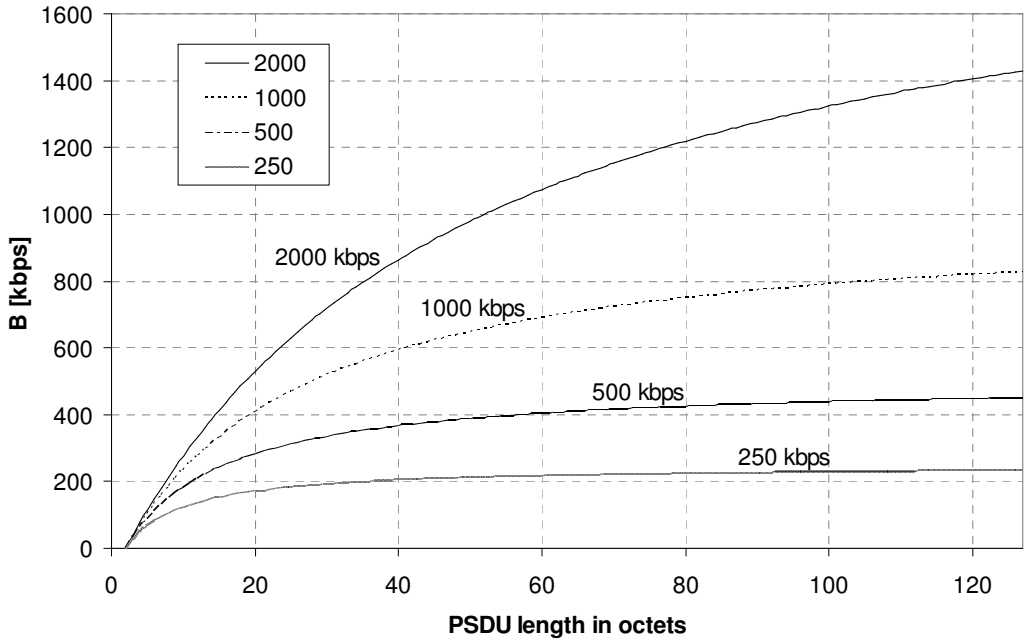
A comparison of the general packet structure for different data rates with an example PSDU length of 80 octets is shown in Figure 9-30 on page 87.

Figure 9-30. High Data Rate Frame Structure



The effective data rate is smaller than the selected data rate due to the overhead caused by the SHR, the PHR and the FCS. The overhead depends further on the length of the PSDU. A graphical representation of the effective data rate is shown in the following figure:

Figure 9-31. Effective Data Rate “B” for High Data Rate Mode



Therefore High Data Rate transmission and reception is useful for large PSDU lengths due to the higher effective data rate or to reduce the power consumption of the system. Furthermore the active on-air time using High Data Rate Modes is significantly reduced.

9.8.2.3 High Data Rate Frame Buffer Access

The Frame Buffer access to read or write frames for High Data Rate communication is similar to the procedure described in "Frame Buffer" on page 77. However the last byte in the Frame Buffer after the PSDU data is the ED value rather than the LQI value.

9.8.2.4 High Data Rate Energy Detection

According to IEEE 802.15.4 the ED measurement duration is 8 symbol periods. For frames operated at higher data rates the automated ED measurement duration is reduced to 32 µs to take the reduced frame length into account ("Energy Detection (ED)" on page 68).

9.8.2.5 High Data Rate Mode Options

Receiver Sensitivity Control

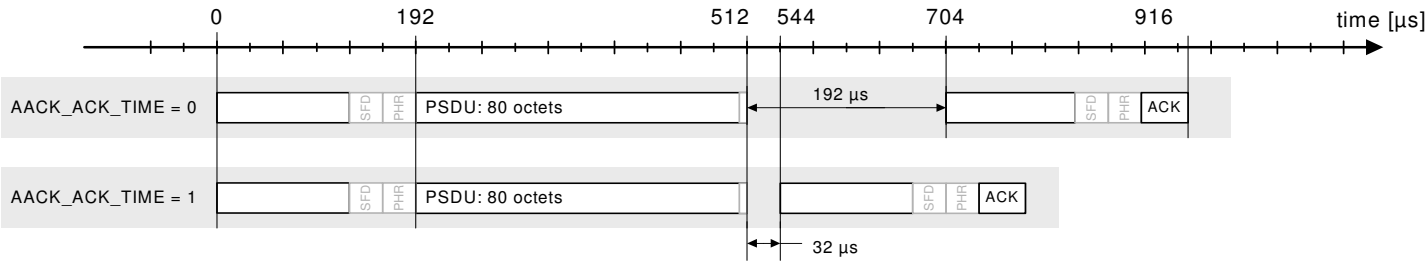
The different data rates between PPDU header (SHR and PHR) and PHY payload (PSDU) cause a different sensitivity between header and payload. This can be adjusted by defining sensitivity threshold levels of the receiver. The receiver does not receive frames with an RSSI level below the defined sensitivity threshold level (register bits RX_PDT_LEVEL > 0). Under these operating conditions the receiver current consumption is reduced by 500 µA (refer to chapter "Current Consumption Specifications" on page 509).

A description of the settings to control the sensitivity threshold with register RX_SYN can be found in section "RX_SYN – Transceiver Receiver Sensitivity Control Register" on page 118.

Reduced Acknowledgment Timing

On higher data rates the IEEE 802.15.4 compliant acknowledgment frame response time of 192 µs significantly reduces the effective data rate of the network. To minimize this influence in Extended Operating Mode RX_AACK (see section "RX_AACK_ON – Receive with Automatic ACK" on page 46), the acknowledgment frame response time can be reduced to 32 µs. Figure 9-32 below illustrates an example for a reception and acknowledgement of a frame with a data rate of 2000 kb/s and a PSDU length of 80 symbols. The PSDU length of the acknowledgment frame is 5 octets according to IEEE 802.15.4.

Figure 9-32. High Data Rate AACK Timing



The acknowledgment time is reduced from 192 µs to 32 µs if bit AACK_ACK_TIME of register XAH_CTRL_1 is set.

9.8.3 Antenna Diversity

The main features of the Antenna Diversity implementation are:

- Improves signal path robustness between nodes;
- Self-contained antenna diversity algorithm of the radio transceiver;
- Direct register based antenna selection;

9.8.3.1 Overview

The receive signal strength may vary and affect the link quality even for small changes of the antenna location due to multipath propagation effects between network nodes. These fading effects can result in an increased error floor or loss of the connection between devices.

Antenna Diversity can be applied to reduce the effects of multipath propagation and fading hence improving the reliability of a RF connection between network nodes.

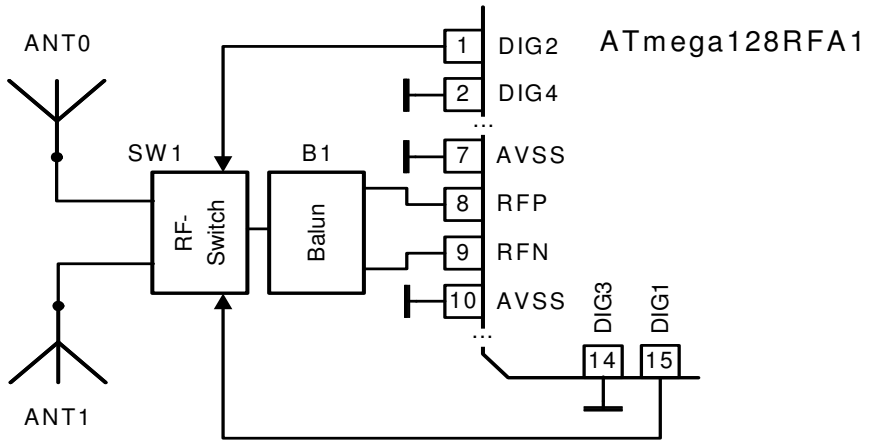
Antenna Diversity uses two antennas to switch to the most reliable RF signal path. This is done by the radio transceiver during RX_LISTEN and RX_AACK_ON state without interaction of the application software. Both antennas should be carefully separated from each other to ensure highly independent receive signals.

Antenna Diversity can be used in Basic and Extended Operating Modes and can also be combined with other features and operating modes like High Data Rate Mode and RX/TX Indication.

9.8.3.2 Antenna Diversity Application Example

A block diagram for an application using an antenna switch is shown in the following figure.

Figure 9-33. External Antenna Diversity – Block Diagram



Generally, the Antenna Diversity algorithm is enabled with bit ANT_DIV_EN=1 in register ANT_DIV. For the External Antenna Diversity the control of the antenna switch (SW1) must be enabled by bit ANT_EXT_SW_EN of register ANT_DIV. Under this condition the control pins DIG1 and DIG2 are configured as outputs. DIG1 and DIG2 are used to feed the antenna switch signal and its inverse to the differential inputs of the RF Switch (SW1).

The selected antenna is indicated by bit ANT_SEL of register ANT_DIV. The antenna selection continues searching for new frames on both antennas after the frame reception is completed. However the register bit ANT_SEL maintains its previous value (from the last received frame) until a new SHR has been found and the selection algorithm locked into one antenna again. Then the register bit ANT_SEL is updated.

The antenna defined by the ANT_CTRL bits of register ANT_DIV is selected for transmission. If for example the same antenna as selected for reception is to be used for transmission, the antenna must be set using the ANT_CTRL bits based on the value read from the ANT_SEL bit. It is recommended to read bit ANT_SEL after the TRX24_RX_START interrupt.

The autonomous search and selection allows the use of Antenna Diversity during reception even if the application software currently does not control the radio transceiver for instance in Extended Operating Mode.

An application software defined selection of a certain antenna can be done by disabling the automatic Antenna Diversity algorithm (ANT_DIV_EN = 0) and selecting one antenna using register bit ANT_CTRL.

If the radio transceiver is not in a receive or transmit state, it is recommended to disable register bit ANT_EXT_SW_EN and to set the port pins DIG1 and DIG2 to output low (DDG1 = 1, PORTG1 = 0, DDF2 = 1, PORTF2 = 0) in order to reduce the power consumption or avoid leakage current of the external RF switch especially during sleep modes.

9.8.3.3 Antenna Diversity with Extended Operation Modes

A combination of Extended Operation Mode and antenna diversity is allowed.

While the radio transceiver is in RX_AACK_ON state, it switches to an antenna with a reliable signal. The receive antenna selection is also used for transmission of an automatic acknowledge frame.

While the radio transceiver is in TX_ARET state, the selected antenna is automatically changed for every frame transmission retry.

9.8.3.4 Antenna Diversity Sensitivity Control

The detection threshold of the receiver has to be adjusted due to a different receive algorithm used by the Antenna Diversity algorithm. It is recommended to set bits PDT_THRES of register RX_CTRL to 3.

9.8.4 RX/TX Indicator

The main features are:

- RX/TX Indicator to control an external RF Front-End;
- Application software independent RF Front-End Control;
- Provide TX Timing Information;

9.8.4.1 Overview

While IEEE 802.15.4 is a low-cost, low-power standard, solutions supporting higher transmit output power are occasionally desirable. A differential control pin pair can indicate that the radio transceiver is currently in transmit mode to simplify the control of an optional external RF front-end.

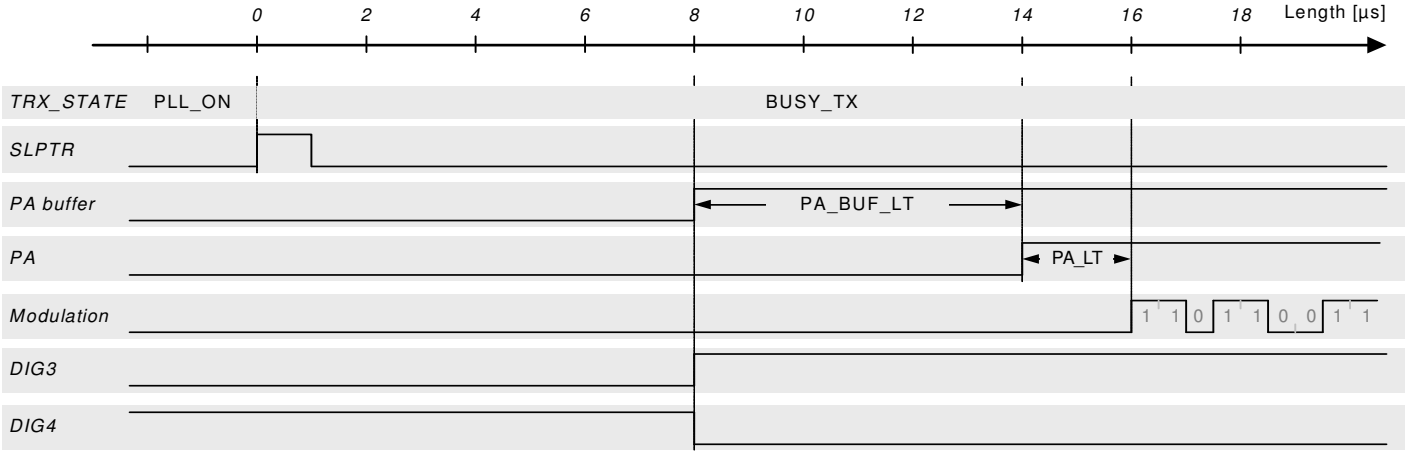
The control of an external RF front-end is done via the digital control pins DIG3/DIG4. The function of this pin pair is enabled with bit PA_EXT_EN of register TRX_CTRL_1. Pin DIG3 is set to low level and DIG4 to high level while the transmitter is turned off. The two pins change the polarity when the radio transceiver starts transmitting. This differential pin pair can be used to control PA, LNA and RF switches.

If the radio transceiver is not in a receive or transmit state, it is recommended to disable register bit PA_EXT_EN and to set the port pins DIG3 and DIG4 to output low (DDG0 = 1, PORTG0 = 0, DDF3 = 1, PORTF3 = 0) in order to reduce the power consumption or avoid leakage current of external RF switches and other building blocks especially during sleep modes.

9.8.4.2 External RF-Front End Control

The setup time of the external power amplifier (PA) relative to the internal building blocks should be adjusted when using an external RF front-end including a power amplifier to optimize the overall power spectral density (PSD) mask.

Figure 9-34. TX Power Ramping Control for RF Front-Ends



The start-up sequence of the individual building blocks of the internal transmitter is shown in the previous figure. The transmission is actually initiated by writing ‘1’ to SLPTR. The radio transceiver state changes from PLL_ON to BUSY_TX and the PLL settles to the transmit frequency within 16 µs (parameter tTR23 at page 42). The modulation starts 16 µs (parameter tTR10 at page 42) after the SLPTR=1. The PA buffer and the internal PA are enabled during this time.

The control of an external PA is done via the differential pin pair DIG3 and DIG4. DIG3 = H / DIG4 = L indicates that the transmission starts and can be used to enable an external PA. The timing of pins DIG3/DIG4 can be adjusted relative to the start of the frame and the activation of the internal PA buffer. This is controlled using the register bits PA_BUF_LT and PA_LT. For details refer to Figure 9-22 on page 77.

9.8.5 RX Frame Time Stamping

To determine the exact timing of an incoming frame e.g. for beaconing networks, the Symbol Counter should be used. SFD Time Stamping is enabled by setting bit SCTSE of the Symbol Counter Control Register SCCR0. The actual 32 Bit Symbol Counter value is captured in the SFD Time Stamp register SCTSR at the time, the SFD has been received. For details see section "SFD and Beacon Timestamp Generation" on page 135.

9.8.6 Configurable Start-Of-Frame Delimiter (SFD)

The SFD is a field indicating the end of the SHR and the start of the packet data. The length of the SFD is 1 octet (2 symbols). This octet is used for byte synchronization only and is not included in the Frame Buffer.

The value of the SFD could be changed if it is needed to operate non IEEE 802.15.4 compliant networks. An IEEE 802.15.4 compliant network node does not synchronize to frames with a different SFD value.

The register SFD_VALUE contains the one octet start-of-frame delimiter (SFD) to synchronize to a received frame. It is not recommended to set the low-order 4 bits to 0 due to the way the SHR is formed.

9.8.7 Dynamic Frame Buffer Protection

The ATmega128RFA1 continues the reception of incoming frames as long as it is in any receive state. When a frame was successfully received and stored into the Frame Buffer, the following frame will overwrite the Frame Buffer content again. To relax the timing requirements for a Frame Buffer read access the Dynamic Frame Buffer

Protection prevents that a new valid frame passes to the Frame Buffer until the buffer protection bit is cleared (RX_SAFE_MODE = 0).

A received frame is automatically protected against overwriting:

- in Basic Operating Mode, if its FCS is valid
- in Extended Operating Mode, if an TRX24_RX_END interrupt is generated

The Dynamic Frame Buffer Protection is enabled, if register bit RX_SAFE_MODE (register TRX_CTRL_2, see "TRX_CTRL_2 – Transceiver Control Register 2" on page 112) is set and the radio transceiver state is RX_ON or RX_AACK_ON.

Note that Dynamic Frame Buffer Protection only prevents write accesses from the air interface not from the application software. The application software may still modify the Frame Buffer content.

9.8.8 Security Module (AES)

The security module (AES) is characterized by:

- Hardware accelerated encryption and decryption;
- Compatible with AES-128 standard (128 bit key and data block size);
- ECB (encryption/decryption) mode and CBC (encryption) mode support;
- Stand-alone operation, independent of other blocks;

See TBD Application regarding the AES security module usage.

9.8.8.1 Overview

The security module is based on an AES-128 core according to the FIPS197 standard [5]. and provides two modes, the Electronic Code Book (ECB) and the Cipher Block Chaining (CBC). The security module works independent of other building blocks of the radio transceiver. Encryption and decryption can be performed in parallel to a frame transmission or reception.

The ECB and CBC modules including the AES core are clocked with the 16 MHz Radio Transceiver Crystal Oscillator.

Controlling the security block is possible over 5 Registers within AVR I/O space:

Table 9-24. Security Module Address Space Overview

Register Name	Description
AES_STATUS	AES status register
AES_CTRL	AES control register
AES_KEY	Access to 16 Byte key buffer
AES_STATE	Access to 16 Byte data buffer

9.8.8.2 Security Module Preparation

The use of the security module requires a configuration of the security engine before starting a security operation. The following steps are required:

Table 9-25. AES Engine Configuration Steps

Step	Description	Description
1	Key Setup	Write encryption or decryption key to KEY buffer (16 consecutive byte writes to AES_KEY)

Step	Description	Description
2	AES configuration	Select AES mode: ECB or CBC Select encryption or decryption Enable the AES Encryption Ready Interrupt AES_READY
3	Write Data	Write plaintext or cipher text to DATA buffer (16 consecutive byte writes to AES_STATE)
4	Start operation	Start AES operation
5	Wait for AES finished: 1. AES_READY IRQ or 2. polling AES_DONE bit (register AES_STATUS) or 3. wait for 24 µs	Wait until AES encryption/decryption is finished successfully
6	Read Data	Read cipher text or plaintext from DATA buffer (16 consecutive byte reads from AES_STATE)

Before starting any security operation a 16 Byte key must be written to the security engine (refer to section "Security Key Setup" below). This can be done by 16 consecutive write accesses to the I/O register AES_KEY. An internal address counter is incremented automatically with every read/ write operation. An AES encryption/ decryption run resets the internal byte counter. If the key and data buffer has not been read or written completely (all 16 Bytes), the following encryption/ decryption operation will finish with an error.

The following step selects either Electronic Code Book (ECB) or Cipher Block Chaining (CBC) as the AES_MODE. These modes are explained in more detail in section "Security Operation Modes" on page 94. Encryption or decryption must be further selected with bit AES_DIR of register AES_CTRL.

If the AES Error or AES Ready IRQ is used, the interrupt must be enabled with bit AES_IM.

Next the 128-bit plain text or cipher text data has to be provided to the AES hardware engine. The 16 data bytes must be consecutively written to the AES_STATE register. The AES_STATE register can be accessed in the same way as the key register (refer to "Security Key Setup" below).

The encryption or decryption is initiated with bit AES_REQUEST = 1.

The operation takes 24 µs and the completed encryption/ decryption is indicated by the AES_READY IRQ and the AES_DONE bit. The internal byte counter of the key and data buffer is cleared and the resulting data can be read out.

For additional information about the key and data buffer please refer to section "AES_KEY – AES Encryption and Decryption Key Buffer Register" on page 102 and "AES_STATE – AES Plain and Cipher Text Buffer Register" on page 102.

Notes:
1. Access to the security block is not possible while the radio transceiver is in state SLEEP.
2. All configurations of the security module, the SRAM content and keys are reset during SLEEP or RESET states.

9.8.8.3 Security Key Setup

The key is stored in a 16 Byte sequential buffer. To read or write the contents of the buffer, 16 consecutive read or write operations to the AES_KEY register are required.

A 16-folded read access to registers AES_KEY returns the last round key of the preceding security operation. This is the key required for the corresponding ECB decryption operation after an ECB encryption operation. However the initial AES key written to the security module in advance of an AES run (see step 1 in Table 9-25 on page 92) is not modified during an AES operation. This initial key is used for the next AES run although it cannot be read from AES_KEY.

Before accessing the Key Buffer it must be ensured, that the internal address counter is initialized correctly. This is the cases after Radio Transceiver Reset (see TRXPR – Transceiver Pin Register on page 169) or a completed AES Encryption/ Decryption operation. After an interrupted buffer read or write access, Address pointer reinitialization is recommended by a simple read access to the AES_CTRL register.

Note: 1. ECB decryption is not required for IEEE 802.15.4 or ZigBee security processing. The radio transceiver provides this functionality as an additional feature.

9.8.8.4 Security Operation Modes

9.8.8.4.1 Electronic Code Book (ECB)

ECB is the basic operating mode of the security module and is configured by the AES_CTRL register. The bit AES_MODE = 0 defines the ECB mode and bit AES_DIR selects the direction to either encryption or decryption. The data to be processed has to be written to registers AES_STATE.

A security operation can be started by writing the start command AES_REQUEST = 1 (AES_CTRL register).

The ECB encryption operation is illustrated in Figure 9-35 below. Figure 9-36 below shows the ECB decryption mode which is supported in a similar way.

Figure 9-35. ECB Mode - Encryption

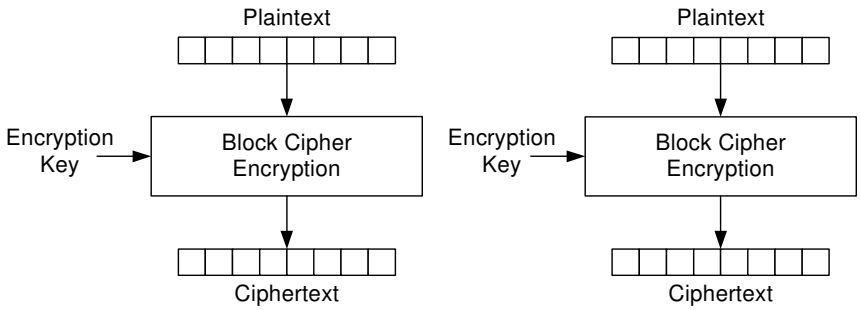
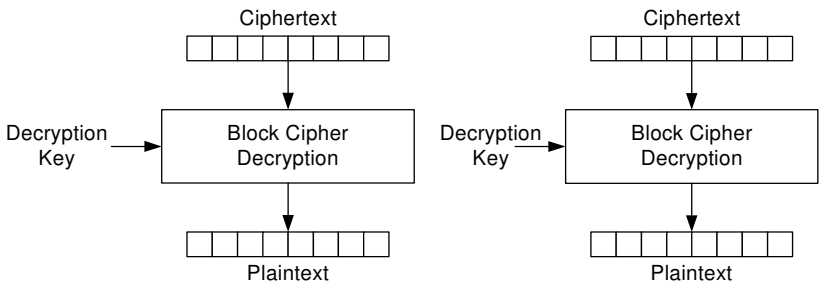


Figure 9-36. ECB Mode - Decryption



Due to the nature of AES algorithm the initial key to be used when decrypting is not the same as the one used for encryption. Instead it is the last round key. This last round

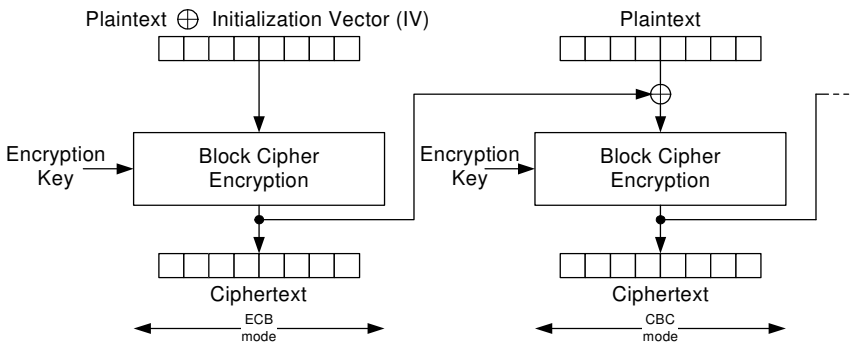
key is the content of the key address space stored after running one full encryption cycle and must be saved for decryption. If the decryption key has not been saved, it has to be recomputed by first running a dummy encryption (of an arbitrary plaintext) using the original encryption key. Then the resulting round key must be fetched from the key memory and written back into the key memory as the decryption key.

ECB decryption is not used by either IEEE 802.15.4 or ZigBee frame security. Both of these standards do not directly encrypt the payload. Instead they protect the payload by applying a XOR operation between the original payload and the resulting (AES-) cipher text with a nonce (number used once). As the nonce is the same for encryption and decryption only ECB encryption is required. Decryption is performed by a XOR operation between the received cipher text and its own encryption result concluding in the original plain text payload upon success.

9.8.8.4.2 Cipher Block Chaining (CBC)

In CBC mode the result of a previous AES operation is XOR-combined with the new incoming vector forming the new plaintext to encrypt as shown in the next figure. This mode is used for the computation of a cryptographic checksum (message integrity code, MIC).

Figure 9-37. CBC Mode - Encryption



After preparing the AES key and defining the AES operation direction register bit AES_DIR, the data has to be provided to the AES engine and the CBC operation can be started.

The first CBC run has to be configured as ECB to process the initial data (plain text XOR with an initialization vector provided by the application software). All succeeding AES runs are to be configured as CBC by setting bit AES_MODE = 1 (AES_CTRL register). Bit AES_DIR (AES_CTRL register) must be set to AES_DIR = 0 to enable AES encryption. The data to be processed has to be transferred to the AES_STATE register. Setting bit AES_REQUEST = 1 (AES_CTRL register) as described in section "Security Operation Modes" on page 94 starts the first encryption. This causes the next 128 bits of plain text data to be XORed with the previous cipher text data, see Figure 9-37 above.

According to IEEE 802.15.4 the input for the very first CBC operation has to be prepared by a XOR operation of the plain text with the initialization vector (IV). The value of the initialization vector is 0. However any other initialization vector can be applied for non-compliant usage. This operation has to be prepared by the application software.

Note that the MIC algorithm of the IEEE 802.15.4-2006 standard requires CBC mode encryption only because it implements a one-way hash function.

The status of the security processing is indicated by register AES_STATUS. After a AES processing time of 24 µs the register bit AES_DONE changes to 1 (register AES_STATUS) indicating that the security operation has finished (see "Digital Interface Timing Characteristics" on page 507).

The end of the AES processing can also be indicated by the AES_READY Interrupt. The bit AES_ER of register AES_STATUS is set if the operation has finished with an error. Otherwise this bit is zero but AES_DONE is '1'.

9.8.8.5 AES Interrupt Handling

The AES Interrupt handling is slightly different from all other IRQ's. If the AES_IM Bit (AES_CTRL Register) and the global interrupt enable flag is set, the AES core can generate an AES Ready Interrupt (AES_READY). If the IRQ is issued, the AES_STATUS register must be read to check the finish status of the last operation. If AES_DONE is set, the last AES operation finished successfully. If AES_ER is set, an error occurred during the last operation. The AES_ER flag is cleared automatically during the read access to the AES_STATUS register. The AES_DONE flag is cleared during the next read or write access to the AES_STATE (AES data) register.

The two status flags must be cleared before a new Interrupt can be issued.

If AES_IM is not set, the processing status can be polled by software (AES_STATUS register), but no Interrupt occurs.

9.9 Continuous Transmission Test Mode

9.9.1 Overview

The 2.4GHz transceiver offers a Continuous Transmission Test Mode to support final application / production tests as well as certification tests. In this test mode the radio transceiver transmits continuously a previously transferred frame (PRBS mode) or a continuous wave signal (CW mode).

In CW mode two different signal frequencies per channel can be transmitted:

- $f_1 = f_{CH} + 0.5$ MHz
- $f_2 = f_{CH} - 0.5$ MHz

Here f_{CH} is the channel center frequency programmed by register PHY_CC_CCA.

Note that in CW mode it is not possible to transmit a RF signal directly on the channel center frequency. PSDU data in the Frame Buffer must contain at least a valid PHR (see section "Introduction – IEEE 802.15.4-2006 Frame Format" on page 61). It is recommended to use a frame of maximum length (127 bytes) and arbitrary PSDU data for the PRBS mode. The SHR and the PHR are not transmitted. The transmission starts with the PSDU data and is repeated continuously.

9.9.2 Configuration

All register configurations shall be setup as follows before enabling Continuous Transmission Test Mode:

- TX channel setting (optional);
- TX output power setting (optional);
- Mode selection (PRBS / CW);

An access to the registers TST_CTRL_DIGI and PART_NUM enables the Continuous Transmission Test Mode.

The transmission is started by enabling the PLL (TRX_CMD = PLL_ON) and writing the TX_START command to register TRX_STATE.

Even for CW signal transmission it is required to write valid PSDU data to the Frame Buffer. For PRBS mode it is recommended to write a frame of maximum length.

The detailed programming sequence is shown in Table 9-26 below. The column R/W informs about writing (W) or reading (R) a register or the Frame Buffer.

Table 9-26. Continuous Transmission Programming Sequence

Step	Action	Register	R/W	Value	Description
1	RESET				Reset the transceiver
2	Register Access	IRQ_MASK	W	0x01	Set IRQ mask register, enable PLL_LOCK interrupt and set global AVR IRQ enable
3	Register Access	TRX_CTRL_1	W	0x00	Disable TX_AUTO_CRC_ON
4	Register Access	TRX_STATE	W	0x03	Set radio transceiver state TRX_OFF
5	Register Access	PHY_CC_CCA	W	0x33	Set IEEE 802.15.4 CHANNEL, e.g. 19
6	Register Access	PHY_TX_PWR	W	0x00	Set TX output power, e.g. to P_{max}
7	Register Access	TRX_STATUS	R	0x08	Verify TRX_OFF state
8	Register Access	TST_CTRL_DIGI	W	0x0F	Enable Continuous Transmission Test Mode – step # 1
9(1)	Register Access	TRX_CTRL_2	W	0x03	Enable High Data Rate Mode, 2 Mb/s
10(1)	Register Access	RX_CTRL	W	0xA7	Configure High Data Rate Mode
11(2)	Frame Buffer Write Access		W		Write PSDU data (even for CW mode), refer to Table 9-27 on page 98
12	Register Access	PART_NUM	W	0x54	Enable Continuous Transmission Test Mode – step # 2
13	Register Access	PART_NUM	W	0x46	Enable Continuous Transmission Test Mode – step # 3
14	Register Access	TRX_STATE	W	0x09	Enable PLL_ON state
15	Interrupt event		R	0x01	Wait for PLL_LOCK interrupt
16	Register Access	TRX_STATE	W	0x02	Initiate Transmission, enter BUSY_TX state
17	Measurement				Perform measurement
18	Register Access	TRX_CTRL_2	W	0x00	Disable Continuous Transmission Test Mode
19	RESET				Reset the transceiver

Notes: 1. Only required for CW mode, do not configure for PRBS mode.
2. Frame Buffer content varies for different modulation schemes.

The content of the Frame Buffer has to be defined for Continuous Transmission PRBS mode or CW mode. To measure the power spectral density (PSD) mask of the transmitter it is recommended to use a random sequence of maximum length for the PSDU data.

To measure CW signals it is necessary to write either 0x00 or 0xFF to the Frame Buffer. For details refer to Table 9-27 below.

Table 9-27. Frame Buffer Content for various Continuous Transmission Modulation Schemes

Step	Action	Frame Content	Comment
11	Frame Buffer Write Access	Random Sequence	modulated RF signal
		0x00 (each byte)	f_{CH} – 0.5 MHz, CW signal
		0xFF (each byte)	f_{CH} + 0.5 MHz, CW signal

9.10 Abbreviations

AACK	-	Automatic acknowledgement
ACK	-	Acknowledgement
ADC	-	Analog-to-digital converter
AD	-	Antenna diversity
AGC	-	Automated gain control
AES	-	Advanced encryption standard
ARET	-	Automatic retransmission
AVREG	-	Voltage regulator for analog building blocks
AWGN	-	Additive White Gaussian Noise
BATMON	-	Battery monitor
BBP	-	Base band processor
BPF	-	Band pass filter
CBC	-	Cipher block chaining
CRC	-	Cyclic redundancy check
CCA	-	Clear channel assessment
CSMA-CA	-	Carrier sense multiple access/Collision avoidance
CW	-	Continuous wave
DVREG	-	Voltage regulator for digital building blocks
ECB	-	Electronic code book
ED	-	Energy detection
ESD	-	Electro static discharge
EVM	-	Error vector magnitude
FCF	-	Frame control field
FCS	-	Frame check sequence
FIFO	-	First in first out
FTN	-	Filter tuning network
GPIO	-	General purpose input output

ISM	-	Industrial, scientific, and medical
LDO	-	Low-drop output
LNA	-	Low-noise amplifier
LO	-	Local oscillator
LQI	-	Link quality indicator
LSB	-	Least significant bit
MAC	-	Medium access control
MFR	-	MAC footer
MHR	-	MAC header
MSB	-	Most significant bit
MSDU	-	MAC service data unit
MPDU	-	MAC protocol data unit
MSK	-	Minimum shift keying
O-QPSK	-	Offset - quadrature phase shift keying
PA	-	Power amplifier
PAN	-	Personal area network
PCB	-	Printed circuit board
PER	-	Packet error rate
PHR	-	PHY header
PHY	-	Physical layer
PLL	-	Phase locked loop
POR	-	Power-on reset
PPF	-	Poly-phase filter
PRBS	-	Pseudo random bit sequence
PSDU	-	PHY service data unit
PSD	-	Power spectral mask
QFN	-	Quad flat no-lead package
RF	-	Radio frequency
RSSI	-	Received signal strength indicator
RX	-	Receiver
SFD	-	Start-of-frame delimiter
SHR	-	Synchronization header
SPI	-	Serial peripheral interface
SRAM	-	Static random access memory
SSBF	-	Single side band filter
TX	-	Transmitter

VCO	-	Voltage controlled oscillator
VREG	-	Voltage regulator
XOSC	-	Crystal oscillator

9.11 Reference Documents

[1] IEEE Std 802.15.4™-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)

[2] IEEE Std 802.15.4™-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)

[3] ANSI / ESD-STM5.1-2001: ESD Association Standard Test Method for electrostatic discharge sensitivity testing – Human Body Model (HBM).

[4] ESD-STM5.3.1-1999: ESD Association Standard Test Method for electrostatic discharge sensitivity testing – Charged Device Model (CDM).

[5] NIST FIPS PUB 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/NIST, November 26, 2001

[6] ATmega128RFA1 Software Programming Model

9.12 Register Description

9.12.1 AES_CTRL – AES Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($13C)	AES_REQUEST	Res	AES_MODE	Res	AES_DIR	AES_IM	Res1	Res0	AES_CTRL
Read/Write	RW	R	RW	R	RW	RW	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register controls the operation of the security module. Do not access this register during AES operation to read the AES core status. A read or write access to the register stops the ongoing processing. To read the AES status use bit AES_DONE of register AES_STATUS. Note that the AES_CTRL register is cleared when entering the radio transceiver SLEEP state.

- **Bit 7 – AES_REQUEST - Request AES Operation.**

A write access with AES_REQUEST = 1 initiates the AES operation.

- **Bit 6 – Res - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 5 – AES_MODE - Set AES Operation Mode**

This register bit sets the AES operation mode (ECB/CBC Mode).

Table 9-28 AES_MODE Register Bits

Register Bits	Value	Description
AES_MODE	0	AES Mode is ECB (Electronic Code Book).
	1	AES Mode is CBC (Cipher Block Chaining).

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 3 – AES_DIR - Set AES Operation Direction**

This register bit sets the AES operation direction to either encryption or decryption.

Table 9-29 AES_DIR Register Bits

Register Bits	Value	Description
AES_DIR	0	AES operation is encryption.
	1	AES operation is decryption.

- **Bit 2 – AES_IM - AES Interrupt Enable**

This register bit is used to enable the AES interrupt.

- **Bit 1:0 – Res1:0 - Reserved Bit**

These bits are reserved for future use. The result of a read access is undefined. The register bits must always be written with the reset value.

9.12.2 AES_STATUS – AES Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($13D)	AES_ER	Res5	Res4	Res3	Res2	Res1	Res0	AES_DONE	AES_STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This read-only register signals the status of the security module and operation. Note that the AES_STATUS register is cleared when entering the radio transceiver SLEEP state.

- **Bit 7 – AES_ER - AES Operation Finished with Error**

This register bit indicates an error during AES module run. An error occurs if accessing AES_CTRL while an AES operation is running or if AES_KEY or AES_STATE Memory is not loaded completely or less than 16 Byte read from AES_STATE.

- **Bit 6:1 – Res5:0 - Reserved**

These bits are reserved for future use.

- **Bit 0 – AES_DONE - AES Operation Finished with Success**

This register bit indicates a successfully finished operation of the AES module.

9.12.3 AES_STATE – AES Plain and Cipher Text Buffer Register

Bit	7	6	5	4	3	2	1	0	
NA ($13E)	AES_STATE7:0								AES_STATE
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The AES_STATE register accesses a 16 byte internal data buffer. The buffer is accessed by reading or writing 16 times to the same address location (AES_STATE). If the buffer is not completely read or written an error occurs when an AES operation is started. Note that the AES_STATE register is cleared when entering the radio transceiver SLEEP state.

- **Bit 7:0 – AES_STATE7:0 - AES Plain and Cipher Text Buffer**

These bits represent the data buffer for the AES operation.

9.12.4 AES_KEY – AES Encryption and Decryption Key Buffer Register

Bit	7	6	5	4	3	2	1	0	
NA ($13F)	AES_KEY7:0								AES_KEY
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The AES key register accesses a 128 Bit internal buffer that holds the Encryption or Decryption Key. The AES_KEY buffer is a 16 Byte buffer. The buffer is accessed by reading or writing 16 fold to the same address location (AES_KEY). A read access to registers AES_KEY returns the last round key of the preceding security operation. This is the key that is required for the corresponding ECB decryption operation after an ECB encryption operation. However, the initial AES key written to the security module in advance of an AES run is not modified during an AES operation. This initial key is used for the next AES run even if it cannot be read from AES_KEY register. Note that the AES_KEY register is cleared when entering the radio transceiver SLEEP state.

- **Bit 7:0 – AES_KEY7:0 - AES Encryption/Decryption Key Buffer**

These bits represent the data buffer for the AES Encryption/Decryption key.

9.12.5 TRX_STATUS – Transceiver Status Register

Bit	7	6	5	4	
NA ($141)	CCA_DONE	CCA_STATUS	TST_STATUS	TRX_STATUS4	TRX_STATUS
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($141)	TRX_STATUS3	TRX_STATUS2	TRX_STATUS1	TRX_STATUS0	TRX_STATUS
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	

This read-only register signals the present state of the radio transceiver as well as the status of the CCA operation. A state change is initiated by writing a state transition

command to the TRX_CMD bits of register TRX_STATE. The register is not accessible in SLEEP state.

- **Bit 7 – CCA_DONE - CCA Algorithm Status**

This bit indicates if a CCA request is completed. This is also indicated by a TRX24_CCA_ED_DONE interrupt. Note that register bit CCA_DONE is cleared in response to a CCA_REQUEST.

Table 9-30 CCA_DONE Register Bits

Register Bits	Value	Description
CCA_DONE	0	CCA calculation not finished
	1	CCA calculation finished

- **Bit 6 – CCA_STATUS - CCA Status Result**

The result of the CCA measurement is available in register bit CCA_STATUS after a CCA request is completed. Note that register bit CCA_STATUS is cleared in response to a CCA_REQUEST.

Table 9-31 CCA_STATUS Register Bits

Register Bits	Value	Description
CCA_STATUS	0	Channel indicated as busy.
	1	Channel indicated as idle.

- **Bit 5 – TST_STATUS - Test mode status**

This bit is reserved for internal use. It indicates the status of the test mode.

Table 9-32 TST_STATUS Register Bits

Register Bits	Value	Description
TST_STATUS	0	Test mode is disabled.
	1	Test mode is active.

- **Bit 4:0 – TRX_STATUS4:0 - Transceiver Main Status**

The register bits TRX_STATUS signal the current radio transceiver status. Do not try to initiate a further state change while the radio transceiver is in STATE_TRANSITION_IN_PROGRESS state. Values not listed in the following table are reserved.

Table 9-33 TRX_STATUS Register Bits

Register Bits	Value	Description
TRX_STATUS4:0		
	0x01	BUSY_RX
	0x02	BUSY_TX
	0x06	RX_ON
	0x08	TRX_OFF
	0x09	PLL_ON
	0x0F	SLEEP
	0x11	BUSY_RX_AACK
	0x12	BUSY_TX_ARET
	0x16	RX_AACK_ON
	0x19	TX_ARET_ON

Register Bits	Value	Description
	0x1F	STATE_TRANSITION_IN_PROGRESS

9.12.6 TRX_STATE – Transceiver State Control Register

Bit	7	6	5	4	
NA ($142)	TRAC_STATUS2	TRAC_STATUS1	TRAC_STATUS0	TRX_CMD4	TRX_STATE
Read/Write	R	R	R	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($142)	TRX_CMD3	TRX_CMD2	TRX_CMD1	TRX_CMD0	TRX_STATE
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

The states of the radio transceiver are controlled via register TRX_STATE using register bits TRX_CMD. The read-only register bits TRAC_STATUS indicate the status or result of an Extended Operating Mode transaction. A successful state transition shall be confirmed by reading register bits TRX_STATUS. This register is used for both Basic and Extended Operating Mode.

- **Bit 7:5 – TRAC_STATUS2:0 - Transaction Status**

The status of the RX_AACK and TX_ARET procedure is indicated by register bits TRAC_STATUS. TRAC_STATUS is only valid in Extended Operating Modes. Details of the algorithm and a description of the status information are given in the RX_AACK_ON and TX_ARET_ON sections of the data-sheet. Even though the reset value for register bits TRAC_STATUS is 0, the RX_AACK and TX_ARET procedures set the register bits to TRAC_STATUS = 7 (INVALID) when it is started. Not all status values are used in both RX_AACK and TX_ARET transactions. In TX_ARET the status SUCCESS_DATA_PENDING indicates a successful reception of an ACK frame with frame pending bit set to 1. In RX_AACK the status SUCCESS_WAIT_FOR_ACK indicates an ACK frame is about to sent in RX_AACK slotted acknowledgment. Slotted acknowledgment operation must be enabled with the SLOTTED_OPERATION bit of register XAH_CTRL_0. The application software must set the SLPTR bit of register TRXPWR at the next back-off slot boundary in order to initiate a transmission of the ACK frame. For details refer to IEEE 802.15.4-2006, chapter 5.5.4.1. Values not listed in the following table are reserved.

Table 9-34 TRAC_STATUS Register Bits

Register Bits	Value	Description
TRAC_STATUS2:0	0	SUCCESS (RX_AACK, TX_ARET)
	1	SUCCESS_DATA_PENDING (TX_ARET)
	2	SUCCESS_WAIT_FOR_ACK (RX_AACK)
	3	CHANNEL_ACCESS_FAILURE (TX_ARET)
	5	NO_ACK (TX_ARET)
	7	INVALID (RX_AACK, TX_ARET)

- **Bit 4:0 – TRX_CMD4:0 - State Control Command**

A write access to register bits TRX_CMD initiates a state transition of the radio transceiver towards the new state as defined by the write access. Do not try to initiate a further state change while the radio transceiver is in STATE_TRANSITION_IN_PROGRESS state (see TRX_STATUS register). FORCE_PLL_ON is not valid for the SLEEP state as well as during STATE_TRANSITION_IN_PROGRESS towards the SLEEP state. Values not listed in the following table are reserved and mapped to NOP.

Table 9-35 TRX_CMD Register Bits

Register Bits	Value	Description
TRX_CMD4:0	0x00	NOP
	0x02	TX_START
	0x03	FORCE_TRX_OFF
	0x04	FORCE_PLL_ON
	0x06	RX_ON
	0x08	TRX_OFF
	0x09	PLL_ON (TX_ON)
	0x16	RX_AACK_ON
	0x19	TX_ARET_ON

9.12.7 TRX_CTRL_0 – Reserved

Bit	7	6	5	4	3	2	1	0	
NA ($143)	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	TRX_CTRL_0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	1	1	0	0	1	

This register is reserved for future use.

- **Bit 7:0 – Res7:0 - Reserved**

These bits are reserved for future use.

9.12.8 TRX_CTRL_1 – Transceiver Control Register 1

Bit	7	6	5	4	
NA ($144)	PA_EXT_EN	IRQ_2_EXT_EN	TX_AUTO_CRC_ON	Res4	TRX_CTRL_1
Read/Write	RW	RW	RW	R	
Initial Value	0	0	1	0	
Bit	3	2	1	0	
NA ($144)	Res3	Res2	Res1	Res0	TRX_CTRL_1
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	

The TRX_CTRL_1 register is a multi purpose register to control various operating modes and settings of the radio transceiver.

- **Bit 7 – PA_EXT_EN - External PA support enable**

This register bit enables pin DIG3 and pin DIG4 to indicate the transmit state of the radio transceiver. The control of the external RF front-end is disabled when this bit is 0. Both pins DIG3 and DIG4 are then low. The control of the external front-end is enabled when this bit is 1. DIG3 and DIG4 then indicate the state of the radio transceiver. Pin DIG3 is high and pin DIG4 is low in the state TX_BUSY. In all other states pin DIG3 is low and pin DIG4 is high. It is recommended to set PA_EXT_EN=1 only in receive or transmit states to reduce the power consumption or avoid leakage current of external RF switches or other building blocks especially during SLEEP state.

- **Bit 6 – IRQ_2_EXT_EN - Connect Frame Start IRQ to TC1**

When this bit is set to one the capture input of Timer/Counter 1 is connected to the RX frame start signal and pin DIG2 becomes an output, driving the RX frame start signal. Antenna Diversity RF switch control (ANT_EXT_SW_EN=1) shall not be used at the same time, because it shares the same device pin. The function IRQ_2_EXT_EN is available for alternate frame time stamping using Timer/Counter 1. In general the preferred method for frame time stamping is using the symbol counter.

- **Bit 5 – TX_AUTO_CRC_ON - Enable Automatic CRC Calculation**

This register bit controls the automatic FCS generation for TX operations. The automatic FCS algorithm is performed autonomously by the radio transceiver if register bit TX_AUTO_CRC_ON=1.

- **Bit 4:0 – Res4:0 - Reserved**

9.12.9 PHY_TX_PWR – Transceiver Transmit Power Control Register

Bit	7	6	5	4	
NA ($145)	PA_BUF_LT1	PA_BUF_LT0	PA_LT1	PA_LT0	PHY_TX_PWR
Read/Write	RW	RW	RW	RW	
Initial Value	1	1	0	0	
Bit	3	2	1	0	
NA ($145)	TX_PWR3	TX_PWR2	TX_PWR1	TX_PWR0	PHY_TX_PWR
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register controls the output power and the ramping of the transmitter.

- **Bit 7:6 – PA_BUF_LT1:0 - Power Amplifier Buffer Lead Time**

These register bits control the enable lead time of the internal PA buffer relative to the enable time of the internal PA. This time is further used to derive a control signal for an external RF front-end to switch between receive and transmit.

Table 9-36 PA_BUF_LT Register Bits

Register Bits	Value	Description
PA_BUF_LT1:0	0	0 µs
	1	2 µs
	2	4 µs
	3	6 µs

- **Bit 5:4 – PA_LT1:0 - Power Amplifier Lead Time**

These register bits control the enable lead time of the internal power amplifier relative to the beginning of the transmitted frame (SHR).

Table 9-37 PA_LT Register Bits

Register Bits	Value	Description
PA_LT1:0	0	2 µs
	1	4 µs
	2	6 µs
	3	8 µs

- **Bit 3:0 – TX_PWR3:0 - Transmit Power Setting**

These register bits determine the TX output power of the radio transceiver.

Table 9-38 TX_PWR Register Bits

Register Bits	Value	Description
TX_PWR3:0	0	3.0 dBm
	1	2.8 dBm
	2	2.3 dBm
	3	1.8 dBm
	4	1.3 dBm
	5	0.7 dBm
	6	0.0 dBm
	7	-1 dBm
	8	-2 dBm
	9	-3 dBm
	10	-4 dBm
	11	-5 dBm
	12	-7 dBm
	13	-9 dBm
	14	-12 dBm
	15	-17 dBm

9.12.10 PHY_RSSI – Receiver Signal Strength Indicator Register

Bit	7	6	5	4	
NA ($146)	RX_CRC_VALID	RND_VALUE1	RND_VALUE0	RSSI4	PHY_RSSI
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($146)	RSSI3	RSSI2	RSSI1	RSSI0	PHY_RSSI
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	

The PHY_RSSI register is a multi purpose register that indicates FCS validity, provides random numbers and shows the current RSSI value.

- **Bit 7 – RX_CRC_VALID - Received Frame CRC Status**

Reading this register bit indicates whether the last received frame has a valid FCS or not. The register bit is updated when issuing a TRX24_RX_END interrupt and remains

valid until the next TRX24_RX_END interrupt is issued, caused by a new frame reception.

Table 9-39 RX_CRC_VALID Register Bits

Register Bits	Value	Description
RX_CRC_VALID	0	CRC (FCS) not valid
	1	CRC (FCS) valid

- **Bit 6:5 – RND_VALUE1:0 - Random Value**

A 2-bit random value can be retrieved by reading register bits RND_VALUE. The value can be used for random numbers for security applications. Note that the radio transceiver shall be in Basic Operating Mode receive state. The values are updated every 1 µs.

- **Bit 4:0 – RSSI4:0 - Receiver Signal Strength Indicator**

The result of the automated RSSI measurement is stored in these register bits. The value is updated every 2µs in receive states. The read value is a number between 0 and 28 indicating the received signal strength as a linear curve on a logarithmic input power scale (dBm) with a resolution of 3 dB. A RSSI value of 0 indicates a RF input power lower than RSSI_BASE_VAL (-90 dBm). A value of 28 marks a power higher or equal to -10 dBm.

Table 9-40 RSSI Register Bits

Register Bits	Value	Description
RSSI4:0	0	Minimum RSSI value: P(RF) < -90 dBm
	1	P(RF) = RSSI_BASE_VAL+3 · (RSSI-1) [dBm]
	2	...
	28	Maximum RSSI value: P(RF) ≥ -10 dBm

9.12.11 PHY_ED_LEVEL – Transceiver Energy Detection Level Register

Bit	7	6	5	4	
NA ($147)	ED_LEVEL7	ED_LEVEL6	ED_LEVEL5	ED_LEVEL4	PHY_ED_LEVEL
Read/Write	R	R	R	R	
Initial Value	1	1	1	1	
Bit	3	2	1	0	
NA ($147)	ED_LEVEL3	ED_LEVEL2	ED_LEVEL1	ED_LEVEL0	PHY_ED_LEVEL
Read/Write	R	R	R	R	
Initial Value	1	1	1	1	

This register contains the result of an Energy Detection measurement.

- **Bit 7:0 – ED_LEVEL7:0 - Energy Detection Level**

The minimum ED value (ED_LEVEL = 0) indicates a receiver power less than or equal to RSSI_BASE_VAL. The range is 84 dB with a resolution of 1 dB and an absolute accuracy of ±5 dB. A manual ED measurement can be initiated by a write access to this register. A value of 0xFF signals that no measurement has yet been started (reset value). The measurement duration is 8 symbol periods (128 µs) for a data rate of 250 kb/s. For High Data Rate Modes the automated measurement duration is reduced to 32

µs. For manually initiated ED measurements in these modes the measurement period is still 128 µs as long as the receiver is in RX_ON state. A value other than 0xFF indicates the result of the last ED measurement.

Table 9-41 ED_LEVEL Register Bits

Register Bits	Value	Description
ED_LEVEL7:0	0x00	Minimum result of last ED measurement
	0x01	P(RF) = RSSI_BASE_VAL+ED [dBm]
	0x02	...
	0x54	Maximum result of last ED measurement
	0xFF	Reset value

9.12.12 PHY_CC_CCA – Transceiver Clear Channel Assessment (CCA) Control Register

Bit	7	6	5	4	
NA ($148)	CCA_REQUEST	CCA_MODE1	CCA_MODE0	CHANNEL4	PHY_CC_CCA
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	1	0	
Bit	3	2	1	0	
NA ($148)	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0	PHY_CC_CCA
Read/Write	RW	RW	RW	RW	
Initial Value	1	0	1	1	

This register is provided to initiate and control a CCA measurement.

- **Bit 7 – CCA_REQUEST - Manual CCA Measurement Request**

A manual CCA measurement is initiated with setting CCA_REQUEST=1. The end of the CCA measurement is indicated by the TRX24_CCA_ED_DONE interrupt. Register bits CCA_DONE and CCA_STATUS of register TRX_STATUS are updated after a CCA_REQUEST. The register bit is automatically cleared after requesting a CCA measurement with CCA_REQUEST=1.

- **Bit 6:5 – CCA_MODE1:0 - Select CCA Measurement Mode**

The CCA mode can be selected using these register bits. Note that IEEE 802.15.4-2006 CCA Mode 3 defines the logical combination of CCA Mode 1 and 2 with the logical operators AND or OR. This can be selected with CCA_MODE=0 for logical operation OR and CCA_MODE=3 for logical operation AND.

Table 9-42 CCA_MODE Register Bits

Register Bits	Value	Description
CCA_MODE1:0	0	Mode 3a, Carrier sense OR energy above threshold
	1	Mode 1, Energy above threshold
	2	Mode 2, Carrier sense only
	3	Mode 3b, Carrier sense AND energy above threshold

- **Bit 4:0 – CHANNEL4:0 - RX/TX Channel Selection**

These register bits define the RX/TX channel. The channel assignment is according to IEEE 802.15.4.

Table 9-43 CHANNEL Register Bits

Register Bits	Value	Description
CHANNEL4:0	11	2405 MHz
	12	2410 MHz
	13	2415 MHz
	14	2420 MHz
	15	2425 MHz
	16	2430 MHz
	17	2435 MHz
	18	2440 MHz
	19	2445 MHz
	20	2450 MHz
	21	2455 MHz
	22	2460 MHz
	23	2465 MHz
	24	2470 MHz
	25	2475 MHz
	26	2480 MHz

9.12.13 CCA_THRES – Transceiver CCA Threshold Setting Register

Bit	7	6	
NA ($149)	CCA_CS_THRES3	CCA_CS_THRES2	CCA_THRES
Read/Write	RW	RW	
Initial Value	1	1	
Bit	5	4	
NA ($149)	CCA_CS_THRES1	CCA_CS_THRES0	CCA_THRES
Read/Write	RW	RW	
Initial Value	0	0	
Bit	3	2	
NA ($149)	CCA_ED_THRES3	CCA_ED_THRES2	CCA_THRES
Read/Write	RW	RW	
Initial Value	0	1	
Bit	1	0	
NA ($149)	CCA_ED_THRES1	CCA_ED_THRES0	CCA_THRES
Read/Write	RW	RW	
Initial Value	1	1	

This register sets the threshold level for the Energy Detection (ED) of the Clear Channel Assessment (CCA).

- **Bit 7:4 – CCA_CS_THRES3:0 - CS Threshold Level for CCA Measurement**

These bits are reserved for internal use.

- **Bit 3:0 – CCA_ED_THRES3:0 - ED Threshold Level for CCA Measurement**

These bits define the received power threshold of the Energy above threshold algorithm. The threshold is calculated by RSSI_BASE_VAL + 2CCA_ED_THRES [dBm]. Any received power above this level is interpreted as a busy channel.

9.12.14 RX_CTRL – Transceiver Receive Control Register

Bit	7	6	5	4	
NA ($14A)	Resx7	Resx6	Resx5	Resx4	RX_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	1	0	1	1	
Bit	3	2	1	0	
NA ($14A)	PDT_THRES3	PDT_THRES2	PDT_THRES1	PDT_THRES0	RX_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	0	1	1	1	

The register controls the sensitivity of the Antenna Diversity Mode. Note that in High Data Rate modes the ACR module will always be disabled.

- **Bit 7:4 – Resx7:4 - Reserved**
- **Bit 3:0 – PDT_THRES3:0 - Receiver Sensitivity Control**

These register bits control the sensitivity of the receiver correlation unit. If the Antenna Diversity algorithm is enabled the value shall be set to PDT_THRES = 3. Otherwise it shall be set back to the reset value. Values not listed in the following table are reserved.

Table 9-44 PDT_THRES Register Bits

Register Bits	Value	Description
PDT_THRES3:0	0x7	Reset value, to be used if Antenna Diversity algorithm is disabled
	0x3	Recommended correlator threshold for Antenna Diversity operation

9.12.15 SFD_VALUE – Start of Frame Delimiter Value Register

Bit	7	6	5	4	3	2	1	0	
NA ($14B)	SFD_VALUE7:0								SFD_VALUE
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	0	1	0	0	1	1	1	

This register contains the one octet start-of-frame delimiter (SFD) to synchronize to a received frame. The lower 4 bits must not be all zero to avoid decoding conflicts.

- **Bit 7:0 – SFD_VALUE7:0 - Start of Frame Delimiter Value**

For compliant IEEE 802.15.4 networks set SFD_VALUE = 0xA7. This is the default value of the register. To establish non IEEE 802.15.4 compliant networks the SFD value can be changed to any other value. If enabled a RX_START interrupt is issued only if the received SFD matches the register content of SFD_VALUE and a valid PHR is received.

Table 9-45 SFD_VALUE Register Bits

Register Bits	Value	Description
SFD_VALUE7:0	0xA7	IEEE 802.15.4 compliant value of the SFD

9.12.16 TRX_CTRL_2 – Transceiver Control Register 2

Bit	7	6	5	4	
NA ($14C)	RX_SAFE_MODE	Res4	Res3	Res2	TRX_CTRL_2
Read/Write	RW	R	R	R	
Initial Value	0	0	0	0	

Bit	3	2	1	0	
NA ($14C)	Res1	Res0	OQPSK_DATA_RATE1	OQPSK_DATA_RATE0	TRX_CTRL_2
Read/Write	R	R	RW	RW	
Initial Value	0	0	0	0	

This register controls the data rate setting of the radio transceiver.

- **Bit 7 – RX_SAFE_MODE - RX Safe Mode**

If this bit is set, the next received frame will be protected and not overwritten by following frames. Set this bit to 0 to release the buffer (and set it again for further protection).

- **Bit 6:2 – Res4:0 - Reserved**
- **Bit 1:0 – OQPSK_DATA_RATE1:0 - Data Rate Selection**

A write access to these register bits sets the OQPSK PSDU data rate used by the radio transceiver. The reset value OQPSK_DATA_RATE = 0 is the PSDU data rate according to IEEE 802.15.4. All other values are used in High Data Rate Modes.

Table 9-46 OQPSK_DATA_RATE Register Bits

Register Bits	Value	Description
OQPSK_DATA_RATE1:0	0	250 kb/s (IEEE 802.15.4 compliant)
	1	500 kb/s
	2	1000 kb/s
	3	2000 kb/s

9.12.17 ANT_DIV – Antenna Diversity Control Register

Bit	7	6	5	4	
NA ($14D)	ANT_SEL	Res2	Res1	Res0	ANT_DIV
Read/Write	R	R	R	R	
Initial Value	0	0	0	0	

Bit	3	2	1	0	
NA ($14D)	ANT_DIV_EN	ANT_EXT_SW_EN	ANT_CTRL1	ANT_CTRL0	ANT_DIV
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	1	1	

This register controls the Antenna Diversity.

- **Bit 7 – ANT_SEL - Antenna Diversity Antenna Status**

This register bit signals the currently selected antenna path. The selection may be based either on the last antenna diversity cycle (ANT_DIV_EN = 1) or on the content of register bits ANT_CTRL.

Table 9-47 ANT_SEL Register Bits

Register Bits	Value	Description
ANT_SEL	0	Antenna 0
	1	Antenna 1

- **Bit 6:4 – Res2:0 - Reserved**
- **Bit 3 – ANT_DIV_EN - Enable Antenna Diversity**

If this register bit is set the Antenna Diversity algorithm is enabled. On reception of a frame the algorithm selects an antenna autonomously during SHR search. This selection is kept until

1. a new SHR search starts or

2. receive states are left or

3. a manually programming of bits ANT_CTRL occurred. If ANT_DIV_EN = 1 the bit ANT_EXT_SW_EN shall also be set to 1.

Table 9-48 ANT_DIV_EN Register Bits

Register Bits	Value	Description
ANT_DIV_EN	0	Antenna Diversity algorithm disabled
	1	Antenna Diversity algorithm enabled

- **Bit 2 – ANT_EXT_SW_EN - Enable External Antenna Switch Control**

If enabled, pin DIG1 and pin DIG2 become output pins and provide a differential control signal for an external Antenna Diversity switch. The selection of a specific antenna is done either by the automatic Antenna Diversity algorithm (ANT_DIV_EN = 1) or according to bits ANT_CTRL if the Antenna Diversity algorithm is disabled. Do not enable Antenna Diversity RF switch control (ANT_EXT_SW_EN = 1) and RX Frame Time Stamping (IRQ_2_EXT_EN = 1, see register TRX_CTRL_1) at the same time. If this bit is set the control pins DIG1/DIG2 are activated in all radio transceiver states as long as bit ANT_EXT_SW_EN is also set. If the radio transceiver is not in a receive or transmit state, it is recommended to disable bit ANT_EXT_SW_EN to reduce the power consumption or avoid leakage current of an external RF switch especially during SLEEP state. The output pins DIG1 and DIG2 are pulled-down to digital ground if bit ANT_EXT_SW_EN = 0.

Table 9-49 ANT_EXT_SW_EN Register Bits

Register Bits	Value	Description
ANT_EXT_SW_EN	0	Antenna Diversity RF switch control disabled
	1	Antenna Diversity RF switch control enabled

- **Bit 1:0 – ANT_CTRL1:0 - Static Antenna Diversity Switch Control**

These bits provide a static control of an Antenna Diversity switch. This register setting defines the selected antenna if ANT_DIV_EN is set to 0 (Antenna Diversity disabled). Register values 1 and 2 are valid for ANT_EXT_SW_EN = 1.

Table 9-50 ANT_CTRL Register Bits

Register Bits	Value	Description
ANT_CTRL1:0	0	Reserved
	1	Antenna 1: DIG1=H, DIG2=L

Register Bits	Value	Description
	2	Antenna 0: DIG1=L, DIG2=H
	3	Default value for ANT_EXT_SW_EN=0; Mandatory setting for applications not using Antenna Diversity

9.12.18 IRQ_MASK – Transceiver Interrupt Enable Register

Bit	7	6	5	4	
NA ($14E)	AWAKE_EN	TX_END_EN	AMI_EN	CCA_ED_DONE_EN	IRQ_MASK
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($14E)	RX_END_EN	RX_START_EN	PLL_UNLOCK_EN	PLL_LOCK_EN	IRQ_MASK
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register is used to enable or disable individual interrupts of the radio transceiver. An interrupt is enabled if the corresponding bit is set to 1. All interrupts are disabled after the power up sequence or reset. If an interrupt is enabled it is recommended to read the interrupt status register IRQ_STATUS first to clear the history.

- **Bit 7 – AWAKE_EN - Awake Interrupt Enable**
- **Bit 6 – TX_END_EN - TX_END Interrupt Enable**
- **Bit 5 – AMI_EN - Address Match Interrupt Enable**
- **Bit 4 – CCA_ED_DONE_EN - End of ED Measurement Interrupt Enable**
- **Bit 3 – RX_END_EN - RX_END Interrupt Enable**
- **Bit 2 – RX_START_EN - RX_START Interrupt Enable**
- **Bit 1 – PLL_UNLOCK_EN - PLL Unlock Interrupt Enable**
- **Bit 0 – PLL_LOCK_EN - PLL Lock Interrupt Enable**

9.12.19 IRQ_STATUS – Transceiver Interrupt Status Register

Bit	7	6	5	4	
NA ($14F)	AWAKE	TX_END	AMI	CCA_ED_DONE	IRQ_STATUS
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($14F)	RX_END	RX_START	PLL_UNLOCK	PLL_LOCK	IRQ_STATUS
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register contains the status of the pending interrupt requests. An interrupt is pending if the associated bit has a value of one. Such a pending interrupts can be manually cleared by writing a 1 to that register bit. Interrupts are automatically cleared when the corresponding interrupt service routine is being executed.

- **Bit 7 – AWAKE - Awake Interrupt Status**

- **Bit 6 – TX_END - TX_END Interrupt Status**
- **Bit 5 – AMI - Address Match Interrupt Status**
- **Bit 4 – CCA_ED_DONE - End of ED Measurement Interrupt Status**
- **Bit 3 – RX_END - RX_END Interrupt Status**
- **Bit 2 – RX_START - RX_START Interrupt Status**
- **Bit 1 – PLL_UNLOCK - PLL Unlock Interrupt Status**
- **Bit 0 – PLL_LOCK - PLL Lock Interrupt Status**

9.12.20 VREG_CTRL – Voltage Regulator Control and Status Register

Bit	7	6	5	4	
NA ($150)	AVREG_EXT	AVDD_OK	Resx5	Resx4	VREG_CTRL
Read/Write	RW	R	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($150)	Resx3	DVDD_OK	Resx1	Resx0	VREG_CTRL
Read/Write	RW	R	RW	RW	
Initial Value	0	0	0	0	

This register controls the use of the voltage regulators and indicates their status.

- **Bit 7 – AVREG_EXT - Use External AVDD Regulator**

If set, this register bit disables the internal analog voltage regulator to apply an external regulated 1.8V supply for the analog building blocks.

Table 9-51 AVREG_EXT Register Bits

Register Bits	Value	Description
AVREG_EXT	0	Internal AVDD voltage regulator for the analog section is enabled.
	1	Internal AVDD voltage regulator is disabled; use external regulated 1.8V supply voltage for the analog section.

- **Bit 6 – AVDD_OK - AVDD Supply Voltage Valid**

This register bit indicates if the internal 1.8V regulated voltage supply AVDD has settled. The bit is set to logic high if AVREG_EXT = 1.

Table 9-52 AVDD_OK Register Bits

Register Bits	Value	Description
AVDD_OK	0	Analog voltage regulator disabled or supply voltage not stable
	1	Analog supply voltage has settled

- **Bit 5:3 – Resx5:3 - Reserved**
- **Bit 2 – DVDD_OK - DVDD Supply Voltage Valid**

This register bit indicates if the internal 1.8V regulated voltage supply DVDD has settled. The bit is set to logic high if DVREG_EXT = 1.

Table 9-53 DVDD_OK Register Bits

Register Bits	Value	Description
DVDD_OK	0	Digital voltage regulator disabled or supply voltage not stable
	1	Digital supply voltage has settled

- **Bit 1:0 – DVREG_TRIM1:0 - Reserved**

Table 9-54 DVREG_TRIM Register Bits

Register Bits	Value	Description
DVREG_TRIM1:0	0	1.80V
	1	1.75V
	2	1.84V
	3	1.88V

9.12.21 BATMON – Battery Monitor Control and Status Register

Bit	7	6	5	4	
NA ($151)	BAT_LOW	BAT_LOW_EN	BATMON_OK	BATMON_HR	BATMON
Read/Write	RW	RW	R	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($151)	BATMON_VTH3	BATMON_VTH2	BATMON_VTH1	BATMON_VTH0	BATMON
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	1	0	

This register configures the battery monitor to observe the supply voltage at EVDD. The status of the EVDD supply voltage is accessible by reading bit BATMON_OK with respect to the actual BATMON settings. Furthermore the Battery Monitor Interrupt can be controlled with the bits BAT_LOW and BAT_LOW_EN similar to the function of the IRQ_STATUS and IRQ_MASK register for other radio transceiver interrupts.

- **Bit 7 – BAT_LOW - Battery Monitor Interrupt Status**

A BATMON Interrupt is pending if this bit is set. Writing one to this bit if it has been at one will clear the interrupt.

- **Bit 6 – BAT_LOW_EN - Battery Monitor Interrupt Enable**

The Battery Monitor Interrupt is enabled if this bit is set to one. The Battery Monitor will not generate an interrupt if this bit is zero.

- **Bit 5 – BATMON_OK - Battery Monitor Status**

The register bit BATMON_OK indicates the level of the external supply voltage with respect to the programmed threshold BATMON_VTH.

Table 9-55 BATMON_OK Register Bits

Register Bits	Value	Description
BATMON_OK	0	The battery voltage is below the threshold.
	1	The battery voltage is above the threshold.

- **Bit 4 – BATMON_HR - Battery Monitor Voltage Range**

This bit sets the range and resolution of the battery monitor.

Table 9-56 BATMON_HR Register Bits

Register Bits	Value	Description
BATMON_HR	0	Enables the low range, see BATMON_VTH
	1	Enables the high range, see BATMON_VTH

- **Bit 3:0 – BATMON_VTH3:0 - Battery Monitor Threshold Voltage**

The threshold values for the battery monitor are set by these register bits according to the following table.

Table 9-57 BATMON_VTH Register Bits

Register Bits	Value	Description
BATMON_VTH3:0	0x0	2.550V (BATMON_HR=1) 1.70V (BATMON_HR=0)
	0x1	2.625V (BATMON_HR=1) 1.75V (BATMON_HR=0)
	0x2	2.700V (BATMON_HR=1) 1.80V (BATMON_HR=0)
	0x3	2.775V (BATMON_HR=1) 1.85V (BATMON_HR=0)
	0x4	2.850V (BATMON_HR=1) 1.90V (BATMON_HR=0)
	0x5	2.925V (BATMON_HR=1) 1.95V (BATMON_HR=0)
	0x6	3.000V (BATMON_HR=1) 2.00V (BATMON_HR=0)
	0x7	3.075V (BATMON_HR=1) 2.05V (BATMON_HR=0)
	0x8	3.150V (BATMON_HR=1) 2.10V (BATMON_HR=0)
	0x9	3.225V (BATMON_HR=1) 2.15V (BATMON_HR=0)
	0xA	3.300V (BATMON_HR=1) 2.20V (BATMON_HR=0)
	0xB	3.375V (BATMON_HR=1) 2.25V (BATMON_HR=0)
	0xC	3.450V (BATMON_HR=1) 2.30V (BATMON_HR=0)
	0xD	3.525V (BATMON_HR=1) 2.35V (BATMON_HR=0)
	0xE	3.600V (BATMON_HR=1) 2.40V (BATMON_HR=0)
	0xF	3.675V (BATMON_HR=1) 2.45V (BATMON_HR=0)

9.12.22 XOSC_CTRL – Crystal Oscillator Control Register

Bit	7	6	5	4	
NA ($152)	XTAL_MODE3	XTAL_MODE2	XTAL_MODE1	XTAL_MODE0	XOSC_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	1	1	1	1	

Bit	3	2	1	0	
NA ($152)	XTAL_TRIM3	XTAL_TRIM2	XTAL_TRIM1	XTAL_TRIM0	XOSC_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register controls the operation of the 16MHz crystal oscillator.

- **Bit 7:4 – XTAL_MODE3:0 - Crystal Oscillator Operating Mode**

These register bits set the operating mode of the 16 MHz crystal oscillator. For normal operation the default value is set to XTAL_MODE = 0xF after reset. For use with an external clock source it is recommended to set XTAL_MODE = 0x4.

Table 9-58 XTAL_MODE Register Bits

Register Bits	Value	Description
XTAL_MODE3:0	0x4	Internal crystal oscillator disabled; use external reference frequency.
	0xF	Internal crystal oscillator enabled; amplitude regulation of oscillation enabled.

- **Bit 3:0 – XTAL_TRIM3:0 - Crystal Oscillator Load Capacitance Trimming**

These register bits control two internal capacitance arrays connected to pins XTAL1 and XTAL2. A capacitance value in the range from 0 pF to 4.5 pF is selectable with a resolution of 0.3 pF.

Table 9-59 XTAL_TRIM Register Bits

Register Bits	Value	Description
XTAL_TRIM3:0	0x0	0.0 pF, trimming capacitors disconnected
	0x1	0.3 pF, trimming capacitor switched on
	0x2	...
	0xF	4.5 pF, trimming capacitor switched on

9.12.23 RX_SYN – Transceiver Receiver Sensitivity Control Register

Bit	7	6	
NA ($155)	RX_PDT_DIS	Res2	RX_SYN
Read/Write	RW	R	
Initial Value	0	0	
Bit	5	4	
NA ($155)	Res1	Res0	RX_SYN
Read/Write	R	R	
Initial Value	0	0	
Bit	3	2	
NA ($155)	RX_PDT_LEVEL3	RX_PDT_LEVEL2	RX_SYN
Read/Write	RW	RW	
Initial Value	0	0	
Bit	1	0	
NA ($155)	RX_PDT_LEVEL1	RX_PDT_LEVEL0	RX_SYN
Read/Write	RW	RW	
Initial Value	0	0	

This register controls the sensitivity threshold of the receiver.

- **Bit 7 – RX_PDT_DIS - Prevent Frame Reception**

RX_PDT_DIS = 1 prevents the reception of a frame even if the radio transceiver is in receive modes. An ongoing frame reception is not affected. This operation mode is independent of the setting of register bits RX_PDT_LEVEL.

- **Bit 6:4 – Res2:0 - Reserved**
- **Bit 3:0 – RX_PDT_LEVEL3:0 - Reduce Receiver Sensitivity**

These register bits reduce the receiver sensitivity such that frames with a RSSI level below the RX_PDT_LEVEL threshold level are not received (RX_PDT_LEVEL>0). The threshold level can be calculated according to the following formula: RX_THRES > RSSI_BASE_VAL+3·(RX_PDT_LEVEL-1), for RX_PDT_LEVEL>0. If register bits RX_PDT_LEVEL>0 the current consumption of the receiver in states RX_ON and RX_AACK_ON is reduced by 500 µA. If register bits RX_PDT_LEVEL=0 (reset value) all frames with a valid SHR and PHR are received, independently of their signal strength. Examples for certain register settings are given in the following table.

Table 9-60 RX_PDT_LEVEL Register Bits

Register Bits	Value	Description
RX_PDT_LEVEL3:0	0x0	RX_THRES ≤ RSSI_BASE_VAL (Reset value); RSSI value not considered
	0x1	RX_THRES > RSSI_BASE_VAL + 0 · 3; RSSI > -90 dBm
	0x2	...
	0xE	RX_THRES > RSSI_BASE_VAL + 13 · 3; RSSI > -51 dBm
	0xF	RX_THRES > RSSI_BASE_VAL + 14 · 3; RSSI > -48 dBm

9.12.24 XAH_CTRL_1 – Transceiver Acknowledgment Frame Control Register 1

Bit	7	6	5	4	
NA ($157)	Res1	Res0	AACK_FLTR_RES_FT	AACK_UPLD_RES_FT	XAH_CTRL_1
Read/Write	R	R	RW	RW	
Initial Value	0	0	0	0	

Bit	3	2	1	0	
NA ($157)	Res	AACK_ACK_TIME	AACK_PROM_MODE	Res	XAH_CTRL_1
Read/Write	R	RW	RW	R	
Initial Value	0	0	0	0	

This register is a multi-purpose control register for various RX_AACK settings.

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 5 – AACK_FLTR_RES_FT - Filter Reserved Frames**

This register bit shall only be set if AACK_UPLD_RES_FT = 1. If AACK_FLTR_RES_FT = 1 reserved frame types are filtered similar to data frames as

specified in IEEE 802.15.4-2006. Reserved frame types are explained in IEEE 802.15.4 section 7.2.1.1.1. If AACK_FLTR_RES_FT = 0 a received, reserved frame is only checked for a valid FCS.

- **Bit 4 – AACK_UPLD_RES_FT - Process Reserved Frames**

If AACK_UPLD_RES_FT = 1 received frames indicated as reserved are further processed. A RX_END interrupt is generated if the FCS of those frames is valid. In conjunction with the configuration bit AACK_FLTR_RES_FT set, these frames are handled like IEEE 802.15.4 compliant data frames during RX_AACK transaction. An AMI interrupt is issued if the address in the received frame matches the node address. That means if a reserved frame passes the third level filter rules, an acknowledgment frame is generated and transmitted if it was requested by the received frame. If this is not wanted bit AACK_DIS_ACK in register CSMA_SEED_1 has to be set.

- **Bit 3 – Res - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 2 – AACK_ACK_TIME - Reduce Acknowledgment Time**

According to IEEE 802.15.4, section 7.5.6.4.2 the transmission of an acknowledgment frame shall commence 12 symbols (aTurnaroundTime) after the reception of the last symbol of a data or MAC command frame. This is achieved with the reset value of the register bit AACK_ACK_TIME. If AACK_ACK_TIME = 1 an acknowledgment frame is alternatively sent already 2 symbol periods (32 µs) after the reception of the last symbol of a data or MAC command frame. This may be applied to proprietary networks or networks using the High Data Rate Modes to increase battery lifetime and to improve the overall data throughput. This setting affects also to acknowledgment frame response time for slotted acknowledgment operation.

Table 9-61 AACK_ACK_TIME Register Bits

Register Bits	Value	Description
AACK_ACK_TIME	0	12 symbols acknowledgment time
	1	2 symbols acknowledgment time

- **Bit 1 – AACK_PROM_MODE - Enable Promiscuous Mode**

This register bit enables the promiscuous mode within the RX_AACK mode; refer to IEEE 802.15.4-2006 chapter 7.5.6.5. If this bit is set, every incoming frame with a valid PHR finishes with a RX_END interrupt even if the third level filter rules do not match or the FCS is not valid. The bit RX_CRC_VALID of register PHY_RSSI is set accordingly. If this bit is set and a frame passes the third level filter rules, an acknowledgment frame is generated and transmitted unless disabled by bit AACK_DIS_ACK of register CSMA_SEED_1.

- **Bit 0 – Res - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

9.12.25 FTN_CTRL – Transceiver Filter Tuning Control Register

Bit	7	6	5	4	
NA ($158)	FTN_START	Resx6	Resx5	Resx4	FTN_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	0	1	0	1	

Bit	3	2	1	0	
NA ($158)	Resx3	Resx2	Resx1	Resx0	FTN_CTRL
Read/Write	RW	RW	RW	RW	
Initial Value	1	0	0	0	

This register controls the operation of the calibration loop of the filter tuning network.

- **Bit 7 – FTN_START - Start Calibration Loop of Filter Tuning Network**

FTN_START = 1 initiates the calibration of the filter tuning network. When the calibration cycle has finished after at most 25 µs the register bit is automatically reset to 0.

- **Bit 6:0 – Resx6:0 - Reserved**

9.12.26 PLL_CF – Transceiver Center Frequency Calibration Control Register

Bit	7	6	
NA ($15A)	PLL_CF_START	Resx6	PLL_CF
Read/Write	RW	RW	
Initial Value	0	1	
Bit	5	4	
NA ($15A)	Resx5	Resx4	PLL_CF
Read/Write	RW	RW	
Initial Value	0	1	
Bit	3	2	
NA ($15A)	Resx3	Resx2	PLL_CF
Read/Write	RW	RW	
Initial Value	0	1	
Bit	1	0	
NA ($15A)	Resx1	Resx0	PLL_CF
Read/Write	RW	RW	
Initial Value	1	1	

This register controls the operation of the center frequency calibration loop.

- **Bit 7 – PLL_CF_START - Start Center Frequency Calibration**

PLL_CF_START = 1 initiates the center frequency calibration. The calibration cycle has finished after 35 µs (typical). The register bit is cleared immediately after finishing the calibration.

- **Bit 6:0 – Resx6:0 - Reserved**

9.12.27 PLL_DCU – Transceiver Delay Cell Calibration Control Register

Bit	7	6	5	4	
NA ($15B)	PLL_DCU_START	Resx6	Resx5	Resx4	PLL_DCU
Read/Write	RW	R	RW	RW	
Initial Value	0	0	1	0	

Bit	3	2	1	0	
NA ($15B)	Resx3	Resx2	Resx1	Resx0	PLL_DCU
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register controls the operation of the calibration loop of the delay cell.

- **Bit 7 – PLL_DCU_START - Start Delay Cell Calibration**

PLL_DCU_START = 1 initiates the delay cell calibration. The calibration cycle has finished after at most 6 µs. The register bit is cleared immediately after finishing the calibration.

- **Bit 6:0 – Resx6:0 - Reserved**

9.12.28 PART_NUM – Device Identification Register (Part Number)

Bit	7	6	5	4	3	2	1	0	
NA ($15C)	PART_NUM7:0								PART_NUM
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	1	0	0	0	0	0	1	1	

This register contains the part number of the device.

- **Bit 7:0 – PART_NUM7:0 - Part Number**

These bits decode the part number of the device according to the following table.

Table 9-62 PART_NUM Register Bits

Register Bits	Value	Description
PART_NUM7:0	0x83	ATmega128RFA1 part number

9.12.29 VERSION_NUM – Device Identification Register (Version Number)

Bit	7	6	5	4	3	2	1	0	
NA ($15D)	VERSION_NUM7:0								VERSION_NUM
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the version number of the device. The device identification overwrites the Reset value.

- **Bit 7:0 – VERSION_NUM7:0 - Version Number**

These bits decode the version number of the device according to the following table.

Table 9-63 VERSION_NUM Register Bits

Register Bits	Value	Description
VERSION_NUM7:0	2	Revision AB
	3	Revision C
	4	Revision D

9.12.30 MAN_ID_0 – Device Identification Register (Manufacture ID Low Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($15E)	MAN_ID_07:00								MAN_ID_0
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	1	1	1	1	1	

Bits [7:0] of the 32-bit JEDEC manufacturer ID are stored in this register. Bits [15:8] are stored in register MAN_ID_1. The highest 16 bits of the JEDEC ID are not stored in registers.

- **Bit 7:0 – MAN_ID_07:00 - Manufacturer ID (Low Byte)**

These bits contain bits [7:0] of the 32-bit JEDEC manufacturer ID.

Table 9-64 MAN_ID_0 Register Bits

Register Bits	Value	Description
MAN_ID_07:00	0x1f	Atmel JEDEC manufacturer ID, bits [7:0] of 32 bit manufacturer ID: 00 00 00 1F

9.12.31 MAN_ID_1 – Device Identification Register (Manufacture ID High Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($15F)	MAN_ID_17:10								MAN_ID_1
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bits [15:8] of the 32-bit JEDEC manufacturer ID are stored in this register. Bits [7:0] are stored in register MAN_ID_0. The highest 16 bits of the JEDEC ID are not stored in registers.

- **Bit 7:0 – MAN_ID_17:10 - Manufacturer ID (High Byte)**

These bits contain bits [15:8] of the 32-bit JEDEC manufacturer ID.

Table 9-65 MAN_ID_1 Register Bits

Register Bits	Value	Description
MAN_ID_17:10	0x00	Atmel JEDEC manufacturer ID, bits [15:8] of 32 bit manufacturer ID: 00 00 00 1F

9.12.32 SHORT_ADDR_0 – Transceiver MAC Short Address Register (Low Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($160)	SHORT_ADDR_07:00								SHORT_ADDR_0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

This register contains the lower 8 bits of the MAC short address for Frame Filter address recognition.

- **Bit 7:0 – SHORT_ADDR_07:00 - MAC Short Address**

These bits contain the bits [7:0] of the MAC short address.

9.12.33 SHORT_ADDR_1 – Transceiver MAC Short Address Register (High Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($161)	SHORT_ADDR_17:10								SHORT_ADDR_1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

This register contains the upper 8 bits of the MAC short address for Frame Filter address recognition.

- **Bit 7:0 – SHORT_ADDR_17:10 - MAC Short Address**

These bits contain the bits [15:8] of the MAC short address.

9.12.34 PAN_ID_0 – Transceiver Personal Area Network ID Register (Low Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($162)	PAN_ID_07:00								PAN_ID_0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

This register contains the lower 8 bits of the MAC PAN ID for Frame Filter address recognition.

- **Bit 7:0 – PAN_ID_07:00 - MAC Personal Area Network ID**

These bits contain the bits [7:0] of the MAC PAN ID.

9.12.35 PAN_ID_1 – Transceiver Personal Area Network ID Register (High Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($163)	PAN_ID_17:10								PAN_ID_1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

This register contains the upper 8 bits of the MAC PAN ID for Frame Filter address recognition.

- **Bit 7:0 – PAN_ID_17:10 - MAC Personal Area Network ID**

These bits contain the bits [15:8] of the MAC PAN ID.

9.12.36 IEEE_ADDR_0 – Transceiver MAC IEEE Address Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($164)	IEEE_ADDR_07:00								IEEE_ADDR_0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [7:0] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_07:00 - MAC IEEE Address**

These bits map to the bits [7:0] of the 64 bit MAC IEEE address.

9.12.37 IEEE_ADDR_1 – Transceiver MAC IEEE Address Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($165)	IEEE_ADDR_17:10								IEEE_ADDR_1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [15:8] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_17:10 - MAC IEEE Address**

These bits map to the bits [15:8] of the 64 bit MAC IEEE address.

9.12.38 IEEE_ADDR_2 – Transceiver MAC IEEE Address Register 2

Bit	7	6	5	4	3	2	1	0	
NA ($166)	IEEE_ADDR_27:20								IEEE_ADDR_2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [23:16] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_27:20 - MAC IEEE Address**

These bits map to the bits [23:16] of the 64 bit MAC IEEE address.

9.12.39 IEEE_ADDR_3 – Transceiver MAC IEEE Address Register 3

Bit	7	6	5	4	3	2	1	0	
NA ($167)	IEEE_ADDR_37:30								IEEE_ADDR_3
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [31:24] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_37:30 - MAC IEEE Address**

These bits map to the bits [31:24] of the 64 bit MAC IEEE address.

9.12.40 IEEE_ADDR_4 – Transceiver MAC IEEE Address Register 4

Bit	7	6	5	4	3	2	1	0	
NA ($168)	IEEE_ADDR_47:40								IEEE_ADDR_4
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [39:32] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_47:40 - MAC IEEE Address**

These bits map to the bits [39:32] of the 64 bit MAC IEEE address.

9.12.41 IEEE_ADDR_5 – Transceiver MAC IEEE Address Register 5

Bit	7	6	5	4	3	2	1	0	
NA ($169)	IEEE_ADDR_57:50								IEEE_ADDR_5
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [47:40] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_57:50 - MAC IEEE Address**

These bits map to the bits [47:40] of the 64 bit MAC IEEE address.

9.12.42 IEEE_ADDR_6 – Transceiver MAC IEEE Address Register 6

Bit	7	6	5	4	3	2	1	0	
NA ($16A)	IEEE_ADDR_67:60								IEEE_ADDR_6
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [55:48] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_67:60 - MAC IEEE Address**

These bits map to the bits [55:48] of the 64 bit MAC IEEE address.

9.12.43 IEEE_ADDR_7 – Transceiver MAC IEEE Address Register 7

Bit	7	6	5	4	3	2	1	0	
NA ($16B)	IEEE_ADDR_77:70								IEEE_ADDR_7
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the bits [63:56] of the MAC IEEE address for Frame Filter address recognition.

- **Bit 7:0 – IEEE_ADDR_77:70 - MAC IEEE Address**

These bits map to the bits [63:56] of the 64 bit MAC IEEE address.

9.12.44 XAH_CTRL_0 – Transceiver Extended Operating Mode Control Register

Bit	7	6	
NA ($16C)	MAX_FRAME_RETRIES3	MAX_FRAME_RETRIES2	XAH_CTRL_0
Read/Write	RW	RW	
Initial Value	0	0	
Bit	5	4	
NA ($16C)	MAX_FRAME_RETRIES1	MAX_FRAME_RETRIES0	XAH_CTRL_0
Read/Write	RW	RW	
Initial Value	1	1	
Bit	3	2	
NA ($16C)	MAX_CSMA_RETRIES2	MAX_CSMA_RETRIES1	XAH_CTRL_0
Read/Write	RW	RW	
Initial Value	1	0	
Bit	1	0	
NA ($16C)	MAX_CSMA_RETRIES0	SLOTTED_OPERATION	XAH_CTRL_0
Read/Write	RW	RW	
Initial Value	0	0	

This register is used to control various settings of the Extended Operating Mode.

- **Bit 7:4 – MAX_FRAME_RETRIES3:0 - Maximum Number of Frame Re-transmission Attempts**

The setting of MAX_FRAME_RETRIES in TX_ARET mode specifies the number of attempts to retransmit a frame when it was not acknowledged by the recipient. The transaction gets canceled if the number of attempts exceeds MAX_FRAME_RETRIES.

Table 9-66 MAX_FRAME_RETRIES Register Bits

Register Bits	Value	Description
MAX_FRAME_RETRIES3:0	0x0	Retransmission of frame is not attempted.
	0x1	Retransmission of frame is attempted once.
	0x2	...
	0xF	Retransmission of frame is attempted 15 times.

- **Bit 3:1 – MAX_CSMA_RETRIES2:0 - Maximum Number of CSMA-CA Procedure Repetition Attempts**

MAX_CSMA_RETRIES specifies the number of retries in TX_ARET mode to repeat the CSMA-CA procedure before the transaction gets canceled. According to IEEE 802.15.4 the valid range of MAX_CSMA_RETRIES is 0 to 5. A value of MAX_CSMA_RETRIES = 7 initiates an immediate frame transmission without performing CSMA-CA. This may especially be required for slotted acknowledgment operation. MAX_CSMA_RETRIES = 6 is reserved.

Table 9-67 MAX_CSMA_RETRIES Register Bits

Register Bits	Value	Description
MAX_CSMA_RETRIES2:0	0x0	No repetition of CSMA-CA procedure
	0x1	One repetition of CSMA-CA procedure
	0x2	...
	0x5	Five repetitions (highest IEEE 802.15.4 compliant value)
	0x6	Reserved
	0x7	Immediate frame re-transmission without performing CSMA-CA

- **Bit 0 – SLOTTED_OPERATION - Set Slotted Acknowledgment**

When using RX_AACK mode in networks operating in beacon or slotted mode according to IEEE 802.15.4-2006, chapter 5.5.1 the register bit SLOTTED_OPERATION indicates that acknowledgment frames are to be sent on back-off slot boundaries (slotted acknowledgment). If this register bit is set the acknowledgment frame transmission has to be initiated by the application software using bit SLPTR of register TRXPR. This waiting state is signaled in sub register TRAC_STATUS of register TRX_STATE with value SUCCESS_WAIT_FOR_ACK.

Table 9-68 SLOTTED_OPERATION Register Bits

Register Bits	Value	Description
SLOTTED_OPERATION	0	The radio transceiver operates in unslotted mode. An acknowledgment frame is automatically sent if requested.
	1	The transmission of an acknowledgment frame has to be controlled by the microcontroller.

9.12.45 CSMA_SEED_0 – Transceiver CSMA-CA Random Number Generator Seed Register

Bit	7	6	5	4	3	2	1	0	
NA ($16D)	CSMA_SEED_07:00								CSMA_SEED_0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	0	1	0	1	0	

This register contains the lower 8 bits of the CSMA_SEED. The upper 3 bits are part of register CSMA_SEED_1. CSMA_SEED is the seed for the random number generation that determines the length of the back-off period in the CSMA-CA algorithm. It is recommended to initialize registers CSMA_SEED by random values. This can be done using the bits RND_VALUE of register PHY_RSSI.

- **Bit 7:0 – CSMA_SEED_07:00 - Seed Value for CSMA Random Number Generator**

These bits contain the bits [7:0] of the CSMA_SEED.

9.12.46 CSMA_SEED_1 – Transceiver Acknowledgment Frame Control Register 2

Bit	7	6	
NA ($16E)	AACK_FVN_MODE1	AACK_FVN_MODE0	CSMA_SEED_1
Read/Write	RW	RW	
Initial Value	0	1	
Bit	5	4	
NA ($16E)	AACK_SET_PD	AACK_DIS_ACK	CSMA_SEED_1
Read/Write	RW	RW	
Initial Value	0	0	
Bit	3	2	
NA ($16E)	AACK_I_AM_COORD	CSMA_SEED_12	CSMA_SEED_1
Read/Write	RW	RW	
Initial Value	0	0	
Bit	1	0	
NA ($16E)	CSMA_SEED_11	CSMA_SEED_10	CSMA_SEED_1
Read/Write	RW	RW	
Initial Value	1	0	

This register is a control register for RX_AACK and contains a part of the CSMA_SEED for the CSMA-CA algorithm.

- **Bit 7:6 – AACK_FVN_MODE1:0 - Acknowledgment Frame Filter Mode**

The frame control field of the MAC header (MHR) contains a frame version subfield. The setting of AACK_FVN_MODE specifies the frame filtering behavior of the radio transceiver. According to the content of these register bits the radio transceiver passes frames with a specific frame version number, number group or independent of the frame version number. Thus the register bits AACK_FVN_MODE define the maximum acceptable frame version. Received frames with a higher frame version number than configured do not pass the address filter and are not acknowledged.

Table 9-69 AACK_FVN_MODE Register Bits

Register Bits	Value	Description
AACK_FVN_MODE1:0	0	Acknowledge frames with version number 0
	1	Acknowledge frames with version number 0 or 1
	2	Acknowledge frames with version number 0 or 1 or 2
	3	Acknowledge frames independent of frame version number

- **Bit 5 – AACK_SET_PD - Set Frame Pending Sub-field**

The content of AACK_SET_PD bit is copied into the frame pending subfield of the acknowledgment frame if the acknowledgment is the answer to a data request MAC command frame. If in addition the bits AACK_FVN_MODE of this register are configured to accept frames with a frame version other than 0 or 1, the content of register bit AACK_SET_PD is also copied into the frame pending subfield of the acknowledgment frame for any MAC command frame with a frame version of 2 or 3 that have the security enabled subfield set to 1. This is done in the assumption that a future version of the IEEE 802.15.4 standard might change the length or structure of the auxiliary security header, so that it is not possible to safely detect whether the MAC command frame is actually a data request command or not.

- **Bit 4 – AACK_DIS_ACK - Disable Acknowledgment Frame Transmission**

If this bit is set no acknowledgment frames are transmitted in RX_AACK Extended Operating Mode even if requested.

- **Bit 3 – AACK_I_AM_COORD - Set Personal Area Network Coordinator**

This register bit has to be set if the node is a PAN coordinator. It is used for address filtering in RX_AACK.

- **Bit 2:0 – CSMA_SEED_12:10 - Seed Value for CSMA Random Number Generator**

These bits contain the bits [10:8] of the CSMA_SEED. The lower part is defined in register CSMA_SEED_0. See register CSMA_SEED_0 for details.

9.12.47 CSMA_BE – Transceiver CSMA-CA Back-off Exponent Control Register

Bit	7	6	5	4	
NA ($16F)	MAX_BE3	MAX_BE2	MAX_BE1	MAX_BE0	CSMA_BE
Read/Write	RW	RW	RW	RW	
Initial Value	0	1	0	1	
Bit	3	2	1	0	
NA ($16F)	MIN_BE3	MIN_BE2	MIN_BE1	MIN_BE0	CSMA_BE
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	1	1	

This register controls the back-off exponent for the CSMA-CA procedure.

- **Bit 7:4 – MAX_BE3:0 - Maximum Back-off Exponent**

These register bits define the maximum back-off exponent used in the CSMA-CA algorithm to generate a pseudo random number for back off the CCA. For details refer to IEEE 802.15.4-2006, section 7.5.1.4. Valid values are 3 to 8.

Table 9-70 MAX_BE Register Bits

Register Bits	Value	Description
MAX_BE3:0	1	This value is not valid for the maximum back-off exponent.
	2	This value is not valid for the maximum back-off exponent.
	3	Minimum, IEEE compliant value for the maximum back-off exponent.
	4	...
	8	Maximum, IEEE compliant value for the maximum back-off exponent.

- **Bit 3:0 – MIN_BE3:0 - Minimum Back-off Exponent**

These register bits define the minimum back-off exponent used in the CSMA-CA algorithm to generate a pseudo random number for back off the CCA. For details refer to IEEE 802.15.4-2006, section 7.5.1.4. Valid values are MAX_BE, MAX_BE-1), ..., 0. If MIN_BE = 0 and MAX_BE = 0 the CCA back off period is always set to 0.

Table 9-71 MIN_BE Register Bits

Register Bits	Value	Description
MIN_BE3:0	0	Minimum value of minimum back-off exponent.
	1	...
	8	Maximum value of minimum back-off exponent. MIN_BE must be smaller or equal to MAX_BE.

9.12.48 TST_CTRL_DIGI – Transceiver Digital Test Control Register

Bit	7	6	
NA ($176)	Resx7	Resx6	TST_CTRL_DIGI
Read/Write	RW	RW	
Initial Value	0	0	
Bit	5	4	
NA ($176)	Resx5	Resx4	TST_CTRL_DIGI
Read/Write	RW	RW	
Initial Value	0	0	
Bit	3	2	
NA ($176)	TST_CTRL_DIG3	TST_CTRL_DIG2	TST_CTRL_DIGI
Read/Write	RW	RW	
Initial Value	0	0	
Bit	1	0	
NA ($176)	TST_CTRL_DIG1	TST_CTRL_DIG0	TST_CTRL_DIGI
Read/Write	RW	RW	
Initial Value	0	0	

This register takes part in the activation sequence of the continuous transmission test mode. Other functionality of this register is reserved for internal use.

- **Bit 7:4 – Resx7:4 - Reserved**
- **Bit 3:0 – TST_CTRL_DIG3:0 - Digital Test Controller Register**

This sub-register selects a test controller function. All values not listed int the following table are reserved for internal use.

Table 9-72 TST_CTRL_DIG Register Bits

Register Bits	Value	Description
TST_CTRL_DIG3:0	0	NORMAL (no test is active)
	15	TST_CONT_TX (continuous transmit)

9.12.49 TST_RX_LENGTH – Transceiver Received Frame Length Register

Bit	7	6	5	4	
NA ($17B)	RX_LENGTH7	RX_LENGTH6	RX_LENGTH5	RX_LENGTH4	TST_RX_LENGTH
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

Bit	3	2	1	0	
NA ($17B)	RX_LENGTH3	RX_LENGTH2	RX_LENGTH1	RX_LENGTH0	TST_RX_LENGTH
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register contains the frame length information of a received frame. This information is not stored in the frame buffer. The frame length information is written to this register after the last received octet.

- **Bit 7:0 – RX_LENGTH7:0 - Received Frame Length**

These bits contain the length of the last received frame.

9.12.50 TRXFBST – Start of frame buffer

Bit	7	6	5	4	3	2	1	0	
NA ($180)	TRXFBST7:0								TRXFBST
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

First byte of the 128 byte long frame buffer of the TRX24.

- **Bit 7:0 – TRXFBST7:0 - Frame Buffer Start Byte**

9.12.51 TRXFBEND – End of frame buffer

Bit	7	6	5	4	3	2	1	0	
NA ($1FF)	TRXFBEND7:0								TRXFBEND
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register is the last byte of the 128 byte long frame buffer of the radio transceiver.

- **Bit 7:0 – TRXFBEND7:0 - Frame Buffer End Byte**

10 MAC Symbol Counter

10.1 Main Features

The MAC symbol counter provides symbol timing information for IEEE 802.15.4 wireless networks. The counter time base can be derived from the 16 MHz crystal or the RTC (32.768 kHz crystal on TOSC) during operation. In deep-sleep mode the counter operates from the RTC clock. The module provides the following features:

- **Backoff slot counter with interrupt generation**
- **Counter clock source selection between XTAL1 (16 MHz) and TOSC1 (RTC)**
- **Automatic RTC clock selection for sleep mode operation and automatic fallback**
- **3 independent compare units with relative and absolute compare mode and interrupt generation (support for slotted operation and superframe handling)**
- **Low-power, deep-sleep mode operation and system wake up with all symbol counter interrupt events**
- **Automatic SFD and incoming beacon timestamping**
- **Manual beacon timestamping**
- **Manual timer synchronization within a 16 µs symbol period by resetting clock prescaler and backoff slot counter**
- **Atomic read/write access for 32 bit registers**

10.2 Clock source selection and Sleep/Active mode operation

The symbol counter can be sourced by the transceiver clock or by the asynchronous Real Time Clock (RTC) oscillator. If the transceiver goes from active mode into sleep mode, the symbol counter clock source is switched to the RTC clock automatically. A clock source change is indicated in the bit SCCKSEL of Register "SCCR0 – Symbol Counter Control Register 0" on page 143 . The bit SCCKSEL can not be written if the radio tranceiver is in SLEEP mode.

After wake up, the counter switches back to the clock source which was selected before going to sleep mode. Switching the clock source from RTC to 16 MHz resets the 16 MHz clock prescaler. This makes sure, that after switching back the clock source, the symbol counter starts counting with a full 16 µs symbol period.

The clock source can be selected with bit SCCKSEL in the SCCR0 Register

10.3 32 bit Register Access (Atomic Read/Write)

All 32 bit registers support atomic read or write operation. That means reading or writing the least significant *xxx*LL byte (the register name ends in LL) updates or captures the complete 32 bit value.

Read Access: 1. Reading the LL-Byte captures the 32 bit value in a temporary register
2. Read the upper 3 bytes

Write Access: 1. Write the upper 3 byte
2. Writing the LL-Byte stores the 32 bit value in the counter registers

The same temporary register is used for all 32 bit register of the MAX symbol counter.

10.4 Symbol Counter (32 bit, SCCNT)

The symbol counter is a 32 bit counter which can be sourced by a 62.5 kHz clock, derived from the 16 MHz system clock or from the RTC (32.768 kHz). If sourced by the RTC, a special control circuitry ensures that the counter error does not exceed one symbol period.

The symbol counter can be set or read from the controller. Reading must start with the least significant byte. If the least significant byte is accessed, all 32 bit of the counter are captured. A read access to SCCNTLL requires a maximum of three AVR clocks. Reading the upper three bytes of the counter requires two CPU clock cycles for each byte.

Writing to the counter should start with the most significant byte. Writing the least significant byte initiates the counter update and the new 32 bit counter value is loaded into the counter with the next available counter clock edge. This can take up to 16 µs beginning from the low byte write operation, if the counter is sourced by the RTC.

If the counter clock is derived from the 16 MHz clock system, the new counter value is stored immediately.

During the counter update cycle, the counter busy flag SCBSY in the SCSR register is set to “1”. As long as this bit is “1”, no further read/write access to the counter should be initiated. The same applies if the AVR is forced to any sleep mode with disabled AVR clock, right after writing to the SCCNT register. If the counter busy flag is not checked before going to sleep, it is possible that the counter register is not updated correctly.

The symbol counter overflow is indicated by a overflow interrupt. The interrupt is generated when the counter turns from 0xFFFFFFFF to 0x00000000.

10.5 Symbol Counter SFD Timestamp Register (32 bit, SCTSR, Read Only)

The SFD timestamp register stores the symbol counter value at the time, the SFD has been detected. The Register value becomes valid if a valid frame length byte (frame length > 0) has been detected, but it is not checked if the received frame is valid (CRC check). Timestamping must be enabled in the control register (Bit SCTSE of Register SCCR0). A read access to SCTSRLL requires a maximum of three AVR clocks. Reading the upper three bytes of the timestamp requires two CPU clock cycles for each byte.

Note that there is no separate interrupt provided for timestamping. Instead the TRX24_RX_START interrupt can be used (see "Interrupt Vectors in ATmega128RFA1" on page 211).

10.6 Symbol Counter Beacon Timestamp Register (32 bit, SCBTSR)

If timestamping is enabled in the SCCR register, the beacon timestamp register is updated with the SFD timestamp at the end of the received frame, if the received frame was a beacon frame with valid FCS and:

- Source PAN identifier == {PAN_ID_1, PAN_ID_0}

or

- {PAN_ID_1, PAN_ID_0} == 0xFFFF

PAN_ID_0 and PAN_ID_1 are register of the radio transceiver, see "PAN_ID_0 – Transceiver Personal Area Network ID Register (Low Byte)" on page 124.

Beacon timestamps can also be generated manually. Writing “1” to SCMBTS of Register SCCR0 captures the current symbol counter value and stores it in the beacon timestamp register. The bit is cleared automatically afterwards.

It is also possible to manually set the register in order to provide a distinct starting value for the relative compare modes (see next section).

10.7 Compare Unit (3x 32 bit, SCOCR1, SCOCR2, SCOCR3)

The compare unit contains 3 independent 32 bit compare modules and is used to compare the current counter value with the value stored in the compare register, and optionally the beacon timestamp register. There are two possible modes available which can be selected separately for all three compare modules:

1. Absolute Compare: In this mode the value stored in the compare register is compared directly with the symbol counter value (SCCNT == SCOCRx). If the values are equal an interrupt is generated.

2. Relative Compare: This mode allows the compare between the current symbol counter value and the compare value plus the beacon timestamp value (SCCNT == SCBTSR + SCOCRx). This mode can be used to generate an interrupt at a time offset relative to the value stored in the beacon timestamp register.

Note that a beacon timestamp is valid after a valid FCS. The relative compare must exceed the beacon length, otherwise no relative compare interrupt will occure.

10.8 Interrupt Control Registers

The interrupt status and mask registers control the interrupt generation. Each interrupt can be enabled in SCIRQM (Symbol Counter IRQ Mask Register). If an interrupt occurs, the appropriate interrupt flag within the interrupt status register is set regardless of the interrupt mask register setting. If the appropriate interrupt is enabled, an interrupt is generated.

The interrupt flags can be cleared either by:

1. Entering the respective interrupt handler, or
2. Writing “one” to the according interrupt flag in the interrupt status register.

All Interrupts can be used to wakeup the controller from any sleep state.

10.9 Backoff Slot Counter

The backoff slot counter can be used to provide accurate MAC protocol timing. The counter is sourced by the transceiver clock and works only if the transceiver clock is running. If the transceiver is disabled or in sleep mode the counter is also disabled.

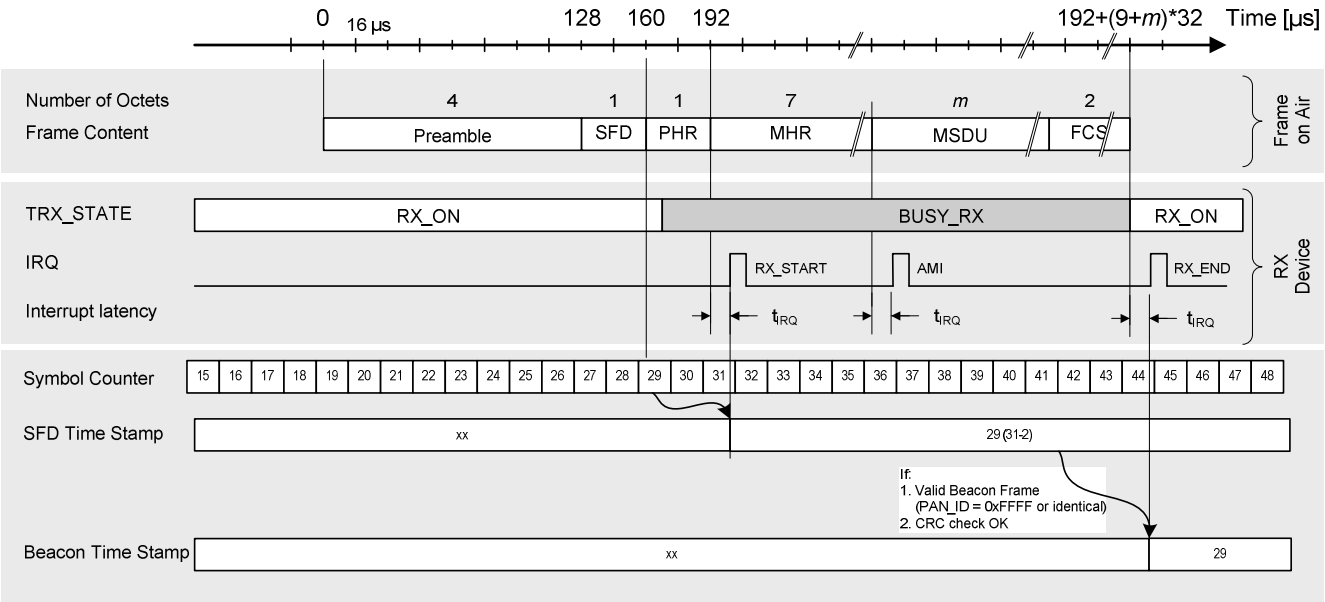
The counter generates periodic Interrupts every 20 symbols, i.e. every 320 µs.

10.10 Symbol Counter Usage

10.10.1 SFD and Beacon Timestamp Generation

The SFD timestamp register is updated with the symbol counter value at the time the SFD value has been received completely. For an incoming frame, the register is valid after the RX_START IRQ was issued until the next RX_START IRQ. SFD timestamps are generated for all incoming frames with valid SFD and length field even if the PSDU is corrupted (invalid FCS).

Figure 10-1. SFD and Beacon Timestamp Generation



Note that Figure 10-1 contains no exact timing information; it is for visualization only.

The beacon timestamp register is updated with the SFD timestamp value at the end of the frame (RX_END IRQ), if the received frame was a beacon frame with valid FCS and expected source PAN identifier or { PAN_ID_1, PAN_ID_0} = 0xFFFF.

The register value is valid until a new beacon frame has been received or the beacon timestamp is updated manually. A manual beacon timestamp can be generated by writing “1” to SCMBTS of the SCCR0 register.

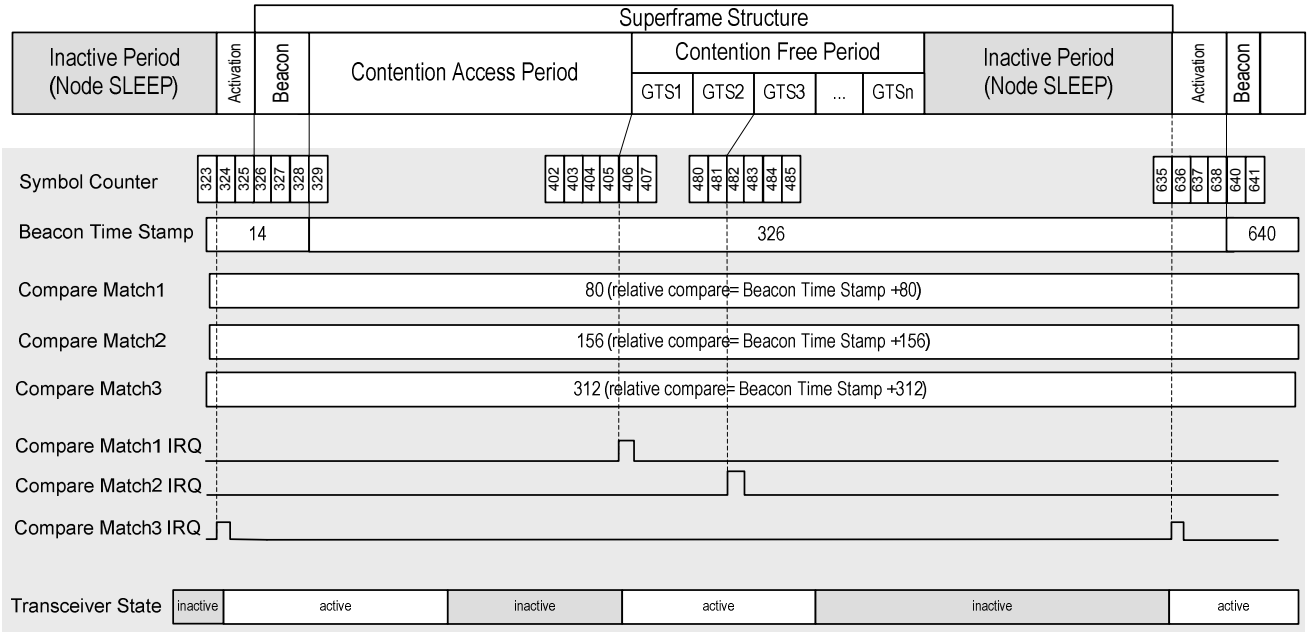
10.10.2 Relative Compare Mode for Superframe Access Timing

The IEEE 802.15.4 describes a superframe structure which contains different time slots where a device can access the channel.

The Symbol Counter together with the three compare units provide support for waking up the device at the right time to receive the beacon for superframe synchronization and at certain times within the superframe.

A typical superframe timing scenario using the symbol counter relative compare mode is shown in Figure 10-2 on page 137. The Symbol Counter values in the figure do not reflect realistic time intervals but demonstrate the principle of operation.

Figure 10-2. Relative Compare Mode



The compare match registers are programmed with symbol intervals relative to the beacon frame SFD timestamp. For instance the SCCMP1 is programmed to 80, because the first Granted Time Slot (GTS1) is expected 80 symbols after the beacon frame. Register SCCMP2 is programmed to 156 to meet GTS3 156 symbols after the beacon frame. SCCMP3 is programmed to 312. This is the time interval where the beacon of the next superframe is expected. Because it requires some time to activate the transceiver and there is also some timing drift possible, the compare interrupt must be programmed to wake up some symbols in advance to make sure the next beacon is not missed.

If the controller receives a compare match wake up event it is activating the transceiver. After the frame operations are finished, the system can go back to sleep until the next compare match event occurs.

10.11 Register Description

10.11.1 SCCNTHH – Symbol Counter Register HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E4)	SCCNTHH7:0								SCCNTHH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit Symbol Counter.

- **Bit 7:0 – SCCNTHH7:0 - Symbol Counter Register HH-Byte**

10.11.2 SCCNTHL – Symbol Counter Register HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E3)	SCCNTHL7:0								SCCNTHL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit Symbol Counter.

- **Bit 7:0 – SCCNTHL7:0 - Symbol Counter Register HL-Byte**

10.11.3 SCCNTLH – Symbol Counter Register LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E2)	SCCNTLH7:0								SCCNTLH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit Symbol Counter.

- **Bit 7:0 – SCCNTLH7:0 - Symbol Counter Register LH-Byte**

10.11.4 SCCNTLL – Symbol Counter Register LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E1)	SCCNTLL7:0								SCCNTLL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit Symbol Counter.

- **Bit 7:0 – SCCNTLL7:0 - Symbol Counter Register LL-Byte**

10.11.5 SCTSRHH – Symbol Counter Frame Timestamp Register HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($EC)	SCTSRHH7:0								SCTSRHH
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit frame (SFD) timestamp register

- **Bit 7:0 – SCTSRHH7:0 - Symbol Counter Frame Timestamp Register HH-Byte**

10.11.6 SCTSRHL – Symbol Counter Frame Timestamp Register HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($EB)	SCTSRHL7:0								SCTSRHL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit Frame (SFD) Timestamp Register

- **Bit 7:0 – SCTSRHL7:0 - Symbol Counter Frame Timestamp Register HL-Byte**

10.11.7 SCTSRLH – Symbol Counter Frame Timestamp Register LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($EA)	SCTSRLH7:0								SCTSRLH
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit Frame (SFD) Timestamp Register

- **Bit 7:0 – SCTSRLH7:0 - Symbol Counter Frame Timestamp Register LH-Byte**

10.11.8 SCTSRLL – Symbol Counter Frame Timestamp Register LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E9)	SCTSRLL7:0								SCTSRLL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit Frame (SFD) Timestamp Register

- **Bit 7:0 – SCTSRLL7:0 - Symbol Counter Frame Timestamp Register LL-Byte**

10.11.9 SCBTSRHH – Symbol Counter Beacon Timestamp Register HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E8)	SCBTSRHH7:0								SCBTSRHH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit Beacon Timestamp Register. The Beacon Timestamp Register is updated with the contents of the Frame Timestamp Register if the received frame was a valid beacon frame with matching source PAN identifier or register {PAN_ID_1, PAN_ID_0} = 0xFFFF.

- **Bit 7:0 – SCBTSRHH7:0 - Symbol Counter Beacon Timestamp Register HH-Byte**

10.11.10 SCBTSRHL – Symbol Counter Beacon Timestamp Register HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E7)	SCBTSRHL7:0								SCBTSRHL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit Beacon Timestamp Register.

- **Bit 7:0 – SCBTSRHL7:0 - Symbol Counter Beacon Timestamp Register HL-Byte**

10.11.11 SCBTSRLH – Symbol Counter Beacon Timestamp Register LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E6)	SCBTSRLH7:0								SCBTSRLH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit Beacon Timestamp Register.

- **Bit 7:0 – SCBTSRLH7:0 - Symbol Counter Beacon Timestamp Register LH-Byte**

10.11.12 SCBTSRLL – Symbol Counter Beacon Timestamp Register LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($E5)	SCBTSRLL7:0								SCBTSRLL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit Beacon Timestamp Register.

- **Bit 7:0 – SCBTSRLL7:0 - Symbol Counter Beacon Timestamp Register LL-Byte**

10.11.13 SCOCR1HH – Symbol Counter Output Compare Register 1 HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F8)	SCOCR1HH7:0								SCOCR1HH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit compare value for the first compare unit

- **Bit 7:0 – SCOCR1HH7:0 - Symbol Counter Output Compare Register 1 HH-Byte**

10.11.14 SCOCR1HL – Symbol Counter Output Compare Register 1 HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F7)	SCOCR1HL7:0								SCOCR1HL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit compare value for the first compare unit

- **Bit 7:0 – SCOCR1HL7:0 - Symbol Counter Output Compare Register 1 HL-Byte**

10.11.15 SCOCR1LH – Symbol Counter Output Compare Register 1 LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F6)	SCOCR1LH7:0								SCOCR1LH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit compare value for the first compare unit

- **Bit 7:0 – SCOCR1LH7:0 - Symbol Counter Output Compare Register 1 LH-Byte**

10.11.16 SCOCR1LL – Symbol Counter Output Compare Register 1 LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F5)	SCOCR1LL7:0								SCOCR1LL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit compare value for the first compare unit

- **Bit 7:0 – SCOCR1LL7:0 - Symbol Counter Output Compare Register 1 LL-Byte**

10.11.17 SCOCR2HH – Symbol Counter Output Compare Register 2 HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F4)	SCOCR2HH7:0								SCOCR2HH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit compare value for the second compare unit

- **Bit 7:0 – SCOCR2HH7:0 - Symbol Counter Output Compare Register 2 HH-Byte**

10.11.18 SCOCR2HL – Symbol Counter Output Compare Register 2 HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F3)	SCOCR2HL7:0								SCOCR2HL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit compare value for the second compare unit

- **Bit 7:0 – SCOCR2HL7:0 - Symbol Counter Output Compare Register 2 HL-Byte**

10.11.19 SCOCR2LH – Symbol Counter Output Compare Register 2 LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F2)	SCOCR2LH7:0								SCOCR2LH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit compare value for the second compare unit

- **Bit 7:0 – SCOCR2LH7:0 - Symbol Counter Output Compare Register 2 LH-Byte**

10.11.20 SCOCR2LL – Symbol Counter Output Compare Register 2 LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F1)	SCOCR2LL7:0								SCOCR2LL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit compare value for the second compare unit

- **Bit 7:0 – SCOCR2LL7:0 - Symbol Counter Output Compare Register 2 LL-Byte**

10.11.21 SCOCR3HH – Symbol Counter Output Compare Register 3 HH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($F0)	SCOCR3HH7:0								SCOCR3HH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the most significant byte of the 32 bit compare value for the third compare unit

- **Bit 7:0 – SCOCR3HH7:0 - Symbol Counter Output Compare Register 3 HH-Byte**

10.11.22 SCOCR3HL – Symbol Counter Output Compare Register 3 HL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($EF)	SCOCR3HL7:0								SCOCR3HL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second most significant byte of the 32 bit compare value for the third compare unit

- **Bit 7:0 – SCOCR3HL7:0 - Symbol Counter Output Compare Register 3 HL-Byte**

10.11.23 SCOCR3LH – Symbol Counter Output Compare Register 3 LH-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($EE)	SCOCR3LH7:0								SCOCR3LH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the second least significant byte of the 32 bit compare value for the third compare unit

- **Bit 7:0 – SCOCR3LH7:0 - Symbol Counter Output Compare Register 3 LH-Byte**

10.11.24 SCOCR3LL – Symbol Counter Output Compare Register 3 LL-Byte

Bit	7	6	5	4	3	2	1	0	
NA ($ED)	SCOCR3LL7:0								SCOCR3LL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register contains the least significant byte of the 32 bit compare value for the third compare unit

- **Bit 7:0 – SCOCR3LL7:0 - Symbol Counter Output Compare Register 3 LL-Byte**

10.11.25 SCCR0 – Symbol Counter Control Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($DC)	SCRES	SCMBTS	SCEN	SCCKSEL	SCTSE	SCCMP3	SCCMP2	SCCMP1	SCCR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Control Register 0 is used to setup the operating mode of the symbol counter and the compare units

- **Bit 7 – SCRES - Symbol Counter Synchronization**

If this bit is set to 1, the 16 MHz clock prescaler as well as the backoff slot counter is cleared. This function can be used to align the symbol timing within one 16 µs symbol period and to restart the backoff slot counter with a complete 320 µs period. This

feature works only if the symbol counter module operates with the 16 MHz clock from XTAL1. After switching to RTC clock source, the symbol period synchronization is lost. This bit is cleared automatically.

- **Bit 6 – SCMBTS - Manual Beacon Timestamp**

With this bit a manual beacon timestamp can be generated. If set to 1, the current symbol counter value is stored into the beacon timestamp register. The bit is cleared afterwards. The manual beacon timestamping can be used in conjunction with the relative compare mode of the three compare units to generate compare match interrupts without having a beacon frame received.

- **Bit 5 – SCEN - Symbol Counter enable**

This bit activates the symbol counter module. If the bit is not set, the counter, backoff slot counter and the compare unit are disabled and disconnected from the clock. In this way the power consumption can be reduced. All registers can be accessed, but write access to the counter register SCCNT is not possible.

- **Bit 4 – SCCKSEL - Symbol Counter Clock Source select**

With this bit the clock source for the symbol counter can be selected. If the bit is one, the RTC clock from TOSC1 is selected, otherwise the symbol counter operates with the clock from XTAL1. During transceiver sleep modes the clock falls back to the RTC clock source, regardless of the selected clock. After wakeup, it switches back to the previosly selected clock source.

- **Bit 3 – SCTSE - Symbol Counter Automatic Timestamping enable**

This bit enables automatic SFD and Beacon Timestamping. If the bit is zero, no automatic timestamp capturing is possible. Only manual beacon timestamping can be used.

- **Bit 2 – SCCMP3 - Symbol Counter Compare Unit 3 Mode select**

This bit enables the relative compare mode for compare unit 3. If enabled, the counter value is compared against the content of the beacon timestamp register plus the content of the compare register 3 (SCCNT == SCBTS+SCOCR3). Otherwise, the counter is compared against the copare register 3 (SCCNT == SCOCR3).

- **Bit 1 – SCCMP2 - Symbol Counter Compare Unit 2 Mode select**

This bit enables the relative compare mode for compare unit 2. If enabled, the counter value is compared against the content of the beacon timestamp register plus the content of the compare register 2 (SCCNT == SCBTS+SCOCR2). Otherwise, the counter is compared against the copare register 2 (SCCNT == SCOCR2).

- **Bit 0 – SCCMP1 - Symbol Counter Compare Unit 1 Mode select**

This bit enables the relative compare mode for compare unit 1. If enabled, the counter value is compared against the content of the beacon timestamp register plus the content of the compare register 1 (SCCNT == SCBTS+SCOCR1). Otherwise, the counter is compared against the copare register 1 (SCCNT == SCOCR1).

10.11.26 SCCR1 – Symbol Counter Control Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($DD)	Res6	Res5	Res4	Resx4	Resx3	Resx2	Resx1	SCENBO	SCCR1
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

This register is used to enable the backoff slot counter.

- **Bit 7:5 – Res6:4 - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 4:1 – Resx4:1 - Reserved**
- **Bit 0 – SCENBO - Backoff Slot Counter enable**

If this bit is set, the backoff slot counter starts working. To enable the corresponding IRQ the SCIRQM register must be updated.

10.11.27 SCSR – Symbol Counter Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($DE)	Res6	Res5	Res4	Res3	Res2	Res1	Res0	SCBSY	SCSR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 – Res6:0 - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 0 – SCBSY - Symbol Counter busy**

This bit is set if a write operation to the symbol counter register is pending. This bit is set after writing the counter low byte (SCCNTLL) until the symbol counter is updated with the new value. This update process can take up to 16 µs and during this time no read or write access to the 32 bit counter register should occure.

10.11.28 SCIRQS – Symbol Counter Interrupt Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($E0)	Res2	Res1	Res0	IRQSBO	IRQSOF	IRQSCP3	IRQSCP2	IRQSCP1	SCIRQS
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Interrupt Status Register indicates pending interrupt requests. If the corresponding interrupt mask bit is set, an interrupt service routine is called and the status bit is cleared automatically. It is also possible to clear the status bit by writing "1" to the selected bit.

- **Bit 7:5 – Res2:0 - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 4 – IRQSBO - Backoff Slot Counter IRQ**

This interrupt is generated every 320 µs, that means every 20 symbols.

- **Bit 3 – IRQSOF - Symbol Counter Overflow IRQ**

This interrupt is generated when the 32 bit counter turns from 0xFFFFFFF to 0x00000000.

- **Bit 2 – IRQSCP3 - Compare Unit 3 Compare Match IRQ**

This interrupt indicates a compare match on compare unit 3.

- **Bit 1 – IRQSCP2 - Compare Unit 2 Compare Match IRQ**

This interrupt indicates a compare match on compare unit 2.

- **Bit 0 – IRQSCP1 - Compare Unit 1 Compare Match IRQ**

This interrupt indicates a compare match on compare unit 1.

10.11.29 SCIRQM – Symbol Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($DF)	Res2	Res1	Res0	IRQMBO	IRQMOF	IRQMCP3	IRQMCP2	IRQMCP1	SCIRQM
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Interrupt Mask Register is used to enable corresponding interrupts. After reset all interrupts are disabled. Disabled interrupts are still captured in the interrupt status register SCIRQS, but no interrupt is requested. Before enabling an interrupt, the corresponding interrupt status bit should be cleared by writing a 1. If the status bit is set and the IRQ gets enabled, the IRQ handler is called immediatly.

- **Bit 7:5 – Res2:0 - Reserved Bit**

This bit is reserved for future use. The result of a read access is undefined. The register bit must always be written with the reset value.

- **Bit 4 – IRQMBO - Backoff Slot Counter IRQ enable**

This bit enables the SCNT_BACKOFF interrupt.

- **Bit 3 – IRQMOF - Symbol Counter Overflow IRQ enable**

This bit enables the SCNT_OVFL interrupt.

- **Bit 2 – IRQMCP3 - Symbol Counter Compare Match 3 IRQ enable**

This bit enables the SCNT_CMP3 interrupt.

- **Bit 1 – IRQMCP2 - Symbol Counter Compare Match 2 IRQ enable**

This bit enables the SCNT_CMP2 interrupt.

- **Bit 0 – IRQMCP1 - Symbol Counter Compare Match 1 IRQ enable**

This bit enables the SCNT_CMP1 interrupt.

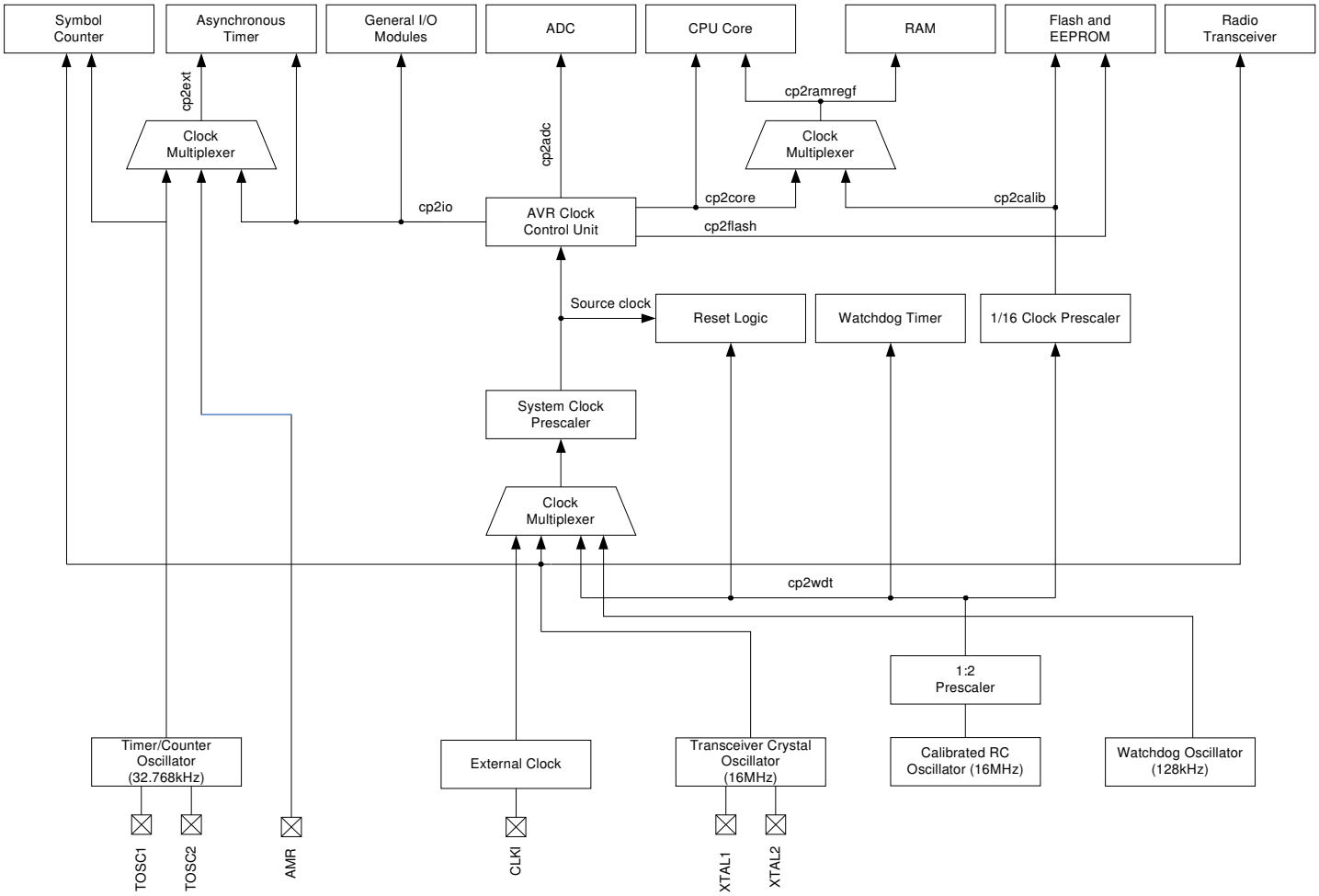
11 System Clock and Clock Options

This section describes the clock options for the AVR microcontroller.

11.1 Overview

Figure 11-1 below 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 chapter "Power Management and Sleep Modes" on page 156. The clock systems are detailed below.

Figure 11-1. Clock Distribution



11.2 Clock Systems and their Distribution

11.2.1 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.

11.2.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. 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. Also note that start condition detection in the 2-wire serial interface (TWI) module is carried out asynchronously when $clk_{I/O}$ is halted. Similar the TWI address recognition in all sleep modes also occurs asynchronously.

11.2.3 Flash Clock – clk_{FLASH}

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

11.2.4 Asynchronous Timer Clock – clk_{ASY}

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external clock or an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even if the device is in sleep mode.

11.2.5 ADC Clock – clkADC

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

11.3 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 11-1. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3:0
Transceiver clock	1111 – 0110
Reserved	0101 - 0100
Internal 128 kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Notes: 1. For all fuses “1” means unprogrammed while “0” means programmed.

11.3.1 Default Clock Source

The device is shipped with internal RC oscillator at 16.0 MHz, the 1:2 prescaler enabled and with the fuse CKDIV8 programmed, resulting in 1.0 MHz system clock. The startup time is set to maximum time. (CKSEL = "0010", SUT = "10", CKDIV8 = "0"). The default setting ensures that all users can make their desired clock source setting using any available programming interface.

11.3.2 Clock Start-up Sequence

Any clock source needs a minimum number of oscillating cycles before it can be considered stable.

To ensure sufficient startup time, the device issues an internal reset with a time-out delay (t_{TOUT}) after the device reset is released by all other reset sources. Section "Power-on Reset" on page 177 describes the start conditions for the internal reset. The delay (t_{TOUT}) is timed from the Watchdog Oscillator and the number of cycles in the delay is set by the SUTx and CKSELx fuse bits. The selectable delays are shown in Table 11-2 below. The frequency of the Watchdog Oscillator is voltage dependent as shown in section "Typical Characteristics" on page 510.

Table 11-2. Number of Watchdog Oscillator Cycles

Typ Time-out	Number of Cycles
0 ms	0
4.0 ms	512
64 ms	8K (8,192)

Main purpose of the delay is to keep the AVR in reset until it is supplied with a stable V_{DEVDD}. The delay will not monitor the actual voltage and it will be required to select a delay longer than the DEVDD rise time. If this is not possible, an internal or external Brown-Out Detection (BOD) circuit should be used. A BOD circuit will ensure sufficient V_{DEVDD} before it releases the reset, and the time-out delay can be disabled. Disabling the time-out delay without utilizing a Brown-Out Detection circuit is not recommended.

The oscillator is required to oscillate for a minimum number of cycles before the clock is considered stable. An internal ripple counter monitors the oscillator output clock, and keeps the internal reset active for a given number of clock cycles. The reset is then released and the device will start to execute. The recommended oscillator start-up time is dependent on the clock type, and varies from 6 cycles for an externally applied clock to 32K cycles for a low frequency crystal.

The start-up sequence for the clock includes both the time-out delay and the start-up time when the device starts up from reset. When starting up from Power-save or Power-down mode, DEVDD is assumed to be at a sufficient level and only the start-up time is included.

11.4 Calibrated Internal RC Oscillator

By default, the Internal RC Oscillator provides an approximate 16 MHz clock. The RC oscillator is voltage and temperature dependent, but can be very accurately calibrated by the user. See chapter "Clock Characteristics" on page 502 and "Internal Oscillator Speed" on page 510 for more details. The device is shipped with the CKDIV8 Fuse and the 1:2 system clock prescaler programmed. See section "System Clock Prescaler" on page 152 for more details.

This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 11-3 on page 150. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. The accuracy of this calibration is shown as Factory calibration in section "Clock Characteristics" on page 502.

By changing the OSCCAL register (see "OSCCAL – Oscillator Calibration Value" on page 153) from Software, it is possible to get a higher calibration accuracy than by using the factory calibration. The accuracy of this calibration is shown as User calibration in section "Clock Characteristics" on page 502.

When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section "Calibration Byte" on page 467.

Table 11-3. Internal Calibrated RC Oscillator Operating Modes[1][2]

Frequency Range (MHz)	CKSEL3:0
9.6 ... 22.4	0010

Notes: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in the following table.

Table 11-4. Start-up times for the internal calibrated RC Oscillator clock selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset	SUT1:0
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4.0 ms	01
Slowly rising power	6 CK	14CK + 64 ms[1]	10
Reserved			11

Notes: 1. The device is shipped with this option selected

11.5 128 kHz Internal Oscillator

The 128 kHz Internal Oscillator is an ultra-low power RC oscillator providing a clock of approximate 128 kHz nominal frequency. This clock may be selected as the system clock by programming the CKSEL Fuses to “0011” as shown in the following table.

Table 11-5. 128 kHz Internal Oscillator Operating Modes[1]

Nominal Frequency	CKSEL3:0
128 kHz	0011

Notes: 1. Note that the 128 kHz oscillator is a very low power clock source, and is not designed for high accuracy

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in the following table.

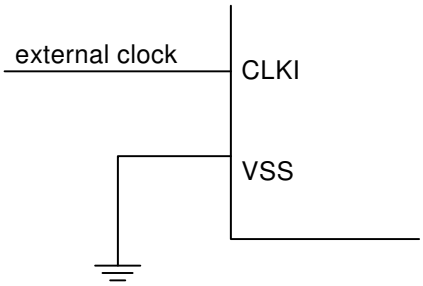
Table 11-6. Start-up Times for the 128 kHz Internal Oscillator

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset	SUT1:0
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4.1 ms	01
Slowly rising power	6 CK	14CK + 64 ms	10
Reserved			11

11.6 External Clock

To drive the device from an external clock source, CLKI should be used as shown in Figure 11-2 on page 151. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000”.

Figure 11-2. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 11-8 below.

Table 11-7. External Clock Frequency

Nominal Frequency	CKSEL3:0
0 – 16 MHz	0000

Table 11-8. Start-up Times for the External Clock Selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset	SUT1:0
BOD enabled	6 CK	14 CK	00
Fast rising power	6 CK	14 CK + 4.0 ms	01
Slowly rising power	6 CK	14 CK + 64 ms	10
Reserved			11

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the microcontroller unit (MCU). A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. If changes of more than 2% are required, ensure that the MCU is kept in Reset during the changes.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to section "System Clock Prescaler" on page 152 for details.

11.7 Transceiver Crystal Oscillator

The integrated crystal oscillator for the radio transceiver generates a low-jitter 16MHz clock frequency. See section "Crystal Oscillator (XOSC)" on page 80 for details about the operation of this oscillator. The AVR core and the radio transceiver operate synchronously on the same clock if this oscillator is selected. If the transceiver crystal oscillator is selected as AVR core clock, it remains enabled even if the radio transceiver is in SLEEP mode or its power reduction bit PRTRX24 is set.

Table 11-9. Transceiver Crystal Clock Operating Mode

Frequency Range (MHz)	CKSEL3:0(1)
16	1111 - 0110

Notes: 1. All CKSEL fuse values have the same significance.

Table 11-10. Start-up Times for the Transceiver Oscillator Clock Selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset	CKSEL0	SUT1:0
fast rising power	258 CK	14CK + 4.1 ms	0	00
slowly rising power	258 CK	14CK + 65 ms	0	01
BOD enabled	1K CK	14CK + 0 ms	0	10
fast rising power	1K CK	14CK + 4.1 ms	0	11
slowly rising power	1K CK	14CK + 65 ms	1	00
BOD enabled	16K CK	14CK + 0 ms	1	01
fast rising power	16K CK	14CK + 4.1 ms	1	10
slowly rising power	16K CK	14CK + 65 ms	1	11

11.8 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT Fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock also will be output during reset, and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC Oscillator, can be selected when the clock is output on CLKO. If the System Clock Prescaler is used, it is the divided system clock that is output.

Special attention is required to prevent unwanted radiation from the connected PCB clock trace. Proper filtering can help to suppress higher harmonics.

11.9 Timer/Counter Oscillator

The device can operate the Timer/Counter2 from the 32.768 kHz crystal oscillator or an external clock source. See section "Application Circuits" on page 493 for the watch crystal connection.

11.10 System Clock Prescaler

The ATmega128RFA1 has a system clock prescaler, and the system clock can be divided by setting the “CLKPR – Clock Prescale Register”. This feature can be used to decrease the system clock frequency and the power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. The clocks $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in CLKPR – Clock Prescale Register on page 154.

The prescaler clock division factor of the internal RC-Oscillator is different from all other clock sources, see register description CLKPR – Clock Prescale Register on page 154

Flash, EEPROM, Fuse- and Lock-bit programming is not allowed while using RC-Oscillator with CLKPS=0xF (clk_{CPU} = 16MHz).

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occur 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 prescaler is implemented as a ripple counter running at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not

possible to determine the state of the prescaler - even if it were readable. The exact time it takes to switch from one clock division to another cannot be exactly predicted. From the time the CLKPS values are written, it takes between $t_1 + t_2$ and $t_1 + 2t_2$ before the new clock frequency is active. In this interval 2 active clock edges are produced. Here t_1 is the previous clock period and t_2 is the clock 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 settings to make sure the write procedure is not interrupted.

It is not required to change the prescaler setting of an existing software package written for an 8MHz internal RC oscillator. The change of the prescaler (additional 1:2 divider) is compensated by doubling the RC oscillator frequency of the ATmega128RFA1.

11.11 Register Description

11.11.1 OSCCAL – Oscillator Calibration Value

Bit	7	6	5	4	3	2	1	0	
NA ($66)	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. A preprogrammed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in the section "Electrical Characteristics". Calibration outside that range is not guaranteed. Note that this oscillator is used to time EEPROM and Flash write accesses and these write times will be affected accordingly. The calibration to very high frequencies can cause EEPROM or Flash erase/write failures. The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80. The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range.

- **Bit 7:0 – CAL7:0 - Oscillator Calibration Tuning Value**

Table 11-11 CAL Register Bits

Register Bits	Value	Description
CAL7:0	0x00	Calibration value for lowest oscillator frequency
	0x7f	End value of low frequency range calibration
	0x80	Start value of high frequency range calibration

Register Bits	Value	Description
	0xff	Calibration value for highest oscillator frequency

11.11.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
NA ($61)	CLKPCE	Res2	Res1	Res0	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
Read/Write	RW	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **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, nor clear the CLKPCE bit.

- **Bit 6:4 – Res2:0 - Reserved**
- **Bit 3:0 – CLKPS3:0 - Clock Prescaler Select Bits**

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 the following table. Note that the factor is different when using the internal 16MHz RC oscillator as the clock source. The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is not programmed, the CLKPS bits will be reset to 0000. If CKDIV8 is programmed, CLKPS bits are reset to 0011 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 11-12 CLKPS Register Bits

Register Bits	Value	Description
CLKPS3:0	0x0	Division factor 1 / RC-Oscillator 2
	0x1	Division factor 2 / RC-Oscillator 4
	0x2	Division factor 4 / RC-Oscillator 8
	0x3	Division factor 8 / RC-Oscillator 16
	0x4	Division factor 16 / RC-Oscillator 32
	0x5	Division factor 32 / RC-Oscillator 64
	0x6	Division factor 64 / RC-Oscillator 128
	0x7	Division factor 128 / RC-Oscillator 256
	0x8	Division factor 256 / RC-Oscillator 512
	0x9	Reserved
	0xA	Reserved

Register Bits	Value	Description
	0xB	Reserved
	0xC	Reserved
	0xD	Reserved
	0xE	Reserved
	0xF	Division factor 1 only permitted for RC-Oscillator. Flash and EEPROM programming is not allowed.

12 Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR microcontroller and the RF transceiver provide various sleep modes allowing the user to tailor the power consumption to the application's requirements.

12.1 Deep-Sleep Mode

When the microcontroller goes into Power-down or Power-save modes while the transceiver is in SLEEP state the device enters the Deep-Sleep mode.

Sending the microcontroller to Power-down or Power-save is not allowed during the wake-up phase of the transceiver. The TRX24_AWAKE interrupt shall be used to wait for the transceiver is operational.

The DVDD voltage regulator and the associated power chain will be switched off. Remaining running logic will then be supplied from the Low Leakage Voltage Regulator. Even the AVDD regulator will switched off. See chapter "Radio Transceiver" on page 160 how to disable the radio transceiver.

The SRAM blocks use the data retention mode to preserve its content while saving leakage power. The Low Leakage Voltage Regulator has only limited driving capabilities, see section "Supply Voltage and Leakage Control" on page 161 for details. Therefore the remaining running logic must be clocked with low frequencies only.

The Deep-Sleep mode can be finished by a wake-up source shown by the Table 12-1 below. Then DVDD voltage regulator and the associated power chain will be switched on. If the power-chain is completely enabled the standard AVR wake-up procedure continues (for details see chapter "Power-chain" on page 161).

Note that the wake-up time from Deep-sleep mode is significantly longer than the wake-up time from the Power-down or Power-save mode because the entire power-chain will be restarted.

Additionally note that if the ADC is enabled and/or running a conversion, while entering Deep-sleep mode, the ADC supply voltage is switched off. Therefore the ADC must be disabled before entering Deep-sleep mode to avoid an undefined ADC operation.

12.2 AVR Microcontroller Sleep Modes

In chapter "System Clock and Clock Options" on page 147 the different clock systems in the ATmega128RFA1, and their distribution were presented. Figure 11-1 on page 147 is helpful in selecting an appropriate sleep mode. The following table shows the different sleep modes and their wake-up sources.

Table 12-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators		Wake-up Sources								
Sleep Mode	**clkCPU**	**clkFLASH**	**clkIO**	**clkADC**	**clkASY**	**Main Clock-source Enabled**	**Timer Oscillator Enabled**	**INT7:0 and Pin Change**	**TWI Address Match**	**Timer/Counter2**	**SPM/EEPROM Ready**	**ADC**	**WDT Interrupt**	**Other I/O**	**Symbol Counter**	**Transceiver**
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		X	X

	Active Clock Domains					Oscillators		Wake-up Sources								
Power-down								X(3)	X				X		X	X
Power-save					X		X(2)	X(3)	X	X			X		X	X
Standby (1)						X		X(3)	X				X		X	X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X	X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.

To enter any of the sleep modes, the SE bit in in the SMCR register (see "SMCR – Sleep Mode Control Register" on page 166) must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the SMCR Register select which sleep mode will be activated by the SLEEP instruction. See chapter "Register Description" on page 166 for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. Note that SRAM data retention must be enabled in some sleep modes to preserve the memory contents (see section "SRAM with Data Retention" on page 163). If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

12.2.1 Idle Mode

When the SM2:0 bits are written to 000 in the SMCR register, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the SPI, USART, Analog Comparator, ADC, 2-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system 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 and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

12.2.2 ADC Noise Reduction Mode

When the SM2:0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode (ADCNRM), stopping the CPU but allowing the ADC, the external interrupts, 2-wire Serial Interface address match, Timer/Counter2 and the Watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart form the ADC Conversion Complete interrupt, only an External Reset, a Watchdog System Reset, a Watchdog interrupt, a Brown-out Reset, a 2-wire serial interface interrupt, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt,

an external level interrupt on INT7:4 or a pin change interrupt can wakeup the MCU from ADC Noise Reduction mode.

12.2.3 Power-down Mode

When the SM2:0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the 16 MHz crystal oscillator is stopped (if selected by CKSEL fuses), while the external interrupts, the 2-wire Serial Interface, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, 2-wire Serial Interface address match, an external level interrupt on INT7:4, an external interrupt on INT3:0, a pin change interrupt, or a symbol counter interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to section "External Interrupts" on page 218 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after have been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period, as described in chapter "System Clock and Clock Options" on page 147.

12.2.4 Power-save Mode

When the SM2:0 bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is enabled, it will keep running during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK2, and the Global Interrupt Enable bit in SREG is set. If Timer/Counter2 is not running, Power-down mode is recommended instead of Power-save mode.

The Timer/Counter2 can be clocked both synchronously and asynchronously in Power-save mode. If the Timer/Counter2 is not using the asynchronous clock, the Timer/Counter Oscillator is stopped during sleep. If the Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Note that even if the synchronous clock is running in Power-save, this clock is only available for the Timer/Counter2. Timer/Counter2 operation is described in detail in section "8-bit Timer/Counter2 with PWM and Asynchronous Operation" on page 309.

12.2.5 Standby Mode

When the SM2:0 bits are 110 and the crystal oscillator of the radio transceiver is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

12.2.6 Extended Standby Mode

When the SM2:0 bits are 111 and the crystal oscillator of the radio transceiver is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-save mode with the exception that the oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles.

12.3 Power Reduction Register

The Power Reduction Register (PRR), see "PRR0 – Power Reduction Register0" on page 167, "PRR1 – Power Reduction Register 1" on page 168 and "PRR2 – Power Reduction Register 2" on page 168, provide 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 unit 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 the shutdown. Exceptions are the SRAM blocks and the radio transceiver. The SRAM is shut down by a DRT switch and the radio transceiver is in reset state if its respective power reduction bit is set.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See chapter "Typical Characteristics" on page 510 for examples. In all other sleep modes, the clock is already stopped.

12.4 Minimizing Power Consumption

There are several issues to consider when trying minimizing 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.

12.4.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. Refer to chapter "ADC – Analog to Digital Converter" on page 410 for details on ADC operation.

12.4.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode the Analog Comparator should also be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "AC – Analog Comparator" on page 407 for details on how to configure the Analog Comparator.

12.4.3 Brown-out Detector

If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be disabled in Deep-sleep mode. Refer to "Brown-out Detection" on page 178 for details on how to configure the Brown-out Detector. It is recommended to enable the Brown-out Detector.

12.4.4 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and not consume power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be

used immediately. Refer to "Internal Voltage Reference" on page 179 for details on the start-up time.

12.4.5 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 180 for details on how to configure the Watchdog Timer.

12.4.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section "I/O-Ports" on page 186 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 DEVDD/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 DEVDD/2 on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the Digital Input Disable Registers DIDR1 and DIDR0. Refer to "DIDR1 – Digital Input Disable Register 1" on page 409 and "DIDR0 – Digital Input Disable Register 0" on page 433 for details.

12.4.7 On-chip Debug System

If the On-chip debug system is enabled by the OCDEN Fuse and the chip enters sleep mode, the main clock source is enabled, and hence, always consumes power. In the deeper sleep modes, this will contribute significantly to the total current consumption. There are three alternative ways to disable the OCD system:

- Disable the OCDEN Fuse.
- Disable the JTAGEN Fuse.
- Write one to the JTD bit in MCUCR.

12.4.8 Symbol Counter

The Symbol Counter acts as a separate counter, which uses either the 16MHz clock from XTAL1/XTAL2 crystal pins or the clock from PG3/PG4 low frequency crystal pins. If the Symbol Counter module is not used, it should be disabled, see section "MAC Symbol Counter" on page 133.

12.4.9 Radio Transceiver

The radio transceiver module is automatically starting its state machine after power on. While the CPU is in any sleep mode, the radio transceiver remains active. This enables the radio transceiver to wakeup the MCU if a pending action is over (frame received or transmission completed). The radio transceiver will be inactive during sleep, if either the its power reduction bit PRTRX24 in register PRR1 is set or it is send into SLEEP mode, see "PRR1 – Power Reduction Register 1" on page 168 for details.

The radio transceiver is derived from a stand alone solution that was partly controlled by external pins. Now the radio transceiver is fully controlled by individual register bits.

The radio transceiver has a separate reset signal. A radio transceiver reset is initiated by setting bit TRXRST in register TRXPR. This bit is self-resetting.

The radio transceiver signal SLPTR can be controlled by the bit SLPTR in register TRXPR and is used to set the radio transceiver into SLEEP mode (assuming TRX_STATE is TRX_OFF). This bit has a multiple function, see section "Low-Power 2.4 GHz Transceiver" on page 29 for a detailed description of the radio transceiver.

12.5 Supply Voltage and Leakage Control

For battery applications using deep sleep periods, the leakage current defines the system life time. Due to the typical strong temperature dependency of the leakage current, major contributors to the leakage budget are turned off:

- Analog and digital voltage regulator,
- Non-volatile memory (NVM),
- SRAM,
- Digital signal processor of the radio transceiver including AES engine.

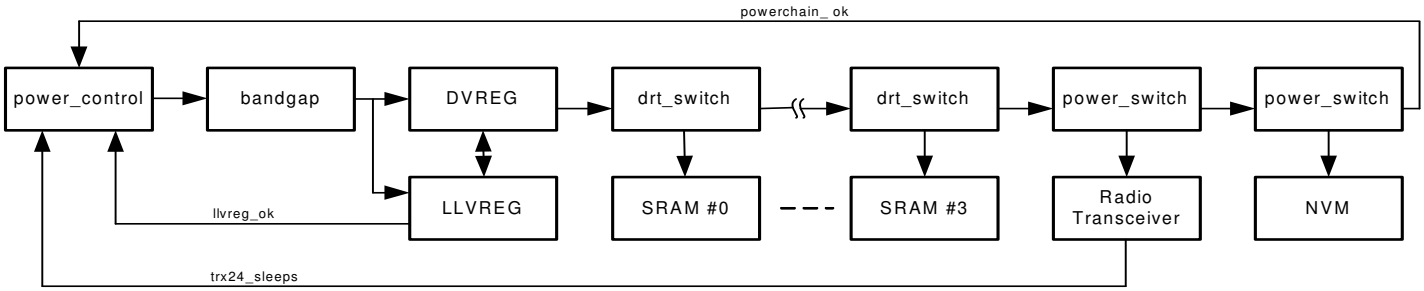
If the CPU uses one of the sleep modes “power-down” or “power-save”, the above mentioned blocks will be switched off by power switches. When the CPU wakes up, the blocks are switched on again. There are some additional exceptions (internal voltage regulator, SRAM, radio transceiver), see section "Power-chain" below .

The supply voltage control is mainly hidden to the application, it is not necessary to configure the supply voltage control. Nevertheless some configurations can be done in order to get the maximum effect and the lowest sleep current, for details see section "SRAM with Data Retention" on page 163.

12.5.1 Power-chain

The following figure shows the major dependencies of the power-chain and how the power switches are situated inside the chain.

Figure 12-1. Power-chain connections



Startup and Wakeup from deep sleep

After power-on reset (POR) or wakeup from deep sleep the power switches of the blocks will be enabled one after another (power-chained) to decrease current peaks. The blocks will be enabled in the following order:

1. Bandgap reference and voltage regulator,
2. Digital voltage regulator (DVREG) and low leakage voltage regulator (LLVREG),
3. SRAM block #0 (lower 4k bytes),
4. SRAM block #1,
5. SRAM block #2,

6. SRAM block #3 (upper 4k bytes),
7. Radio transceiver including AES engine,
8. Non-volatile memory.

If the power-chain is completely enabled the standard AVR wake-up procedure continues.

Figure 12-2 shows the chained startup procedure after power up. A module is only switched on if it is not deselected by power reduction register (PRR1 or PRR2). This is possible for SRAM blocks and radio transceiver power switch. At the end of the startup, the pin RSTON is enabled.

Figure 12-3 shows a similar procedure, the startup from deep sleep.

For further timing information see "Power Management Electrical Characteristics" on page 503.

Figure 12-2. Timing visualization of power up

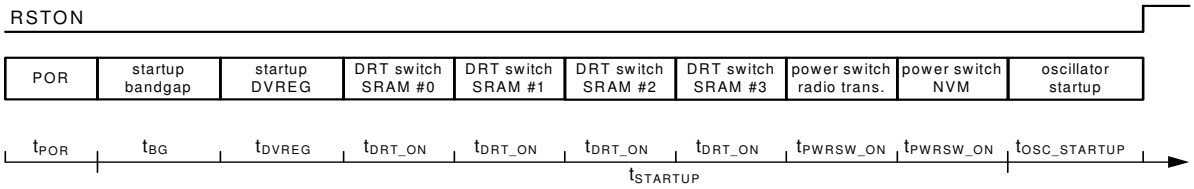
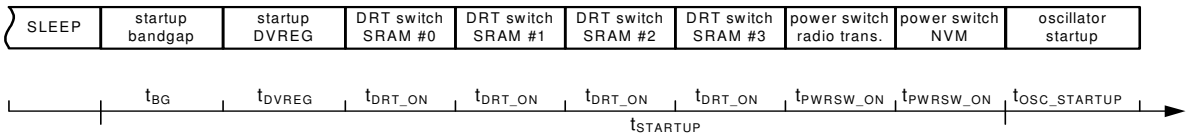


Figure 12-3. Timing visualization of wakeup from deep sleep



Sleep

Six sleep modes are defined for the CPU. Disabling the power-chain and thus switching off of the above mentioned blocks makes only sense for the modes “power-down” and “power-save”. Also an enabled radio transceiver prevents the power-chain from being disabled.

In order to disable the power-chain, one of the following conditions must fit:

- The radio transceiver has to be disabled (power reduction register PRR1 bit PRTRX24).
- The radio transceiver is sent into SLEEP mode (register TRXPR bit SLPTR).

The SRAM blocks may be configured separately to decrease their leakage current (see section "SRAM with Data Retention" on page 163).

The following table shows the different implemented sleep modes and the behavior of the power-chain depending on the current state of the radio transceiver.

Table 12-2. Power states of microcontroller and radio transceiver

AVR State	Radio Transceiver State	Powerchain
ON	ON	ON
ON	off (SLEEP or power reduction)	ON
off (1...6)	ON	ON
off (1,4...6)	off (SLEEP or power reduction)	ON

AVR State	Radio Transceiver State	Powerchain
off [2,3] DEEP SLEEP	off (SLEEP or power reduction)	off [7]

Notes:
1. Idle
2. Power Down
3. Power Save
4. ADC Noise Reduction Mode
5. Standby
6. Extended Standby
7.

12.5.2 SRAM with Data Retention

It is necessary to prevent any data loss of the SRAM when setting the CPU in one of the sleep modes. For that purpose the SRAM blocks will not be completely switched off if the power-chain is disabled. Instead the supply voltage for any individual SRAM block is decreased to reduce its leakage current but guaranteeing its data retention.

The SRAM memory is divided into four separate blocks. Each block can be fully switched off by setting the correspondent bit (PRRAM0 ... PRRAM3) in register PRR2 (see "PRR2 – Power Reduction Register 2" on page 168). This enables the application software to switch off unused SRAM memory to save power and to reduce leakage currents.

Every SRAM block can be enabled again by resetting the respective bit (PRRAM0 ... PRRAM3) of register PRR2. For each SRAM block *n* the bit DRTSWOK of the corresponding register DRTRAM*n* shows the state of the DRT switch (logic high means SRAM block can be accessed).

If the power-chain is switched off during deep-sleep modes, the content of the SRAM blocks must be sustained. To provide data retention and lowest leakage current, a data retention block controls the SRAM behavior during deep-sleep. Since the leakage current is dramatically depending from the voltage of the SRAM, the supply voltage can be decreased by enabling the data retention mode DRT.

Every SRAM block *n* is controlled by its assigned register DRTRAM*n*. The bit ENDRT enables the data retention mode during deep-sleep. If this bit is zero, the respective SRAM block is completely switched off.

Table 12-3. SRAM behavior while in deep-sleep mode

ENDRT	Power-chain	SRAM supply voltage
1	ON	1.8V (DVDD)
0	ON	1.8V (DVDD)
1	off	Reduced
0	off	Disconnected

The lower 4-bit of the register DRTRAM*n* are reserved and should not be changed. The reset value of the DRT voltage settings are preprogrammed during the manufacturing process and need not to be changed.

12.5.3 Voltage Regulators (AVREG, DVREG)

The main features of the Voltage Regulator blocks are:

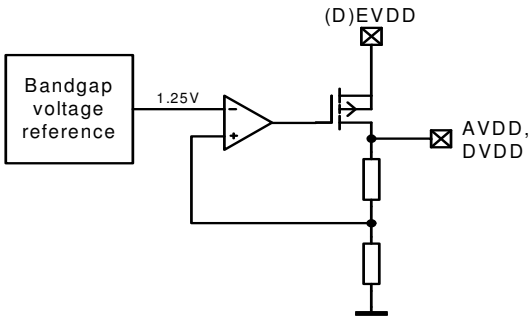
- Bandgap stabilized 1.8V supply for analog and digital domain;

- Low dropout (LDO) voltage regulator;
- Configurable to use an external voltage regulator;

The internal voltage regulators supply a stabilized voltage to the ATmega128RFA1. The AVREG provides the regulated 1.8V supply voltage for the analog section and the DVREG supplies the 1.8V supply voltage for the digital section. The DVREG is enabled during startup and is switched off if the power-chain is disabled. The AVREG is enabled only on request by either the A/D converter or the radio transceiver.

A simplified schematic of the internal voltage regulator is shown in Figure 12-4 below.

Figure 12-4. Simplified Schematic of AVREG/DVREG



The voltage regulators require bypass capacitors for stable operation. The value of the bypass capacitors determines their settling time. The bypass capacitors shall be placed as close as possible to the pins and shall be connected to ground with the shortest possible traces.

The voltage regulators can be configured with the register VREG_CTRL. It is recommended to use the internal regulators but it is also possible to supply the low voltage domains by an external voltage supply. For this configuration the internal regulators need to be switched off by setting the register bits to the values AVREG_EXT = 1 and DVREG_EXT = 1 (see "VREG_CTRL – Voltage Regulator Control and Status Register" on page 115). A regulated external supply voltage of 1.8V must then be connected to the pins 13, 14 (DVDD) and pin 29 (AVDD). When turning on the external supply ensure a sufficiently long stabilization time before interacting with the ATmega128RFA1.

The status bits AVDD_OK = 1 and DVDD_OK = 1 of register VREG_CTRL indicate an enabled and stable internal supply voltage. Reading value 0 indicates that the internal supply voltage is disabled or not yet settled to the final value.

Note that disabling the digital voltage regulator by DVREG_EXT bit is for internal use only.

12.5.4 Low Leakage Voltage Regulator (LLVREG)

The main digital voltage regulator (DVREG) will be switched off during the deep sleep modes “power-down” and “power-save”. The Low Leakage Voltage Regulator will then keep the digital supply voltage to provide data retention. No application software control is required.

During the active power states, when the main voltage regulator supplies the chip, the Low Leakage Voltage Regulator is digitally calibrated. Its output voltage is adjusted to match the output voltage of the main regulator. This fixed calibration result is stored and used when the chip enters a power-down state where the main regulator is switched off.

Because the calibration setting is fixed, temperature and load current variations during the following Deep Sleep period are not regulated out. Thus the output voltage may drift away from the target value. However the design guarantees that for allowed operating conditions the output voltage will stay within valid limits. After every wake-up a new calibration cycle is initiated.

The output driving capability of the Low Leakage Voltage Regulator is limited. Its main purpose is to provide the leakage current of the connected analog and digital blocks.

At least one full calibration cycle of the Low Leakage Voltage Regulator has to be completed before the power-chain can be disabled. Therefore if the CPU uses one of the deep sleep modes “power down” or “power save”, the power-chain is not disabled before the Low Leakage Voltage Regulator completed this first calibration cycle.

By default the LLVREG automatically starts the calibration after finishing the power-on reset and the wake-up/start-up procedures (see section "Low Leakage Voltage Regulator Control" below for a detailed description of the Low Leakage Voltage Regulator).

12.5.5 Low Leakage Voltage Regulator Control

The three register LLCR, LLDRL and LLDRH allow the software to monitor the calibration process and to modify or correct the calibration results. The automatic calibration is the normal operation mode. It is an internal process that does not require any software interaction. Nevertheless the calibration is transparent for the user through LLCR, LLDRL and LLDRH (control and data register respectively).

Before the device can enter the sleep mode “power down” or “power save” the first calibration cycle of the Low Leakage Voltage Regulator must be completed to get valid data in LLDRL and LLDRH. The cycle time $t_{LLVREG\ CALIB}$ is not fixed. It depends on the temperature, manufacturing process and the frequency of the 128 kHz RC oscillator (independent of the Watchdog setting).

Systems that require very short power-up times may temporarily disable the calibration process by setting bit LLENCAL to 0.

The output voltage of the Low Leakage Voltage Regulator in sleep mode will be the most accurate if constantly calibrated to compensate for any environmental changes (e.g. temperature). However these changes may be slow enough to skip the calibration during some power-up cycles (e.g. calibrate only every 10th power-up time and use the old calibration results during all other times).

After the completion of the power-up process the calibration will start automatically if bit LLENCAL in the control register LLCR is 1 (default). The completion of a calibration cycle is indicated by the bit LLDONE in that same register. After the first cycle the calibration will continue to run until either the device goes into a sleep mode (“power down” or “power save”) or by setting the LLENCAL bit to 0. The output voltage of the Low Leakage Voltage Regulator is then defined by the values in the data register LLDRL and LLDRH and by the bits LLTCO and LLSHORT of the control register.

Write access to the three register is granted when the bit LLENCAL is set to 0. The application software can then modify the calibration results. Higher values in the data register generate lower output voltages in the sleep modes. In general it is not recommended nor required to alter the automatically generated calibration result.

The write access to the three register must follow a certain scheme to be successful. The registers are implemented in the I/O clock domain while the logic of the Low Leakage Voltage Regulator runs with 64 kHz (clock output of the 128 kHz RC oscillator divided by 2). It takes at least two 64 kHz clock cycles before the data written to the

register take effect in the regulator circuit. The write access from the software must be aware of this process. Furthermore the value of LLDRH must be written first followed by LLDRL. Otherwise the LLDRH write access will be ignored. The following Assembler code fragment shows a working example. Note the polling of bit 3 *LLCAL* of the LLCR register to verify the completion of the synchronization process.

Assembly Code Example

```
    …
    clr r20
    IOST LLDRH,r18    ; write LLDRH first
    IOST LLDRL,r19    ; write LLDRL second
    IOST LLCR,r20     ; bit 0 cleared = disable automatic calibration
; poll LLCAL bit of LLCR to check if automatic calibration is
; turned of
wait_calib:
    IOLD r20,LLCR
    sbrc r20,3
    rjmp wait_calib   ; not executed if bit 3 of LLCR is cleared
    …
```

12.6 Register Description

12.6.1 SMCR – Sleep Mode Control Register

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	Res3	Res2	Res1	Res0	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Sleep Mode Control Register contains control bits for power management.

- **Bit 7:4 – Res3:0 - Reserved**
- **Bit 3:1 – SM2:0 - Sleep Mode Select bit 2**

These bits select between the five available sleep modes. Standby modes are only recommended for use with external crystals or resonators.

Table 12-4 SM Register Bits

Register Bits	Value	Description
SM2:0	0x00	Idle
	0x01	ADC Noise Reduction (If Available)
	0x02	Power Down
	0x03	Power Save
	0x04	Reserved
	0x05	Reserved
	0x06	Standby
	0x07	Extended Standby

- **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 programmers 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.

12.6.2 PRR0 – Power Reduction Register0

Bit	7	6	5	4	3	2	1	0	
NA ($64)	PRTWI	PRTIM2	PRTIM0	PRPGA	PRTIM1	PRSPI	PRUSART0	PRADC	PRR0
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 – PRTWI - Power Reduction TWI**

Writing a logic one to this bit shuts down the TWI by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

- **Bit 6 – PRTIM2 - Power Reduction Timer/Counter2**

Writing a logic one to this bit shuts down the Timer/Counter2 module. When the Timer/Counter2 is enabled, operation will continue like before the shutdown.

- **Bit 5 – 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.

- **Bit 4 – PRPGA - Power Reduction PGA**

Writing a logic one to this bit reduced the power consumption of the programmable gain amplifier. The block is not turned off. Only the current levels in the amplifiers are reduced. Reducing the PGA current levels is only recommended for slow ADC clock frequencies. A new ADC conversion using the PGA should be delayed by a default start-up time after changing (setting or resetting) this bit.

- **Bit 3 – 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 2 – PRSPI - Power Reduction Serial Peripheral Interface**

Writing a 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 re initialized to ensure proper operation.

- **Bit 1 – PRUSART0 - Power Reduction USART**

Writing a logic one to this bit shuts down the USART0 by stopping the clock to the module. When waking up the USART0 again, the USART0 should be reinitialized to ensure proper operation.

- **Bit 0 – PRADC - Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled (reset ADEN bit in register ADCSRA) before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

12.6.3 PRR1 – Power Reduction Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($65)	Res	PRTRX24	PRTIM5	PRTIM4	PRTIM3			PRUSART1	PRR1
Read/Write	R	RW	RW	RW	RW			RW	
Initial Value	0	0	0	0	0			0	

- **Bit 7 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 6 – PRTRX24 - Power Reduction Transceiver**

Writing a logic one to this bit shuts down the transceiver (disconnect from the power supply). In power-down and power-save modes the power-chain will be disabled when this bit is one. Writing a logic zero to this bit will re-enable the transceiver.

- **Bit 5 – PRTIM5 - Power Reduction Timer/Counter5**

Writing a logic one to this bit shuts down the Timer/Counter5 module. When the Timer/Counter5 is enabled, operation will continue like before the shutdown.

- **Bit 4 – PRTIM4 - Power Reduction Timer/Counter4**

Writing a logic one to this bit shuts down the Timer/Counter4 module. When the Timer/Counter4 is enabled, operation will continue like before the shutdown.

- **Bit 3 – PRTIM3 - Power Reduction Timer/Counter3**

Writing a logic one to this bit shuts down the Timer/Counter3 module. When the Timer/Counter3 is enabled, operation will continue like before the shutdown.

- **Bit 0 – PRUSART1 - Power Reduction USART1**

Writing a logic one to this bit shuts down the USART1 by stopping the clock to the module. When waking up the USART1 again, the USART1 should be reinitialized to ensure proper operation.

12.6.4 PRR2 – Power Reduction Register 2

Bit	7	6	5	4	3	2	1	0	
NA ($63)	Res3	Res2	Res1	Res0	PRRAM3	PRRAM2	PRRAM1	PRRAM0	PRR2
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Power Reduction Register PRR2 allows to individually disable all four SRAM blocks. Setting any PRRAM3:0 bit to one will completely switch off (disconnect from the power supply) the corresponding SRAM block. This enables the application to disable unused SRAM memory to save power. Every SRAM block can be re-enabled by reseting the appropriate PRRAM3:0 bit.

- **Bit 7:4 – Res3:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – PRRAM3 - Power Reduction SRAM 3**

Setting this bit to one will disable the SRAM block 3. Setting this bit to zero will enable the SRAM block 3.

- **Bit 2 – PRRAM2 - Power Reduction SRAM 2**

Setting this bit to one will disable the SRAM block 2. Setting this bit to zero will enable the SRAM block 2.

- **Bit 1 – PRRAM1 - Power Reduction SRAM 1**

Setting this bit to one will disable the SRAM block 1. Setting this bit to zero will enable the SRAM block 1.

- **Bit 0 – PRRAM0 - Power Reduction SRAM 0**

Setting this bit to one will disable the SRAM block 0. Setting this bit to zero will enable the SRAM block 0.

12.6.5 TRXPR – Transceiver Pin Register

Bit	7	6	5	4	3	2	1	0	
NA ($139)	Res3	Res2	Res1	Res0	Resx3	Resx2	SLPTR	TRXRST	TRXPR
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The register TRXPR allows to control basic actions of the radio transceiver like reset or state transitions. The register bit functionality is inherited from the external pins of the stand-alone radio transceiver.

- **Bit 7:4 – Res3:0 - Reserved**
- **Bit 3:2 – Resx3:2 - Reserved**
- **Bit 1 – SLPTR - Multi-purpose Transceiver Control Bit**

The bit SLPTR is a multi-functional bit to control transceiver state transitions. Dependent on the radio transceiver state, a rising edge of bit SLPTR causes the following state transitions: TRX_OFF => SLEEP (level sensitive), PLL_ON => BUSY_TX. Whereas the falling edge of bit SLPTR causes the following state transition: SLEEP => TRX_OFF (level sensitive). When the radio transceiver is in TRX_OFF state the microcontroller forces the transceiver to SLEEP by setting SLPTR = H. The Transceiver awakes when the microcontroller releases the bit SLPTR. In states PLL_ON and TX_ARET_ON, bit SLPTR is used as trigger input to initiate a TX transaction. Here SLPTR is sensitive on rising edge only. After initiating a state change by a rising edge at Bit SLPTR in radio transceiver states TRX_OFF, RX_ON or RX_AACK_ON, the radio transceiver remains in the new state as long as the pin is logical high and returns to the preceding state with the falling edge.

- **Bit 0 – TRXRST - Force Transceiver Reset**

The RESET state is used to set back the state machine and to reset all registers of the transceiver to their default values. A reset forces the radio transceiver into the TRX_OFF state and resets all transceiver register to their default values. A reset is initiated with bit TRXRST = H. The bit is cleared automatically During transceiver reset the microcontroller has to set the radio transceiver control bit SLPTR to the default value.

12.6.6 DRTRAM0 – Data Retention Configuration Register of SRAM 0

Bit	7	6	5	4	3	2	1	0	
NA ($135)	Res1	Res0	DRTSWOK	ENDRT	Resx3	Resx2	Resx1	Resx0	DRTRAM0
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DRTRAM0 register controls the behavior of SRAM block 0 in the power-states "power-save" and "power-down". To prevent any data loss the SRAM will not completely disconnected from the power supply. Reserved bits will be overwritten during chip reset by the factory calibration and should not be modified.

- **Bit 7:6 – Res1:0 - Reserved**
- **Bit 5 – DRTSWOK - DRT Switch OK**

This bit indicates the status of the SRAM power-switch. A logical one indicates that the SRAM supply voltage is fully available and the memory may be accessed normally.

- **Bit 4 – ENDRT - Enable SRAM Data Retention**

During "Deep-Sleep" each SRAM block will either be switched off or provides data retention of its memory content. This bit must set to one if data retention mode should be used. Otherwise the SRAM is switched off (disconnected from the power supply) and all its data are lost.

- **Bit 3:0 – Resx3:0 - Reserved**

12.6.7 DRTRAM1 – Data Retention Configuration Register of SRAM 1

Bit	7	6	5	4	3	2	1	0	
NA ($134)	Res1	Res0	DRTSWOK	ENDRT	Resx3	Resx2	Resx1	Resx0	DRTRAM1
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DRTRAM1 register controls the behavior of SRAM block 1 in the power-states "power-save" and "power-down". To prevent any data loss the SRAM will not completely disconnected from the power supply. Reserved bits will be overwritten during chip reset by the factory calibration and should not be modified.

- **Bit 7:6 – Res1:0 - Reserved**
- **Bit 5 – DRTSWOK - DRT Switch OK**

This bit indicates the status of the SRAM power-switch. A logical one indicates that the SRAM supply voltage is fully available and the memory may be accessed normally.

- **Bit 4 – ENDRT - Enable SRAM Data Retention**

During "Deep-Sleep" each SRAM block will either be switched off or provides data retention of its memory content. This bit must set to one if data retention mode should be used. Otherwise the SRAM is switched off (disconnected from the power supply) and all its data are lost.

- **Bit 3:0 – Resx3:0 - Reserved**

12.6.8 DRTRAM2 – Data Retention Configuration Register of SRAM 2

Bit	7	6	5	4	3	2	1	0	
NA ($133)	Resx7	Res	DRTSWOK	ENDRT	Resx3	Resx2	Resx1	Resx0	DRTRAM2
Read/Write	RW	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DRTRAM2 register controls the behavior of SRAM block 2 in the power-states "power-save" and "power-down". To prevent any data loss the SRAM will not completely disconnected from the power supply. Reserved bits will be overwritten during chip reset by the factory calibration and should not be modified.

- **Bit 7 – Resx7 - Reserved**
- **Bit 6 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – DRTSWOK - DRT Switch OK**

This bit indicates the status of the SRAM power-switch. A logical one indicates that the SRAM supply voltage is fully available and the memory may be accessed normally.

- **Bit 4 – ENDRT - Enable SRAM Data Retention**

During "Deep-Sleep" each SRAM block will either be switched off or provides data retention of its memory content. This bit must set to one if data retention mode should be used. Otherwise the SRAM is switched off (disconnected from the power supply) and all its data are lost.

- **Bit 3:0 – Resx3:0 - Reserved**

12.6.9 DRTRAM3 – Data Retention Configuration Register of SRAM 3

Bit	7	6	5	4	3	2	1	0	
NA ($132)	Res1	Res0	DRTSWOK	ENDRT	Resx3	Resx2	Resx1	Resx0	DRTRAM3
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DRTRAM3 register controls the behavior of SRAM block 3 in the power-states "power-save" and "power-down". To prevent any data loss the SRAM will not completely disconnected from the power supply. Reserved bits will be overwritten during chip reset by the factory calibration and should not be modified.

- **Bit 7:6 – Res1:0 - Reserved**
- **Bit 5 – DRTSWOK - DRT Switch OK**

This bit indicates the status of the SRAM power-switch. A logical one indicates that the SRAM supply voltage is fully available and the memory may be accessed normally.

- **Bit 4 – ENDRT - Enable SRAM Data Retention**

During "Deep-Sleep" each SRAM block will either be switched off or provides data retention of its memory content. This bit must set to one if data retention mode should be used. Otherwise the SRAM is switched off (disconnected from the power supply) and all its data are lost.

- **Bit 3:0 – Resx3:0 - Reserved**

12.6.10 LLCR – Low Leakage Voltage Regulator Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($12F)	Res1	Res0	LLDONE	LLCOMP	LLCAL	LLTCO	LLSHORT	LLENCAL	LLCR
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	1	

This register allows to monitor and to control the calibration process of the low-leakage voltage regulator. The automatic calibration is the normal operation mode. However, certain circumstances may require to disable this automatic process for instance to save power-up time. The results of the automatic calibration can also be modified when required by the application for instance to get a higher or lower output voltage.

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – LLDONE - Calibration Done**

This bit indicates the last state of the calibration algorithm. The data register contents is updated with new calibration data after the bit changed to 1. The bit will only be high for one 64kHz clock period, because a new calibration loop is started automatically.

- **Bit 4 – LLCOMP - Comparator Output**

This bit indicates the output state of the comparator of the low-leakage voltage regulator. In this way the calibration progress can be directly monitored for debug purposes. The state of the bit changes at most every 64kHz clock period.

- **Bit 3 – LLCAL - Calibration Active**

This bit indicates that the automatic calibration is in progress. The analog part of the calibration circuit is powered up if the bit is 1.

- **Bit 2 – LLTCO - Temperature Coefficient of Current Source**

This bit shows the status of the selection of the temperature coefficient. The state of the bit is updated in the course of the automatic calibration. A valid value is present after the LLDONE bit is 1 for the first time. Write access is only enabled when the automatic calibration is turned off (LLENCAL is 0). This bit should not be changed without further information.

- **Bit 1 – LLSHORT - Short Lower Calibration Circuit**

This bit shows the status of the short switch for the lower calibration circuit. The state of the bit is updated in the course of the automatic calibration. A valid value is present after the LLDONE bit is 1 for the first time. If this bit is set to 1 register LLDRL has no function. Write access is only possible when the automatic calibration is turned off (LLENCAL is 0). This bit should not be changed without further information.

- **Bit 0 – LLENCAL - Enable Automatic Calibration**

This bit enables the automatic calibration. The automatic calibration runs if the state of the bit is 1. Write access to the two data register and the bits LLSHORT and LLTCO is then denied. If the state of LLENCAL is 0 then the calibration algorithm is stopped and the output voltage of the low-leakage voltage regulator is defined by the values in the two data register LLDRL and LLDRH and by the bits LLSHORT and LLTCO.

12.6.11 LLDRH – Low Leakage Voltage Regulator Data Register (High-Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($131)	Res2	Res1	Res0	LLDRH4	LLDRH3	LLDRH2	LLDRH1	LLDRH0	LLDRH
Read/Write	R	R	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The high-byte of the calibration data can be accessed through this register. Write access is only enabled when the bit LLENCAL of the LLCR register is 0. Then the data bits LLDRH4:0 directly control the output voltage of the low-leakage voltage regulator. Higher numbers generate lower voltages. If the bit LLENCAL is 1 then the results of the automatic calibration are stored.

- **Bit 7:5 – Res2:0 - Reserved**

These bits are reserved for future use.

- **Bit 4:0 – LLDRH4:0 - High-Byte Data Register Bits**

Value of the high-byte calibration result

Table 12-5 LLDRH Register Bits

Register Bits	Value	Description
LLDRH4:0	0x00	Calibration limit for fast process corner/high output voltage
	0x10	Calibration limit for slow process corner/low output voltage

12.6.12 LLDRL – Low Leakage Voltage Regulator Data Register (Low-Byte)

Bit	7	6	5	4	3	2	1	0	
NA ($130)	Res3	Res2	Res1	Res0	LLDRL3	LLDRL2	LLDRL1	LLDRL0	LLDRL
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The low-byte of the calibration data can be accessed through this register. Write access is only enabled when the bit LLENCAL of the LLCR register is 0. Then the data bits LLDRL3:0 directly control the output voltage of the low-leakage voltage regulator. Higher numbers generate lower voltages. The contents of this register is meaningless when the bit LLSHORT of the LLCR register is 1. If the bit LLENCAL is 1 then the results of the automatic calibration are stored.

- **Bit 7:4 – Res3:0 - Reserved**

These bits are reserved for future use.

- **Bit 3:0 – LLDRL3:0 - Low-Byte Data Register Bits**

Value of the low-byte calibration result

Table 12-6 LLDRL Register Bits

Register Bits	Value	Description
LLDRL3:0	0x00	Calibration limit for fast process corner/high output voltage
	0x08	Calibration limit for slow process corner/low output voltage

12.6.13 DPDS0 – Port Driver Strength Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($136)	PFDRV1	PFDRV0	PEDRV1	PEDRV0	PDDRV1	PDDRV0	PBDRV1	PBDRV0	DPDS0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The output driver strength can be set individually for each digital I/O port. The following tables show output current levels for a typical supply voltage of DEVDD = 3.3V. Refer to section "Electrical Characteristics" for details.

- **Bit 7:6 – PFDRV1:0 - Driver Strength Port F**

Table 12-7 PFDRV Register Bits

Register Bits	Value	Description
PFDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 5:4 – PEDRV1:0 - Driver Strength Port E**

Table 12-8 PEDRV Register Bits

Register Bits	Value	Description
PEDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 3:2 – PDDRV1:0 - Driver Strength Port D**

Table 12-9 PDDRV Register Bits

Register Bits	Value	Description
PDDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 1:0 – PBDRV1:0 - Driver Strength Port B**

Table 12-10 PBDRV Register Bits

Register Bits	Value	Description
PBDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

12.6.14 DPDS1 – Port Driver Strength Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($137)	Res5	Res4	Res3	Res2	Res1	Res0	PGDRV1	PGDRV0	DPDS1
Read/Write	R	R	R	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The output driver strength can be set individually for each digital I/O port. The following table shows output current levels for a typical supply voltage of DEVDD = 3.3V. Refer to section "Electrical Characteristics" for details.

- **Bit 7:2 – Res5:0 - Reserved**
- **Bit 1:0 – PGDRV1:0 - Driver Strength Port G**

Driver strength can be set for port G except the port pins PG3 and PG4. The leakage current of the ports PG3 and PG4 is reduced.

Table 12-11 PGDRV Register Bits

Register Bits	Value	Description
PGDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

13 System Control and Reset

13.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 13-1 on page 177 shows the reset logic."System and Reset Characteristics" on page 502 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

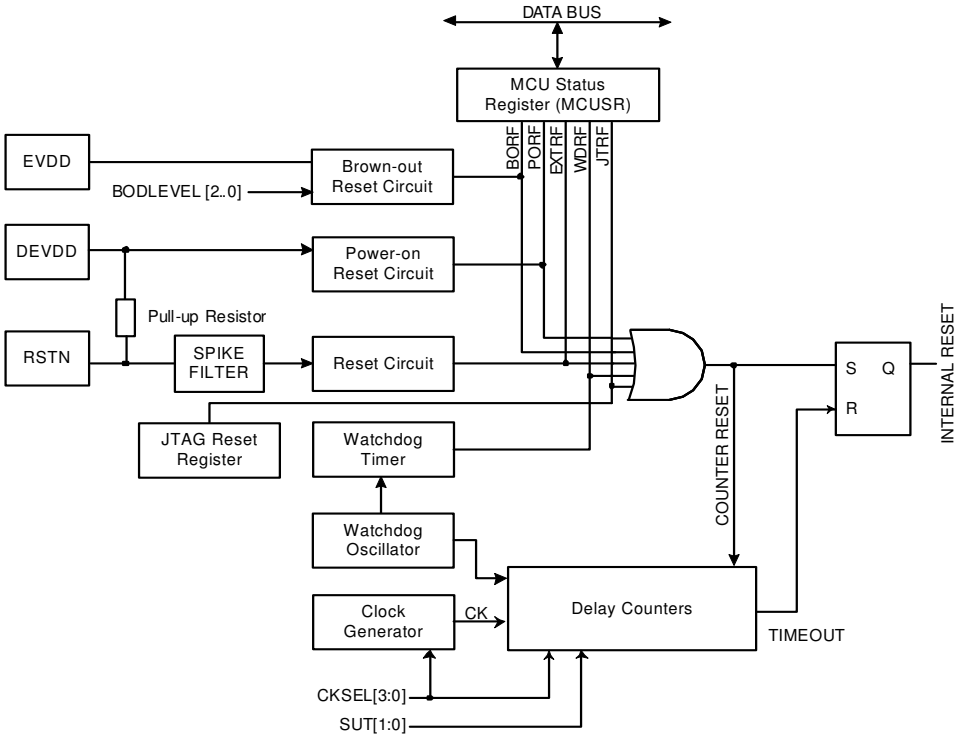
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 148.

13.2 Reset Sources

The ATmega128RFA1 has five sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the RSTN 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 the supply voltage EVDD is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.
- JTAG AVR Reset. The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. Refer to the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 441 for details.

Figure 13-1. Reset Logic



13.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by a dynamic, on-chip detection circuit. The POR is active when DEVDD is rising. The electrical characteristics are defined in "System and Reset Characteristics" on page 502. The POR circuit can be used to trigger the start-up reset. To detect a failure in the supply voltage (e.g. a voltage drop) the brown-own detector should be used.

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 the DEVDD rise. The RESET signal is activated again without any delay, when DEVDD decreases below the detection level.

Figure 13-2. MCU Start-up, RSTN Tied to DEVDD

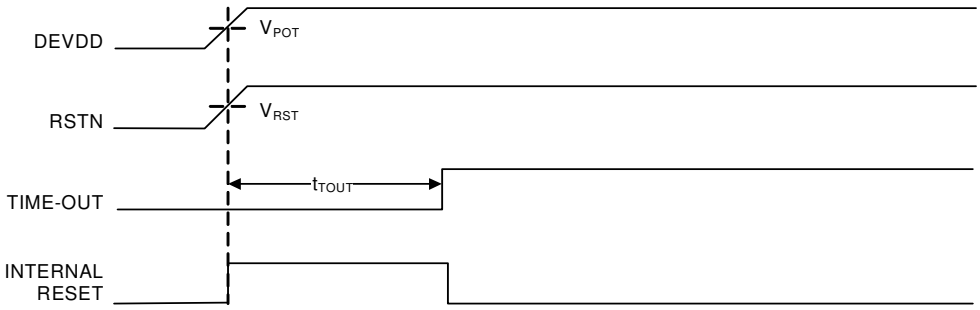
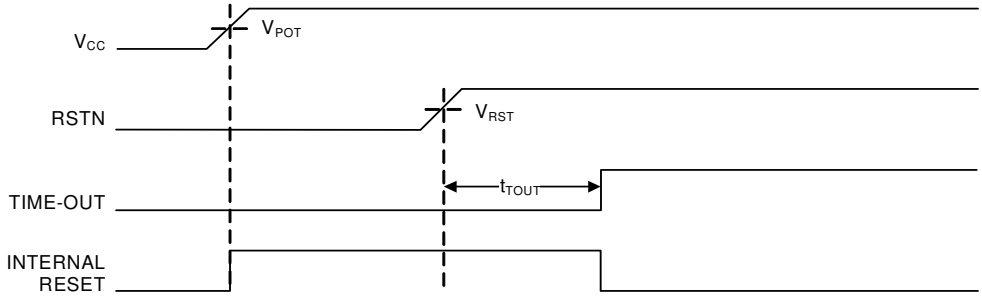


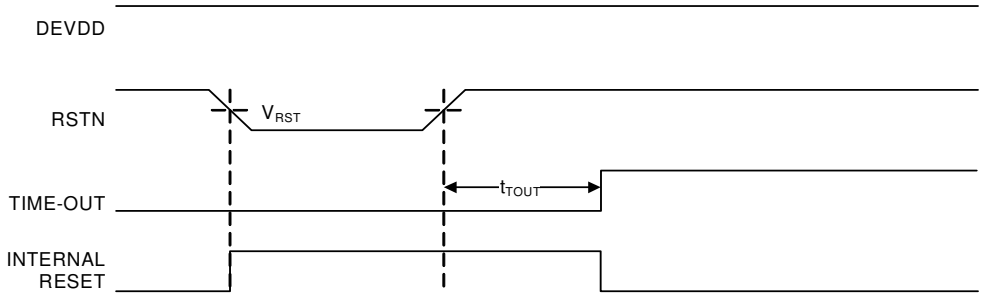
Figure 13-3. MCU Start-up, RSTN Extended Externally



13.2.2 External Reset

An External Reset is generated by a low level on the RSTN pin. Reset pulses longer than the minimum pulse width (see "System and Reset Characteristics" on page 502) 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 13-4. Reset During Operation



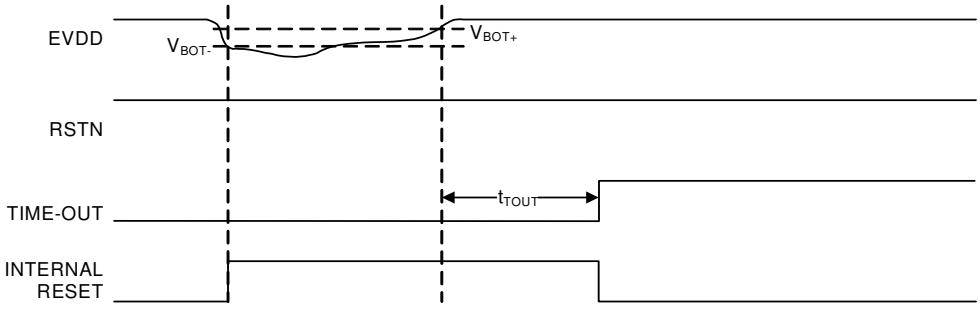
13.2.3 Brown-out Detection

ATmega128RFA1 has an On-chip Brown-out Detection (BOD) circuit for monitoring the EVDD level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

When the BOD is enabled, and EVDD decreases to a value below the trigger level (V_{BOT-} in Figure 13-5 on page 179), the Brown-out Reset is immediately activated. When EVDD increases above the trigger level (V_{BOT+} in Figure 13-5 on page 179), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in EVDD if the voltage stays below the trigger level for longer than t_{BOD} given in "System and Reset Characteristics" on page 502.

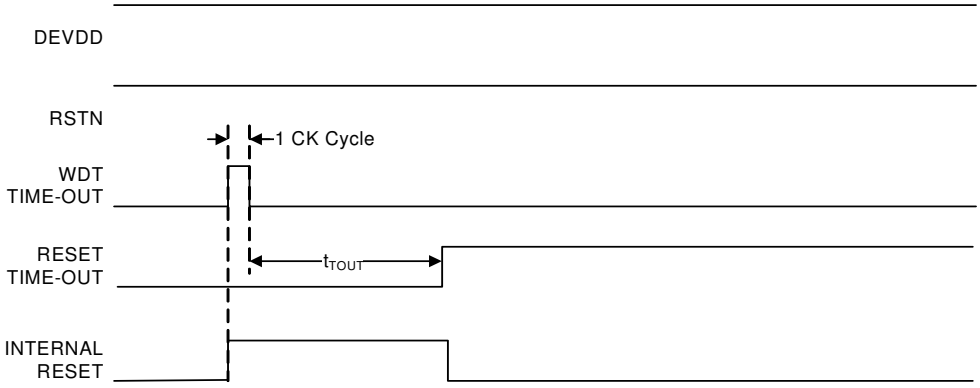
Figure 13-5. Brown-out Reset During Operation



13.2.4 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}. See "Watchdog Timer" on page 180. for details on operation of the Watchdog Timer.

Figure 13-6. Watchdog Reset During Operation



13.3 Internal Voltage Reference

ATmega128RFA1 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC.

Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in "System and Reset Characteristics" on page 502. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2:0] Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

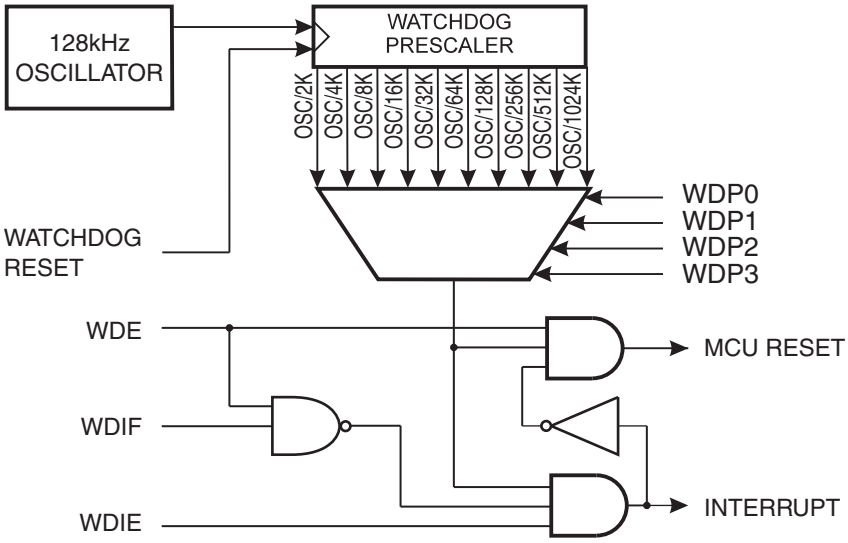
Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

13.4 Watchdog Timer

13.4.1 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**
- **Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode**

Figure 13-7. Watchdog Timer



13.4.2 Overview

ATmega128RFA1 has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of a separate on-chip 128 kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR -Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

In Interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In System Reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, Interrupt and System Reset mode, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure

program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and Watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example(1)

```
WDT_off:
 ; Turn off global interrupt
 cli
 ; Reset Watchdog Timer
 wdr
 ; Clear WDRF in MCUSR
 in    r16, MCUSR
 andi  r16, (0xff & (0<<WDRF))
 out   MCUSR, r16
 ; Write logical one to WDCE and WDE
 ; Keep old prescaler setting to prevent unintentional time-out
 ldi r16, WDTCSR
 ori r16, (1<<WDCE) | (1<<WDE)
 sts WDTCSR, r16
 ; Turn off WDT
 ldi r16, (0<<WDE)
 sts WDTCSR, r16
 ; Turn on global interrupt
 sei
 ret
```

C Code Example(1)

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Clear WDRF in MCUSR*/
  MCUSR &= ~(1<<WDRF);
  /* Write logical one to WDCE and WDE */
  /* Keep old prescaler setting to prevent unintentional time-out */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

Note: 1. The example code assumes that the part specific header file is included.

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(1,2)

```
WDT_Prescaler_Change:
 ; Turn off global interrupt cli
 ; 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 = 64K cycles (~0.5 s)
 ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
 out   WDTCSR, r16
 ; --Finished setting new values, used 2 cycles -
 ; Turn on global interrupt
 sei
 ret
```

C Code Example(1,2)

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed  equence */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note: 1. The example code assumes that the part specific header file is included.
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.

13.5 Register Description

13.5.1 MCUSR – MCU Status Register

Bit	7	6	5	4	3	2	1	0	
$34 ($54)	Res2	Res1	Res0	JTRF	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	RW	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The MCU Status Register provides information on which reset source caused an MCU reset. To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags. Note, after power on the bit EXTRF has to be ignored.

- **Bit 7:5 – Res2:0 - Reserved**
- **Bit 4 – JTRF - JTAG Reset Flag**

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This 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 – BORF - Brown-out Reset Flag**

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

- **Bit 1 – EXTRF - External Reset Flag**

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

- **Bit 0 – PORF - Power-on Reset Flag**

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

13.5.2 WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – WDIF - Watchdog Timeout 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 Timeout Interrupt Enable**

When this bit is written to one and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if time-out in the Watchdog Timer occurs. If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode). This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 13-1. 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 and “1” means un-programmed.

- **Bit 4 – WDCE - Watchdog Change Enable**

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set. Once written to one, hardware will clear WDCE after four clock cycles.

- **Bit 3 – WDE - Watch Dog Enable**

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set (one). To disable an enabled watchdog timer, the following procedure must be followed: 1. In the same operation, write a logical one to WDTOE and WDE. A logical one must be written to WDE even though it is set to one before the disable operation starts. 2. Within the next four clock cycles, write a logical 0 to WDE. This disables the watchdog. WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

- **Bit 5, 2:0 – WDP3:0 – Watchdog Timer Prescaler 3, 2, 1 and 0**

The WDP3:0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is running.

Table 13-2. WDP Register Bits

Register Bits	Value	Description
WDP3:0	0x00	Oscillator Cycles 2k, (16ms)
	0x01	Oscillator Cycles 4k, (32ms)
	0x02	Oscillator Cycles 8k, (64ms)
	0x03	Oscillator Cycles 16k, (0.125s)

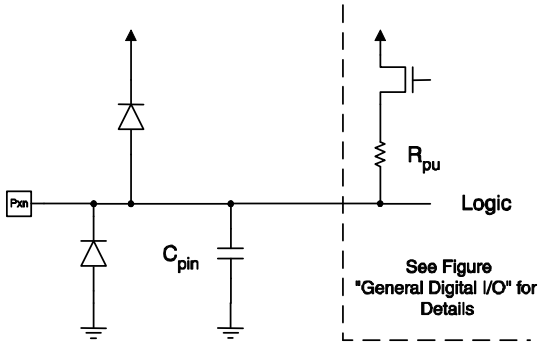
Register Bits	Value	Description
	0x04	Oscillator Cycles 32k, (0.25s)
	0x05	Oscillator Cycles 64k, (0.5s)
	0x06	Oscillator Cycles 128k, (1.0s)
	0x07	Oscillator Cycles 256k, (2.0s)
	0x08	Oscillator Cycles 512k, (4.0s)
	0x09	Oscillator Cycles 1024k, (8.0s)

14 I/O-Ports

14.1 Introduction

All ATmega128RFA1 ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both configurable sink and source capability. Every port is individually configurable in four different drive strengths. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both DEVDD and DVSS as indicated in Figure 14-1 below. Refer to "Electrical Characteristics" on page 501 for a complete list of parameters.

Figure 14-1. 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.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

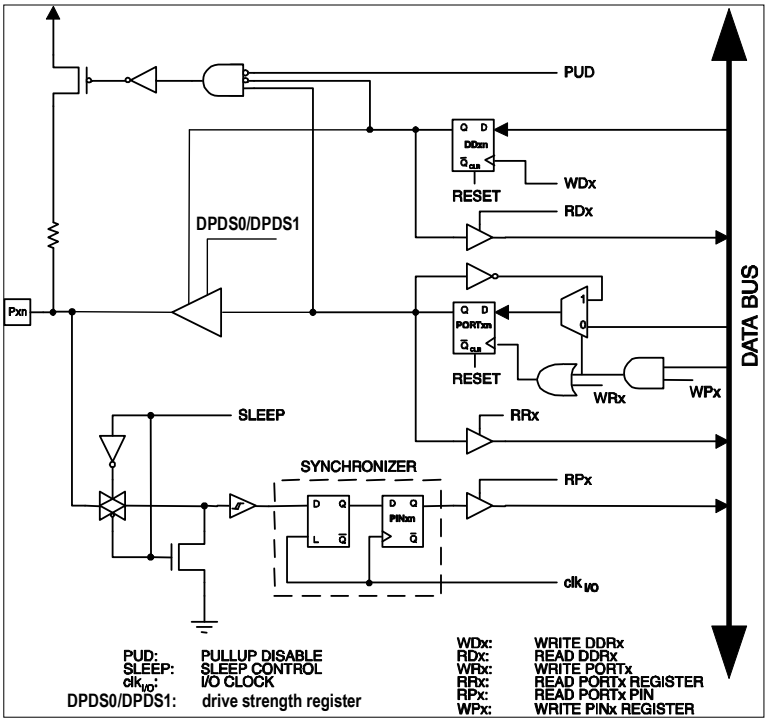
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 187. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 191. 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.

14.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 14-2 below shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 14-2. 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.

14.2.1 Configuring the Port

Drive strength of output buffers is configurable port-wise. Source/sink capably of 2mA, 4mA, 6mA or 8mA is selectable through registers DPDS1 and DPDS0. Note that pins PG3 and PG4 of PORTG have fixed drive strength of 2mA to enable the operation of the low power crystal oscillator.

14.2.2 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. 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).

14.2.3 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.

14.2.4 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.

The following table summarizes the control signals for the pin value.

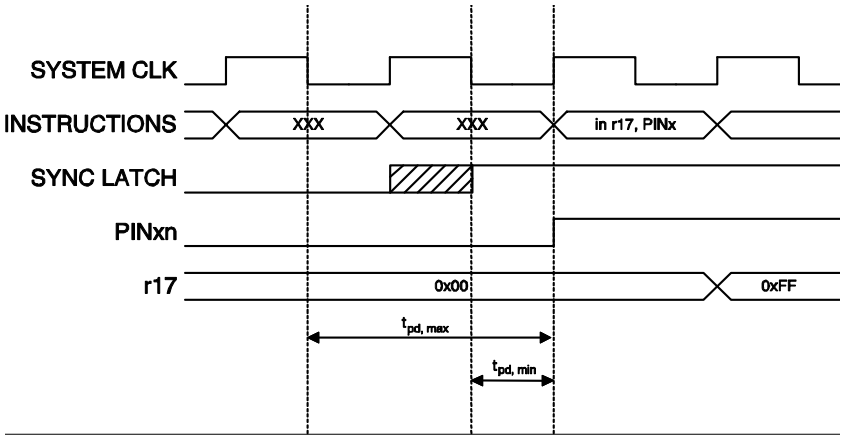
Table 14-1. Port Pin Configurations

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)

14.2.5 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 14-2 on page 187, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid meta-stability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 14-3 on page 189 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.

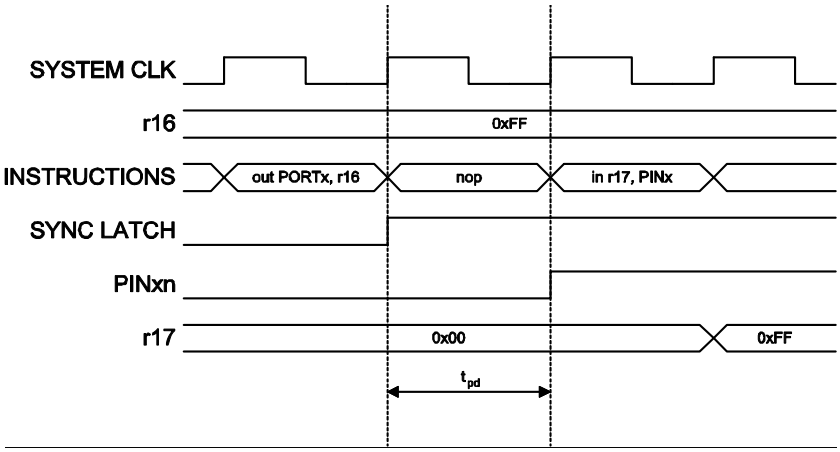
Figure 14-3. Synchronization when reading an external applied pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows $t_{PD,MAX}$ and $t_{PD,MIN}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a NOP instruction must be inserted as indicated in Figure 14-4 below. The out instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay t_{PD} through the synchronizer is 1 system clock period.

Figure 14-4. Synchronization when reading 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*/
__no_operation();
/* 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.

14.2.6 Digital Input Enable and Sleep Modes

As shown in Figure 14-2 on page 187, the digital input signal can be clamped to ground at the input of the Schmitt-Trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode, Power-save mode, and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{DEVDD}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in "Alternate Port Functions" on page 191.

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.

14.2.7 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

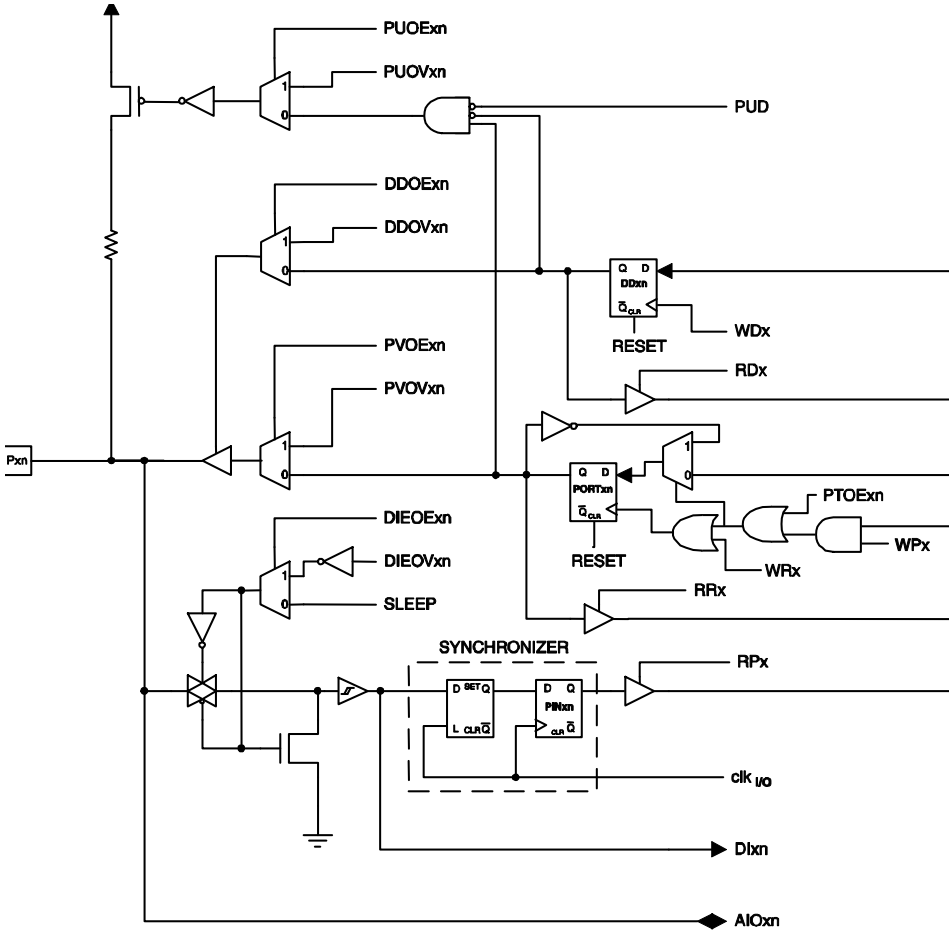
described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset-, Active- and Idle-mode).

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 DEVDD or DVSS is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

14.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/O ports. Figure 14-5 below shows how the port pin control signals from the simplified Figure 14-2 on page 187 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 14-5. Alternate Port Functions (1)



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN

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.

The following table summarizes the function of the overriding signals. The pin and port indexes from Figure 14-5 on page 191 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 14-2. 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.

14.3.1 Alternate Functions of Port B

The Port B pins with alternate functions are shown in the following table.

Table 14-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	OC0A/OC1C/PCINT7 (Output Compare and PWM Output A for Timer/Counter0, Output Compare and PWM Output C for Timer/Counter1 or Pin Change Interrupt 7)
PB6	OC1B/PCINT6 (Output Compare and PWM Output B for Timer/Counter1 or Pin Change Interrupt 6)
PB5	OC1A/PCINT5 (Output Compare and PWM Output A for Timer/Counter1 or Pin Change Interrupt 5)
PB4	OC2A/PCINT4 (Output Compare and PWM Output A for Timer/Counter2 or Pin Change Interrupt 4)
PB3	MISO/PDO/PCINT3 (SPI Bus Master Input/Slave Output, Programming Data Output or Pin Change Interrupt 3)
PB2	MOSI/PDI/PCINT2 (SPI Bus Master Output/Slave Input , Programming Data Input or Pin Change Interrupt 2)
PB1	SCK/PCINT1 (SPI Bus Serial Clock or Pin Change Interrupt 1)
PB0	SS/PCINT0 (SPI Slave Select input or Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- **OC0A/OC1C/PCINT7, Bit 7**

OC0A, Output Compare Match A output: The PB7 pin can serve as an external output for the Timer/Counter0 Output Compare. The pin has to be configured as an output (DDB7 set “one”) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

OC1C, Output Compare Match C output: The PB7 pin can serve as an external output for the Timer/Counter1 Output Compare C. The pin has to be configured as an output (DDB7 set (one)) to serve this function. The OC1C pin is also the output pin for the PWM mode timer function.

PCINT7, Pin Change Interrupt source 7: The PB7 pin can serve as an external interrupt source.

- **OC1B/PCINT6, Bit 6**

OC1B, Output Compare Match B output: The PB6 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDB6 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT6, Pin Change Interrupt source 6: The PB6 pin can serve as an external interrupt sourceOC1A, Output Compare Match A output: The PB5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDB5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT5, Pin Change Interrupt source 5: The PB5 pin can serve as an external interrupt source.

- **OC2A/PCINT4, Bit 4**

OC2A, Output Compare Match output: The PB4 pin can serve as an external output for the Timer/Counter2 Output Compare. The pin has to be configured as an output (DDB4 set (one)) to serve this function. The OC2A pin is also the output pin for the PWM mode timer function.

PCINT4, Pin Change Interrupt source 4: The PB4 pin can serve as an external interrupt source.

- **MISO/PDO/PCINT3 – Port B, Bit 3**

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 DDB3. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB3 bit.

PDO, SPI Serial Programming Data Output. During Serial Program Downloading, this pin is used as data output line (see section "Serial Downloading" on page 477 for details).

PCINT3, Pin Change Interrupt source 3: The PB3 pin can serve as an external interrupt source.

- **MOSI/PDI/PCINT2 – Port B, Bit 2**

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 DDB2. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB2 bit.

PDI, SPI Serial Programming Data Input. During Serial Program Downloading, this pin is used as data input line (see section "Serial Downloading" on page 477 for details).

PCINT2, Pin Change Interrupt source 2: The PB2 pin can serve as an external interrupt source.

- **SCK/PCINT1 – Port B, Bit 1**

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 DDB1. When the SPI0 is enabled as a master, the data direction of this pin is controlled by DDB1. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB1 bit.

PCINT1, Pin Change Interrupt source 1: The PB1 pin can serve as an external interrupt source.

- **SS/PCINT0 – Port B, Bit 0**

SS: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB0. 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 DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 bit.

Table 14-4 below and Table 14-5 on page 195 relate the alternate functions of Port B to the overriding signals shown in Figure 14-5 on page 191. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

PCINT0, Pin Change Interrupt source 0: The PB0 pin can serve as an external interrupt source.

Table 14-4. Overriding Signals for Alternate Functions in PB7:PB4

Signal Name	PB7/OC0A/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC2A
PUOE	0	0	0	0

Signal Name	PB7/OC0A/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC2A
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0/OC1C ENABLE	OC1B ENABLE	OC1A ENABLE	OC2A ENABLE
PVOV	OC0/OC1C	OC1B	OC1A	OC2A
DIEOE	PCINT7•PCIE0	PCINT6•PCIE0	PCINT5•PCIE0	PCINT4•PCIE0
DIEOV	1	1	1	1
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT	PCINT4 INPUT
AIO	–	–	–	–

Table 14-5. Overriding Signals for Alternate Functions in PB3:PB0

Signal Name	PB3/MISO/PDO	PB2/MOSI/PDI	PB1/SCK	PB0/SS
PUOE	SPE•MSTR	SPE•(~MSTR)	SPE•(~MSTR)	SPE•(~MSTR)
PUOV	PORTB3•(~PUD)	PORTB2•(~PUD)	PORTB1•(~PUD)	PORTB0•(~PUD)
DDOE	SPE•MSTR	SPE•(~MSTR)	SPE•(~MSTR)	SPE•(~MSTR)
DDOV	0	0	0	0
PVOE	SPE•(~MSTR)	SPE•MSTR	SPE•MSTR	0
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	0
DIEOE	PCINT3•PCIE0	PCINT2•PCIE0	PCINT1•PCIE0	PCINT0•PCIE0
DIEOV	1	1	1	1
DI	SPI MSTR INPUT PCINT3 INPUT	SPI SLAVE INPUT PCINT2 INPUT	SCK INPUT PCINT1 INPUT	SPI SS PCINT0 INPUT
AIO	–	–	–	–

14.3.2 Alternate Functions of Port D

The Port D pins with alternate functions are shown in the following table.

Table 14-6. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	T0 (Timer/Counter0 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1 (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1Input Capture Trigger)
PD3	INT3/TXD1 (External Interrupt3 Input or USART1 Transmit Pin)
PD2	INT2/RXD 1(External Interrupt2 Input or USART1 Receive Pin)
PD1	INT1/SDA (External Interrupt1 Input or TWI Serial Data)
PD0	INT0/SCL (External Interrupt0 Input or TWI Serial Clock)

The alternate pin configuration is as follows:

- **T0 – Port D, Bit 7**

T0, this is Timer/Counter0 counter source.

- **T1 – Port D, Bit 6**

T1, this is Timer/Counter1 counter source.

- **XCK1 – Port D, Bit 5**

XCK1, USART1 External clock: The Data Direction Register (DDD5) controls whether the clock is output (DDD5 set) or input (DDD5 cleared). The XCK1 pin is active only when the USART1 operates in Synchronous mode.

- **ICP1 – Port D, Bit 4**

ICP1 – Input Capture Pin 1: The PD4 pin can act as an input capture pin for Timer/Counter1.

- **INT3/TXD1 – Port D, Bit 3**

INT3, External Interrupt source 3: The PD3 pin can serve as an external interrupt source to the MCU.

TXD1, Transmit Data (Data output pin for the USART1). When the USART1 Transmitter is enabled, this pin is configured as an output regardless of the value of DDD3.

- **INT2/RXD1 – Port D, Bit 2**

INT2, External Interrupt source 2: The PD2 pin can serve as an External Interrupt source to the MCU.

RXD1, Receive Data (Data input pin for the USART1). When the USART1 receiver is enabled this pin is configured as an input regardless of the value of DDD2. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD2 bit.

- **INT1/SDA – Port D, Bit 1**

INT1, External Interrupt source 1: The PD1 pin can serve as an external interrupt source to the MCU.

SDA, 2-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PD1 is disconnected from the port and becomes the Serial Data I/O pin for the 2-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew rate limitation.

- **INT0/SCL – Port D, Bit 0**

INT0, External Interrupt source 0: The PD0 pin can serve as an external interrupt source to the MCU.

SCL, 2-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PD0 is disconnected from the port and becomes the Serial Clock I/O pin for the 2-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

Table 14-7 below and Table 14-8 on page 197 relates the alternate functions of Port D to the overriding signals shown in Figure 14-5 on page 191.

Table 14-7. Overriding Signals for Alternate Functions PD7:PD4

Signal Name	PD7/T0	PD6/T1	PD5/XCK1	PD4/ICP1
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	XCK1 OUTPUT ENABLE	0

Signal Name	PD7/T0	PD6/T1	PD5/XCK1	PD4/ICP1
DDOV	0	0	1	0
PVOE	0	0	XCK1 OUTPUT ENABLE	0
PVOV	0	0	XCK1 OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	T0 INPUT	T1 INPUT	XCK1 INPUT	ICP1 INPUT
AIO	–	–	–	–

Table 14-8. Overriding Signals for Alternate Functions PD3:PD0

Signal Name	PD3/INT3/TXD1	PD2/INT2/RXD1	PD1/INT1/SDA	PD0/INT0/SCL
PUOE	TXEN1	RXEN1	TWEN	TWEN
PUOV	0	PORTD2&(~PUD)	PORTD1&(~PUD)	PORTD0&(~PUD)
DDOE	TXEN1	RXEN1	TWEN	TWEN
DDOV	1	0	SDA_OUT	SCL_OUT
PVOE	TXEN1	0	TWEN	TWEN
PVOV	TXD1	0	0	0
DIEOE	INT3 ENABLE	INT2 ENABLE	INT1 ENABLE	INT0 ENABLE
DIEOV	1	1	1	1
DI	INT3 INPUT	INT2 INPUT/RXD1	INT1 INPUT	INT0 INPUT
AIO	-	-	SDA INPUT	SCL INPUT

Note: 1. When enabled, the 2-wire Serial Interface enables Slew-Rate controls on the output pins PD0 and PD1. This is not shown in this table. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

14.3.3 Alternate Functions of Port E

The Port E pins with alternate functions are shown in the following table.

Table 14-9. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE7	INT7/ICP3/CLK0 (External Interrupt7 Input, Timer/Counter3 Input Capture Trigger or Divided System Clock)
PE6	INT6/T3 (External Interrupt6 Input or Timer/Counter3 Clock Input)
PE5	INT5/OC3C (External Interrupt5 Input or Output Compare and PWM Output C for Timer/Counter3)
PE4	INT4/OC3B (External Interrupt4 Input or Output Compare and PWM Output B for Timer/Counter3)
PE3	AIN1/OC3A (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
PE2	AIN0/XCK0 (Analog Comparator or Positive Input or USART0 external clock input/output)
PE1	TXD0 (USART0 Transmit Pin)

Port Pin	Alternate Function
PE0	RXD0/PCINT8 (USART0 Receive Pin or Pin Change Interrupt8)

- **INT7/ICP3/CLKO – Port E, Bit 7**

INT7, External Interrupt source 7: The PE7 pin can serve as an external interrupt source.

ICP3, Input Capture Pin 3: The PE7 pin can act as an input capture pin for Timer/Counter3.

CLKO - Divided System Clock: The divided system clock can be output on the PE7 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTE7 and DDE7 settings. It will also be output during reset.

- **INT6/T3 – Port E, Bit 6**

INT6, External Interrupt source 6: The PE6 pin can serve as an external interrupt source.

T3, this is the Timer/Counter3 counter source.

- **INT5/OC3C – Port E, Bit 5**

INT5, External Interrupt source 5: The PE5 pin can serve as an External Interrupt source.

OC3C, Output Compare Match C output: The PE5 pin can serve as an External output for the Timer/Counter3 Output Compare C. The pin has to be configured as an output (DDE5 set “one”) to serve this function. The OC3C pin is also the output pin for the PWM mode timer function.

- **INT4/OC3B – Port E, Bit 4**

INT4, External Interrupt source 4: The PE4 pin can serve as an External Interrupt source.

OC3B, Output Compare Match B output: The PE4 pin can serve as an External output for the Timer/Counter3 Output Compare B. The pin has to be configured as an output (DDE4 set (one)) to serve this function. The OC3B pin is also the output pin for the PWM mode timer function.

- **AIN1/OC3A – Port E, Bit 3**

AIN1 – Analog Comparator Negative input. This pin is directly connected to the negative input of the Analog Comparator.

OC3A, Output Compare Match A output: The PE3 pin can serve as an External output for the Timer/Counter3 Output Compare A. The pin has to be configured as an output (DDE3 set “one”) to serve this function. The OC3A pin is also the output pin for the PWM mode timer function.

- **AIN0/XCK0 – Port E, Bit 2**

AIN0 – Analog Comparator Positive input. This pin is directly connected to the positive input of the Analog Comparator.

XCK0, this is the USART0 External clock. The Data Direction Register (DDE2) controls whether the clock is output (DDE2 set) or input (DDE2 cleared). The XCK0 pin is active only when the USART0 operates in Synchronous mode.

- **TXD0 – Port E, Bit 1**

TXD0, this is the USART0 Transmit pin.

- **RXD0/PCINT8 – Port E, Bit 0**

RXD0, USART0 Receive Pin. Receive Data (Data input pin for the USART0). When the

USART0 receiver is enabled this pin is configured as an input regardless of the value of DDRE0. When the USART0 forces this pin to be an input, a logical one in PORTE0 will turn on the internal pull-up.

PCINT8, Pin Change Interrupt source 8: The PE0 pin can serve as an external interrupt source.

Table 14-10 below and Table 14-11 below relates the alternate functions of Port E to the overriding signals shown in Figure 14-5 on page 191.

Table 14-10. Overriding Signals for Alternate Functions PE7:PE4

Signal Name	PE7/INT7/ICP3	PE6/INT6/T3	PE5/INT5/OC3C	PE4/INT4/OC3B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	OC3C ENABLE	OC3B ENABLE
PVOV	0	0	OC3C	OC3B
DIEOE	INT7 ENABLE	INT6 ENABLE	INT5 ENABLE	INT4 ENABLE
DIEOV	1	1	1	1
DI	INT7 INPUT / ICP3 INPUT	INT7 INPUT / T3 INPUT	INT5 INPUT	INT4 INPUT
AIO	–	–	–	–

Table 14-11. Overriding Signals for Alternate Functions PE3:PE0

Signal Name	PE3/AIN1/OC3A	PE2/AIN0/XCK0	PE1/TXD0	PE0 / RXD0/PCINT8
PUOE	0	0	TXEN0	RXEN0
PUOV	0	0	0	PORTE0 & (~PUD)
DDOE	0	XCK0 OUTPUT ENABLE	TXEN0	RXEN0
DDOV	0	1	1	0
PVOE	OC3BENABLE	XCK0 OUTPUT ENABLE	TXEN0	0
PVOV	OC3B	XCK0 OUTPUT	TXD0	0
DIEOE	0	0	0	PCINT8 & PCIE1
DIEOV	0	0	0	1
DI	0	XCK0 INPUT	–	RXD0
PE0	0	0	0	PCINT8 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	-	-

14.3.4 Alternate Functions of Port F

The Port F has an alternate function as analog input for the ADC as shown in Table 14-12 on page 200. If some Port F pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a Reset occurs.

Table 14-12. Port F Pins Alternate Functions

Port Pin	Alternate Function
PF7	ADC7/TDI (ADC input channel 7 or JTAG Test Data Input)
PF6	ADC6/TDO (ADC input channel 6 or JTAG Test Data Output)
PF5	ADC5/TMS (ADC input channel 5 or JTAG Test Mode Select)
PF4	ADC4/TCK (ADC input channel 4 or JTAG Test Clock)
PF3	ADC3/DIG4 (ADC input channel 3 or Radio Transceiver RX/TX Indicator Output)
PF2	ADC2/DIG2 (ADC input channel 2 or Radio Transceiver Antenna Diversity Control Output)
PF1	ADC1 (ADC input channel 1)
PF0	ADC0 (ADC input channel 0)

- **TDI, ADC7 – Port F, Bit 7**

ADC7, Analog to Digital Converter, Channel 7.

TDI, JTAG Test Data In: Serial input data to be shifted in to the Instruction Register or Data Register (scan chains). When the JTAG interface is enabled, this pin can not be used as an I/O pin.

- **TDO, ADC6 – Port F, Bit 6**

ADC6, Analog to Digital Converter, Channel 6.

TDO, JTAG Test Data Out: Serial output data from Instruction Register or Data Register. When the JTAG interface is enabled, this pin can not be used as an I/O pin. The TDO pin is tri-stated unless TAP states that shift out data are entered.

- **TMS, ADC5 – Port F, Bit 5**

ADC5, Analog to Digital Converter, Channel 5.

TMS, JTAG Test Mode Select: This pin is used for navigating through the TAP-controller state machine. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

- **TCK, ADC4 – Port F, Bit 4**

ADC4, Analog to Digital Converter, Channel 4.

TCK, JTAG Test Clock: JTAG operation is synchronous to TCK. When the JTAG interface is enabled, this pin can not be used as an I/O pin.

- **DIG4, ADC3 – Port F, Bit 3**

ADC3, Analog to Digital Converter, Channel 3.

DIG4, Radio Transceiver RX/TX Indicator Output: If the bit PA_EXT_EN in TRX_CTRL_1 is set to one then the PF3 pin serves as the Radio Transceiver receive/transmit indicator output to control an external RF front-end.

- **DIG2, ADC2 – Port F, Bit 2**

ADC2, Analog to Digital Converter, Channel 2.

DIG2, Radio Transceiver Antenna Diversity Control Output: If the bit ANT_EXT_SW_EN in ANT_DIV is set to one then the PF2 pin serves as a Radio Transceiver output to control External Antenna Diversity.

- **ADC1 – ADC0 – Port F, Bit 1:0**

Analog to Digital Converter, Channel 1:0.

Table 14-13. Overriding Signals for Alternate Functions PF7:PF4

Signal Name	PF7/ADC7/TDI	PF6/ADC6/TDO	PF5/ADC5/TMS	PF4/ADC4/TCK
PUOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
PUOV	1	0	1	1
DDOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DDOV	0	SHIFT_IR+SHIFT_DR	0	0
PVOE	0	JTAGEN	0	0
PVOV	0	TDO	0	0
DIEOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	TDI/ADC7 INPUT	ADC6 INPUT	TMS/ADC5 INPUT	TCK/ADC4 INPUT

Table 14-14. Overriding Signals for Alternate Functions PF3:PF0

Signal Name	PF3/ADC3/DIG4	PF2/ADC2/DIG2	PF1/ADC1	PF0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	PA_EXT_EN	ANT_EXT_SW_EN	0	0
DDOV	PA_EXT_EN	ANT_EXT_SW_EN	0	0
PVOE	PA_EXT_EN	ANT_EXT_SW_EN	0	0
PVOV	DIG4	DIG2	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

14.3.5 Alternate Functions of Port G

The Port G alternate pin configuration is as follows:

Table 14-15. Port G Pins Alternate Functions

Port Pin	Alternate Function
PG5	OC0B (Output Compare and PWM Output B for Timer/Counter0)
PG4	TOSC1 (RTC Oscillator Timer/Counter2)
PG3	TOSC2 (RTC Oscillator Timer/Counter2)
PG2	AMR (Automated Meter Reading - Counter Input for Timer/Counter2)
PG1	DIG1 (Radio Transceiver Antenna Diversity Control Output)
PG0	DIG3 (Radio Transceiver RX/TX Indicator Output)

- **OC0B – Port G, Bit 5**

OC0B, Output Compare match B output: The PG5 pin can serve as an external output for the TImer/Counter0 Output Compare. The pin has to be configured as an output (DDG5 set) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.

- **TOSC1 – Port G, Bit 4**

TOSC2, Timer Oscillator pin 1: Setting the AS2 bit to one and the EXCLKAMR bit to zero in ASSR, enables asynchronous clocking of Timer/Counter2 by a Crystal Oscillator. The pin PG4 is disconnected from the port, and becomes the input of the inverting Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

TOSC2 – Port G, Bit 3

TOSC2, Timer Oscillator pin 2: Setting the AS2 bit to one and the EXCLKAMR bit to zero in ASSR, enables asynchronous clocking of Timer/Counter2 by a Crystal Oscillator. The pin PG3 is disconnected from the port, and becomes the inverting output of the Oscillator amplifier. In this mode, a Crystal Oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- **AMR – Port G, Bit 2**

AMR, Automated Meter Reading Input: Setting the AS2 and the EXCLKAMR bits in ASSR to one, enables asynchronous clocking of Timer/Counter2 by the AMR pin

- **DIG1 – Port G, Bit 1**

DIG1, Radio Transceiver Antenna Diversity Control Output: If the bit ANT_EXT_SW_EN in ANT_DIV is set to one then the PG1 pin serves as a Radio Transceiver output to control External Antenna Diversity.

- **DIG3 – Port G, Bit 0**

DIG3, Radio Transceiver RX/TX Indicator Output: If the bit PA_EXT_EN in TRX_CTRL_1 is set to one then the PG0 pin serves as the Radio Transceiver receive/transmit indicator output to control an external RF front-end.

Table 14-16 below relates the alternate functions of Port G to the overriding signals shown in Figure 14-5 on page 191.

Table 14-16. Overriding Signals for Alternate Functions PG5:PG2

Signal Name	PG5/OC0B	PG4/TOSC1	PG3/TOSC2	PG2/AMR
PUOE	–	AS2 & (~EXCLKAMR)	AS2 & (~EXCLKAMR) & (~EXCLK)	AS2 & EXCLKAMR
PUOV	–	0	0	0
DDOE	–	AS2 & (~EXCLKAMR)	AS2 & (~EXCLKAMR) & (~EXCLK)	AS2 & EXCLKAMR
DDOV	–	0	0	0
PVOE	OC0B Enable	0	0	0
PVOV	OC0B	0	0	0
DIEOE	–	AS2 & (~EXCLKAMR)	AS2 & (~EXCLKAMR) & (~EXCLK)	AS2 & EXCLKAMR
DIEOV	–	EXCLK	0	1
DI	–	–	–	AMR
AIO	–	T/C2 OSC INPUT	T/C2 OSC OUTPUT	–

Table 14-17. Overriding Signals for Alternate Functions PG1:PG0

Signal Name	PG1/DIG1	PG0/DIG3
PUOE	0	0
PUOV	0	0
DDOE	ANT_EXT_SW_EN	PA_EXT_EN
DDOV	ANT_EXT_SW_EN	PA_EXT_EN
PVOE	ANT_EXT_SW_EN	PA_EXT_EN
PVOV	DIG1	DIG3
DIEOE	0	0
DIEOV	0	0
DI	–	–
AIO	–	–

14.4 Register Description

14.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
$35 ($55)				PUD					MCUCR
Read/Write				RW					
Initial Value				0					

The MCU Control Register contains control bits of the general Microcontroller Unit functions.

- **Bit 4 – PUD - Pull-up Disable**

When this bit is written to one, the I/O ports pull-up resistors are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-up resistor ({DDxn, PORTxn} = 2'b01). See section "Ports as General Digital I/O" for more details about this feature.

14.4.2 DPDS0 – Port Driver Strength Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($136)	PFDRV1	PFDRV0	PEDRV1	PEDRV0	PDDRV1	PDDRV0	PBDRV1	PBDRV0	DPDS0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The output driver strength can be set individually for each digital I/O port. The following tables show output current levels for a typical supply voltage of DEVDD = 3.3V. Refer to section "Electrical Characteristics" for details.

- **Bit 7:6 – PFDRV1:0 - Driver Strength Port F**

Table 14-18 PFDRV Register Bits

Register Bits	Value	Description
PFDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 5:4 – PEDRV1:0 - Driver Strength Port E**

Table 14-19 PEDRV Register Bits

Register Bits	Value	Description
PEDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 3:2 – PDDRV1:0 - Driver Strength Port D**

Table 14-20 PDDRV Register Bits

Register Bits	Value	Description
PDDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

- **Bit 1:0 – PBDRV1:0 - Driver Strength Port B**

Table 14-21 PBDRV Register Bits

Register Bits	Value	Description
PBDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

14.4.3 DPDS1 – Port Driver Strength Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($137)	Res5	Res4	Res3	Res2	Res1	Res0	PGDRV1	PGDRV0	DPDS1
Read/Write	R	R	R	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The output driver strength can be set individually for each digital I/O port. The following table shows output current levels for a typical supply voltage of DEVDD = 3.3V. Refer to section "Electrical Characteristics" for details.

- **Bit 7:2 – Res5:0 - Reserved**
- **Bit 1:0 – PGDRV1:0 - Driver Strength Port G**

Driver strength can be set for port G except the port pins PG3 and PG4. The leakage current of the ports PG3 and PG4 is reduced.

Table 14-22 PGDRV Register Bits

Register Bits	Value	Description
PGDRV1:0	0	2 mA
	1	4 mA
	2	6 mA
	3	8 mA

14.4.4 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
$05 ($25)	PORTB7:0								PORTB
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

If PORTBn is written logic one when the PORTB pin n is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTBn has to be written

logic zero or the pin has to be configured as an output pin. If PORTBn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTBn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

- **Bit 7:0 – PORTB7:0 - Port B Data Register Value**

14.4.5 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$04 ($24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDBn bit in the DDRB Register selects the direction of the PORTB pin n. If DDBn is written logic one, PBn is configured as an output pin. If DDBn is written logic zero, PBn is configured as an input pin.

- **Bit 7:0 – DDB7:0 - Port B Data Direction Register Value**

14.4.6 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$03 ($23)	PINB7:0								PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register allows access to the PORTB pins independent of the setting of the Data Direction bit DDBn. The port pin can be read through the PINBn Register bit, and writing a logic one to PINBn toggles the value of PORTBn.

- **Bit 7:0 – PINB7:0 - Port B Input Pins Value**

14.4.7 PORTD – Port D Data Register

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	PORTD7:0								PORTD
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

If PORTDn is written logic one when the PORTD pin n is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTDn has to be written logic zero or the pin has to be configured as an output pin. If PORTDn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTDn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

- **Bit 7:0 – PORTD7:0 - Port D Data Register Value**

14.4.8 DDRD – Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDDn bit in the DDRD Register selects the direction of the PORTD pin n. If DDDn is written logic one, PDn is configured as an output pin. If DDDn is written logic zero, PDn is configured as an input pin.

- **Bit 7:0 – DDD7:0 - Port D Data Direction Register Value**

14.4.9 PIND – Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$09 ($29)	PIND7:0								PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register allows access to the PORTD pins independent of the setting of the Data Direction bit DDDn. The port pin can be read through the PINDn Register bit, and writing a logic one to PINDn toggles the value of PORTDn.

- **Bit 7:0 – PIND7:0 - Port D Input Pins Value**

14.4.10 PORTE – Port E Data Register

Bit	7	6	5	4	3	2	1	0	
$0E ($2E)	PORTE7:0								PORTE
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

If PORTEn is written logic one when the PORTE pin n is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTEn has to be written logic zero or the pin has to be configured as an output pin. If PORTEn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTEn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

- **Bit 7:0 – PORTE7:0 - Port E Data Register Value**

14.4.11 DDRE – Port E Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	DDRE
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDEn bit in the DDRE Register selects the direction of the PORTE pin n. If DDEn is written logic one, PEn is configured as an output pin. If DDEn is written logic zero, PEn is configured as an input pin.

- **Bit 7:0 – DDE7:0 - Port E Data Direction Register Value**

14.4.12 PINE – Port E Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	PINE7:0								PINE
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register allows access to the PORTE pins independent of the setting of the Data Direction bit DDEn. The port pin can be read through the PINEn Register bit, and writing a logic one to PINEn toggles the value of PORTEn.

- **Bit 7:0 – PINE7:0 - Port E Input Pins Value**

14.4.13 PORTF – Port F Data Register

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	PORTF7:0								PORTF
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

If PORTFn is written logic one when the PORTF pin n is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTFn has to be written logic zero or the pin has to be configured as an output pin. If PORTFn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTFn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

- **Bit 7:0 – PORTF7:0 - Port F Data Register Value**

14.4.14 DDRF – Port F Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	DDRF
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDFn bit in the DDRF Register selects the direction of the PORTF pin n. If DDFn is written logic one, PFn is configured as an output pin. If DDFn is written logic zero, PFn is configured as an input pin.

- **Bit 7:0 – DDF7:0 - Port F Data Direction Register Value**

14.4.15 PINF – Port F Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$0F ($2F)	PINF7:0								PINF
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register allows access to the PORTF pins independent of the setting of the Data Direction bit DDFn. The port pin can be read through the PINFn Register bit, and writing a logic one to PINFn toggles the value of PORTFn.

- **Bit 7:0 – PINF7:0 - Port F Input Pins Value**

14.4.16 PORTG – Port G Data Register

Bit	7	6	5	4	3	2	1	0	
$14 ($34)	Res1	Res0	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	PORTG
Read/Write	R	R	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

If PORTGn is written logic one when the PORTG pin n is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTGn has to be written logic zero or the pin has to be configured as an output pin. If PORTGn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTGn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5:0 – PORTG5:0 - Port G Data Register Value**

14.4.17 DDRG – Port G Data Direction Register

Bit	7	6	5	4	3	2	1	0	
$13 ($33)	Res1	Res0	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDRG
Read/Write	R	R	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The DDGn bit in the DDRG Register selects the direction of the PORTG pin n. If DDGn is written logic one, PGn is configured as an output pin. If DDGn is written logic zero, PGn is configured as an input pin.

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5:0 – DDG5:0 - Port G Data Direction Register Value**

14.4.18 PING – Port G Input Pins Address

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	Res1	Res0	PING5	PING4	PING3	PING2	PING1	PING0	PING
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

This register allows access to the PORTG pins independent of the setting of the Data Direction bit DDGn. The port pin can be read through the PINGn Register bit, and writing a logic one to PINGn toggles the value of PORTGn.

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5:0 – PING5:0 - Port G Input Pins Value**

15 Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega128RFA1. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 15.

15.1 Interrupt Vectors in ATmega128RFA1

Table 15-1. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
0	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
1	$0002	INT0	External Interrupt Request 0
2	$0004	INT1	External Interrupt Request 1
3	$0006	INT2	External Interrupt Request 2
4	$0008	INT3	External Interrupt Request 3
5	$000A	INT4	External Interrupt Request 4
6	$000C	INT5	External Interrupt Request 5
7	$000E	INT6	External Interrupt Request 6
8	$0010	INT7	External Interrupt Request 7
9	$0012	PCINT0	Pin Change Interrupt Request 0
10	$0014	PCINT1	Pin Change Interrupt Request 1
11	$0016(3)	PCINT2	Pin Change Interrupt Request 2
12	$0018	WDT	Watchdog Time-out Interrupt
13	$001A	TIMER2_COMPA	Timer/Counter2 Compare Match A
14	$001C	TIMER2_COMPB	Timer/Counter2 Compare Match B
15	$001E	TIMER2_OVF	Timer/Counter2 Overflow
16	$0020	TIMER1_CAPT	Timer/Counter1 Capture Event
17	$0022	TIMER1_COMPA	Timer/Counter1 Compare Match A
18	$0024	TIMER1_COMPB	Timer/Counter1 Compare Match B
19	$0026	TIMER1_COMPC	Timer/Counter1 Compare Match C
20	$0028	TIMER1_OVF	Timer/Counter1 Overflow
21	$002A	TIMER0_COMPA	Timer/Counter0 Compare Match A
22	$002C	TIMER0_COMPB	Timer/Counter0 Compare match B
23	$002E	TIMER0_OVF	Timer/Counter0 Overflow
24	$0030	SPI_STC	SPI Serial Transfer Complete
25	$0032	USART0_RX	USART0 Rx Complete
26	$0034	USART0_UDRE	USART0 Data Register Empty
27	$0036	USART0_TX	USART0 Tx Complete
28	$0038	ANALOG_COMP	Analog Comparator

Vector No.	Program Address(2)	Source	Interrupt Definition
29	$003A	ADC	ADC Conversion Complete
30	$003C	EE_READY	EEPROM Ready
31	$003E	TIMER3_CAPT	Timer/Counter3 Capture Event
32	$0040	TIMER3_COMPA	Timer/Counter3 Compare Match A
33	$0042	TIMER3_COMPB	Timer/Counter3 Compare Match B
34	$0044	TIMER3_COMPC	Timer/Counter3 Compare Match C
35	$0046	TIMER3_OVF	Timer/Counter3 Overflow
36	$0048	USART1_RX	USART1 Rx Complete
37	$004A	USART1_UDRE	USART1 Data Register Empty
38	$004C	USART1_TX	USART1 Tx Complete
39	$004E	TWI	2-wire Serial Interface
40	$0050	SPM_READY	Store Program Memory Ready
41	$0052(3)	TIMER4_CAPT	Timer/Counter4 Capture Event
42	$0054	TIMER4_COMPA	Timer/Counter4 Compare Match A
43	$0056	TIMER4_COMPB	Timer/Counter4 Compare Match B
44	$0058	TIMER4_COMPC	Timer/Counter4 Compare Match C
45	$005A	TIMER4_OVF	Timer/Counter4 Overflow
46	$005C(3)	TIMER5_CAPT	Timer/Counter5 Capture Event
47	$005E	TIMER5_COMPA	Timer/Counter5 Compare Match A
48	$0060	TIMER5_COMPB	Timer/Counter5 Compare Match B
49	$0062	TIMER5_COMPC	Timer/Counter5 Compare Match C
50	$0064	TIMER5_OVF	Timer/Counter5 Overflow
51	$0066(3)	Reserved	
52	$0068(3)	Reserved	
53	$006A(3)	Reserved	
54	$006C(3)	Reserved	
55	$006E(3))	Reserved	
56	$0070(3)	Reserved	
57	$0072	TRX24_PLL_LOCK	Transceiver PLL Lock
58	$0074	TRX24_PLL_UNLOCK	Transceiver PLL Unlock
59	$0076	TRX24_RX_START	Transceiver Receive Start
60	$0078	TRX24_RX_END	Transceiver Receive End
61	$007A	TRX24_CCA_ED_DONE	Transceiver CCAED Meassurement finished
62	$007C	TRX24_XAH_AMI	Transceiver Frame Address Match
63	$007E	TRX24_TX_END	Transceiver Transmit End
64	$0080	TRX24_AWAKE	Transceiver Wakeup finished

Vector No.	Program Address(2)	Source	Interrupt Definition
65	$0082	SCNT_CMP1	Symbol Counter Compare Match 1
66	$0084	SCNT_CMP 2	Symbol Counter Compare Match 2
67	$0086	SCNT_CMP 3	Symbol Counter Compare Match 3
68	$0088	SCNT_OVFL	Symbol Counter Overflow
69	$008A	SCNT_BACKOFF	Symbol Counter Backoff Slot Counter
70	$008C	AES_READY	AES Encryption Ready
71	$008E	BAT_LOW	Batterie Monitor Allert

Note:
1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Memory Programming" on page 464.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.
3. Not usefull in ATmega128RFA1 due to limited pin count.

15.2 Reset and Interrupt Vector Placement

Table 15-2 below 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 15-2. 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 Address is shown in Table 30-7 on page 461 through Table 30-6 on page 461. 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 ATmega128RFA1 is:

```
Address   Labels   Code                    Comments
0x0000             jmp    RESET            ;Reset Handler
0x0002             jmp    INT0             ;IRQ0 Handler
0x0004             jmp    INT1             ;IRQ1 Handler
0x0006             jmp    INT2             ;IRQ2 Handler
0x0008             jmp    INT3             ;IRQ3 Handler
0x000A             jmp    INT4             ;IRQ4 Handler
0x000C             jmp    INT5             ;IRQ5 Handler
0x000E             jmp    INT6             ;IRQ6 Handler
0x0010             jmp    INT7             ;IRQ7 Handler
0x0012             jmp    PCINT0           ;PCINT0 Handler
0x0014             jmp    PCINT1           ;PCINT1 Handler
0x0016             jmp    PCINT2           ;PCINT2 Handler
```

```
0X0018          jmp     WDT                 ;Watchdog Timeout Handler
0x001A          jmp     TIM2_COMPA          ;Timer2 CompareA Handler
0x001C          jmp     TIM2_COMPB          ;Timer2 CompareB Handler
0x001E          jmp     TIM2_OVF            ;Timer2 Overflow Handler
0x0020          jmp     TIM1_CAPT           ;Timer1 Capture Handler
0x0022          jmp     TIM1_COMPA          ;Timer1 CompareA Handler
0x0024          jmp     TIM1_COMPB          ;Timer1 CompareB Handler
0x0026          jmp     TIM1_COMPC          ;Timer1 CompareC Handler
0x0028          jmp     TIM1_OVF            ;Timer1 Overflow Handler
0x002A          jmp     TIM0_COMPA          ;Timer0 CompareA Handler
0x002C          jmp     TIM0_COMPB          ;Timer0 CompareB Handler
0x002E          jmp     TIM0_OVF            ;Timer0 Overflow Handler
0x0030          jmp     SPI_STC             ;SPI Transfer Complete Handler
0x0032          jmp     USART0_RX           ;USART0 RX Complete Handler
0x0034          jmp     USART0_UDRE         ;USART0,UDR Empty Handler
0x0036          jmp     USART0_TX           ;USART0 TX Complete Handler
0x0038          jmp     ANA_COMP            ;Analog Comparator Handler
0x003A          jmp     ADC                 ;ADC Conversion Complete Handler
0x003C          jmp     EE_RDY              ;EEPROM Ready Handler
0x003E          jmp     TIM3_CAPT           ;Timer3 Capture Handler
0x0040          jmp     TIM3_COMPA          ;Timer3 CompareA Handler
0x0042          jmp     TIM3_COMPB          ;Timer3 CompareB Handler
0x0044          jmp     TIM3_COMPC          ;Timer3 CompareC Handler
0x0046          jmp     TIM3_OVF            ;Timer3 Overflow Handler
0x0048          jmp     USART1_RX           ;USART1 RX Complete Handler
0x004A          jmp     USART1_UDRE         ;USART1,UDR Empty Handler
0x004C          jmp     USART1_TX           ;USART1 TX Complete Handler
0x004E          jmp     TWI                 ;2-wire Serial Handler
0x0050          jmp     SPM_RDY             ;SPM Ready Handler
0x0052          jmp     TIM4_CAPT           ;Timer4 Capture Handler
0x0054          jmp     TIM4_COMPA          ;Timer4 CompareA Handler
0x0056          jmp     TIM4_COMPB          ;Timer4 CompareB Handler
0x0058          jmp     TIM4_COMPC          ;Timer4 CompareC Handler
0x005A          jmp     TIM4_OVF            ;Timer4 Overflow Handler
0x005C          jmp     TIM5_CAPT           ;Timer5 Capture Handler
0x005E          jmp     TIM5_COMPA          ;Timer5 CompareA Handler
0x0060          jmp     TIM5_COMPB          ;Timer5 CompareB Handler
0x0062          jmp     TIM5_COMPC          ;Timer5 CompareC Handler
0x0064          jmp     TIM5_OVF            ;Timer5 Overflow Handler
0x0066          jmp     0x15e               ;0x15e <__bad_interrupt>
0x0068          jmp     0x15e               ;0x15e <__bad_interrupt>
0x006A          jmp     0x15e               ;0x15e <__bad_interrupt>
0x006C          jmp     0x15e               ;0x15e <__bad_interrupt>
0x006E          jmp     0x15e               ;0x15e <__bad_interrupt>
0x0070          jmp     0x15e               ;0x15e <__bad_interrupt>
0x0072          jmp     TRX24_PLL_LOCK      ;Transceiver PLL Lock Handler
0x0074          jmp     TRX24_PLL_UNLOCK    ;Transceiver PLL Unlock Handler
0x0076          jmp     TRX24_RX_START      ;Transceiver RX Start Handler
0x0078          jmp     TRX24_RX_END        ;Transceiver RX End Handler
0x007A          jmp     TRX24_CCA_ED_DONE   ;Transceiver CCAED DONE Handler
0x007C          jmp     TRX24_XAH_AMI       ;Transceiver Addr. Match Handler
0x007E          jmp     TRX24_TX_END        ;Transceiver Transmit End Handler
0x0080          jmp     TRX24_AWAKE         ;Transceiver Wake Up Handler
0x0082          jmp     SCNT_CMP1           ;Symbol Counter Compare Match 1
0x0084          jmp     SCNT_CMP2           ;Symbol Counter Compare Match 2
0x0086          jmp     SCNT_CMP3           ;Symbol Counter Compare Match 3
0x0088          jmp     SCNT_OVFL           ;Symbol Counter Overflow Handler
0x008A          jmp     SCNT_BACKOFF        ;Symbol Backoff Slot Counter H.
```

```
0x008C              jmp     AES_READY          ;Encryption/Decryption Ready H.
0x008E              jmp     BAT_LOW            ;Batterie Monitor Alert Handler
  ;
0x0090   RESET:     ldi     r16, high(RAMEND)  ;Main program start
0x0091              out     SPH,r16            ;Set Stack Pointer to top of RAM
0x0092              ldi     r16, low(RAMEND)
0x0093              out     SPL,r16
0x0094              sei                        ;Enable interrupts
0x0095            <instr>   xxx
 ...     ...        ...     ...
```

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 8KBytes 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 0xF002
 0xF002             jmp EXT_INT0          ;IRQ0 Handler
 0xF004             jmp EXT_INT1          ;IRQ1 Handler
 ... ... ... ;
 0xFO70             jmp USART3_TXC        ;USART3 TX Complete Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 8KBytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address Labels      Code                  Comments
.org 0x0002
 0x0002             jmp EXT_INT0          ;IRQ0 Handler
 0x0004             jmp EXT_INT1          ;IRQ1 Handler
 ... ... ... ;
.org 0xF000
 0xF000  RESET:     ldi r16,high(RAMEND)  ;Main program start
 0xF001             out SPH,r16           ;Set Stack Pointer to top of RAM
 0xF002             ldi r16,low(RAMEND)
 0xF003             out SPL,r16
 0xF004             sei                   ;Enable interrupts
 0xF005             <instr> xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 8KBytes 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 0xF000
 0xF000             jmp RESET             ;Reset handler
 0xF002             jmp EXT_INT0          ;IRQ0 Handler
 0xF004             jmp EXT_INT1          ;IRQ1 Handler
 ... ... ... ;
 0xF072  RESET:     ldi r16,high(RAMEND)  ; Main program start
```

```
0xF073          out SPH,r16          ;Set Stack Pointer to top of RAM
0xF074          ldi r16,low(RAMEND)
0xF075          out SPL,r16
0xF076          sei                  ;Enable interrupts
0xFO77          <instr> xxx
```

15.3 Moving Interrupts Between Application and Boot Section

The MCU Control Register controls the placement of the Interrupt Vector table, see Code Example below. For more details, see "Reset and Interrupt Handling" on page 15.

Assembly Code Example

```
Move_interrupts:
  ; Get MCUCR
  in r16, MCUCR
  mov r17, r16
  ; Enable change of Interrupt Vectors
  ori r16, (1<<IVCE)
  out MCUCR, r16
  ; Move interrupts to Boot Flash section
  ori r16, (1<<IVSEL)
  out MCUCR, r17
  ret
```

C Code Example

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

15.4 Register Description

15.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	JTD	Res1	Res0	PUD	Res1	Res0	IVSEL	IVCE	MCUCR
Read/Write	RW	R	R	RW	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The MCU Control Register contains control bits for general Microcontroller Unit functions.

- **Bit 7 – JTD - JTAG Interface Disable**

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

- **Bit 6:5 – Res1:0 - Reserved**
- **Bit 4 – PUD - Pull-up Disable**

When this bit is written to one, the I/O ports pull-up resistors are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-up resistor ({DDxn, PORTxn} = 2'b01). See section "Ports as General Digital I/O" for more details about this feature.

- **Bit 3:2 – Res1:0 - Reserved**
- **Bit 1 – IVSEL - Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to the section "Memory Programming" for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit (see section "Moving Interrupts Between Application and Boot Section" for details): 1. Write the Interrupt Vector Change Enable (IVCE) bit to one; 2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE. Interrupts will be automatically disabled while this sequence is executed. Interrupts are disabled in the same 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 that 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.

- **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.

16 External Interrupts

The External Interrupts are triggered by the INT7:0 pin or any of the PCINT8:0 pins. Observe that if enabled, the interrupts will trigger even if the INT7:0 or PCINT8:0 pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin Change Interrupt PCI0 will trigger if any enabled PCINT7:0 pin toggles, Pin change interrupt PCI1 if the enabled PCINT8 toggles. PCINT23:9 have no function inside the ATmega128RFA1. Their corresponding I/O port are not implemented. PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. PCI2 and PCMSK2 associated to PCINT23:16 have no task in this design. Pin change interrupts on PCINT8:0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

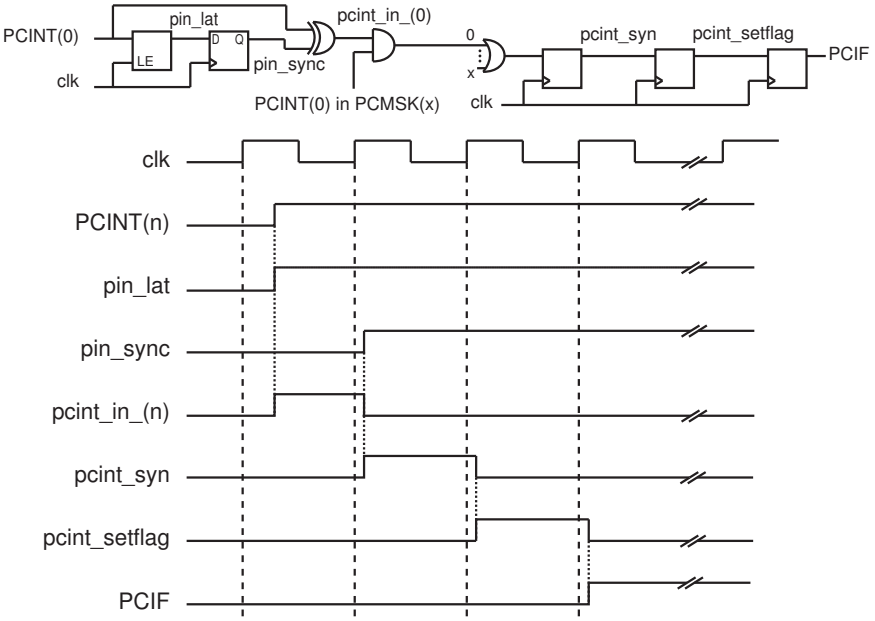
The External Interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Registers – EICRA (INT3:0) and EICRB (INT7:4). When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT7:4 requires the presence of an I/O clock, described in "Overview" on page 3. Low level interrupts and the edge interrupt on INT3:0 are detected asynchronously. This implies that these interrupts 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 and CKSEL Fuses as described in "Clock Sources" on page 148.

16.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 16-1 below.

Figure 16-1. Normal Pin Change Interrupt



16.2 Register Description

16.2.1 EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($69)	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The External Interrupts 3 - 0 are activated by the external pins INT3:0 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in the following tables. Edges on INT3:0 are registered asynchronously. Pulses on INT3:0 pins wider than the minimum pulse width of typical 50 ns 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. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low. When changing the ISCn bit, an interrupt can occur. Therefore, it is recommended to first disable INTn by clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISCn bit can be changed. Finally, the INTn interrupt flag should be cleared by writing a logical one to its Interrupt Flag bit (INTFn) in the EIFR Register before the interrupt is re-enabled. When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

- **Bit 7:6 – ISC31:30 - External Interrupt 3 Sense Control Bit**

Table 16-126 ISC3 Register Bits

Register Bits	Value	Description
ISC31:30	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 5:4 – ISC21:20 - External Interrupt 2 Sense Control Bit**

Table 16-127 ISC2 Register Bits

Register Bits	Value	Description
ISC21:20	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 3:2 – ISC11:10 - External Interrupt 1 Sense Control Bit**

Table 16-128 ISC1 Register Bits

Register Bits	Value	Description
ISC11:10	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 1:0 – ISC01:00 - External Interrupt 0 Sense Control Bit**

Table 16-129 ISC0 Register Bits

Register Bits	Value	Description
ISC01:00	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

16.2.2 EICRB – External Interrupt Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($6A)	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The External Interrupts 7 - 4 are activated by the external pins INT7:4 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in the following tables. Edges on INT7:4 are registered asynchronously. Pulses on INT7:4 pins wider than the minimum pulse width of typical 50 ns 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. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low. When changing the ISCn bit, an interrupt can occur. Therefore, it is recommended to first disable INTn by clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISCn bit can be changed. Finally, the INTn interrupt flag should be cleared by writing a logical one to its Interrupt Flag bit (INTFn) in the EIFR Register before the interrupt is re-enabled. When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

- **Bit 7:6 – ISC71:70 - External Interrupt 7 Sense Control Bit**

Table 16-130 ISC7 Register Bits

Register Bits	Value	Description
ISC71:70	0x00	The low level of INTn generates an interrupt

Register Bits	Value	Description
		request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 5:4 – ISC61:60 - External Interrupt 6 Sense Control Bit**

Table 16-131 ISC6 Register Bits

Register Bits	Value	Description
ISC61:60	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 3:2 – ISC51:50 - External Interrupt 5 Sense Control Bit**

Table 16-132 ISC5 Register Bits

Register Bits	Value	Description
ISC51:50	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

- **Bit 1:0 – ISC41:40 - External Interrupt 4 Sense Control Bit**

Table 16-133 ISC4 Register Bits

Register Bits	Value	Description
ISC41:40	0x00	The low level of INTn generates an interrupt request.
	0x01	Any edge of INTn generates asynchronously an interrupt request.
	0x02	The falling edge of INTn generates asynchronously an interrupt request.
	0x03	The rising edge of INTn generates asynchronously an interrupt request.

16.2.3 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

When an INT7:0 bit is written to one and the I-bit in the Status Register (SREG) is set (one), the corresponding external pin interrupt is enabled. The Interrupt Sense Control bits in the External Interrupt Control Registers EICRA and EICRB define whether the External Interrupt is activated on rising or falling edge or level sensed. Activity on any of these pins will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

- **Bit 7:0 – INT7:0 - External Interrupt Request Enable**

Table 16-134 INT Register Bits

Register Bits	Value	Description
INT7:0	0x00	All external pin interrupts are disabled.
	0xff	All external pin interrupts are enabled.

16.2.4 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$1C ($3C)	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

When an edge or logic change on the INT7:0 pin triggers an interrupt request, INTF7:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit INT7:0 in EIMSK are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. These flags are always cleared when INT7:0 are configured as level interrupt. Note that when entering sleep mode with the INT3:0 interrupts disabled, the input buffers on these pins will be disabled. This may cause a logic change in internal signals which will set the INTF3:0 flags. See "Digital Input Enable and Sleep Modes" for more information.

- **Bit 7:0 – INTF7:0 - External Interrupt Flag**

Table 16-135 INTF Register Bits

Register Bits	Value	Description
INTF7:0	0x00	No edge or logic change on INT7:0 occurred.
	0x01	A edge or logic change on INT0 occurred and triggered an interrupt request.
	0x02	...
	0x80	A edge or logic change on INT7 occurred and triggered an interrupt request.

16.2.5 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($68)	Res4	Res3	Res2	Res1	Res0	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – PCIE2 - Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23:16 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT23:16 pins are enabled individually by the PCMSK2 Register. Note that the I/O ports corresponding to PCINT23:16 are not implemented. Therefore PCIE2 has no function in this device.

- **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 PCINT15:8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT15:8 pins are enabled individually by the PCMSK1 Register. Note that the I/O ports corresponding to PCINT15:9 are not implemented.

- **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 PCINT7:0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT7:0 pins are enabled individually by the PCMSK0 Register.

16.2.6 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	Res4	Res3	Res2	Res1	Res0	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – PCIF2 - Pin Change Interrupt Flag 2**

When a logic change on any PCINT23:16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 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. Note that the I/O ports corresponding to PCINT23:16 are not implemented. Therefore PCIF2 has no function in this device.

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

When a logic change on any PCINT15:8 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. Note that the I/O ports corresponding to PCINT15:9 are not implemented.

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

When a logic change on any PCINT7:0 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.

16.2.7 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	
NA ($6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCMSK2
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($6D)	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

Note that the PCMSK2 register has no function in this device. The I/O ports associated to PCINT23:16 are not implemented. Normally each bit PCINT23:16 selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT23:16 is set and the PCIE2 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT23:16 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

- **Bit 7:0 – PCINT23:16 - Pin Change Enable Mask**

16.2.8 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	
NA ($6C)	PCINT15	PCINT14	PCINT13	PCINT12	PCMSK1
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($6C)	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

Bit PCINT8 selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT8 is set and the PCIE1 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT8 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

- **Bit 7:1 – PCINT15:9 - Pin Change Enable Mask**

Bits 15:9 of the PCMSK1 register have no function in this device. The I/O ports associated to PCINT15:9 are not implemented.

- **Bit 0 – PCINT8 - Pin Change Enable Mask 8**

If this bit is set to one the pin change interrupt on the corresponding I/O pin is enabled. If this bit is set to zero the pin change interrupt is disabled.

16.2.9 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

Each bit PCINT7:0 selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is set and the PCIE0 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

- **Bit 7:0 – PCINT7:0 - Pin Change Enable Mask**

17 8-bit Timer/Counter0 with PWM

17.1 Features

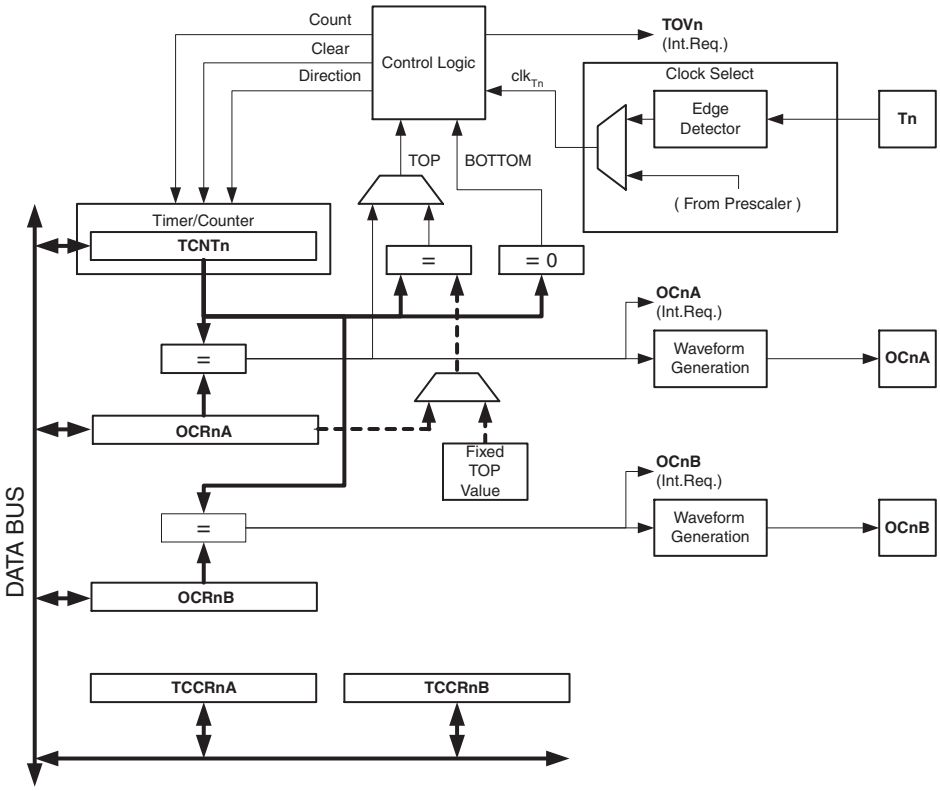
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **Three Independent Interrupt Sources (TOV0, OCF0A, and OCF0B)**

17.2 Overview

Timer/Counter0 is a general purpose 8-bit Timer/Counter module with two independent Output Compare Units and with PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 17-1. For the actual placement of I/O pins refer to section "Pin Configurations" on page 2. 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 "Register Description" on page 238.

Figure 17-1. 8-bit Timer/Counter Block Diagram



17.2.1 Registers

The Timer/Counter (TCNT0) and Output Compare Registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request signals (abbreviated to *Int.Req.* in the figure) are all visible in the Timer Interrupt Flag Register (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B); see "Output Compare Unit" on page 228 for details. The Compare Match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

17.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case "*n*" replaces the Timer/Counter number (in this case 0). A lower case "*x*" replaces the Output Compare Unit (in this case Compare Unit A or Compare Unit B). However when using the register or bit defines in a program, the precise form must be used i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 17-1 are also used extensively throughout the document.

Table 17-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
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 (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

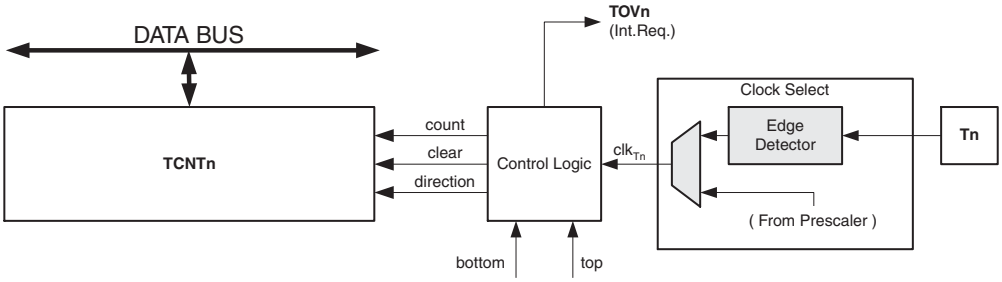
17.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler see Timer/Counter 0, 1, 3, 4, and 5 Prescaler on page 304

17.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 17-2 shows a block diagram of the counter and its surroundings.

Figure 17-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1;
direction	Select between increment and decrement;
clear	Clear TCNT0 (set all bits to zero);
clk$_{Tn}$	Timer/Counter clock referred to as clk$_{T0}$ in the following text;
top	Signalize that TCNT0 has reached maximum value;
bottom	Signalize that TCNT0 has reached minimum value (zero);

Depending of the mode of operation used, the counter is cleared, incremented or decremented at each timer clock (clk$_{T0}$). clk$_{T0}$ can be generated from an external or internal clock source selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU regardless of whether clk$_{T0}$ is present or not. A CPU write access overrides (has priority over) all counter clear or count operations.

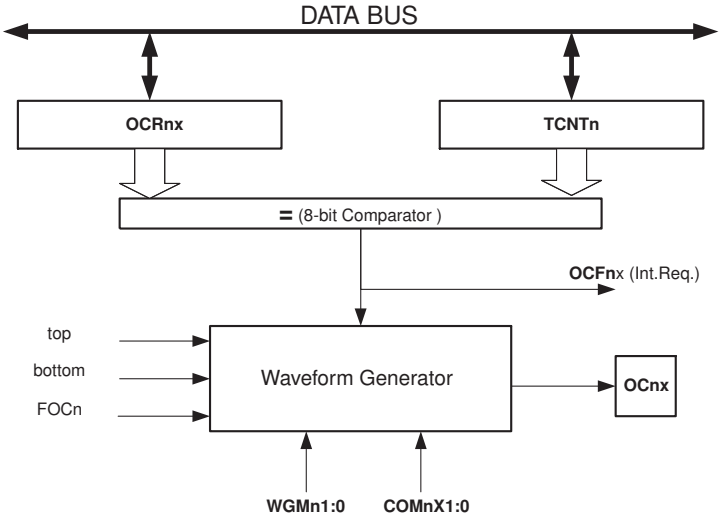
The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 232.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

17.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). The comparator signals a match whenever TCNT0 equals OCR0A or OCR0B. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next clock cycle of the timer. 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. The flag can alternatively be software-cleared by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to the operating mode set by the WGM02:0 bits and Compare Output mode (COM0*x*1:0) bits. The MAX and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (refer to "Modes of Operation" on page 232).

Figure 17-3. Output Compare Unit, Block Diagram



The OCR0*x* Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0*x* Compare Registers to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses and thereby making the output glitch-free.

The OCR0*x* Register access may seem complex, but this is not the case. When the double buffering is enabled, the CPU has access to the OCR0*x* Buffer Register. If double buffering is disabled the CPU will access the OCR0*x* directly.

17.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0*x*) bit. Forcing Compare Match will not set the OCF0*x* Flag or reload/clear the timer, but the OC0*x* pin will be updated as if a real Compare Match had occurred (the COM0*x*1:0 bits settings define whether the OC0*x* pin is set, cleared or toggled).

17.5.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0*x* to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

17.5.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNT0 while using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0*x* value, the Compare Match will be missed resulting in an incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down-counting.

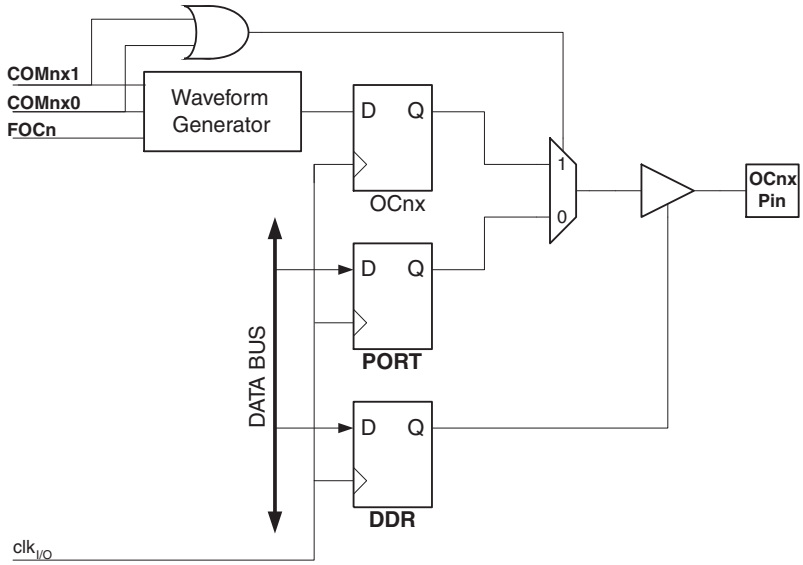
The setup of the OC0*x* should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0*x* value is to use the Force Output Compare (FOC0*x*) strobe bits in Normal mode. The OC0*x* Registers keep their values even when changing between Waveform Generation modes.

Be aware that the COM0*x*1:0 bits are not double buffered together with the compare value. A Change of the COM0*x*1:0 bits will take effect immediately.

17.6 Compare Match Output Unit

The Compare Output mode (COM0*x*1:0) bits have two functions. The Waveform Generator uses the COM0*x*1:0 bits for defining the Output Compare (OC0*x*) state at the next Compare Match. The COM0x1:0 bits control also the OC0x pin output source. Figure 17-4 shows a simplified schematic of the logic affected by the COM0*x*1:0 bit setting. The I/O Registers, I/O bits and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) affected by the COM0*x*1:0 bits are shown. When referring to the OC0*x* state, the reference is to the internal OC0*x* Register and not to the OC0*x* pin. The OC0*x* Register is reset to “0” if a system reset occurs.

Figure 17-4. Compare Match Output Unit Schematic



The general I/O port function is overridden by the Output Compare (OC0*x*) from the Waveform Generator if either of the COM0*x*1:0 bits are set. However the OC0*x* pin direction (input or output) is still controlled by the Data Direction Register (DDR) of the port pin. The Data Direction Register bit of the OC0*x* pin (DDR_OC0*x*) must be set as output before the OC0*x* value is visible at the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initializing the OC0*x* state before the output is enabled. Note that some COM0*x*1:0 bit settings are reserved for certain modes of operation (see "Register Description" on page 238).

17.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0*x*1:0 bits differently in Normal, CTC and PWM modes. A setting of COM0*x*1:0 = 0 tells the Waveform Generator in all modes that no action on the OC0*x* Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 17-2. For fast PWM mode refer to Table 17-3 and for phase correct PWM refer to Table 17-4.

A state change of the COM0*x*1:0 bits will have effect at the first Compare Match after the bits are written. For non-PWM modes the action can be forced to have immediate effect by using the FOC0*x* strobe bits.

The following table shows the COM0*x*1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 17-2. Compare Output Mode, non-PWM Mode

COM0A1 COM0B1	COM0A0 COM0B0	Description
0	0	Normal port operation, OC0*x* disconnected;
0	1	Toggle OC0*x* on Compare Match;
1	0	Clear OC0*x* on Compare Match;
1	1	Set OC0*x* on Compare Match;

Table 17-3 shows the COM0*x*1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 17-3. Compare Output Mode, Fast PWM Mode

COM0A1 COM0B1	COM0A0 COM0B0	Description
0	0	Normal port operation, OC0*x* disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match. OC0B: not applicable, reserved function;
1	0	Clear OC0*x* on Compare Match, set OC0*x* at BOTTOM, (non-inverting mode).
1	1	Set OC0*x* on Compare Match, clear OC0*x* at BOTTOM, (inverting mode).

Note: A special case occurs when OCR0*x* equals TOP and COM0*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 233.

Table 17-4 shows the COM0*x*1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 17-4. Compare Output Mode, Phase Correct PWM Mode

COM0A1 COM0B1	COM0A0 COM0B0	Description
0	0	Normal port operation, OC0*x* disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match. OC0B: not applicable, reserved function;
1	0	Clear OC0*x* on Compare Match when up-counting. Set OC0*x* on Compare Match when down-counting.
1	1	Set OC0*x* on Compare Match when up-counting. Clear OC0*x* on Compare Match when down-counting.

Note: A special case occurs when OCR0*x* equals TOP and COM0*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 233 for more details.

17.7 Modes of Operation

The mode of operation i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM02:0) and Compare Output mode (COM0*x*1:0) bits. The Compare Output mode bits do not affect the counting sequence while the Waveform Generation mode bits do. The COM0*x*1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0*x*1:0 bits control whether the output should be set, cleared, or toggled at a Compare Match (see "Output Compare Unit" on page 228).

For detailed timing information see "Timer/Counter Timing Diagrams" on page 236.

Table 17-5 shows the function of the WGM2:0 bits of registers TCCR0A and TCCR0B. These bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used.

Table 17-5. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRX at	TOV Flag Set on(0,0)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

17.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM02:0 = 0). In this mode the counting direction is always up (incrementing) and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the BOTTOM (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set at the same timer clock cycle when the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a 9th bit, except that it is only set and not cleared. However, the timer resolution can be increased by software utilizing the timer overflow interrupt that automatically clears the TOV0 Flag. There are no special cases to consider in the Normal mode. A new counter value can be written at anytime.

The Output Compare Unit can be used to generate interrupts at some given time. It is not recommended to use the Output Compare for waveform generation in Normal mode, since this will occupy too much CPU time.

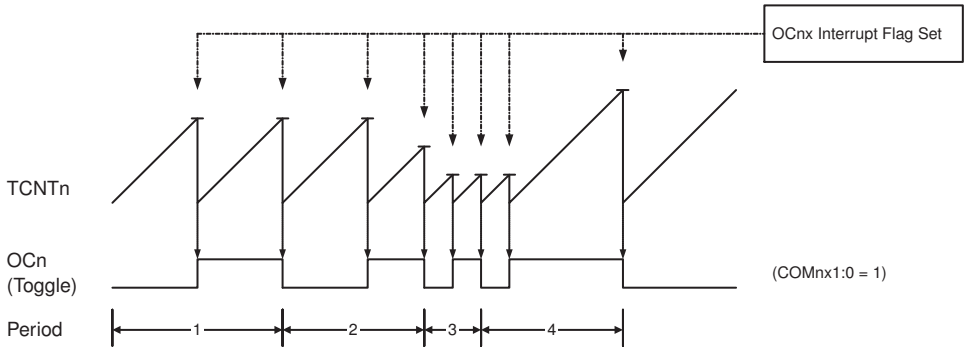
17.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare (CTC) mode (WGM02:0 = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches OCR0A. The OCR0A value 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 17-5. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0A. The counter (TCNT0) is then cleared.

Figure 17-5. CTC Mode Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A Flag. If the interrupt is enabled, the interrupt handler routine can update the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with no 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 OCR0A is lower than the current value of TCNT0, 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.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction of the pin is set to output. The generated waveform will have a maximum frequency of $f_{OC0} = f_{clk_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OC0x} = \frac{f_{CLK_I/O}}{2 \cdot N \cdot (1 + OCR0x)}$$

The N variable represents the prescale factor (1, 8, 64, 256 or 1024).

As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter changes from MAX to 0x00.

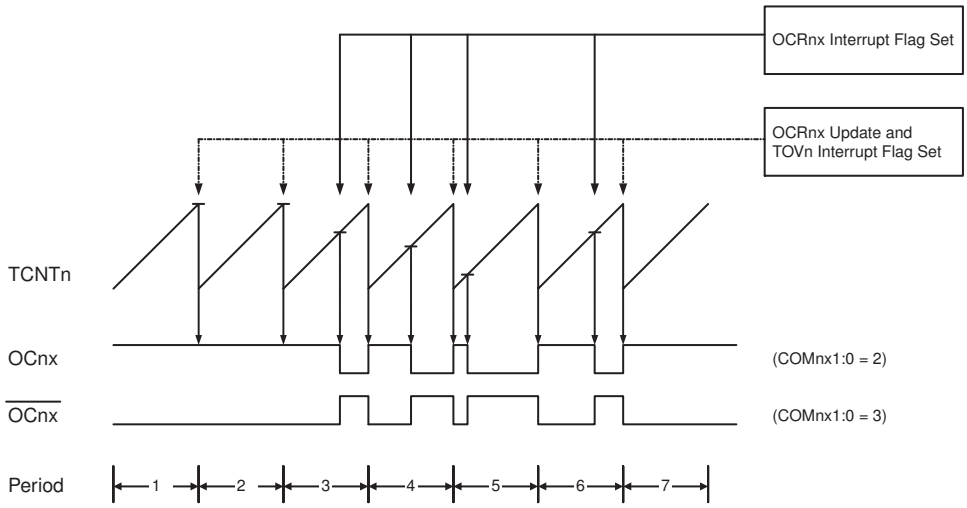
17.7.3 Fast PWM Mode

The fast Pulse Width Modulation (PWM) mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM mode differs from the other PWM modes by its single-slope operation. The counter counts from BOTTOM to TOP and then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting Compare Output mode the Output Compare (OC0*x*) is cleared on the Compare Match between TCNT0 and OCR0*x* and set at BOTTOM. In inverting Compare Output mode the output is set on Compare Match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as in the phase correct PWM mode that uses dual-slope operation. This high frequency operation makes the fast

PWM mode well suited for power regulation, rectification and DAC applications. The high frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 17-6. The TCNT0 value is shown in the timing diagram as a histogram illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0*x* and TCNT0.

Figure 17-6. Fast PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. The interrupt handler routine can be used for updating the compare value if the interrupt is enabled.

In fast PWM mode the compare unit allows generating PWM waveforms on the OC0*x* pins. Setting the COM0*x*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0*x*1:0 to 3. Setting the COM0A1:0 bits to 1 allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 17-3 on page 231). The actual OC0*x* value will only be visible at the port pin if the data direction of the port pin is set to output. The PWM waveform is generated by setting (or clearing) the OC0*x* Register at the Compare Match between OCR0*x* and TCNT0, and by clearing (or setting) the OC0*x* Register at the timer clock cycle when the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output $f_{OC0xPWM}$ can be calculated with the following equation:

$$f_{OC0xPWM} = \frac{f_{clk_I/O}}{N \cdot 256}$$

The N variable represents the pre-scale factor (1, 8, 64, 256 or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A

equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

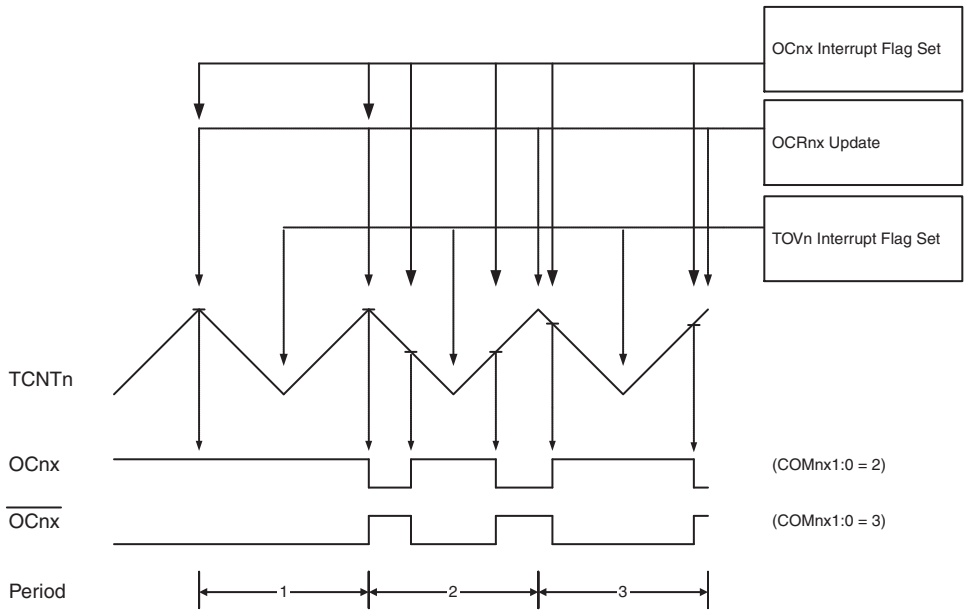
A frequency with 50% duty cycle waveform output in fast PWM mode can be achieved by setting OC0*x* to toggle its logical level on each Compare Match (COM0*x*1:0 = 1). The generated waveform will have a maximum frequency of $f_{OC0xPWM} = f_{clk_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except that in the fast PWM mode the double buffer feature of the Output Compare unit is enabled.

17.7.4 Phase Correct PWM Mode

The phase correct pulse-width modulation (PWM) mode (WGM02:0 = 1 or 5) provides a phase-correct, high-resolution PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 1 and TOP = OCR0A when WGM2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the Compare Match between TCNT0 and OCR0*x* while up-counting, and OC0*x* is set on the Compare Match while down-counting. The operation is inverted in inverting Output Compare mode. The dual-slope operation has a lower maximum operation frequency than single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. The counter changes the direction when reaching TOP. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown in Figure 17-7 below. The TCNT0 value is shown in the timing diagram as a histogram illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0*x* and TCNT0.

Figure 17-7. Phase Correct PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generating PWM waveforms on the OC0*x* pins. Setting the COM0*x*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0*x*1:0 to 3. Setting the COM0A0 bits to 1 allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 17-4 on page 231). The actual OC0*x* value will only be visible at the port pin if the data direction for the port pin is set to output. The PWM waveform is generated by clearing (or setting) the OC0*x* Register at the Compare Match between OCR0*x* and TCNT0 when the counter increments, and by setting (or clearing) the OC0*x* Register at Compare Match between OCR0*x* and TCNT0 when the counter decrements. The PWM frequency for the output $f_{OC0xPCPWM}$ when using phase-correct PWM can be calculated with the following equation:

$$f_{OC0xPCPWM} = \frac{f_{clk_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256 or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the phase-correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 17-7 OC*nx* has a transition from high to low even though there is no Compare Match. The reason of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

- OCR0*x* changes its value from MAX like in Figure 17-7 on page 235. When the OCR0*x* value is MAX the OC0*x* pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OC0*x* value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR0*x*. For that reason it misses the Compare Match and hence the OC0*x* change that would have happened on the way up.

17.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 17-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 17-8. Timer/Counter Timing Diagram, no Prescaling

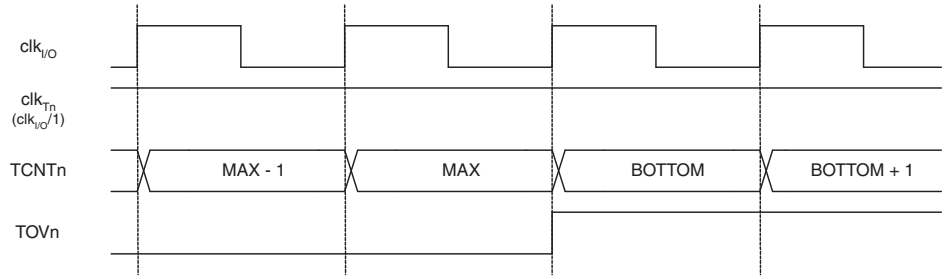


Figure 17-9 shows the same timing data, but with the prescaler enabled.

Figure 17-9. Timer/Counter Timing Diagram with Prescaler ($f_{clk_I/O}/8$)

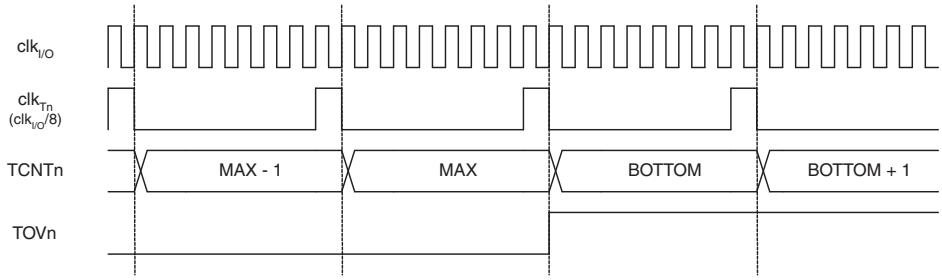


Figure 17-10 shows the setting of OCF0B and OCF0A in all modes except CTC and PWM mode, where OCR0A is TOP.

Figure 17-10. Timer/Counter Timing Diagram, setting of OCF0*x* with Prescaler ($f_{clk_I/O}/8$)

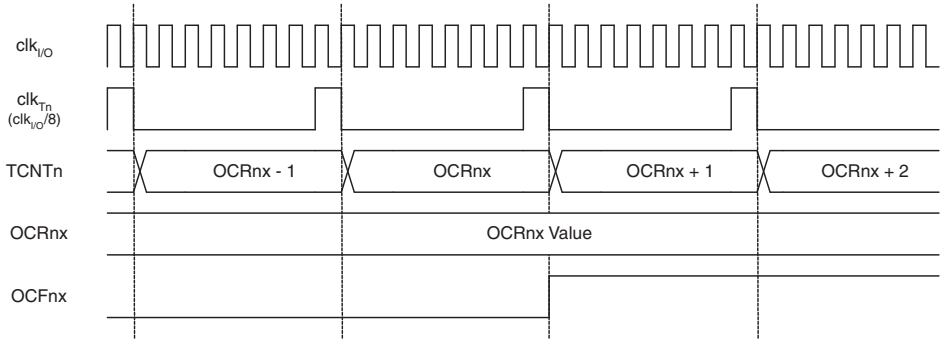
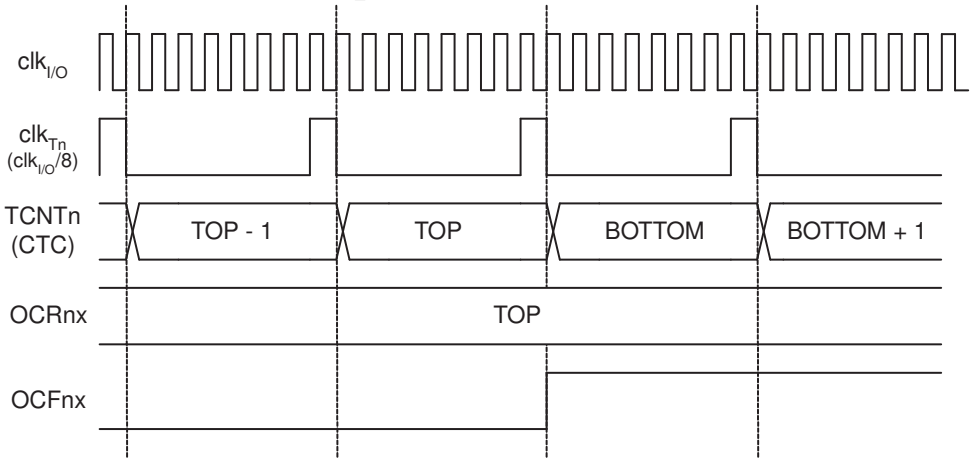


Figure 17-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 17-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode with Prescaler ($f_{clk_I/O}/8$)



17.9 Register Description

17.9.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	TSM	Res4	Res3	Res2	Res1	Res0	PSRASY	PSRSYNC	GTCCR
Read/Write	RW	R	R	R	R	R	R	RW	
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 PSRASY and PSRSYNC bits 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 the configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware and the Timer/Counters simultaneously start counting.

- **Bit 6:2 – Res4:0 - Reserved**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1 – PSRASY - Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

- **Bit 0 – PSRSYNC - Prescaler Reset for Synchronous Timer/Counters**

When this bit is one, the Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 share the same prescaler and a reset of this prescaler will affect all timers.

17.9.2 TCCR0A – Timer/Counter0 Control Register A

Bit	7	6	5	4	3	2	1	0	
$24 ($44)	COM0A1	COM0A0	COM0B1	COM0B0	Res1	Res0	WGM01	WGM00	TCCR0A
Read/Write	RW	RW	RW	RW	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM0A1:0 - Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver. When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. The following shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM). For the functionality in other modes refer to section "Operating Modes".

Table 17-6 COM0A Register Bits

Register Bits	Value	Description
COM0A1:0	0	Normal port operation, OC0A disconnected
	1	Toggle OC0A on Compare Match
	2	Clear OC0A on Compare Match
	3	Set OC0A on Compare Match

- **Bit 5:4 – COM0B1:0 - Compare Match Output B Mode**

These bits control the Output Compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver. When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. The following shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM). For the functionality in other modes refer to section "Operating Modes".

Table 17-7 COM0B Register Bits

Register Bits	Value	Description
COM0B1:0	0	Normal port operation, OC0B disconnected
	1	Toggle OC0B on Compare Match
	2	Clear OC0B on Compare Match
	3	Set OC0B on Compare Match

- **Bit 3:2 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1:0 – WGM01:00 - Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used according to the following table. Modes of operation supported by the Timer/Counter0 unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see section "Modes of Operation" for details).

Table 17-8 WGM0 Register Bits

Register Bits	Value	Description
WGM01:00	0x0	Normal mode of operation
	0x1	PWM, phase correct, TOP=0xFF
	0x2	CTC, TOP = OCRA
	0x3	Fast PWM, TOP=0xFF
	0x4	Reserved
	0x5	PWM, Phase correct, TOP = OCRA
	0x6	Reserved
	0x7	Fast PWM, TOP=OCRA

17.9.3 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	FOC0A	FOC0B	Res1	Res0	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A - Force Output Compare A**

The FOC0A bit is only active when the WGM02:0 bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written in a PWM operation mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare. A FOC0A strobe will not generate any interrupt nor will it clear the timer in CTC mode using OCR0A as TOP. The FOC0A bit is always read as zero.

- **Bit 6 – FOC0B - Force Output Compare B**

The FOC0B bit is only active when the WGM02:0 bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written in a PWM operation mode. When writing a logical one to the FOC0B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0B output is changed according to its COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B1:0 bits that determines the effect of the forced compare. A FOC0B strobe will not generate any interrupt nor will it clear the timer in CTC mode using OCR0B as TOP. The FOC0B bit is always read as zero.

- **Bit 5:4 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – WGM02 -**

Combined with the WGM01:0 bit found in the TCCR0A Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see section "Modes of Operation").

- **Bit 2:0 – CS02:00 - Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter0 according to the following table.If external pin modes are used for Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 17-9 CS0 Register Bits

Register Bits	Value	Description
CS02:00	0x00	No clock source (Timer/Counter0 stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)

Register Bits	Value	Description
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	External clock source on T0 pin, clock on falling edge
	0x07	External clock source on T0 pin, clock on rising edge

17.9.4 TCNT0 – Timer/Counter0 Register

Bit	7	6	5	4	
$26 ($46)	TCNT0_7	TCNT0_6	TCNT0_5	TCNT0_4	TCNT0
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
$26 ($46)	TCNT0_3	TCNT0_2	TCNT0_1	TCNT0_0	TCNT0
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter0 unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

- **Bit 7:0 – TCNT0_7:0 - Timer/Counter0 Byte**

17.9.5 OCR0A – Timer/Counter0 Output Compare Register

Bit	7	6	5	4	
$27 ($47)	OCR0A_7	OCR0A_6	OCR0A_5	OCR0A_4	OCR0A
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
$27 ($47)	OCR0A_3	OCR0A_2	OCR0A_1	OCR0A_0	OCR0A
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

- **Bit 7:0 – OCR0A_7:0 - Output Compare Register**

17.9.6 OCR0B – Timer/Counter0 Output Compare Register B

Bit	7	6	5	4	
$28 ($48)	OCR0B_7	OCR0B_6	OCR0B_5	OCR0B_4	OCR0B
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
$28 ($48)	OCR0B_3	OCR0B_2	OCR0B_1	OCR0B_0	OCR0B
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0B pin.

- **Bit 7:0 – OCR0B_7:0 - Output Compare Register**

17.9.7 TIMSK0 – Timer/Counter0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($6E)	Res4	Res3	Res2	Res1	Res0	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCIE0B - Timer/Counter0 Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0B bit is set in the Timer/Counter0 Interrupt Flag Register TIFR0.

- **Bit 1 – OCIE0A - Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter0 Interrupt Flag Register TIFR0.

- **Bit 0 – TOIE0 - Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs i.e., when the TOV0 bit is set in the Timer/Counter0 Interrupt Flag Register TIFR0.

17.9.8 TIFR0 – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$15 ($35)	Res4	Res3	Res2	Res1	Res0	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCF0B - Timer/Counter0 Output Compare B Match Flag**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0B Output Compare Register. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable) and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

- **Bit 1 – OCF0A - Timer/Counter0 Output Compare A Match Flag**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A Output Compare Register. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter Compare A Match Interrupt Enable) and OCF0A are set, the Timer/Counter Compare Match Interrupt is executed.

- **Bit 0 – TOV0 - Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable) and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed. The setting of this flag is dependent of the WGM02:0 bit setting.

18 16-bit Timer/Counter (Timer/Counter 1, 3, 4, and 5)

18.1 Features

- **True 16-bit Design (i.e., allows 16-bit PWM)**
- **Three independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceller**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Numerous independent interrupt sources**
    - **TOV1, OCF1A, OCF1B, OCF1C, ICF1**
    - **TOV3, OCF3A, OCF3B, OCF3C, ICF3**
    - **TOV4, OCF4A, OCF4B, OCF4C**
    - **TOV5, OCF5A, OCF5B, OCF5C**

18.2 Overview

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation and signal timing measurement.

Most register and bit references in this section are written in general form. A lower case "*n*" replaces the Timer/Counter number, and a lower case "*x*" replaces the Output Compare unit channel. However when using the register or bit defines in a program, the precise form must be used i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

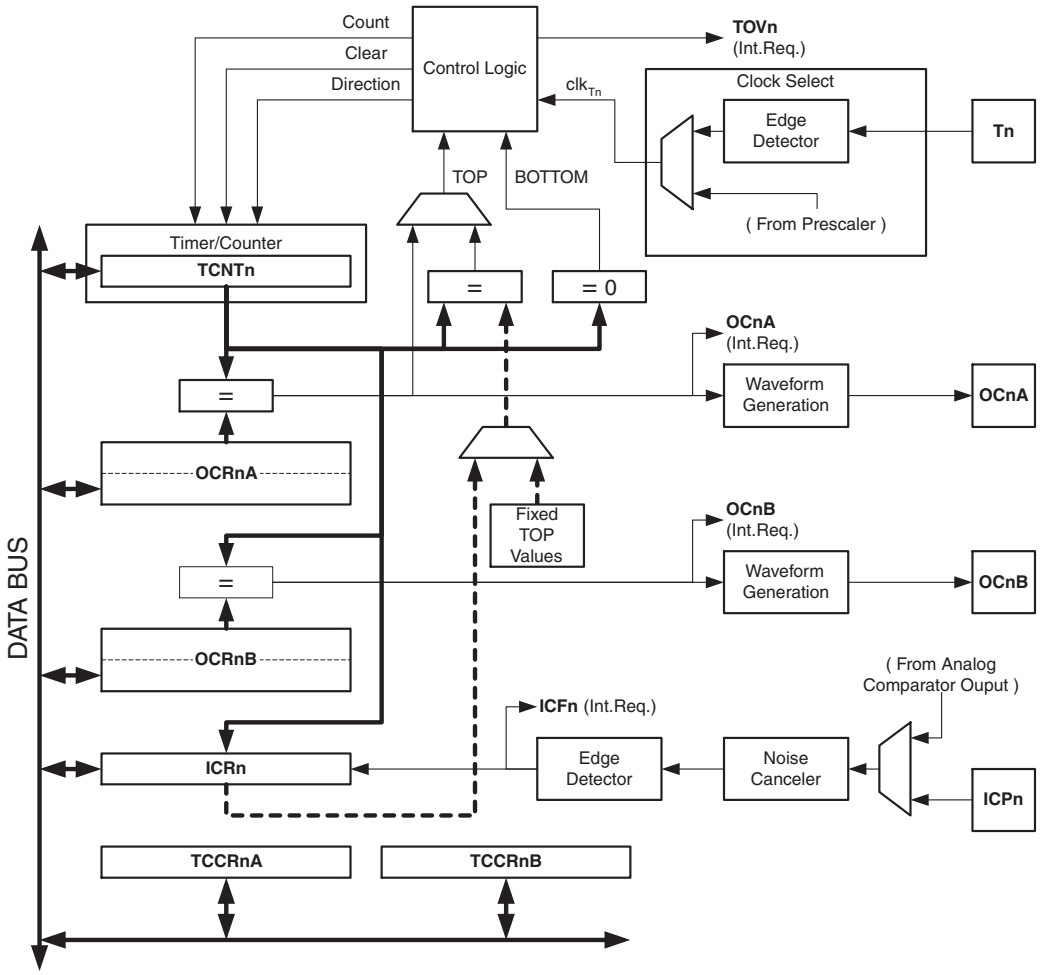
A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 18-1. For the actual placement of I/O pins, see section "Pin Configurations" on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins are shown in bold. The device-specific I/O Register and bit locations are listed in the section "Register Description" on page 266.

The Power Reduction Timer/Counter1 bit, PRTIM1, in "PRR0 – Power Reduction Register0" on page 167 must be written to zero to enable Timer/Counter1 module.

The Power Reduction bits of Timer/Counter3 (PRTIM3), Timer/Counter4 bit (PRTIM4) and Timer/Counter5 (PRTIM5) in "PRR1 – Power Reduction Register 1" on page 168 must be written to zero to enable the respective Timer/Counter module.

Note, note the complete Timer/Counter I/O functionality is provided for each Timer/Counter module depending on the available I/O pins.

Figure 18-1. 16-bit Timer/Counter Block Diagram[1]



Notes: 1. Refer to Figure 1-1 on page 2, Table 14-3 on page 193 and Table 14-9 on page 197 for Timer/Counter1, 2 and 3 pin placements and description.

18.2.1 Registers

The Timer/Counter (TCNT*n*) Output Compare Registers (OCR*n*A/B/C) and Input Capture Register (ICR*n*) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section "Accessing 16-bit Registers" on page 246. The Timer/Counter Control Registers (TCCR*n*A/B/C) are 8-bit registers and have no CPU access restrictions. Interrupt requests (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR*n*). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK*n*). TIFR*n* and TIMSK*n* are not shown in the figure since these registers are shared by other timer units.

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 which clock edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{Tn}).

The double buffered Output Compare Registers (OCR*n*A/B/C) are compared with the

Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC*n*A/B/C). See section "Output Compare Units" on page 252 for details. The compare match event will also set the Compare Match Flag (OCF*n*A/B/C) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP*n*) or on the Analog Comparator pins (see "AC – Analog Comparator" on page 407). The Input Capture unit includes a digital filtering unit (Noise Canceller) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR*n*A Register, the ICR*n* Register or by a set of fixed values. When using OCR*n*A as TOP value in a PWM mode, the OCR*n*A Register can not be used for generating a PWM output. However the TOP value will in this case be double buffered allowing the TOP value to be changed at run time. If a fixed TOP value is required, the ICR*n* Register can be used as an alternative, freeing the OCR*n*A to be used as PWM output.

18.2.2 Definitions

The following definitions are used extensively throughout the document:

Table 18-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, 0x03FF or to the value stored in the OCR*n*A or ICR*n* Register. The assignment is dependent of the mode of operation.

18.3 Accessing 16-bit Registers

The TCNT*n*, OCR*n*A/B/C and ICR*n* are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same Temporary Register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the written low byte and the high byte stored in the Temporary Register 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.

Not all 16-bit accesses use the Temporary Register for the high byte. Reading the OCR*n*A/B/C 16-bit registers does not involve using 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 interrupt updates the temporary register. The same principle can be used directly for accessing the OCR*n*A/B/C and ICR*n* Registers. Note that when using the C-programming language, the compiler handles the 16-bit access.

Assembly Code Examples[1]

```
...
; 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 Examples[1]

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

Notes: 1. See "About Code Examples" on page 7.

The assembly code example returns the TCNT*n* 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 the main code must disable the interrupts during the 16-bit access when both the main code and the interrupt code update the temporary register.

The following code examples show how to do an atomic read of the TCNT*n* Register contents. Reading any of the OCR*n*A/B/C or ICR*n* Registers can be done by using the same principle.

The assembly code example returns the TCNT*n* value in the r17:r16 register pair.

Assembly Code Examples[1]

```
TIM16_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 Examples[1]

```
unsigned int TIM16_ReadTCNTn( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Read TCNTn into i */
  i = TCNTn;
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Notes: 1. See "About Code Examples" on page 7 .

The following code examples show how to do an atomic write of the TCNT*n* Register contents. Writing any of the OCR*n*A/B/C or ICR*n* Registers can be done by using the same principle.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT*n*.

Assembly Code Examples[1]

```
TIM16_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 Examples[1]

```
void TIM16_WriteTCNTn( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Set TCNTn to i */
  TCNTn = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

Notes: 1. See "About Code Examples" on page 7 .

18.3.1 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.

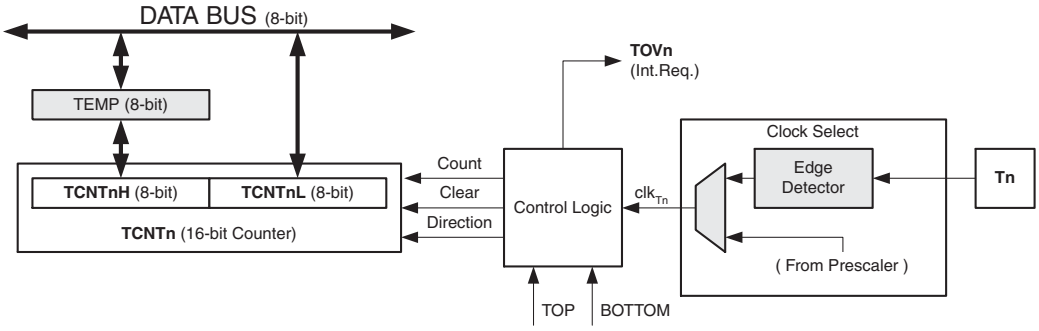
18.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS*n*2:0) bits located in the Timer/Counter control Register B (TCCR*n*B). For details on clock sources and prescaler, see "Timer/Counter 0, 1, 3, 4, and 5 Prescaler" on page 304.

18.5 Counter Unit

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

Figure 18-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT*n* by 1;
Direction	Select between increment and decrement;

Clear	Clear TCNT*n* (set all bits to zero);
clk_{Tn}	Timer/Counter clock;
TOP	Signalize that TCNT*n* has reached maximum value;
BOTTOM	Signalize that TCNT*n* has reached minimum value (zero);

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT*n*H) contains the upper eight bits of the counter and Counter Low (TCNT*n*L) contains the lower eight bits. The TCNT*n*H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT*n*H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT*n*H value when the TCNT*n*L is read and TCNT*n*H is updated with the temporary register value when TCNT*n*L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT*n* Register giving unpredictable results when the counter is running. These special cases are described in the sections of their importance.

Depending on the mode of operation, the counter is cleared, incremented or decremented at each timer clock (clk_{Tn}). The clk_{Tn} can be generated from an external or internal clock source selected by the Clock Select bits (CS*n*2:0). The timer is stopped when no clock source is selected (CS*n*2:0 = 0). However, the TCNT*n* value can be accessed by the CPU independent of whether clk_{Tn} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the settings of the Waveform Generation mode bits (WGM*n*3:0) located in the Timer/Counter Control Registers A and B (TCCR*n*A and TCCR*n*B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC*n*x. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 256.

The Timer/Counter Overflow Flag (TOV*n*) is set according to the mode of operation selected by the WGM*n*3:0 bits. TOV*n* can be used for generating a CPU interrupt.

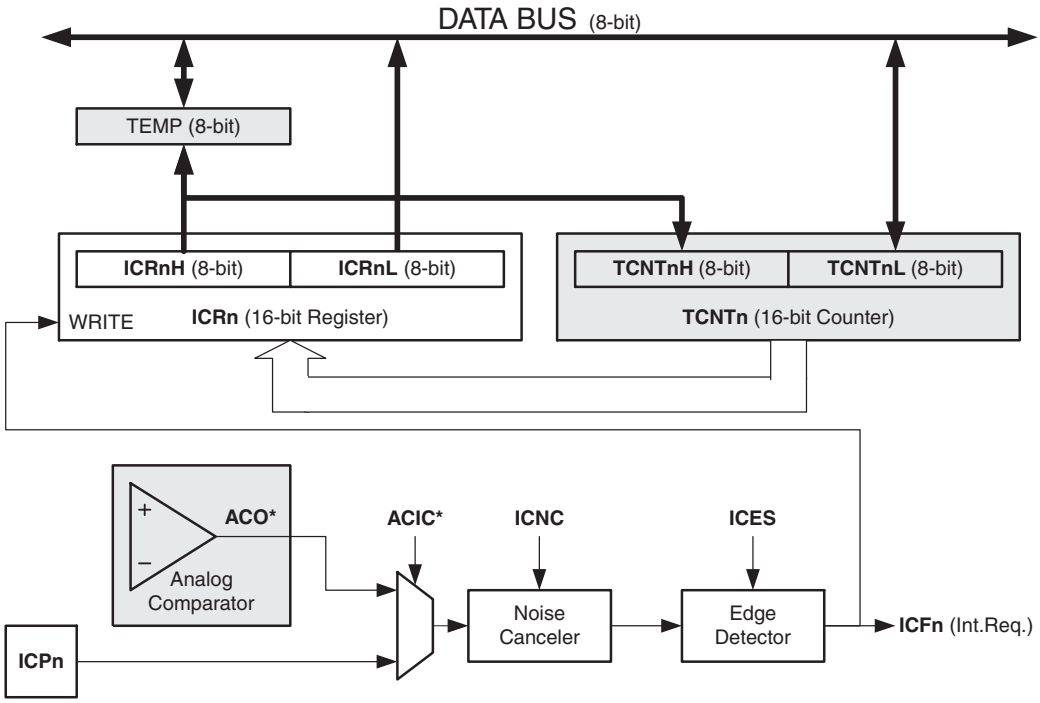
18.6 Input Capture Unit

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 indicating an event, or multiple events, can be applied via the ICP*n* pin or alternatively, for the Timer/Counter1 only, via the Analog Comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 18-3. The elements of the block diagram not direct parts of the input capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

A capture will be triggered when a change of the logic level (an event) occurs on the Input Capture Pin (ICP*n*), or alternatively on the analog Comparator output (ACO), and this change matches the setting of the edge detector. When a capture is triggered, the 16-bit value of the counter (TCNT*n*) is written to the Input Capture Register (ICR*n*). The Input Capture Flag (ICF*n*) is set at the same system clock as the TCNT*n* value is copied into ICR*n* Register. If enabled (TICIE*n* = 1), the input capture flag generates an input capture interrupt. The ICF*n* flag is automatically cleared when the interrupt is executed. Alternatively the ICF*n* flag can be software-cleared by writing a logical one to its I/O bit location.

Figure 18-3. Input Capture Unit Block Diagram



Note: 1. The Analog Comparator Output (ACO) can only trigger the Timer/Counter1 ICP – not Timer/Counter3, 4 or 5.

Reading the 16-bit value in the Input Capture Register (ICR*n*) is done by first reading the low byte (ICR*n*L) and then the high byte (ICR*n*H). When the low byte is read the high byte is copied into the high byte Temporary Register (TEMP). The CPU will access the TEMP Register when reading the ICR*n*H I/O location.

The ICR*n* Register can only be written when using a Waveform Generation mode that utilizes the ICR*n* Register for defining the counter's TOP value. In these cases the Waveform Generation mode (WGMn3:0) bits must be set before the TOP value can be written to the ICR*n* Register. When writing the ICR*n* Register the high byte must be written to the ICR*n*H I/O location before the low byte is written to ICR*n*L.

For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 246.

18.6.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the Input Capture Pin (ICP*n*). Timer/Counter1 can alternatively use the analog comparator output as trigger source for the input capture unit. The Analog Comparator is selected as trigger source by setting the analog Comparator Input Capture (ACIC) bit in the Analog Comparator Control and Status Register (ACSR). Be aware that changing trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

Both the Input Capture Pin (ICP*n*) and the Analog Comparator output (ACO) inputs are sampled using the same technique as for the T*n* pin (Figure 19-1 on page 304). The edge detector is also identical. However, when the noise canceller is enabled, additional logic is inserted before the edge detector increasing the delay by four system

clock cycles. Note that the input of the noise canceller and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR*n* to define TOP.

An input capture can be software-triggered by controlling the port of the ICP*n* pin.

18.6.2 Noise Canceller

The noise canceller improves noise immunity by using a simple digital filtering scheme. The noise canceller 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 canceller is enabled by setting the Input Capture Noise Canceller (ICNC*n*) bit in Timer/Counter Control Register B (TCCR*n*B). When enabled the noise canceller introduces additional four system clock cycles of delay from a change applied to the input to the update of the ICR*n* Register. The noise canceller uses the system clock and is therefore not affected by the prescaler.

18.6.3 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. The ICR*n* will be overwritten with a new value if the processor has not read the captured value in the ICR*n* Register before the next event occurs. In this case the result of the capture will be incorrect.

When using the Input Capture interrupt, the ICR*n* Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

It is not recommended to use the Input Capture unit in any mode of operation where the TOP value (resolution) is actively changed while counting.

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

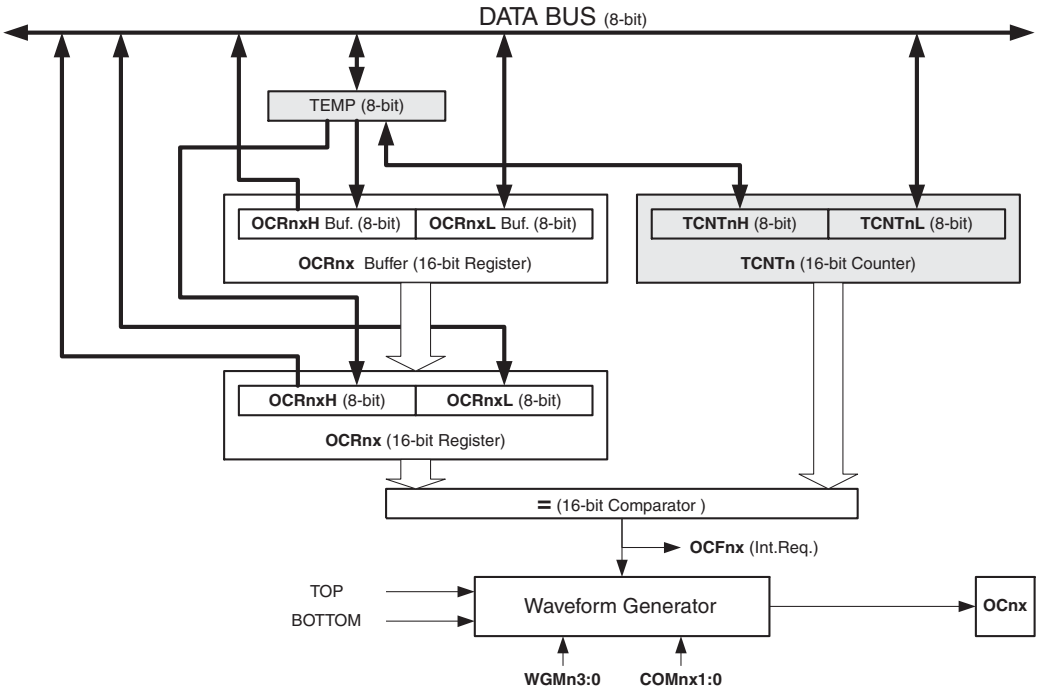
18.7 Output Compare Units

The 16-bit comparator continuously compares TCNT*n* with the Output Compare Register (OCR*nx*). If TCNTn equals OCR*nx* the comparator signals a match. A match will set the Output Compare Flag (OCF*nx*) at the next clock cycle of the timer. If enabled (OCIE*nx* = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF*nx* Flag is automatically cleared when the interrupt is executed. Alternatively the OCF*nx* Flag can be software-cleared by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to the Waveform Generation mode bits (WGM*n*3:0) and Compare Output mode bits (COM*nx*1:0). The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (see "Modes of Operation" on page 256).

A special feature of Output Compare unit A allows it to define the Timer/Counter TOP value i.e., the counter resolution. In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the Waveform Generator.

Figure 18-4 shows a block diagram of the Output Compare unit. The small "*n*" in the register and bit names indicates the device number (*n* = Timer/Counter *n*), and the "*x*" indicates Output Compare unit A, B or C. The elements of the block diagram not direct parts of the Output Compare unit are gray shaded.

Figure 18-4. Output Compare Unit Block Diagram



The OCR*nx* Register is double buffered when using any of the twelve Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR*nx* Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR*nx* Register access may seem complex, but this is not the case. When the double buffering is enabled, the CPU has access to the OCR*nx* Buffer Register. If double buffering is disabled the CPU will access the OCR*nx* directly. The content of the OCR1*x* (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1*x* is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first similar to accessing other 16-bit registers. Writing the OCR*nx* Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR*nx*H) has to be written first. The TEMP Register will be updated with the value written by the CPU to the high byte I/O location. Then when the low byte (OCR*nx*L) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR*nx* buffer or OCR*nx* Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 246.

18.7.1 Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC*nx*) bit. Forcing compare match will not set the OCF*nx* Flag or reload/clear the timer, but the OC*nx* pin will be updated as if a real compare match had occurred (the COM*n*1:0 bits settings define whether the OC*nx* pin is set, cleared or toggled).

18.7.2 Compare Match Blocking by TCNTn Write

All CPU writes to the TCNT*n* Register will block any compare match that occurs in the next clock cycle of the timer even when the timer is stopped. This feature allows OCR*nx* to be initialized to the same value as TCNT*n* without triggering an interrupt when the Timer/Counter clock is enabled.

18.7.3 Using the Output Compare Unit

Since writing TCNT*n* in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT*n* using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT*n* equals the OCR*nx* value, the compare match will be missed resulting in incorrect waveform generation. Do not write the TCNT*n* equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT*n* value equal to BOTTOM when the counter is down-counting.

The setup of the OC*nx* should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC*nx* value is to use the Force Output Compare (FOC*nx*) strobe bits in Normal mode. The OC*nx* Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM*nx*1:0 bits are not double buffered together with the compare value. A change of the COM*nx*1:0 bits will immediately take effect.

18.8 Compare Match Output Unit

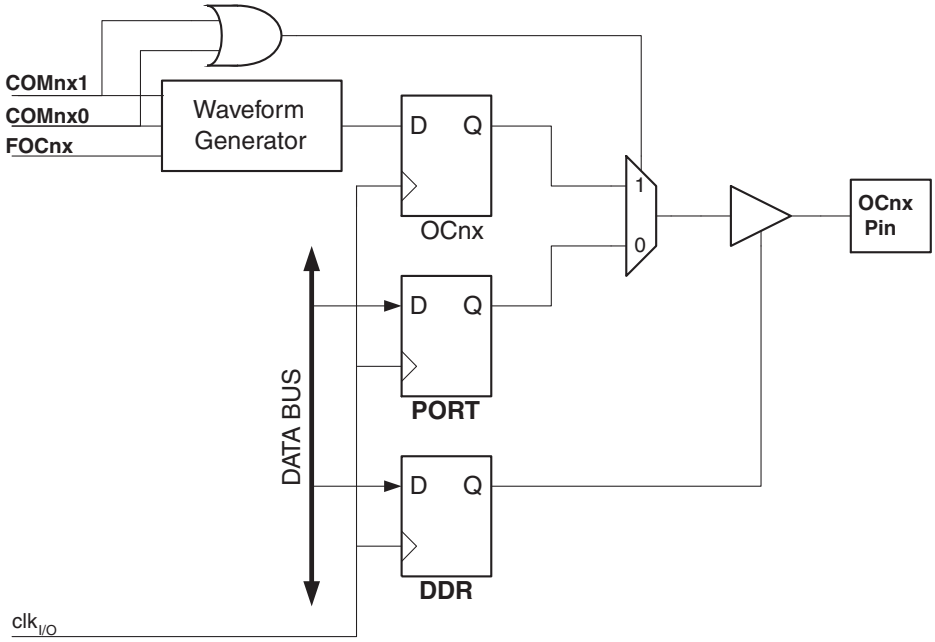
The Compare Output mode (COM*nx*1:0) bits have two functions. The Waveform Generator uses the COM*nx*1:0 bits for defining the Output Compare (OC*nx*) state at the next compare match. Secondly the COM*nx*1:0 bits control the OC*nx* pin output source. Figure 18-5 shows a simplified schematic of the logic affected by the COM*nx*1:0 bit setting. The I/O Registers, I/O bits and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM*nx*1:0 bits are shown. When referring to the OC*nx* state, the reference is to the internal OC*nx* Register and not to the OC*nx* pin. After a system reset the OC*nx* Register will have a value of “0”.

The general I/O port function is overridden by the Output Compare (OC*nx*) from the Waveform Generator if either of the COM*nx*1:0 bits are set. However, the OC*nx* pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC*nx* pin (DDR_OC*nx*) must be set as output before the OC*nx* value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 18-2, Table 18-3 and Table 18-4 on page 256 for details.

The design of the Output Compare pin logic allows initialization of the OC*nx* state before the output is enabled. Note that some COM*nx*1:0 bit settings are reserved for certain modes of operation (see section "Register Description" on page 266).

The COM*nx*1:0 bits have no effect on the Input Capture unit.

Figure 18-5. Compare Match Output Unit, Schematic



18.8.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM*nx*1:0 bits differently in normal, CTC and PWM modes. A setting of COM*nx*1:0 = 0 tells the Waveform Generator in all modes that no action on the OC*nx* Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 18-2. For fast PWM mode refer to Table 18-3 and for phase-correct and phase-and-frequency-correct PWM refer to Table 18-4.

A change of the COM*nx*1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC*nx* strobe bits.

Table 18-2 shows the COM*nx*1:0 bit functionality when the WGM*n*3:0 bits are set to a normal or a CTC mode (non-PWM).

Table 18-2. Compare Output Mode, non-PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OC*n*A/OC*n*B/OC*n*C disconnected.
0	1	Toggle OC*n*A/OC*n*B/OC*n*C on compare match.
1	0	Clear OC*n*A/OC*n*B/OC*n*C on compare match (set output to low level).
1	1	Set OC*n*A/OC*n*B/OC*n*C on compare match (set output to high level).

Table 18-3 shows the COM*nx*1:0 bit functionality when the WGM*n*3:0 bits are set to the fast PWM mode.

Table 18-3. Compare Output Mode, Fast PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OC*n*A/OC*n*B/OC*n*C disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1	0	Clear OC*n*A/OC*n*B/OC*n*C on compare match; set OC*n*A/OC*n*B/OC*n*C at BOTTOM (non-inverting mode).
1	1	Set OC*n*A/OC*n*B/OC*n*C on compare match, clear OC*n*A/OC*n*B/OC*n*C at BOTTOM (inverting mode).

Note: 1. A special case occurs when OCR*n*A/OCR*n*B/OCR*n*C equals TOP and COM*n*A1/COM*n*B1/COM*n*C1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 258 for more details.

Table 18-4 shows the COM*nx*1:0 bit functionality when the WGM*n*3:0 bits are set to the phase correct and phase and frequency correct PWM mode.

Table 18-4. Compare Output Mode, Phase Correct and Phase/Frequency Correct PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OC*n*A/OC*n*B/OC*n*C disconnected.
0	1	WGM13:0 =9 or 11: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1	0	Clear OC*n*A/OC*n*B/OC*n*C on compare match when up-counting. Set OC*n*A/OC*n*B/OC*n*C on compare match when down-counting.
1	1	Set OC*n*A/OC*n*B/OC*n*C on compare match when up-counting. Clear OC*n*A/OC*n*B/OC*n*C on compare match when down-counting.

Note: 1. A special case occurs when OCR*n*A/OCR*n*B/OCR*n*C equals TOP and COM*n*A1/COM*n*B1/COM*n*C1 is set. See "Phase and Frequency Correct PWM Mode" on page 262 for more details.

18.9 Modes of Operation

The mode of operation i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM*n*3:0) and the Compare Output mode (COM*nx*1:0) bits. The Compare Output mode bits do not affect the counting sequence, but the Waveform Generation mode bits do. The COM*nx*1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM*nx*1:0 bits control if the output should be set, cleared or toggle at a compare match (See "Compare Match Output Unit" on page 254)

Table 18-5. Waveform Generation Mode Bit Description [1]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0) (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x3FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR*n*A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR*n*	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR*n*A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR*n*	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR*n*A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR*n*	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR*n*	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR*n*A	BOTTOM	TOP

Notes: 1. The CTC*n* and PWM*n*1:0 bit definition names are obsolete. Use the WGM*n*2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 264.

18.9.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM*n*3:0 = 0). In this mode the counting direction is always up (incrementing) and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the Timer/Counter Overflow Flag (TOV*n*) will be set in the same timer clock cycle as the TCNT*n* becomes zero. In this case the TOV*n* Flag behaves like a 17th bit, except that it is only set and not cleared. However the timer resolution can be increased by software when combined with the timer overflow interrupt that automatically clears the TOV*n* Flag. There are no special cases to consider in the Normal mode. A new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However it is important to note that the maximum interval between the external events must not exceed the resolution of the counter. The timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit if the intervals between events are too long.

The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended because this will occupy too much CPU time.

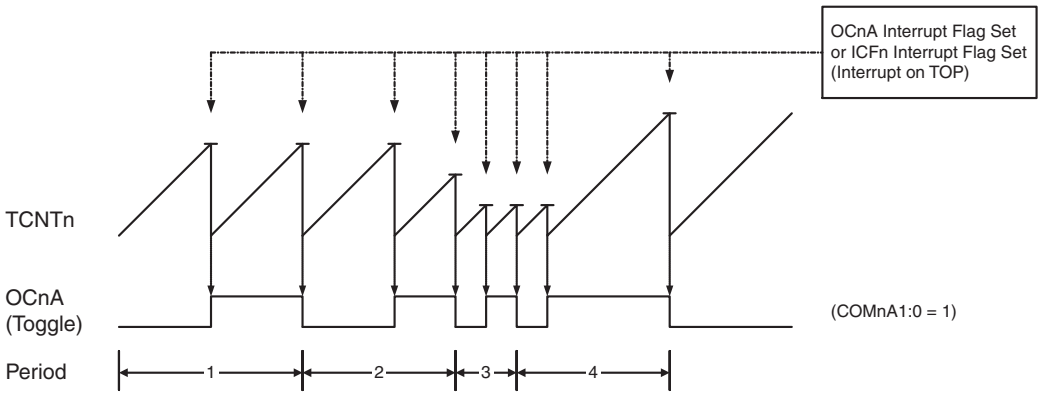
18.9.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare (CTC) mode (WGM*n*3:0 = 4 or 12), the OCR*n*A or ICR*n* Register are used to manipulate the counter resolution. In CTC mode the counter is

cleared to zero when the counter value (TCNT*n*) matches either the OCR*n*A (WGM*n*3:0 = 4) or the ICR*n* (WGM*n*3:0 = 12). The OCR*n*A or ICR*n* define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in the following figure. The counter value (TCNT*n*) increases until a compare match occurs with either OCR*n*A or ICR*n*, and then counter (TCNT*n*) is cleared.

Figure 18-6. CTC Mode Timing Diagram



Each time the counter reaches the TOP value an interrupt can be generated by either the OCF*n*A or ICF*n* Flag according to the register used to define the TOP value. 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 no or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. The counter will miss the compare match if the new value written to OCR*n*A or ICR*n* is lower than the current value of TCNT*n*. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. The fast PWM mode is available as an alternative using OCR*n*A for defining TOP (WGMn3:0 = 15). The OCR*n*A then will be double buffered.

For generating a waveform output in CTC mode, the OC*n*A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM*n*A1:0 = 1). The OC*n*A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC*n*A = 1). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk_I/O}/2$ when OCR*n*A is set to zero (0x0000). The waveform frequency is given by the following equation:

$$f_{OCnA} = \frac{f_{CLK_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

The N variable represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV*n* Flag is set in the same clock cycle of the timer when the counter changes from MAX to 0x0000.

18.9.3 Fast PWM Mode

The fast Pulse Width Modulation (PWM) mode (WGM*n*3:0 = 5, 6, 7, 14 or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC*nx*) is cleared on the compare match between TCNT*n* and OCR*nx*, and

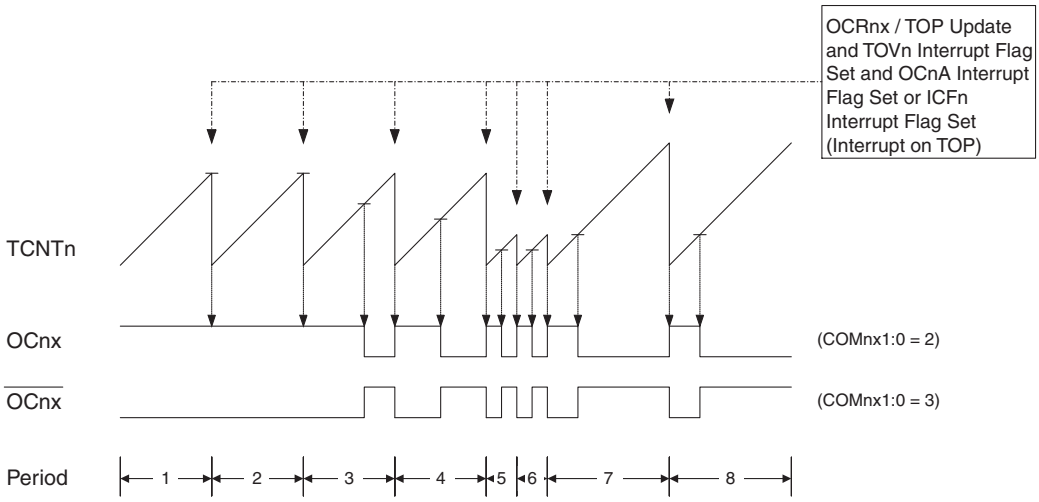
OC*nx* is set at BOTTOM. In inverting Compare Output mode output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase-correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reducing total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR*n* or OCR*n*A. The minimum resolution allowed is 2-bit (ICR*n* or OCR*n*A set to 0x0003), and the maximum resolution is 16-bit (ICR*n* or OCR*n*A set to MAX). The PWM resolution R_{FPWM} in bits can be calculated with the following equation:

$$R_{FPWM} = \frac{\log(TOP+1)}{\log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF or 0x03FF (WGM*n*3:0 = 5, 6 or 7), the value in ICR*n* (WGM*n*3:0 = 14) or the value in OCR*n*A (WGM*n*3:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 18-7. The figure shows fast PWM mode when OCR*n*A or ICR*n* is used to define TOP. The TCNT*n* value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT*n* slopes represent compare matches between OCR*nx* and TCNT*n*. The OC*nx* Interrupt Flag will be set when a compare match occurs.

Figure 18-7. Fast PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV*n*) is set each time the counter reaches TOP. In addition the OC*n*A or ICF*n* Flag is set at the same timer clock cycle as TOV*n* is set when either OCR*n*A or ICR*n* is used to define the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be utilized for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. A compare match will never occur between the TCNT*n* and the OCR*nx* if the TOP value is lower than any of the Compare Registers. Note that when working with fixed TOP values the unused bits are masked to zero when any of the OCR*nx* Registers are written.

The procedure for updating ICR*n* differs from updating OCR*n*A when used for defining the TOP value. The ICR*n* Register is not double buffered. This means that if ICR*n* is changed to a low value while the counter is running with no or a low prescaler value, there is a risk that the newly written ICR*n* value is lower than the current value of TCNT*n*. In consequence the counter will miss the compare match at the TOP value. The counter must then count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR*n*A Register is double buffered though. This feature allows writing the OCR*n*A I/O location at anytime. When the OCR*n*A I/O location is written the new value will be put first into the OCR*n*A Buffer Register. The OCR*n*A Compare Register will then be updated with the value in the Buffer Register at the next clock cycle of the timer when TCNT*n* matches TOP. The update is done at the same timer clock cycle as the TCNT*n* is cleared and the TOV*n* Flag is set.

The definition of TOP with the ICR*n* Register works well for fixed TOP values. Combined with ICR*n*, the OCR*n*A Register is free to be used for generating a PWM output on OC*n*A. However, if the base PWM frequency is actively changed (by modifying the TOP value), working with the OCR*n*A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode the compare units allow the generation of PWM waveforms on the OC*nx* pins. Setting the COM*nx*1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM*nx*1:0 to 3 (see Table 18-3 on page 256). The actual OC*nx* value will only be visible on the port pin if the data direction of the port pin is set to output (DDR_OC*nx*). The PWM waveform is generated by setting (or clearing) the OC*nx* Register at the compare match between OCR*nx* and TCNT*n*, and by clearing (or setting) the OC*nx* Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency of the output $f_{OCnxPWM}$ can be calculated with the following equation:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256 or 1024).

The extreme values for the OCR*nx* Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR*nx* is set equal to BOTTOM (0x0000), the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR*nx* equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM*nx*1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC*n*A to toggle its logical level on each compare match (COM*n*A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk_I/O}/2$ when OCR*n*A is set to zero (0x0000). This feature is similar to the OC*n*A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

18.9.4 Phase Correct PWM Mode

The phase correct Pulse Width Modulation (PWM) mode (WGM*n*3:0 = 1, 2, 3, 10 or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC*nx*) is cleared on the compare match between TCNT*n* and OCR*nx* while

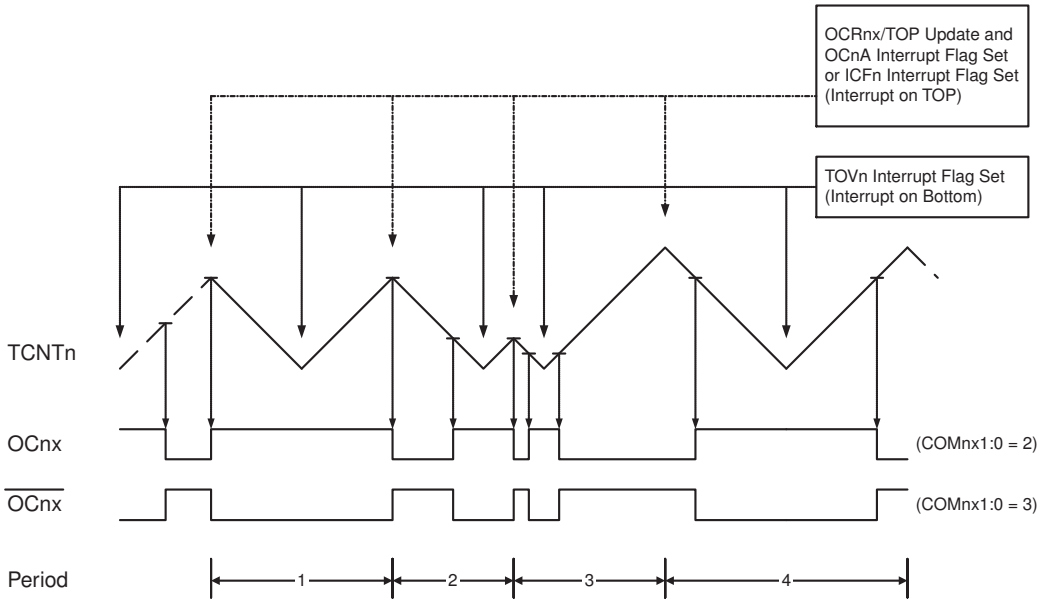
up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has a lower maximum operation frequency than single slope operation. However these modes are preferred for motor control applications due to the symmetric feature of the dual-slope PWM modes.

The PWM resolution for the phase correct PWM mode can be fixed to 8, 9 or 10 bit, or be defined by either ICR*n* or OCR*n*A. The minimum resolution allowed is 2 bit (ICR*n* or OCRnA set to 0x0003), and the maximum resolution is 16-bit (ICR*n* or OCR*n*A set to MAX). The PWM resolution R_{PCPWM} in bits can be calculated with the following equation:

$$R_{PCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF or 0x03FF (WGM*n*3:0 = 1, 2, or 3), the value in ICR*n* (WGM*n*3:0 = 10) or the value in OCR*n*A (WGM*n*3:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT*n* value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 18-8 below. The figure shows phase correct PWM mode when OCR*n*A or ICR*n* is used to define TOP. The TCNT*n* value is shown in the timing diagram as a histogram illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT*n* slopes represent compare matches between OCR*nx* and TCNT*n*. The OC*nx* Interrupt Flag will be set when a compare match occurs.

Figure 18-8. Phase Correct PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV*n*) is set each time the counter reaches BOTTOM. When either OCR*n*A or ICR*n* is used for defining the TOP value, the OC*n*A or ICF*n* Flag is set accordingly at the same timer clock cycle as the OCR*nx* Registers are updated with the double buffer value (at TOP). The Interrupt Flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the

TCNT*n* and the OCR*nx*. Note that when working with fixed TOP values, the unused bits are masked to zero when any of the OCR*nx* Registers are written. As the third period shown in Figure 18-8 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an asymmetrical output. The reason for this can be found in the update time of the OCR*nx* Register. Since the OCR*nx* update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values are not equal the two slopes of the period will differ in length. The difference in length gives the asymmetrical result of the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generating PWM waveforms on the OC*nx* pins. Setting the COM*nx*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM*nx*1:0 to 3 (see Table 18-4 on page 256). The actual OC*nx* value will only be visible on the port pin if the data direction of the port pin is set to output (DDR_OC*nx*). The PWM waveform is generated by setting (or clearing) the OC*nx* Register at the compare match between OCR*nx* and TCNT*n* when the counter increments, and by clearing (or setting) the OC*nx* Register at compare match between OCR*nx* and TCNT*n* when the counter decrements. The PWM frequency of the output $f_{OCnxPCPWM}$ when using phase-correct PWM can be calculated with the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR*nx* Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR*nx* is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

18.9.5 Phase and Frequency Correct PWM Mode

The phase and frequency correct Pulse Width Modulation (PWM) mode (WGM*n*3:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC*nx*) is cleared on the compare match between TCNT*n* and OCR*nx* while up-counting, and set on the compare match while down-counting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However these modes are preferred for motor control applications due to the symmetric feature of the dual-slope PWM modes.

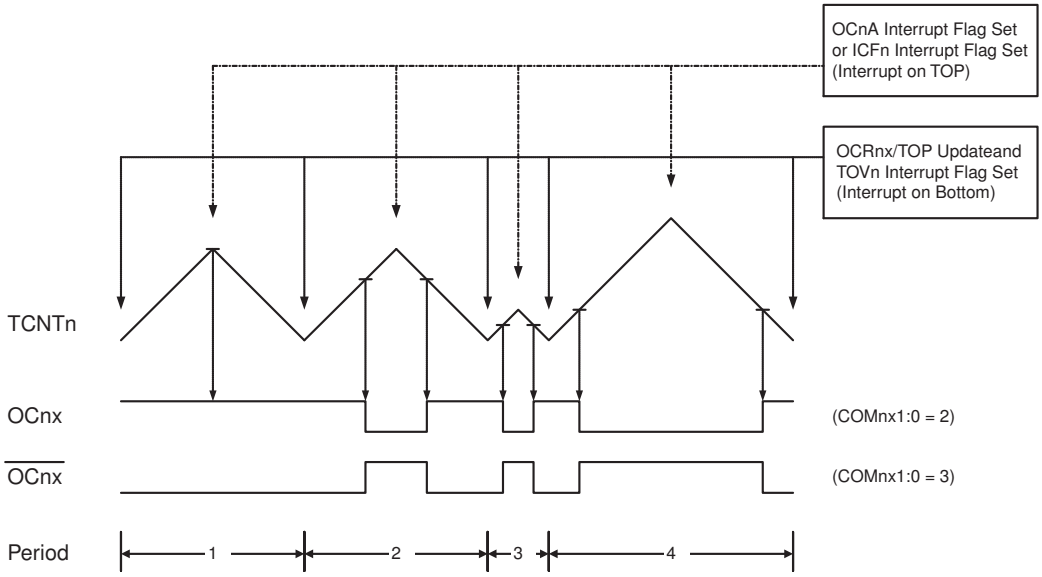
The main difference between the phase correct and the phase and frequency correct PWM mode is the time the OCR*nx* Register is updated by the OCR*nx* Buffer Register, (see Figure 18-8 on page 261 and Figure 18-9 on page 263).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR*n* or OCR*n*A. The minimum resolution allowed is 2 bit (ICR*n* or OCR*n*A set to 0x0003), and the maximum resolution is 16 bit (ICR*n* or OCR*n*A set to MAX). The PWM resolution R_{PFCPWM} in bits can be calculated with the following equation:

$$R_{PFCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR*n* (WGM*n*3:0 = 8), or the value in OCR*n*A (WGM*n*3:0 = 9). The counter has then reached TOP and changes the count direction. The TCNT*n* value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown in Figure 18-9 below. The figure shows phase and frequency correct PWM mode when OCR*n*A or ICR*n* is used to define TOP. The TCNT*n* value is shown in the timing diagram as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT*n* slopes represent compare matches between OCR*nx* and TCNT*n*. The OC*nx* Interrupt Flag will be set when a compare match occurs.

Figure 18-9. Phase and Frequency Correct PWM Mode Timing Diagram



The Timer/Counter Overflow Flag (TOV*n*) is set at the timer clock cycle when the OCR*nx* Registers are updated with the double-buffered value (at BOTTOM). The OC*n*A or ICF*n* Flag is set after TCNT*n* has reached TOP when either OCR*n*A or ICR*n* is used for defining the TOP value. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT*n* and the OCR*nx*.

As Figure 18-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR*nx* Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

The definition of TOP with the ICR*n* Register works well when using fixed TOP values. Combined with ICR*n* the OCR*n*A Register is available for generating a PWM output on OC*n*A. However, if the base PWM frequency is actively changed by modifying the TOP value, using the OCR*n*A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generating PWM waveforms on the OC*nx* pins. Setting the COM*nx*1:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM*nx*1:0 to 3 (see Table 18-4 on page 256). The actual OC*nx* value will only be visible at the port pin if the data direction of the port pin is set to output (DDR_OC*nx*). The PWM waveform is generated by setting (or clearing) the OC*nx* Register at the compare match between OCR*nx* and TCNT*n* when the counter increments, and by clearing (or setting) the OC*nx* Register at compare match between OCR*nx* and TCNT*n* when the counter decrements. The PWM frequency of the output $f_{OCnxPFCPWM}$ when using phase and frequency correct PWM can be calculated with the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR*nx* Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR*nx* is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

18.10 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 and when the OCR*nx* Register is updated with the OCR*nx* buffer value (only for modes utilizing double buffering). Figure 18-10 shows a timing diagram for the setting of OCF*nx*.

Figure 18-10. Timer/Counter Timing Diagram, Setting of OCF*nx*, no Prescaling

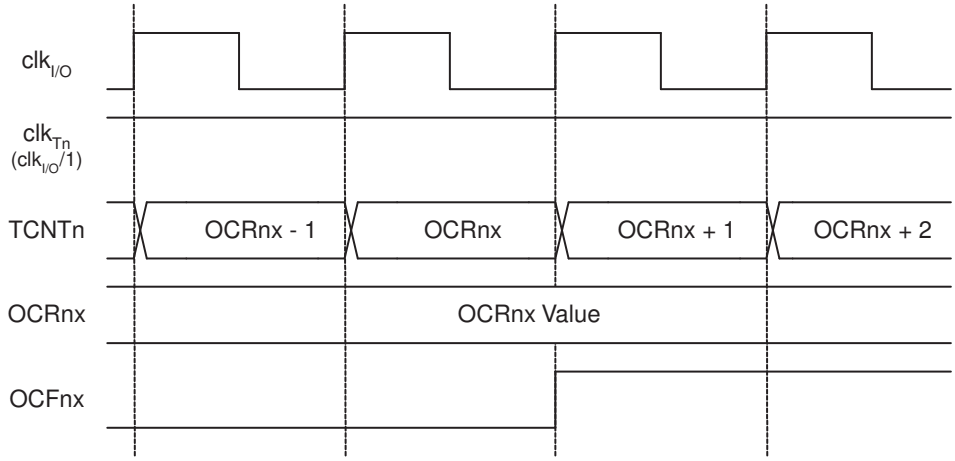


Figure 18-11 shows the same timing data, but with the prescaler enabled.

Figure 18-11. Timer/Counter Timing Diagram, Setting of OCF*nx* with Prescaler ($f_{clk_I/O}/8$)

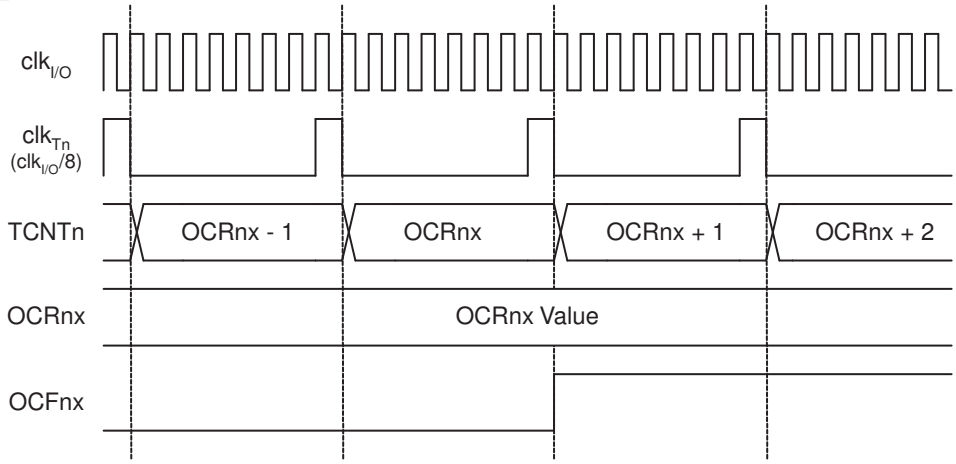


Figure 18-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR*nx* Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV*n* Flag at BOTTOM.

Figure 18-12. Timer/Counter Timing Diagram, no Prescaling

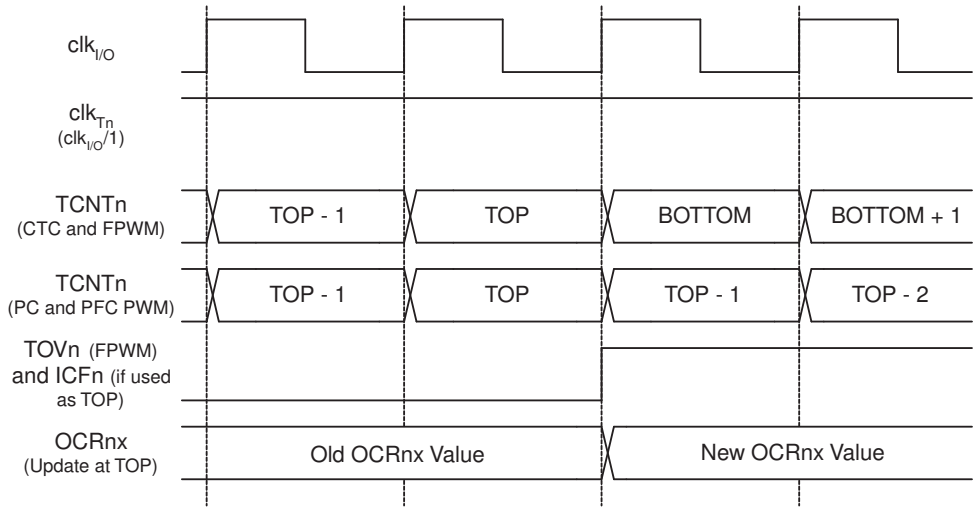
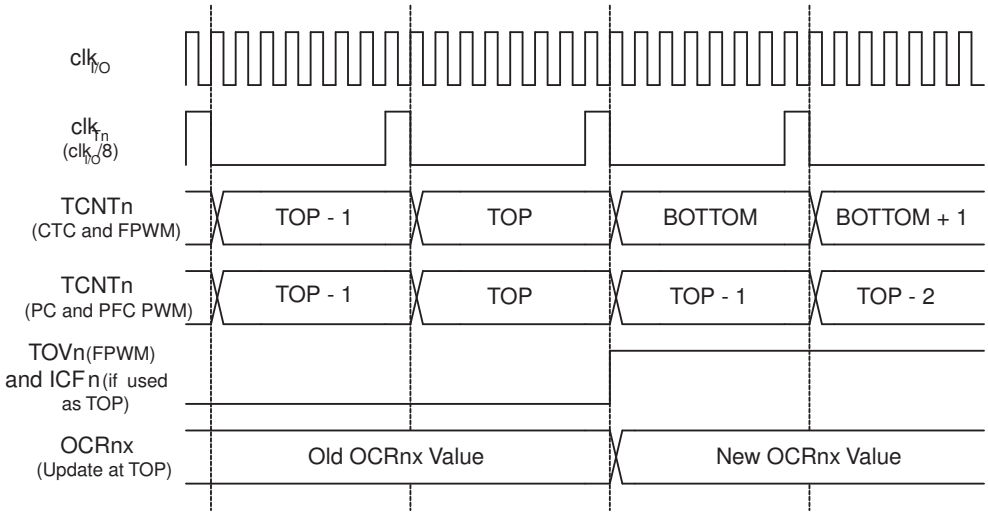


Figure 18-13 shows the same timing data, but with the prescaler enabled.

Figure 18-13. Timer/Counter Timing Diagram with Prescaler ($f_{clk_I/O}/8$)



18.11 Register Description

18.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($80)	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM1A1:0 - Compare Output Mode for Channel A**

The COM1A1:0 bits control the output compare behavior of pin OC1A. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1A pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1A1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1A1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-6 COM1A Register Bits

Register Bits	Value	Description
COM1A1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 5:4 – COM1B1:0 - Compare Output Mode for Channel B**

The COM1B1:0 bits control the output compare behavior of pin OC1B. If one or both of the COM1B1:0 bits are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1B pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1B1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1B1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-7 COM1B Register Bits

Register Bits	Value	Description
COM1B1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 3:2 – COM1C1:0 - Compare Output Mode for Channel C**

The COM1C1:0 bits control the output compare behavior of pin OC1C. If one or both of the COM1C1:0 bits are written to one, the OC1C output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC1C pin must be set in order to enable the output driver. When the OC1A is connected to the pin, the function of the COM1C1:0 bits is dependent of the WGM13:0 bits setting. The following table shows the COM1C1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-8 COM1C Register Bits

Register Bits	Value	Description
COM1C1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 1:0 – WGM11:10 - Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation".

Table 18-9 WGM1 Register Bits

Register Bits	Value	Description
WGM11:10	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit

Register Bits	Value	Description
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($81)	ICNC1	ICES1	Res	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1 - Input Capture 1 Noise Canceller**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The input capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES1 - Input Capture 1 Edge Select**

This bit selects which edge on the Input Capture Pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger. When the ICES1 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture Register (ICR1). The event will also set the Input Capture Flag (ICF1). This can be used to cause an Input Capture Interrupt, if this interrupt is enabled. When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM11:10 - Waveform Generation Mode**

Combined with the WGM11:0 bits found in the TCCR1A Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation".

Table 18-10 WGM1 Register Bits

Register Bits	Value	Description
WGM11:10	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS12:10 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter1 according to the following table. If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 18-11 CS1 Register Bits

Register Bits	Value	Description
CS12:10	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	External clock source on Tn pin, clock on falling edge
	0x07	External clock source on Tn pin, clock on rising edge

18.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($82)	FOC1A	FOC1B	FOC1C	Res4	Res3	Res2	Res1	Res0	TCCR1C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A - Force Output Compare for Channel A**

The FOC1A bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1A bit, an immediate compare match is forced on the waveform generation unit. The OC1A output is changed according to its COM1A1:0 bits setting. Note that the FOC1A bits are implemented as strobes. Therefore it is the value present in the COM1A1:0 bits that determine the effect of the forced compare. A FOC1A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1A as TOP. The FOC1A bits are always read as zero.

- **Bit 6 – FOC1B - Force Output Compare for Channel B**

The FOC1B bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1B output is changed according to its COM1B1:0 bits setting. Note that the FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1B1:0 bits that determine the effect of the forced compare. A FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1B as TOP. The FOC1B bits are always read as zero.

- **Bit 5 – FOC1C - Force Output Compare for Channel C**

The FOC1C bit is only active when the WGM13:0 bits specify a non-PWM mode. When writing a logical one to the FOC1C bit, an immediate compare match is forced on the waveform generation unit. The OC1C output is changed according to its COM1C1:0 bits setting. Note that the FOC1C bits are implemented as strobes. Therefore it is the value present in the COM1C1:0 bits that determine the effect of the forced compare. A FOC1C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1C as TOP. The FOC1C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.4 TCNT1H – Timer/Counter1 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($85)	TCNT1H7:0								TCNT1H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

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

16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers. Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT1H7:0 - Timer/Counter1 High Byte**

18.11.5 TCNT1L – Timer/Counter1 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($84)	TCNT1L7:0								TCNT1L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers. Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT1L7:0 - Timer/Counter1 Low Byte**

18.11.6 OCR1AH – Timer/Counter1 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($89)	OCR1AH7:0								OCR1AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1AH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.7 OCR1AL – Timer/Counter1 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($88)	OCR1AL7:0								OCR1AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1AL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.8 OCR1BH – Timer/Counter1 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8B)	OCR1BH7:0								OCR1BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1BH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.9 OCR1BL – Timer/Counter1 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8A)	OCR1BL7:0								OCR1BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1BL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.10 OCR1CH – Timer/Counter1 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8D)	OCR1CH7:0								OCR1CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1CH7:0 - Timer/Counter1 Output Compare Register High Byte**

18.11.11 OCR1CL – Timer/Counter1 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($8C)	OCR1CL7:0								OCR1CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR1CL7:0 - Timer/Counter1 Output Compare Register Low Byte**

18.11.12 ICR1H – Timer/Counter1 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($87)	ICR1H7:0								ICR1H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin or on the Analog Comparator output. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR1H7:0 - Timer/Counter1 Input Capture Register High Byte**

18.11.13 ICR1L – Timer/Counter1 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($86)	ICR1L7:0								ICR1L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin or on the Analog Comparator output. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR1L7:0 - Timer/Counter1 Input Capture Register Low Byte**

18.11.14 TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($6F)	Res1	Res0	ICIE1	Res	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE1 - Timer/Counter1 Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF1 Flag, located in TIFR1, is set.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE1C - Timer/Counter1 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1C Flag, located in TIFR1, is set.

- **Bit 2 – OCIE1B - Timer/Counter1 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF1B Flag, located in TIFR1, is set.

- **Bit 1 – OCIE1A - Timer/Counter1 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled.

The corresponding Interrupt Vector is executed when the OCF1A Flag, located in TIFR1, is set.

- **Bit 0 – TOIE1 - Timer/Counter1 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV1 Flag, located in TIFR1, is set.

18.11.15 TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	Res1	Res0	ICF1	Res	OCF1C	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

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

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value. ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF1C - Timer/Counter1 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register C (OCR1C). Note that a Forced Output Compare (FOC1C) strobe will not set the OCF1C Flag. OCF1C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF1C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF1B - Timer/Counter1 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B). Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B Flag. OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF1A - Timer/Counter1 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A). Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag. OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

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

The setting of this flag is dependent of the WGM13:0 bits setting of the Timer/Counter1 Control Register. In Normal and CTC modes, the TOV1 Flag is set when the timer

overflows. TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

18.11.16 TCCR3A – Timer/Counter3 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($90)	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM3A1:0 - Compare Output Mode for Channel A**

The COM3A1:0 bits control the output compare behavior of pin OC3A. If one or both of the COM3A1:0 bits are written to one, the OC3A output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3A pin must be set in order to enable the output driver. When the OC3A is connected to the pin, the function of the COM3A1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3A1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-12 COM3A Register Bits

Register Bits	Value	Description
COM3A1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 5:4 – COM3B1:0 - Compare Output Mode for Channel B**

The COM3B1:0 bits control the output compare behavior of pin OC3B. If one or both of the COM3B1:0 bits are written to one, the OC3B output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3B pin must be set in order to enable the output driver. When the OC3B is connected to the pin, the function of the COM3B1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3B1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-13 COM3B Register Bits

Register Bits	Value	Description
COM3B1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match

Register Bits	Value	Description
		(set output to high level).

- **Bit 3:2 – COM3C1:0 - Compare Output Mode for Channel C**

The COM3C1:0 bits control the output compare behavior of pin OC3C. If one or both of the COM3C1:0 bits are written to one, the OC3C output overrides the normal port functionality of the I/O pin it is connected to. However note that the Data Direction Register (DDR) bit corresponding to the OC3C pin must be set in order to enable the output driver. When the OC3C is connected to the pin, the function of the COM3C1:0 bits is dependent of the WGM33:0 bits setting. The following table shows the COM3C1:0 bit functionality when the WGM33:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-14 COM3C Register Bits

Register Bits	Value	Description
COM3C1:0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
	1	Toggle OCnA/OCnB/OCnC on Compare Match.
	2	Clear OCnA/OCnB/OCnC on Compare Match (set output to low level).
	3	Set OCnA/OCnB/OCnC on Compare Match (set output to high level).

- **Bit 1:0 – WGM31:30 - Waveform Generation Mode**

Combined with the WGM33:2 bits found in the TCCR3B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation".

Table 18-15 WGM3 Register Bits

Register Bits	Value	Description
WGM31:30	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved

Register Bits	Value	Description
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.17 TCCR3B – Timer/Counter3 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($91)	ICNC3	ICES3	Res	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC3 - Input Capture 3 Noise Canceller**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICP3) is filtered. The filter function requires four successive equal valued samples of the ICP3 pin for changing its output. The input capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES3 - Input Capture 3 Edge Select**

This bit selects which edge on the Input Capture Pin (ICP3) that is used to trigger a capture event. When the ICES3 bit is written to zero, a falling (negative) edge is used as trigger. When the ICES3 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES3 setting, the counter value is copied into the Input Capture Register (ICR3). The event will also set the Input Capture Flag (ICF3). This can be used to cause an Input Capture Interrupt, if this interrupt is enabled. When the ICR3 is used as TOP value (see description of the WGM33:0 bits located in the TCCR3A and the TCCR3B Register), the ICP3 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM31:30 - Waveform Generation Mode**

Combined with the WGM31:0 bits found in the TCCR3A Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation".

Table 18-16 WGM3 Register Bits

Register Bits	Value	Description
WGM31:30	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit

Register Bits	Value	Description
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS32:30 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter3 according to the following table. If external pin modes are used for the Timer/Counter3, transitions on the T3 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 18-17 CS3 Register Bits

Register Bits	Value	Description
CS32:30	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	External clock source on Tn pin, clock on falling edge
	0x07	External clock source on Tn pin, clock on rising edge

18.11.18 TCCR3C – Timer/Counter3 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($92)	FOC3A	FOC3B	FOC3C	Res4	Res3	Res2	Res1	Res0	TCCR3C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC3A - Force Output Compare for Channel A**

The FOC3A bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3A bit, an immediate compare match is forced on the waveform generation unit. The OC3A output is changed according to its COM3A1:0 bits setting. Note that the FOC3A bits are implemented as strobes. Therefore it is the value present in the COM3A1:0 bits that determine the effect of the forced compare. A FOC3A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR3A as TOP. The FOC3A bits are always read as zero.

- **Bit 6 – FOC3B - Force Output Compare for Channel B**

The FOC3B bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3B bit, an immediate compare match is forced on the waveform generation unit. The OC3B output is changed according to its COM3B1:0 bits setting. Note that the FOC3B bits are implemented as strobes. Therefore it is the value present in the COM3B1:0 bits that determine the effect of the forced compare. A FOC3B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR1B as TOP. The FOC3B bits are always read as zero.

- **Bit 5 – FOC3C - Force Output Compare for Channel C**

The FOC3C bit is only active when the WGM33:0 bits specify a non-PWM mode. When writing a logical one to the FOC3C bit, an immediate compare match is forced on the waveform generation unit. The OC3C output is changed according to its COM3C1:0 bits setting. Note that the FOC3C bits are implemented as strobes. Therefore it is the value present in the COM3C1:0 bits that determine the effect of the forced compare. A FOC3C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR3C as TOP. The FOC3C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.19 TCNT3H – Timer/Counter3 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($95)	TCNT3H7:0								TCNT3H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT3H and TCNT3L, combined TCNT3) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCR3x Registers. Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT3H7:0 - Timer/Counter3 High Byte**

18.11.20 TCNT3L – Timer/Counter3 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($94)	TCNT3L7:0								TCNT3L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT3H and TCNT3L, combined TCNT3) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT3) while the counter is running introduces a risk of missing a compare match between TCNT3 and one of the OCR3x Registers. Writing to the TCNT3 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT3L7:0 - Timer/Counter3 Low Byte**

18.11.21 OCR3AH – Timer/Counter3 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($99)	OCR3AH7:0								OCR3AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3AH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.22 OCR3AL – Timer/Counter3 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($98)	OCR3AL7:0								OCR3AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3A pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3AL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.23 OCR3BH – Timer/Counter3 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9B)	OCR3BH7:0								OCR3BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3BH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.24 OCR3BL – Timer/Counter3 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9A)	OCR3BL7:0								OCR3BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3B pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3BL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.25 OCR3CH – Timer/Counter3 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9D)	OCR3CH7:0								OCR3CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3CH7:0 - Timer/Counter3 Output Compare Register High Byte**

18.11.26 OCR3CL – Timer/Counter3 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($9C)	OCR3CL7:0								OCR3CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT3). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC3C pin. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR3CL7:0 - Timer/Counter3 Output Compare Register Low Byte**

18.11.27 ICR3H – Timer/Counter3 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($97)	ICR3H7:0								ICR3H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT3) value each time an event occurs on the ICP3 pin. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR3H7:0 - Timer/Counter3 Input Capture Register High Byte**

18.11.28 ICR3L – Timer/Counter3 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($96)	ICR3L7:0								ICR3L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Input Capture Register is updated with the counter (TCNT3) value each time an event occurs on the ICP3 pin. The Input Capture Register can be used for defining the counter TOP value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR3L7:0 - Timer/Counter3 Input Capture Register Low Byte**

18.11.29 TIMSK3 – Timer/Counter3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($71)	Res1	Res0	ICIE3	Res	OCIE3C	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE3 - Timer/Counter3 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/Counter3 Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICF3 Flag, located in TIFR3, is set.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE3C - Timer/Counter3 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3C Flag, located in TIFR3, is set.

- **Bit 2 – OCIE3B - Timer/Counter3 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3B Flag, located in TIFR3, is set.

- **Bit 1 – OCIE3A - Timer/Counter3 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF3A Flag, located in TIFR3, is set.

- **Bit 0 – TOIE3 - Timer/Counter3 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter3 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV3 Flag, located in TIFR3, is set.

18.11.30 TIFR3 – Timer/Counter3 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	Res1	Res0	ICF3	Res	OCF3C	OCF3B	OCF3A	TOV3	TIFR3
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF3 - Timer/Counter3 Input Capture Flag**

This flag is set when a capture event occurs on the ICP3 pin. When the Input Capture Register (ICR3) is set by the WGM33:0 to be used as the TOP value, the ICF3 Flag is set when the counter reaches the TOP value. ICF3 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF3 can be cleared by writing a logic one to its bit location.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF3C - Timer/Counter3 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register C (OCR3C). Note that a Forced Output Compare (FOC3C) strobe will not set the OCF3C Flag. OCF3C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF3C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF3B - Timer/Counter3 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register B (OCR3B). Note that a Forced Output Compare (FOC3B) strobe will not set the OCF3B Flag. OCF3B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF3B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF3A - Timer/Counter3 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT3) value matches the Output Compare Register A (OCR3A). Note that a Forced Output Compare (FOC3A) strobe will not set the OCF3A Flag. OCF3A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF3A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV3 - Timer/Counter3 Overflow Flag**

The setting of this flag is dependent of the WGM33:0 bits setting of the Timer/Counter3 Control Register. In Normal and CTC modes, the TOV3 Flag is set when the timer overflows. TOV3 is automatically cleared when the Timer/Counter3 Overflow Interrupt Vector is executed. Alternatively, TOV3 can be cleared by writing a logic one to its bit location.

18.11.31 TCCR4A – Timer/Counter4 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($A0)	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	TCCR4A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM4A1:0 - Compare Output Mode for Channel A**

The Timer/Counter4 has only limited functionality. Therefore the COM4A1:0 bits do not control the output compare behavior of any pin. The following table shows the

COM4A1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-18 COM4A Register Bits

Register Bits	Value	Description
COM4A1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 5:4 – COM4B1:0 - Compare Output Mode for Channel B**

The Timer/Counter4 has only limited functionality. Therefore the COM4B1:0 bits do not control the output compare behavior of any pin. The following table shows the COM4B1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-19 COM4B Register Bits

Register Bits	Value	Description
COM4B1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 3:2 – COM4C1:0 - Compare Output Mode for Channel C**

The Timer/Counter4 has only limited functionality. Therefore the COM4C1:0 bits do not control the output compare behavior of any pin. The following table shows the COM4C1:0 bit functionality when the WGM43:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-20 COM4C Register Bits

Register Bits	Value	Description
COM4C1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 1:0 – WGM41:40 - Waveform Generation Mode**

Combined with the WGM43:2 bits found in the TCCR4B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter4 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-21 WGM4 Register Bits

Register Bits	Value	Description
WGM41:40	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA

Register Bits	Value	Description
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.32 TCCR4B – Timer/Counter4 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($A1)	ICNC4	ICES4	Res	WGM43	WGM42	CS42	CS41	CS40	TCCR4B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC4 - Input Capture 4 Noise Canceller**

Timer/Counter4 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 6 – ICES4 - Input Capture 4 Edge Select**

Timer/Counter4 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM41:40 - Waveform Generation Mode**

Combined with the WGM41:0 bits found in the TCCR4A Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter4 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-22 WGM4 Register Bits

Register Bits	Value	Description
WGM41:40	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit

Register Bits	Value	Description
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS42:40 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter4 according to the following table. External pin modes cannot be used for the Timer/Counter4.

Table 18-23 CS4 Register Bits

Register Bits	Value	Description
CS42:40	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	Reserved
	0x07	Reserved

18.11.33 TCCR4C – Timer/Counter4 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($A2)	FOC4A	FOC4B	FOC4C	Res4	Res3	Res2	Res1	Res0	TCCR4C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC4A - Force Output Compare for Channel A**

The FOC4A bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4A bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4A bits are implemented as strobes. Therefore it is the value present in the COM4A1:0 bits that determine the effect of the forced compare. A FOC4A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on

Compare Match (CTC) mode using OCR4A as TOP. The FOC4A bits are always read as zero.

- **Bit 6 – FOC4B - Force Output Compare for Channel B**

The FOC4B bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4B bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4B bits are implemented as strobes. Therefore it is the value present in the COM4B1:0 bits that determine the effect of the forced compare. A FOC4B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR4B as TOP. The FOC4B bits are always read as zero.

- **Bit 5 – FOC4C - Force Output Compare for Channel C**

The FOC4C bit is only active when the WGM43:0 bits specify a non-PWM mode. When writing a logical one to the FOC4C bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter4 the match has no direct impact on any output pin. Note that the FOC4C bits are implemented as strobes. Therefore it is the value present in the COM4C1:0 bits that determine the effect of the forced compare. A FOC4C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR4C as TOP. The FOC4C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.34 TCNT4H – Timer/Counter4 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A5)	TCNT4H7:0								TCNT4H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT4H and TCNT4L, combined TCNT4) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT4) while the counter is running introduces a risk of missing a compare match between TCNT4 and one of the OCR4x Registers. Writing to the TCNT4 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT4H7:0 - Timer/Counter4 High Byte**

18.11.35 TCNT4L – Timer/Counter4 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A4)	TCNT4L7:0								TCNT4L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT4H and TCNT4L, combined TCNT4) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT4) while the counter is running introduces a risk of missing a compare match between TCNT4 and one of the OCR4x Registers. Writing to the TCNT4 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT4L7:0 - Timer/Counter4 Low Byte**

18.11.36 OCR4AH – Timer/Counter4 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A9)	OCR4AH7:0								OCR4AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4AH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.37 OCR4AL – Timer/Counter4 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A8)	OCR4AL7:0								OCR4AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4AL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.38 OCR4BH – Timer/Counter4 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AB)	OCR4BH7:0								OCR4BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4BH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.39 OCR4BL – Timer/Counter4 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AA)	OCR4BL7:0								OCR4BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4BL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.40 OCR4CH – Timer/Counter4 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AD)	OCR4CH7:0								OCR4CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4CH7:0 - Timer/Counter4 Output Compare Register High Byte**

18.11.41 OCR4CL – Timer/Counter4 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($AC)	OCR4CL7:0								OCR4CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT4). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR4CL7:0 - Timer/Counter4 Output Compare Register Low Byte**

18.11.42 ICR4H – Timer/Counter4 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A7)	ICR4H7:0								ICR4H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter4 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT4) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR4H7:0 - Timer/Counter4 Input Capture Register High Byte**

18.11.43 ICR4L – Timer/Counter4 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($A6)	ICR4L7:0								ICR4L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter4 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT4) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR4L7:0 - Timer/Counter4 Input Capture Register Low Byte**

18.11.44 TIMSK4 – Timer/Counter4 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($72)	Res1	Res0	ICIE4	Res	OCIE4C	OCIE4B	OCIE4A	TOIE4	TIMSK4
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE4 - Timer/Counter4 Input Capture Interrupt Enable**

The Timer/Counter4 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE4C - Timer/Counter4 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4C Flag, located in TIFR4, is set.

- **Bit 2 – OCIE4B - Timer/Counter4 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4B Flag, located in TIFR4, is set.

- **Bit 1 – OCIE4A - Timer/Counter4 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF4A Flag, located in TIFR4, is set.

- **Bit 0 – TOIE4 - Timer/Counter4 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter4 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV4 Flag, located in TIFR4, is set.

18.11.45 TIFR4 – Timer/Counter4 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	Res1	Res0	ICF4	Res	OCF4C	OCF4B	OCF4A	TOV4	TIFR4
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF4 - Timer/Counter4 Input Capture Flag**

The Timer/Counter4 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF4C - Timer/Counter4 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register C (OCR4C). Note that a Forced Output Compare (FOC4C) strobe will not set the OCF4C Flag. OCF4C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF4C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF4B - Timer/Counter4 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register B (OCR4B). Note that a Forced Output Compare (FOC4B) strobe will not set the OCF4B Flag. OCF4B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF4B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF4A - Timer/Counter4 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT4) value matches the Output Compare Register A (OCR4A). Note that a Forced Output Compare (FOC4A) strobe will not set the OCF4A Flag. OCF4A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF4A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV4 - Timer/Counter4 Overflow Flag**

The setting of this flag is dependent of the WGM43:0 bits setting of the Timer/Counter4 Control Register. In Normal and CTC modes, the TOV4 Flag is set when the timer overflows. TOV4 is automatically cleared when the Timer/Counter4 Overflow Interrupt Vector is executed. Alternatively, TOV4 can be cleared by writing a logic one to its bit location.

18.11.46 TCCR5A – Timer/Counter5 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($120)	COM5A1	COM5A0	COM5B1	COM5B0	COM5C1	COM5C0	WGM51	WGM50	TCCR5A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM5A1:0 - Compare Output Mode for Channel A**

The Timer/Counter5 has only limited functionality. Therefore the COM5A1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5A1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-24 COM5A Register Bits

Register Bits	Value	Description
COM5A1:0	0	Normal operation

Register Bits	Value	Description
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 5:4 – COM5B1:0 - Compare Output Mode for Channel B**

The Timer/Counter5 has only limited functionality. Therefore the COM5B1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5B1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-25 COM5B Register Bits

Register Bits	Value	Description
COM5B1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 3:2 – COM5C1:0 - Compare Output Mode for Channel C**

The Timer/Counter5 has only limited functionality. Therefore the COM5C1:0 bits do not control the output compare behavior of any pin. The following table shows the COM5C1:0 bit functionality when the WGM53:0 bits are set to a normal or a CTC mode (non-PWM). For the other functionality refer to section "Modes of Operation".

Table 18-26 COM5C Register Bits

Register Bits	Value	Description
COM5C1:0	0	Normal operation
	1	Reserved
	2	Reserved
	3	Reserved

- **Bit 1:0 – WGM51:50 - Waveform Generation Mode**

Combined with the WGM53:2 bits found in the TCCR5B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter5 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-27 WGM5 Register Bits

Register Bits	Value	Description
WGM51:50	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit
	0x7	Fast PWM, 10-bit

Register Bits	Value	Description
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

18.11.47 TCCR5B – Timer/Counter5 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($121)	ICNC5	ICES5	Res	WGM53	WGM52	CS52	CS51	CS50	TCCR5B
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC5 - Input Capture 5 Noise Canceller**

Timer/Counter5 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 6 – ICES5 - Input Capture 5 Edge Select**

Timer/Counter5 has only limited functionality. It is not connected to any Input Capture Pin. Therefore this bit has no meaningful function.

- **Bit 5 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 4:3 – WGM51:50 - Waveform Generation Mode**

Combined with the WGM51:0 bits found in the TCCR5A Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. For more information on the different modes see section "Modes of Operation". Note that Timer/Counter5 has only limited functionality. It cannot be connected to any I/O pin.

Table 18-28 WGM5 Register Bits

Register Bits	Value	Description
WGM51:50	0x0	Normal mode of operation
	0x1	PWM, phase correct, 8-bit
	0x2	PWM, phase correct, 9-bit
	0x3	PWM, phase correct, 10-bit
	0x4	CTC, TOP = OCRnA
	0x5	Fast PWM, 8-bit
	0x6	Fast PWM, 9-bit

Register Bits	Value	Description
	0x7	Fast PWM, 10-bit
	0x8	PWM, Phase and frequency correct, TOP = ICRn
	0x9	PWM, Phase and frequency correct, TOP = OCRnA
	0xA	PWM, Phase correct, TOP = ICRn
	0xB	PWM, Phase correct, TOP = OCRnA
	0xC	CTC, TOP = OCRnA
	0xD	Reserved
	0xE	Fast PWM, TOP = ICRn
	0xF	Fast PWM, TOP = OCRnA

- **Bit 2:0 – CS52:50 - Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter5 according to the following table. External pin modes cannot be used for the Timer/Counter5.

Table 18-29 CS5 Register Bits

Register Bits	Value	Description
CS52:50	0x00	No clock source (Timer/Counter stopped)
	0x01	clk_IO/1 (no prescaling)
	0x02	clk_IO/8 (from prescaler)
	0x03	clk_IO/64 (from prescaler)
	0x04	clk_IO/256 (from prescaler)
	0x05	clk_IO/1024 (from prescaler)
	0x06	Reserved
	0x07	Reserved

18.11.48 TCCR5C – Timer/Counter5 Control Register C

Bit	7	6	5	4	3	2	1	0	
NA ($122)	FOC5A	FOC5B	FOC5C	Res4	Res3	Res2	Res1	Res0	TCCR5C
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC5A - Force Output Compare for Channel A**

The FOC5A bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5A bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5A bits are implemented as strobes. Therefore it is the value present in the COM5A1:0 bits that determine the effect of the forced compare. A FOC5A strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5A as TOP. The FOC5A bits are always read as zero.

- **Bit 6 – FOC5B - Force Output Compare for Channel B**

The FOC5B bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5B bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5B bits are implemented as strobes. Therefore it is the value present in the COM5B1:0 bits that determine the effect of the forced compare. A FOC5B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5B as TOP. The FOC5B bits are always read as zero.

- **Bit 5 – FOC5C - Force Output Compare for Channel C**

The FOC5C bit is only active when the WGM53:0 bits specify a non-PWM mode. When writing a logical one to the FOC5C bit, an immediate compare match is forced. Due to the limited functionality of the Timer/Counter5 the match has no direct impact on any output pin. Note that the FOC5C bits are implemented as strobes. Therefore it is the value present in the COM5C1:0 bits that determine the effect of the forced compare. A FOC5C strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare Match (CTC) mode using OCR5C as TOP. The FOC5C bits are always read as zero.

- **Bit 4:0 – Res4:0 - Reserved**

These bits are reserved for future use.

18.11.49 TCNT5H – Timer/Counter5 High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($125)	TCNT5H7:0								TCNT5H
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT5H and TCNT5L, combined TCNT5) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT5) while the counter is running introduces a risk of missing a compare match between TCNT5 and one of the OCR5x Registers. Writing to the TCNT5 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT5H7:0 - Timer/Counter5 High Byte**

18.11.50 TCNT5L – Timer/Counter5 Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($124)	TCNT5L7:0								TCNT5L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The two Timer/Counter I/O locations (TCNT5H and TCNT5L, combined TCNT5) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit

counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details. Modifying the counter (TCNT5) while the counter is running introduces a risk of missing a compare match between TCNT5 and one of the OCR5x Registers. Writing to the TCNT5 Register blocks (removes) the compare match on the following timer clock for all compare units.

- **Bit 7:0 – TCNT5L7:0 - Timer/Counter5 Low Byte**

18.11.51 OCR5AH – Timer/Counter5 Output Compare Register A High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($129)	OCR5AH7:0								OCR5AH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5AH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.52 OCR5AL – Timer/Counter5 Output Compare Register A Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($128)	OCR5AL7:0								OCR5AL
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5AL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.53 OCR5BH – Timer/Counter5 Output Compare Register B High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12B)	OCR5BH7:0								OCR5BH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5BH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.54 OCR5BL – Timer/Counter5 Output Compare Register B Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12A)	OCR5BL7:0								OCR5BL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5BL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.55 OCR5CH – Timer/Counter5 Output Compare Register C High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12D)	OCR5CH7:0								OCR5CH
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5CH7:0 - Timer/Counter5 Output Compare Register High Byte**

18.11.56 OCR5CL – Timer/Counter5 Output Compare Register C Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($12C)	OCR5CL7:0								OCR5CL
Read/Write	R	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT5). A match can be used to generate an Output Compare interrupt. The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – OCR5CL7:0 - Timer/Counter5 Output Compare Register Low Byte**

18.11.57 ICR5H – Timer/Counter5 Input Capture Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($127)	ICR5H7:0								ICR5H
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter5 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT5) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR5H7:0 - Timer/Counter5 Input Capture Register High Byte**

18.11.58 ICR5L – Timer/Counter5 Input Capture Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($126)	ICR5L7:0								ICR5L
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter5 has only limited functionality. It is not connected to any I/O pin. Therefore the contents of this register is never updated with the counter (TCNT5) value. The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See section "Accessing 16-bit Registers" for details.

- **Bit 7:0 – ICR5L7:0 - Timer/Counter5 Input Capture Register Low Byte**

18.11.59 TIMSK5 – Timer/Counter5 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($73)	Res1	Res0	ICIE5	Res	OCIE5C	OCIE5B	OCIE5A	TOIE5	TIMSK5
Read/Write	R	R	RW	R	R	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICIE5 - Timer/Counter5 Input Capture Interrupt Enable**

The Timer/Counter5 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCIE5C - Timer/Counter5 Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5C Flag, located in TIFR5, is set.

- **Bit 2 – OCIE5B - Timer/Counter5 Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5B Flag, located in TIFR5, is set.

- **Bit 1 – OCIE5A - Timer/Counter5 Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCF5A Flag, located in TIFR5, is set.

- **Bit 0 – TOIE5 - Timer/Counter5 Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter5 Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOV5 Flag, located in TIFR5, is set.

18.11.60 TIFR5 – Timer/Counter5 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	Res1	Res0	ICF5	Res	OCF5C	OCF5B	OCF5A	TOV5	TIFR5
Read/Write	R	R	RW	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – Res1:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 5 – ICF5 - Timer/Counter5 Input Capture Flag**

The Timer/Counter5 has only limited functionality. It does not have an Input Capture pin. Therefore this bit has no useful meaning.

- **Bit 4 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3 – OCF5C - Timer/Counter5 Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register C (OCR5C). Note that a Forced Output Compare (FOC5C) strobe will not set the OCF5C Flag. OCF5C is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCF5C can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCF5B - Timer/Counter5 Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register B (OCR5B). Note that a Forced Output Compare (FOC5B) strobe will not set the OCF5B Flag. OCF5B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF5B can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF5A - Timer/Counter5 Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT5) value matches the Output Compare Register A (OCR5A). Note that a Forced Output Compare (FOC5A) strobe will not set the OCF5A Flag. OCF5A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF5A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV5 - Timer/Counter5 Overflow Flag**

The setting of this flag is dependent of the WGM53:0 bits setting of the Timer/Counter5 Control Register. In Normal and CTC modes, the TOV5 Flag is set when the timer overflows. TOV5 is automatically cleared when the Timer/Counter5 Overflow Interrupt Vector is executed. Alternatively, TOV5 can be cleared by writing a logic one to its bit location.

19 Timer/Counter 0, 1, 3, 4, and 5 Prescaler

Timer/Counter 0, 1, 3, 4, and 5 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to all Timer/Counters. T*n* is used as a general name, *n* = 0, 1, 3, 4, or 5.

19.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CS*n*2:0 = 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 $f_{CLK_I/O}/1024$.

19.2 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 the Timer/Counter T*n*. 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 > CS*n*2:0 > 1). The number of system clock cycles from the moment the timer is enabled until 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).

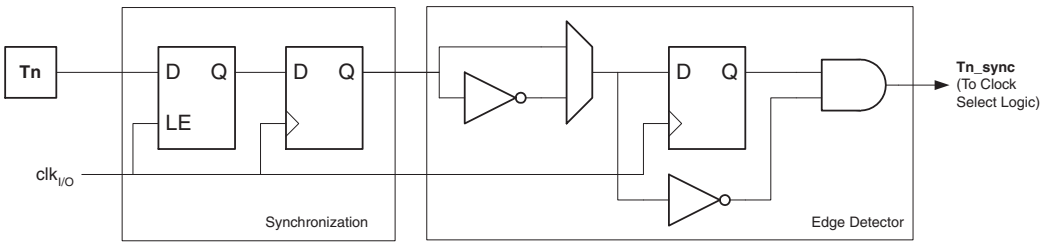
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all connected Timer/Counters.

19.3 External Clock Source

An external clock source applied to the T*n* pin can be used as Timer/Counter clock (clk_{Tn}). The T*n* 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 19-1 shows a functional equivalent block diagram of the T*n* 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 (CS*n*2:0 = 7) or negative (CS*n*2:0 = 6) edge it detects.

Figure 19-1. Tn/T0 Pin Sampling

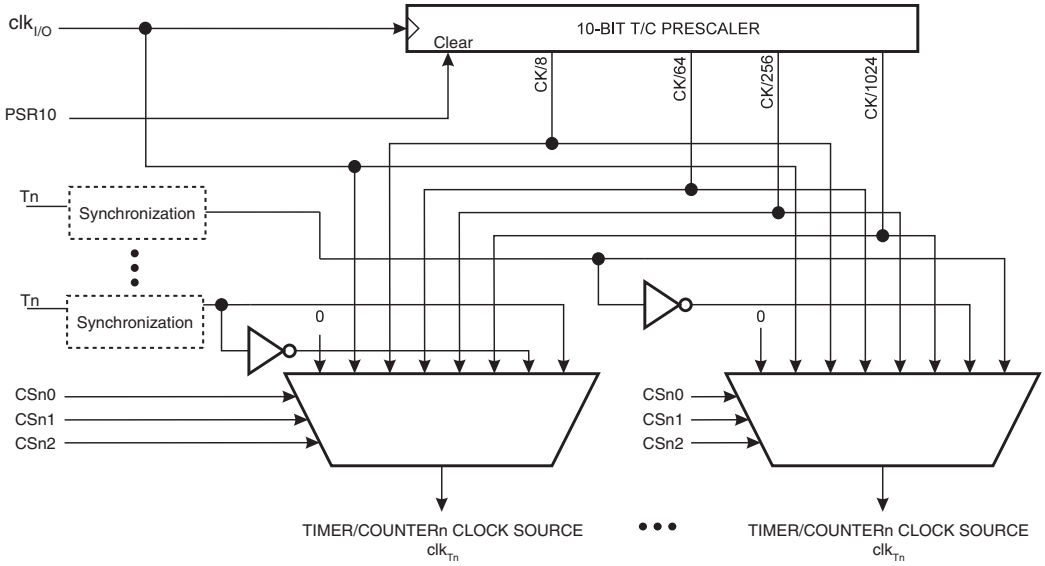


The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge applied to the T*n* pin to the counter being updated.

Enabling and disabling of the clock input must be done when T*n* has been stable for at least one system clock cycle. Otherwise there is a risk of generating a false Timer/Counter clock pulse.

Each half period of the applied, external clock 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 a detectable external clock 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 to limit the maximum frequency of an external clock source to less than $f_{clk_I/O}/2.5$. An external clock source can not be prescaled.

Figure 19-2. Prescaler for synchronous Timer/Counters



19.4 Register Description

19.4.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	TSM	Res4	Res3	Res2	Res1	Res0	PSRASY	PSRSYNC	GTCCR
Read/Write	RW	R	R	R	R	R	R	RW	
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 PSRASY and PSRSYNC bits 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 the configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware and the Timer/Counters simultaneously start counting.

- **Bit 6:2 – Res4:0 - Reserved**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1 – PSRASY - Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

- **Bit 0 – PSRSYNC - Prescaler Reset for Synchronous Timer/Counters**

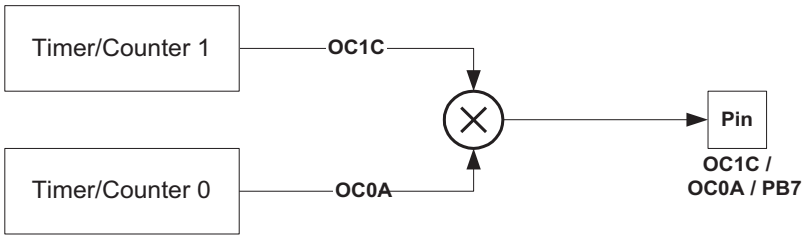
When this bit is one, the Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter0, Timer/Counter1, Timer/Counter3, Timer/Counter4 and Timer/Counter5 share the same prescaler and a reset of this prescaler will affect all timers.

20 Output Compare Modulator (OCM1C0A)

20.1 Overview

The Output Compare Modulator (OCM) allows generation of waveforms modulated with a carrier frequency. The modulator uses the outputs from the Output Compare Unit C of the 16-bit Timer/Counter1 and the Output Compare Unit of the 8-bit Timer/Counter0. For more details about these Timer/Counters see "Timer/Counter 0, 1, 3, 4, and 5 Prescaler" on page 304 and "8-bit Timer/Counter2 with PWM and Asynchronous Operation" on page 309.

Figure 20-1. Output Compare Modulator, Block Diagram



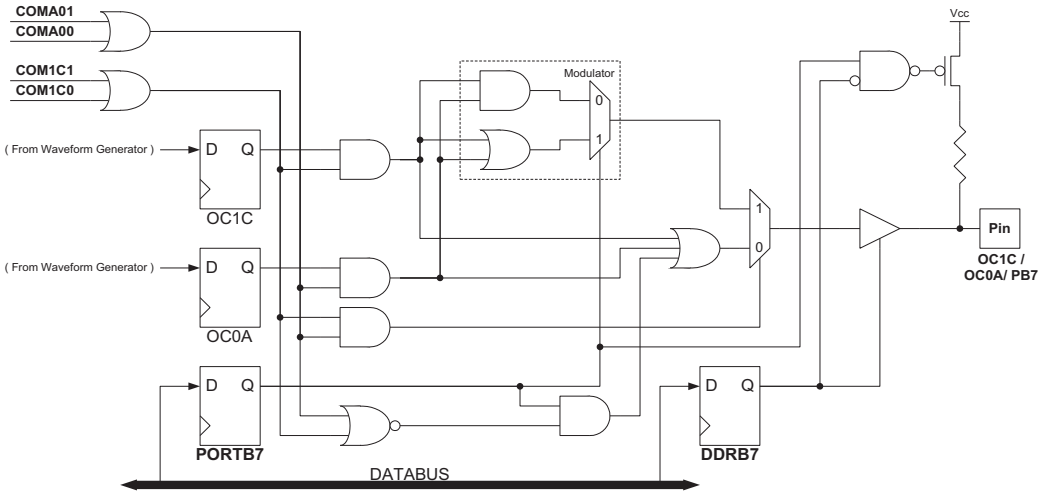
When the modulator is enabled, the two output compare channels are modulated together as shown in the block diagram (Figure 20-1 above).

20.2 Description

The Output Compare unit 1C and Output Compare unit 2 share the PB7 port pin for output. The outputs of the Output Compare units (OC1C and OC0A) override the normal PORTB7 Register when one of them is enabled (i.e., when COMnx1:0 is not equal to zero). When both OC1C and OC0A are enabled at the same time, the modulator is automatically enabled.

The functional equivalent schematic of the modulator is shown on in the following figure. The schematic includes part of the Timer/Counter units and the port B bit 7 output driver circuit.

Figure 20-2. Output Compare Modulator, Schematic

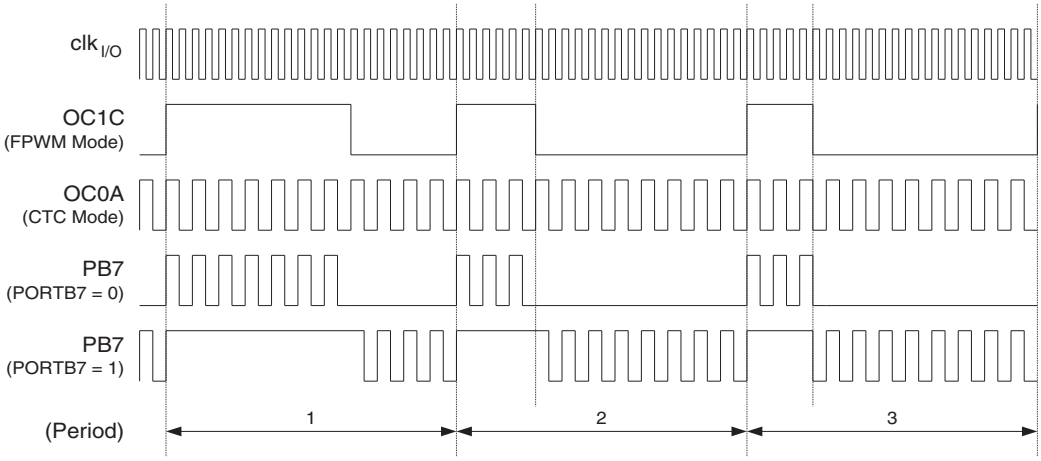


When the modulator is enabled the type of modulation (logical AND or OR) can be selected by the PORTB7 Register. Note that the DDRB7 controls the direction of the port independent of the COMnx1:0 bit setting.

20.3 Timing Example

Figure 20-3 below illustrates the modulator in action. In this example the Timer/Counter1 is set to operate in fast PWM mode (non-inverted) and Timer/Counter0 uses CTC waveform mode with toggle Compare Output mode (COMnx1:0 = 1).

Figure 20-3. Output Compare Modulator, Timing Diagram



In this example Timer/Counter2 provides the carrier while the modulating signal is generated by the Output Compare unit C of the Timer/Counter1.

The resolution of the PWM signal (OC1C) is reduced by the modulation. The reduction factor is equal to the number of system clock cycles of one period of the carrier (OC0A). In this example the resolution is reduced by a factor of two. The reason for the reduction is illustrated in Figure 20-3 above at the second and third period of the PB7 output when PORTB7 equals zero. The period 2 high time is one cycle longer than the period 3 high time, but the result on the PB7 output is equal in both periods.

21 8-bit Timer/Counter2 with PWM and Asynchronous Operation

21.1 Features

Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:

- **Single channel counter**
- **Clear timer on compare match (auto reload)**
- **Glitch-free, phase-correct pulse-width modulator (PWM)**
- **Frequency generator**
- **10 bit clock prescaler**
- **Overflow and compare match interrupt sources (TOV2, OCF2A and OCF2B)**
- **Able to run with external 32 kHz watch crystal independent of the I/O clock**

21.2 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown on Figure 21-1 on page 310. For the current placement of I/O pins, see chapter "Pin Configurations" on page 2. 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 "Register Description" on page 323.

The Power Reduction Timer/Counter2 bit PRTIM2 in register PRR0 (see "PRR0 – Power Reduction Register0" on page 167) must be written to zero to enable Timer/Counter2 module.

Note: OC2B is implemented but not routed to a pin and for this reason it can't be used.

21.2.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8 bit registers. Interrupt request (abbreviated to Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, asynchronously clocked from the TOSC1/2 pins or alternatively from the Automated Meter Reading (AMR) pin as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).

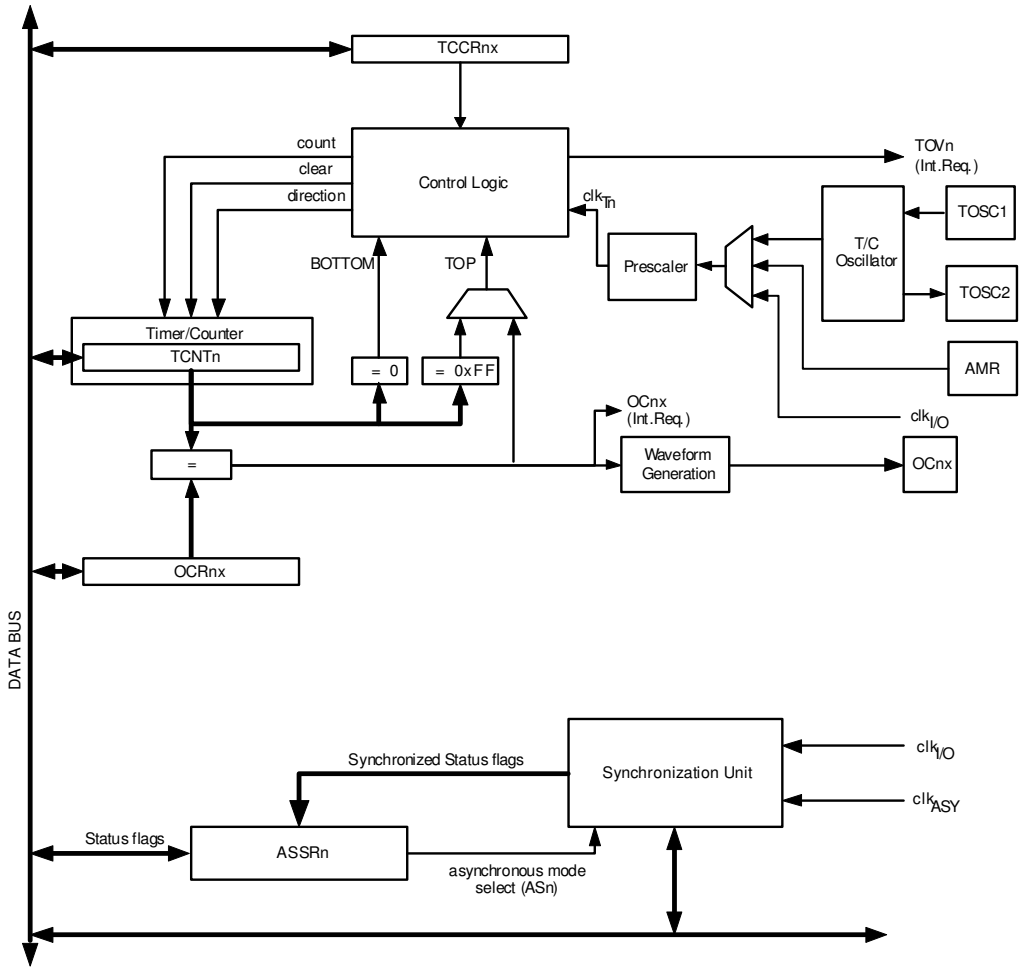
The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See chapter "Output Compare Unit" on page 316 for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

21.2.2 Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the

register or bit defines in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value and so on.

Figure 21-1. 8-bit Timer/Counter Block Diagram



The definitions in Table Table 21-1 below are also used extensively throughout the section.

Table 21-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
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 (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

21.3 Timer/Counter Clock Sources

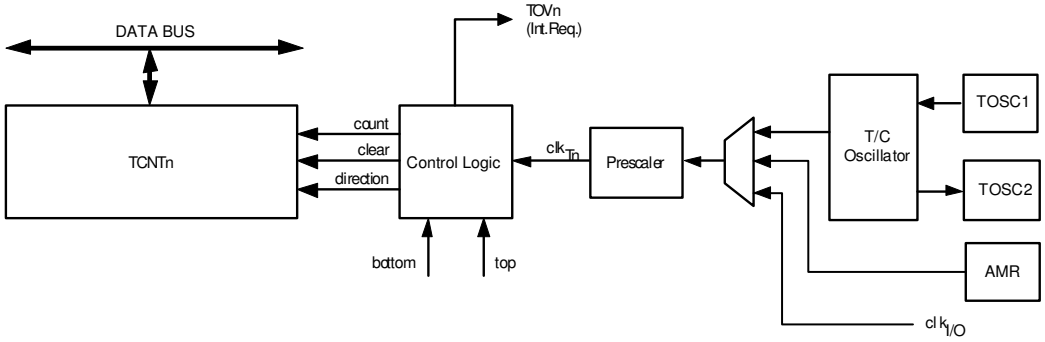
The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is either taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2 or from the AMR pin. For details on asynchronous operation, see section

"Asynchronous Operation of Timer/Counter2" on page 320. For details on clock sources and prescaler, see section "Timer/Counter Prescaler" on page 322.

21.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 21-2 below shows a block diagram of the counter and its surrounding environment.

Figure 21-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2A) and the WGM22 located in the Timer/Counter Control Register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see chapter "Modes of Operation" below.

The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

21.5 Modes of Operation

The mode of operation, i.e., the behaviour of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM22:0) and Compare Output mode (COM2x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM2x1:0 bits control whether the PWM output generated should be inverted or

not (inverted or non-inverted PWM). For non-PWM modes the COM2x1:0 bits control whether the output should be set, cleared, or toggled at a compare match (see chapter "Compare Match Output Unit" on page 317).

For detailed timing information refer to chapter "Timer/Counter Timing Diagrams" on page 319.

The following table shows the function of the WGM22:0 bits of registers TCCR2A and TCCR2B. These bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used.

Table 21-2. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRX at	TOV Flag Set on(1,2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

21.5.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM22:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8 bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 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 TOV2 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.

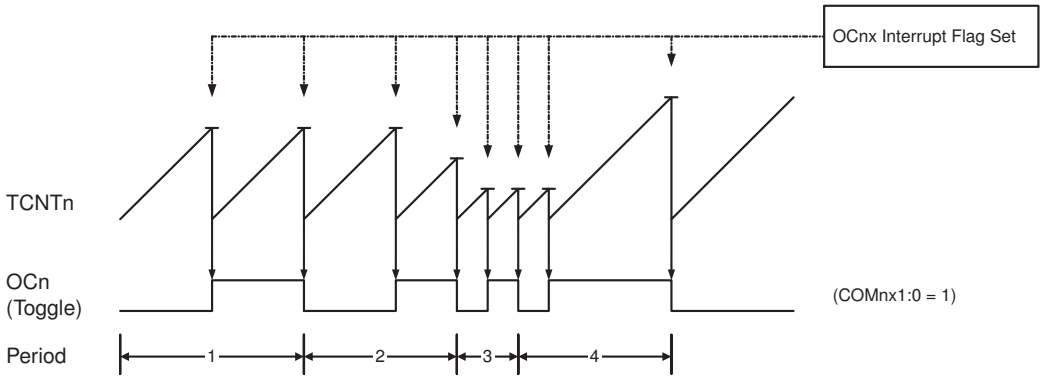
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

21.5.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM22:0 = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A 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 Table 20-3. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.

Figure 21-3. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A 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 OCR2A is lower than the current value of TCNT2, 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.

For generating a waveform output in CTC mode, the OC2A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM2A1:0 = 1). The OC2A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC2A} = f_{clk_I/O}/2$ when OCR2A is set to zero (0x00). The waveform frequency is defined by the following equation

$$f_{OCnx} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

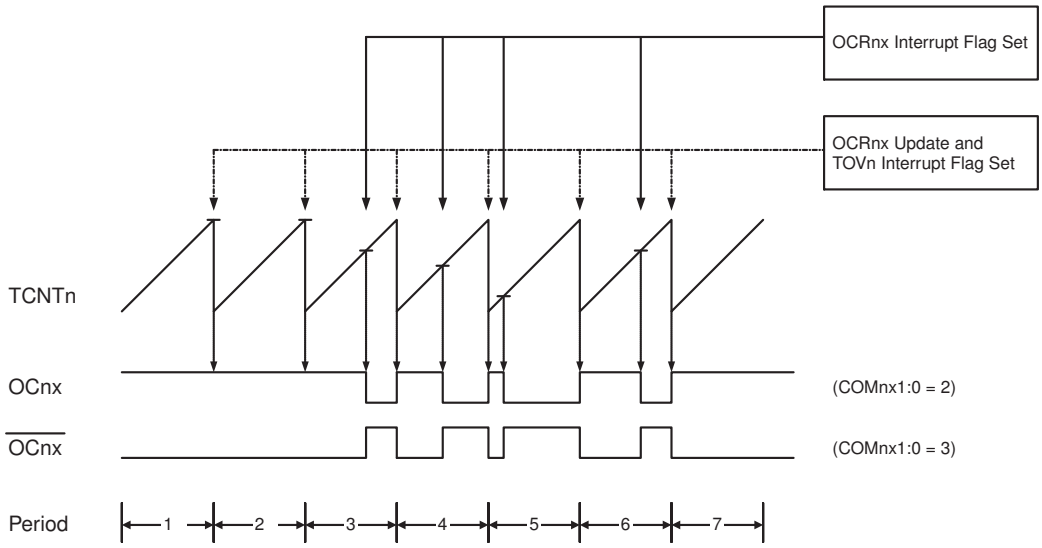
As for the Normal mode of operation, the TOV2 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

21.5.3 Fast PWM Mode

The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM22:0 = 3, and OCR2A when WGM22:0 = 7 (see section "Register Description" on page 323 for register TCCR2A). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2x Register at the compare match between OCR2x and TCNT2, and clearing (or setting) the OC2x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

Figure 21-4. Fast PWM Mode, Timing Diagram



The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot 256}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the fast PWM mode. If the OCR2A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR2A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM2A1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2x to toggle its logical level on each compare match (COM2x1:0 = 1). The waveform generated will have a maximum frequency of $f_{oc2} = f_{clk_I/O}/2$ when OCR2A is set to zero. This feature is similar to the OC2A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

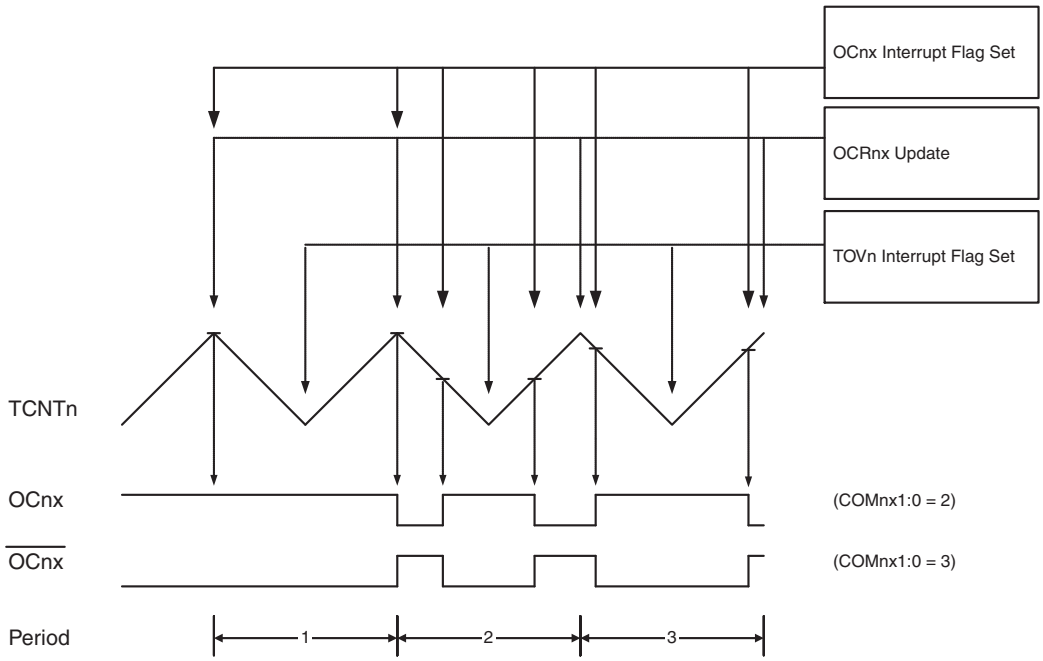
21.5.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM22:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM22:0 = 1, and OCR2A when WGM22:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC2x) is cleared on the compare match between TCNT2 and OCR2x while up-counting, and set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT2 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 21-5 on page 315. The TCNT2 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram

includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slopes represent compare matches between OCR2x and TCNT2.

Figure 21-5**.** Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC2x pin. Setting the COM2x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM2x1:0 to three. TOP is defined as 0xFF when WGM22:0 = 3, and OCR2A when WGM22:0 = 7 (see section "Register Description" on page 323 for register TCCR2A). The actual OC2x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2x Register at the compare match between OCR2x and TCNT2 when the counter increments, and setting (or clearing) the OC2x Register at compare match between OCR2x and TCNT2 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk_I/O}}{N \cdot 510}$$

The N variable represents the pre-scale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR2A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 21-5 above OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

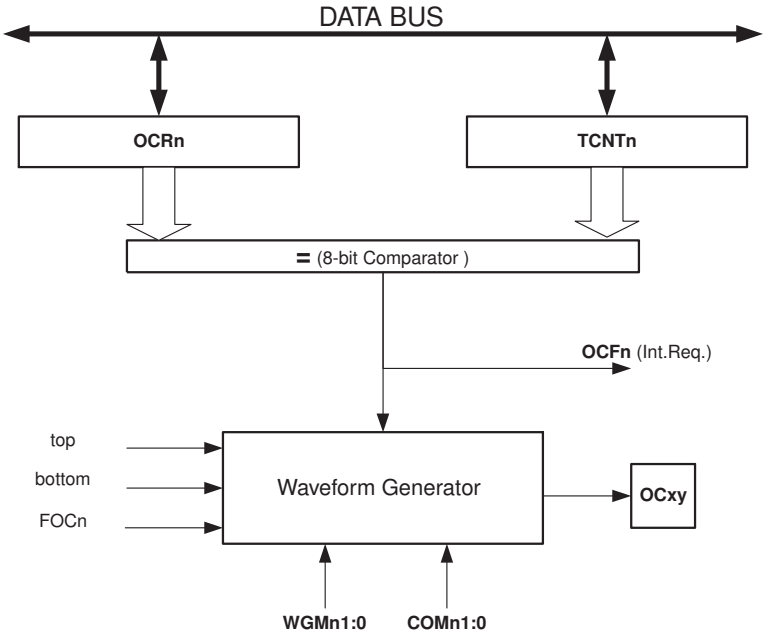
- OCR2A changes its value from MAX, like in Figure 21-5 on page 315. When the OCR2A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

21.6 Output Compare Unit

The 8 bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the Output Compare Flag (OCF2A or OCF2B) at the next timer clock cycle. 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 Output Compare Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM22:0 bits and Compare Output mode (COM2x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (chapter "Modes of Operation" on page 311).

Figure 21-6 below shows a block diagram of the Output Compare unit.

Figure 21-6**.** Output Compare Unit, Block Diagram



The OCR2x Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2x Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2x Register access may seem complex, but this is not the case. When the double buffering is enabled, the CPU has access to the OCR2x Buffer Register, and if double buffering is disabled the CPU will access the OCR2x directly.

21.6.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2x) bit. Forcing compare match will not set the OCF2x Flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x1:0 bits settings define whether the OC2x pin is set, cleared or toggled).

21.6.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

21.6.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is down-counting.

The setup of the OC2x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2x value is to use the Force Output Compare (FOC2x) strobe bit in Normal mode. The OC2x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM2x1:0 bits are not double buffered together with the compare value. A change of the COM2x1:0 bits will take effect immediately.

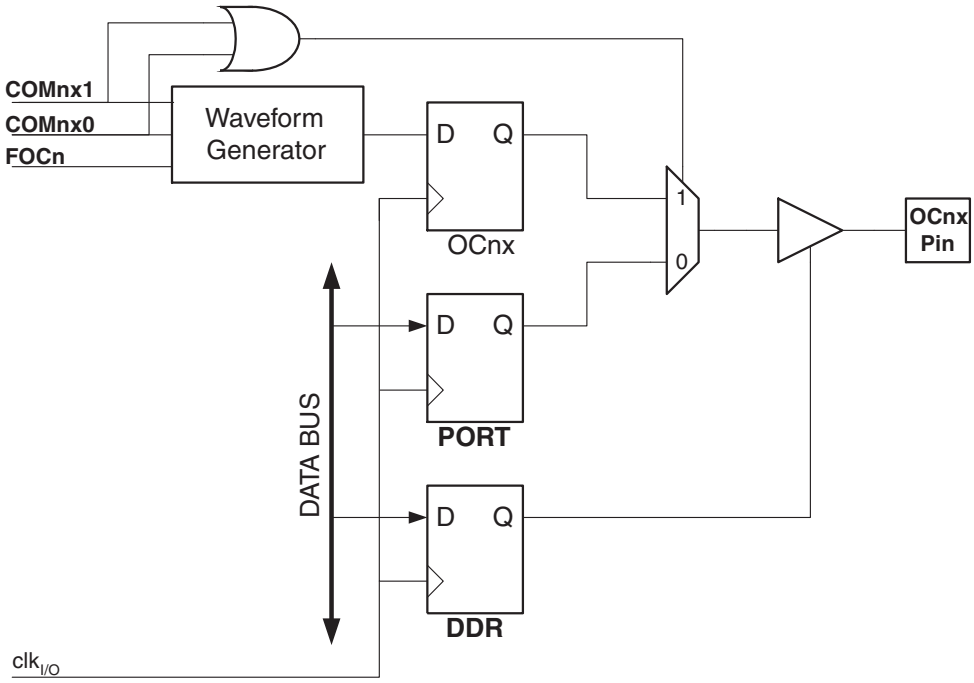
21.7 Compare Match Output Unit

The Compare Output mode (COM2x1:0) bits have two functions. The Waveform Generator uses the COM2x1:0 bits for defining the Output Compare (OC2x) state at the next compare match. Also, the COM2x1:0 bits control the OC2x pin output source. Figure 20-7 shows a simplified schematic of the logic affected by the COM2x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM2x1:0 bits are shown. When referring to the OC2x state, the reference is for the internal OC2x Register, not the OC2x pin.

The general I/O port function is overridden by the Output Compare (OC2x) from the Waveform Generator if either of the COM2x1:0 bits are set. However, the OC2x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2x pin (DDR_OC2x) must be set as output before the OC2x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC2x state before the output is enabled. Note that some COM2x1:0 bit settings are reserved for certain modes of operation. See section "Register Description" on page 323 for details.

Figure 21-7. Compare Match Output Unit, Schematic



21.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM2x1:0 bits differently in normal, CTC, and PWM modes. Setting the COM2x1:0 = 0 for all modes tells the Waveform Generator that no action on the OC2x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes for fast PWM mode and for phase correct PWM refer to section "Register Description" on page 323 for register TCCR2A**.** A change of the COM2x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC2x strobe bits.

The following table shows the COM2*x*1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 21-3. Compare Output Mode, non-PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected;
0	1	Toggle OC2*x* on Compare Match;
1	0	Clear OC2*x* on Compare Match;
1	1	Set OC2*x* on Compare Match;

Table 17-3 shows the COM2*x*1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

Table 21-4. Compare Output Mode, Fast PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match. OC2B: not applicable, reserved function;

COM2x1	COM2x0	Description
1	0	Clear OC2*x* on Compare Match, set OC2*x* at BOTTOM, (non-inverting mode).
1	1	Set OC2*x* on Compare Match, clear OC2*x* at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR2*x* equals TOP and COM2*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 313.

Table 17-4 shows the COM2*x*1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 21-5. Compare Output Mode, Phase Correct PWM Mode

COM2x1	COM2x0	Description
0	0	Normal port operation, OC2*x* disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match. OC2B: not applicable, reserved function;
1	0	Clear OC2*x* on Compare Match when up-counting. Set OC2*x* on Compare Match when down-counting.
1	1	Set OC2*x* on Compare Match when up-counting. Clear OC2*x* on Compare Match when down-counting.

Note: 1. A special case occurs when OCR2*x* equals TOP and COM2*x*1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 314 for more details.

21.8 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 21-8 below contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 21-8**.** Timer/Counter Timing Diagram, no Prescaling

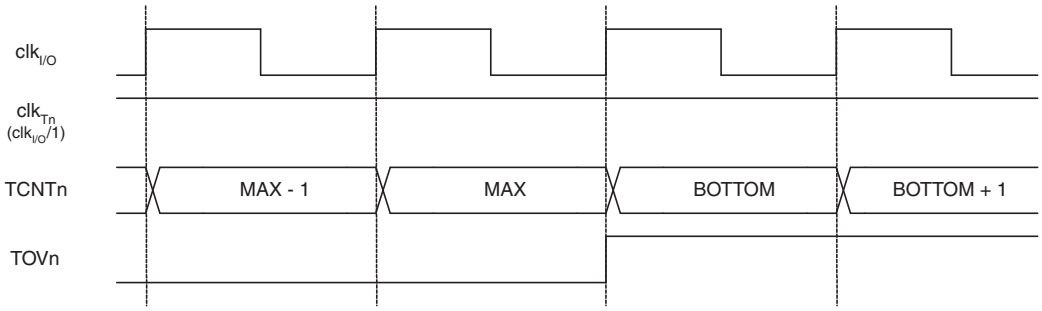


Figure 21-9 below shows the same timing data, but with the prescaler enabled.

Figure 21-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_I/O}/8$)

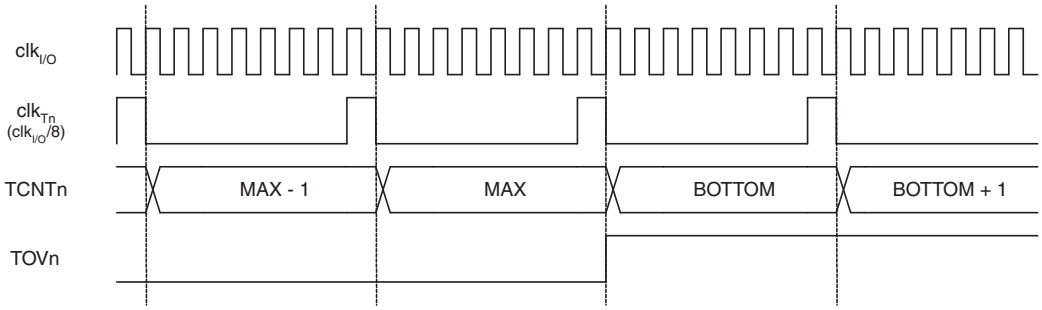


Figure 21-10 below shows the setting of OCF2A in all modes except CTC mode.

Figure 21-10. Timer/Counter Timing Diagram, Setting of OCF2A, with Prescaler ($f_{clk_I/O}/8$)

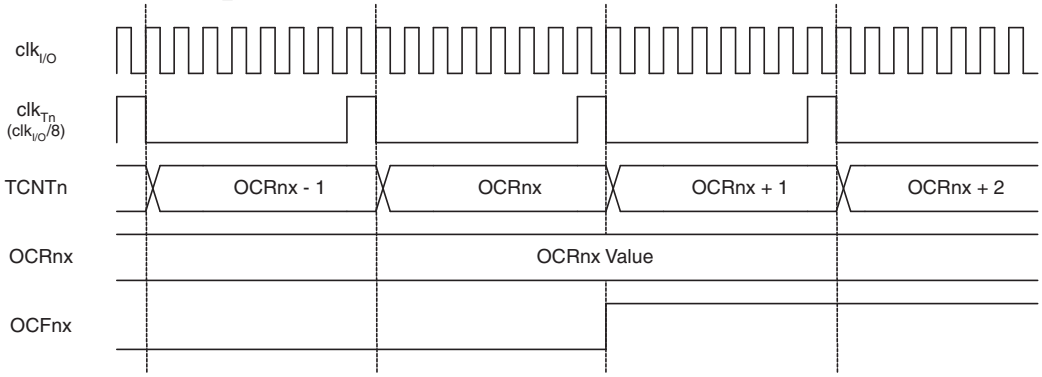
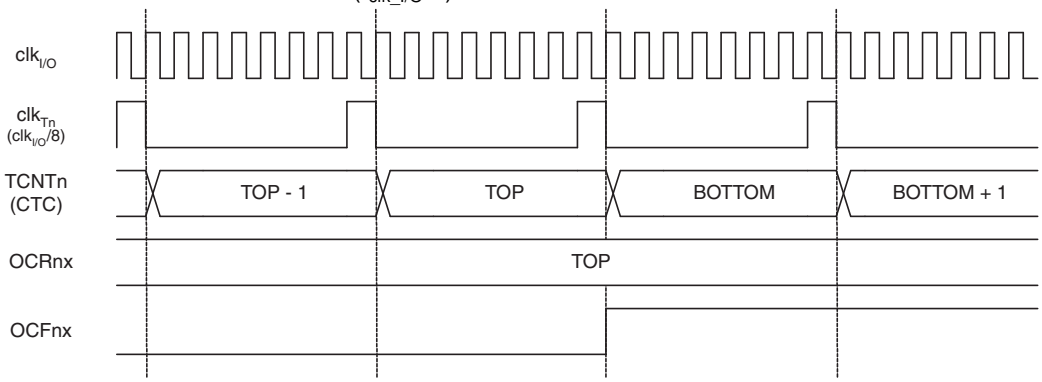


Figure 21-11 below shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 21-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk_I/O}/8$)



21.9 Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCIE2x and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2x, and TCCR2x.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2xUB, and TCR2xUB.
  5. Clear the Timer/Counter2 Interrupt Flags.
  6. Enable interrupts, if needed.
- The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2x, or TCCR2x, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the five mentioned registers have their individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2x write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save or ADC Noise Reduction mode after having written to TCNT2, OCR2x, or TCCR2x, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if any of the Output Compare2 interrupt is used to wake up the device, since the Output Compare function is disabled during writing to OCR2x or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the corresponding OCR2xUB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or ADC Noise Reduction mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering Powersave or ADC Noise Reduction mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  1. Write a value to TCCR2x, TCNT2, or OCR2x.
  2. Wait until the corresponding Update Busy Flag in ASSR returns to zero. .
  3. Enter Power-save or ADC Noise Reduction mode.
- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.
- Description of wake up from Power-save or ADC Noise Reduction mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always

advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.

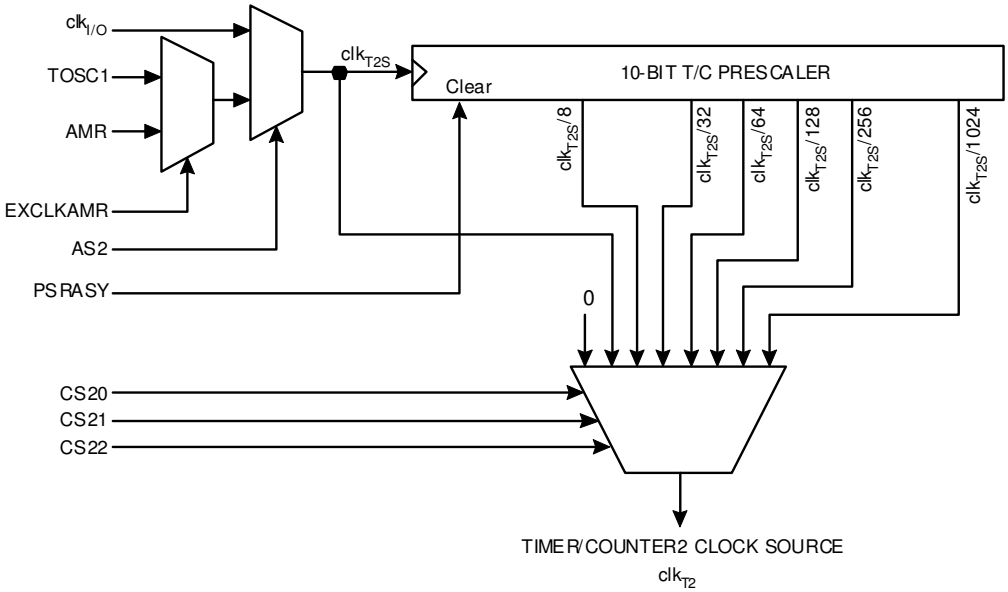
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Powersave mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:

  1. Write any value to either of the registers OCR2x or TCCR2x.
  2. Wait for the corresponding Update Busy Flag to be cleared.
  3. Read TCNT2.

- During asynchronous operation, the synchronization of the Interrupt Flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The Output Compare pin is changed on the timer clock and is not synchronized to the processor clock.
- If the CPU wakes up from asynchronous timer and goes back to sleep again, it may wakeup multiple times or the IRQ is called multiple times. This may be avoided if the CPU waits with the next sleep instruction until the next asynchronous clock arrives.

21.10 Timer/Counter Prescaler

Figure 21-12. Prescaler for Timer/Counter2



The register ASSR defines the clock source for the asynchronous Timer/Counter2. The clock source for Timer/Counter2 is named clk_{T2S}. clk_{T2S} is by default connected to the main system I/O clock clk_{IO}. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked either from the TOSC1 or from the AMR pin. This enables the use of Timer/Counter2 as a Real Time Counter (RTC).

The TOSC1 pin is selected by setting the EXCLKAMR bit in the ASSR register to logic zero. Under this condition TOSC1 and TOSC2 are disconnected from Port G and a crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The Oscillator is optimized for use with a 32.768 kHz crystal. By setting the EXCLK bit in the ASSR, a 32 kHz external clock can be applied on TOSC1.

Setting the EXCLKAMR bit to logic one selects the AMR pin as the Timer/Counter2 clock source. Thus the 32 kHz oscillator can be used by the MAC symbol counter while the Timer/Counter2 uses pin AMR as clock source, see "MAC Symbol Counter" on page 133.

A complete overview of the implemented asynchronous clock sources can be found in Table 21-6 below. The last column mentions which pins are available for GPIO functionality. For details about the ASSR register refer to section "Register Description" below.

For Timer/Counter2, the possible pre-scaled selections are: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, and $clk_{T2S}/1024$. Additionally, clk_{T2S} as well as 0 (stop) may be selected. Setting the PSRASY bit in GTCCR resets the prescaler. This allows the user to operate with a predictable prescaler.

Table 21-6. Asynchronous clock selection for Timer/Counter2 and Symbol-Counter

AS2	EXCLK	EXCLKAMR	Timer/Counter2 clock source	32 kHz crystal Osc. (TOSC1/TOSC2)	PG2, PG3, PG4 as GPIOs
0	0	0	cp2io	off	PG2, PG3, PG4
0	1	0	not defined	not defined	not defined
1	0	0	32 kHz crystal Osc	on	PG2
1	1	0	TOSC1 (PG4)	off	PG2, PG3
0	0	1	cp2io	off	PG2, PG3, PG4
0	1	1	not defined	not defined	not defined
1	0	1	AMR (PG2)	on	
1	1	1	AMR (PG2)	off	PG3, PG4

21.11 Register Description

21.11.1 TIMSK2 – Timer/Counter Interrupt Mask register

Bit	7	6	5	4	3	2	1	0	
NA ($70)	Res4	Res3	Res2	Res1	Res0	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCIE2B - Timer/Counter2 Output Compare Match B Interrupt Enable**

When the OCIE2B bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2B bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

- **Bit 1 – OCIE2A - Timer/Counter2 Output Compare Match A Interrupt Enable**

When the OCIE2A bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCF2A bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

- **Bit 0 – TOIE2 - Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is written to one and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs i.e., when the TOV2 bit is set in the Timer/Counter2 Interrupt Flag Register TIFR2.

21.11.2 TIFR2 – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	Res4	Res3	Res2	Res1	Res0	OCF2B	OCF2A	TOV2	TIFR2
Read/Write	R	R	R	R	R	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – Res4:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 2 – OCF2B - Output Compare Flag 2 B**

The OCF2B bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2B Output Compare Register2. OCF2B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2B (Timer/Counter2 Compare Match Interrupt Enable), and OCF2B are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 1 – OCF2A - Output Compare Flag 2 A**

The OCF2A bit is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2A Output Compare Register2. OCF2A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE2A (Timer/Counter2 Compare Match Interrupt Enable), and OCF2A are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 0 – TOV2 - Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE2A (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 changes counting direction at 0x00.

21.11.3 TCCR2A – Timer/Counter2 Control Register A

Bit	7	6	5	4	3	2	1	0	
NA ($B0)	COM2A1	COM2A0	COM2B1	COM2B0	Res1	Res0	WGM21	WGM20	TCCR2A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM2A1:0 - Compare Match Output A Mode**

These bits control the Output Compare pin (OC2A) behavior. If one or both of the COM2A1:0 bits are set, the OC2A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2A pin must be set in order to enable the output driver. When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM22:0 bit setting. The following table shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM). Refer to section "Compare Match Output Unit" for a description of the functionality in the other modes.

Table 21-7 COM2A Register Bits

Register Bits	Value	Description
COM2A1:0	0	Normal port operation, OC2A disconnected
	1	Toggle OC2A on Compare Match
	2	Clear OC2A on Compare Match
	3	Set OC2A on Compare Match

- **Bit 5:4 – COM2B1:0 - Compare Match Output B Mode**

These bits control the Output Compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC2B pin must be set in order to enable the output driver. When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. The following table shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM). Refer to section "Compare Match Output Unit" for a description of the functionality in the other modes.

Table 21-8 COM2B Register Bits

Register Bits	Value	Description
COM2B1:0	0	Normal port operation, OC2B disconnected
	1	Toggle OC2B on Compare Match
	2	Clear OC2B on Compare Match
	3	Set OC2B on Compare Match

- **Bit 3:2 – Res1:0 - Reserved**
- **Bit 1:0 – WGM21:20 - Waveform Generation Mode**

Combined with the WGM22 bit found in the TCCR2B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter2 unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see section "Modes of Operation" for details).

Table 21-9 WGM2 Register Bits

Register Bits	Value	Description
WGM21:20	0x0	Normal mode of operation
	0x1	PWM, phase correct, TOP=0xFF
	0x2	CTC, TOP = OCRA
	0x3	Fast PWM, TOP=0xFF
	0x4	Reserved
	0x5	PWM, Phase correct, TOP = OCRA
	0x6	Reserved
	0x7	Fast PWM, TOP=OCRA

21.11.4 TCCR2B – Timer/Counter2 Control Register B

Bit	7	6	5	4	3	2	1	0	
NA ($B1)	FOC2A	FOC2B	Res1	Res0	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC2A - Force Output Compare A**

The FOC2A bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written in PWM mode operation. When writing a logical one to the FOC2A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2A output is changed according to its COM2A1:0 bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A1:0 bits that determines the effect of the forced compare. A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP. The FOC2A bit is always read as zero.

- **Bit 6 – FOC2B - Force Output Compare B**

The FOC2B bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2B is written in PWM mode operation. When writing a logical one to the FOC2B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2B output is changed according to its COM2B1:0 bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B1:0 bits that determines the effect of the forced compare. A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP. The FOC2B bit is always read as zero.

- **Bit 5:4 – Res1:0 - Reserved**
- **Bit 3 – WGM22 - Waveform Generation Mode**

Combined with the WGM21:0 bits found in the TCCR2A Register, this bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. See description of "TCCR2A - Timer/Counter2 Control Register A" for details.

- **Bit 2:0 – CS22:20 - Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter2. If external pin modes are used for the Timer/Counter2, transitions on the T2 pin will clock

the counter even if the pin is configured as an output. This feature allows software control of the counting.

Table 21-10 CS2 Register Bits

Register Bits	Value	Description
CS22:20	0x00	No clock source (Timer/Counter2 stopped)
	0x01	clk_T2S/1 (no prescaling)
	0x02	clk_T2S/8 (from prescaler)
	0x03	clk_T2S/32 (from prescaler)
	0x04	clk_T2S/64 (from prescaler)
	0x05	clk_T2S/128 (from prescaler)
	0x06	clk_T2S/256 (from prescaler)
	0x07	clk_T2S/1024 (from prescaler)

21.11.5 TCNT2 – Timer/Counter2

Bit	7	6	5	4	3	2	1	0	
NA ($B2)	TCNT27:20								TCNT2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the 8-bit counter unit of the Timer/Counter2. Writing to the TCNT2 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a Compare Match between TCNT2 and the OCR2x Registers.

- **Bit 7:0 – TCNT27:20 - Timer/Counter2 Byte**

21.11.6 OCR2A – Timer/Counter2 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
NA ($B3)	OCR2A7:0								OCR2A
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
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 (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2A pin.

- **Bit 7:0 – OCR2A7:0 - Output Compare Register**

21.11.7 OCR2B – Timer/Counter2 Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
NA ($B4)	OCR2B7:0								OCR2B
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
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 (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2B pin.

- **Bit 7:0 – OCR2B7:0 - Output Compare Register**

21.11.8 ASSR – Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($B6)	EXCLKAMR	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	ASSR
Read/Write	RW	RW	RW	R	R	R	R	R	
Initial	0	0	0	0	0	0	0	0	

The register ASSR controls the asynchronous clocks for Timer/Counter2 and enables the asynchronous 32kHz clock for the symbol counter. Three bits (AS2,EXCLK,EXCLKAMR) are used to control the clocks. Note, to prevent clock spikes on asynchronous clock wires, every access to ASSR should change only one of the three bits.

- **Bit 7 – EXCLKAMR - Enable External Clock Input for AMR**

The bit EXCLKAMR extends the available clock sources for Timer/Counter2. If this bit is written to one, and asynchronous clock is selected (bit AS2 set), AMR functionality is enabled and Timer/Counter2 is clocked by pin AMR.

- **Bit 6 – EXCLK - Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on Timer Oscillator 1 (TOSC1) pin instead of a 32 kHz crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal Oscillator will only run when this bit is zero.

- **Bit 5 – AS2 - Timer/Counter2 Asynchronous Mode**

When AS2 is written to zero, Timer/Counter2 is clocked from the I/O clock, clkI/O. When AS2 is written to one, Timer/Counter2 is clocked from a crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2A, OCR2B, TCCR2A and TCCR2B might be corrupted.

- **Bit 4 – TCN2UB - Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 3 – OCR2AUB - Timer/Counter2 Output Compare Register A Update Busy**

When Timer/Counter2 operates asynchronously and OCR2A is written, this bit becomes set. When OCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2A is ready to be updated with a new value.

- **Bit 2 – OCR2BUB - Timer/Counter2 Output Compare Register B Update Busy**

When Timer/Counter2 operates asynchronously and OCR2B is written, this bit becomes set. When OCR2B has been updated from the temporary storage register,

this bit is cleared by hardware. A logical zero in this bit indicates that OCR2B is ready to be updated with a new value.

- **Bit 1 – TCR2AUB - Timer/Counter2 Control Register A Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2A is written, this bit becomes set. When TCCR2A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2A is ready to be updated with a new value.

- **Bit 0 – TCR2BUB - Timer/Counter2 Control Register B Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2B is written, this bit becomes set. When TCCR2B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR2B is ready to be updated with a new value.

21.11.9 GTCCR – General Timer Counter Control register

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	TSM						PSRASY		GTCCR
Read/Write	RW						RW		
Initial Value	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 PSRASY and PSRSYNC bits 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 the configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware and the Timer/Counters simultaneously start counting.

- **Bit 1 – PSRASY - Prescaler Reset Timer/Counter2**

When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

22 SPI- Serial Peripheral Interface

22.1 Features

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega128RFA1 and peripheral devices or between several AVR devices.

The ATmega128RFA1 SPI includes the following features:

- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

22.2 Functional Description

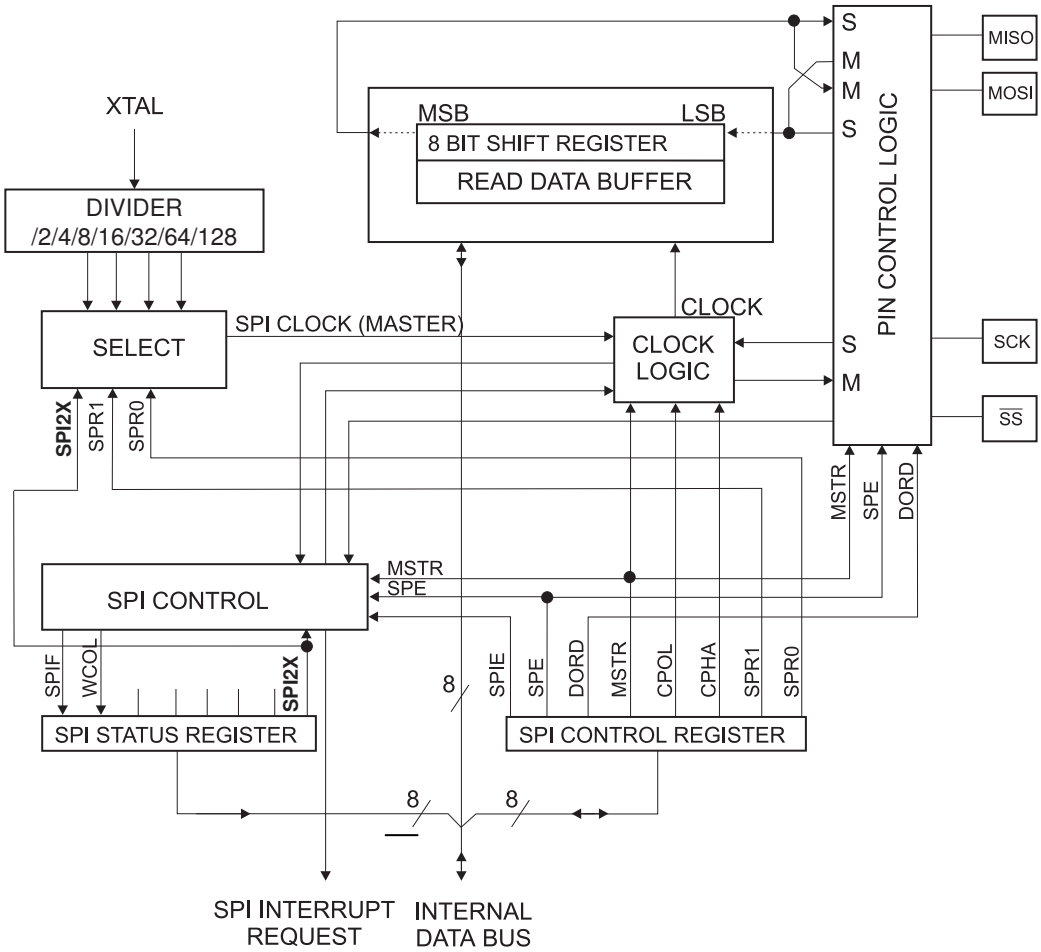
USART can also be used in Master SPI mode, see "USART in SPI Mode" on page 368. The Power Reduction SPI bit, PRSPI, in "PRR0 – Power Reduction Register0" on page 167 must be written to zero to enable SPI module. The block diagram of the SPI interface is shown in Figure 22-1 on page 331.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 22-2 on page 331. 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.

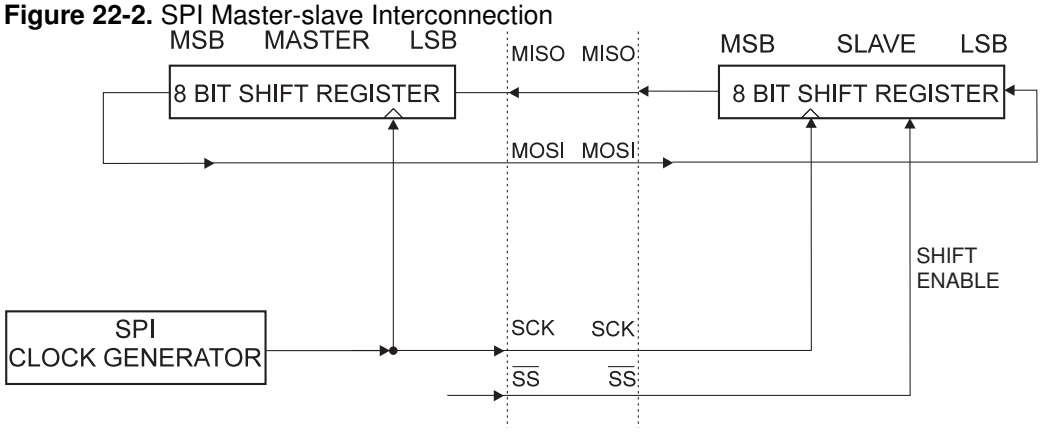
When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the 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 22-1. SPI Block Diagram[1]



Note: 1. Refer to Figure 1-1 on page 2 and Table 14-3 on page 193 for SPI pin placement.

Figure 22-2. 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 minimum low and high periods should be:

Low period: longer than 2 CPU clock cycles
High period: longer than 2 CPU clock cycles

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 21-1. For more details on automatic port overrides, refer to "Alternate Port Functions" on page 191.

Table 22-1. Pin Overrides[1]

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See "Alternate Functions of Port B" on page 192 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 "About Code Examples" on page 7

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 "About Code Examples" on page 7;

22.3 $\overline{SS}$ Pin Functionality

22.3.1 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.

22.3.2 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.

22.3.3 Data Mode

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 22-3 below and Figure 22-4 below. 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 in the summary of Table 22-2 below:

Table 22-2. CPOL Functionality

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

Figure 22-3. SPI Transfer Format with CPHA = 0

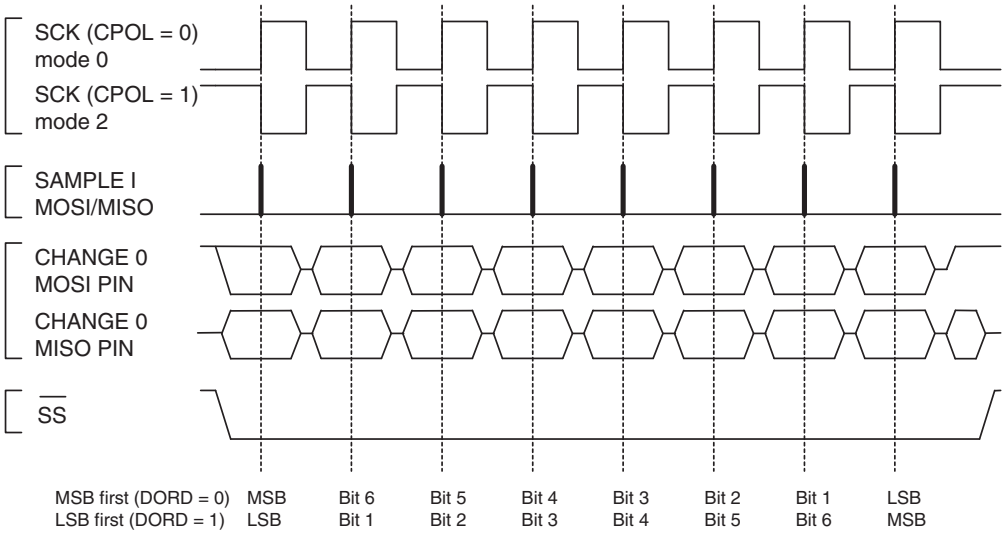
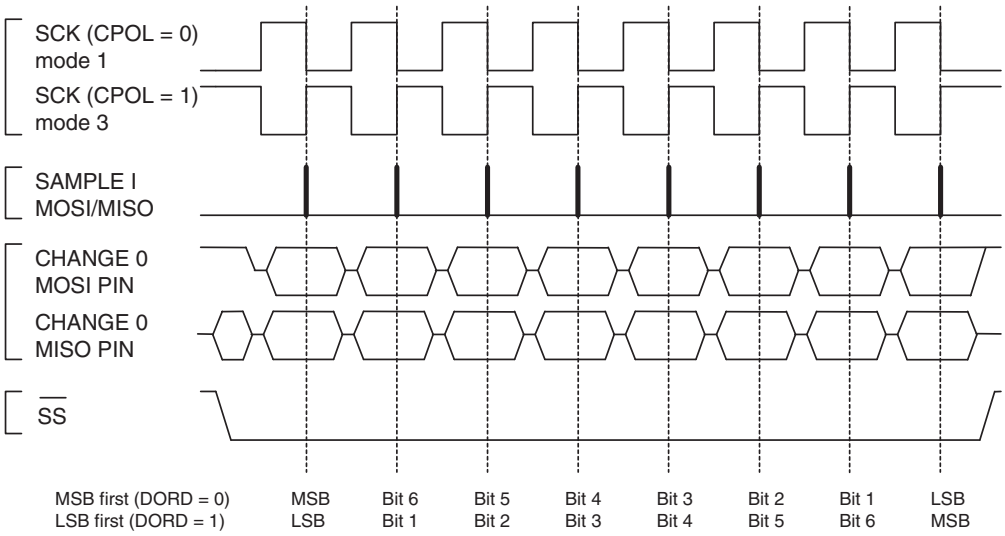


Figure 22-4. SPI Transfer Format with CPHA = 1



22.4 Register Description

22.4.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
$2C ($4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
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.s

- **Bit 6 – SPE - SPI Enable**

When the SPE bit is set (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 the Slave Select pin is configured as an input and is driven low while MSTR is set, MSTR will be cleared and SPIF in SPSR are 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 the "Data Modes" section for an example. The CPOL functionality is summarized below.

Table 22-3 CPOL Register Bits

Register Bits	Value	Description
CPOL	0	Rising (Leading Edge), Falling (Trailing Edge)
	1	Falling (Leading Egde), Rising (Trailing Edge)

- **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 the "Data Modes" section for an example. The CPOL functionality is summarized below.

Table 22-4 CPHA Register Bits

Register Bits	Value	Description
CPHA	0	Sample (Leading Edge), Setup (Trailing Edge)
	1	Setup (Leading Edge), Sample (Trailing Edge)

- **Bit 1:0 – SPR1:0 - 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 fosc is shown in the following table.

Table 22-5 SPR Register Bits

Register Bits	Value	Description
SPR1:0	0x00	fosc/4
	0x01	fosc/16
	0x02	fosc/64
	0x03	fosc/128
	0x04	fosc/2
	0x05	fosc/8
	0x06	fosc/32
	0x07	fosc/64

22.4.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
$2D ($4D)	SPIF	WCOL	Res4	Res3	Res2	Res1	Res0	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	RW	
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. The SPIF Flag is also set if the Slave Select pin is an input and is driven low when the SPI is in Master mode. 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 and 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 – Res4:0 - Reserved**
- **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. 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 fosc/4 or lower. The SPI interface on the ATmega128RFA1 is also used for program memory and EEPROM downloading or uploading. See section "Serial Downloading" for serial programming and verification.

22.4.3 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	SPDR7:0								SPDR
Read/Write	RW	RW	RW	RW	RW	RW	R	R	
Initial Value	X	X	X	X	X	X	0	0	

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.

- **Bit 7:0 – SPDR7:0 - SPI Data Register**

23 USART

23.1 Features

- **Full duplex operation (independent serial receive and transmit registers)**
- **Asynchronous or synchronous operation**
- **Master or slave clocked synchronous operation**
- **High resolution baud rate generator**
- **Supports serial frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits**
- **Odd or even parity generation and parity check supported by hardware**
- **Data overrun detection**
- **Framing error detection**
- **Noise filtering includes false start bit detection and digital low pass filter**
- **3 separate interrupts on TX complete, TX data register empty and RX complete**
- **Multi-processor communication mode**
- **Double speed, asynchronous communication mode**

23.2 Overview

The Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

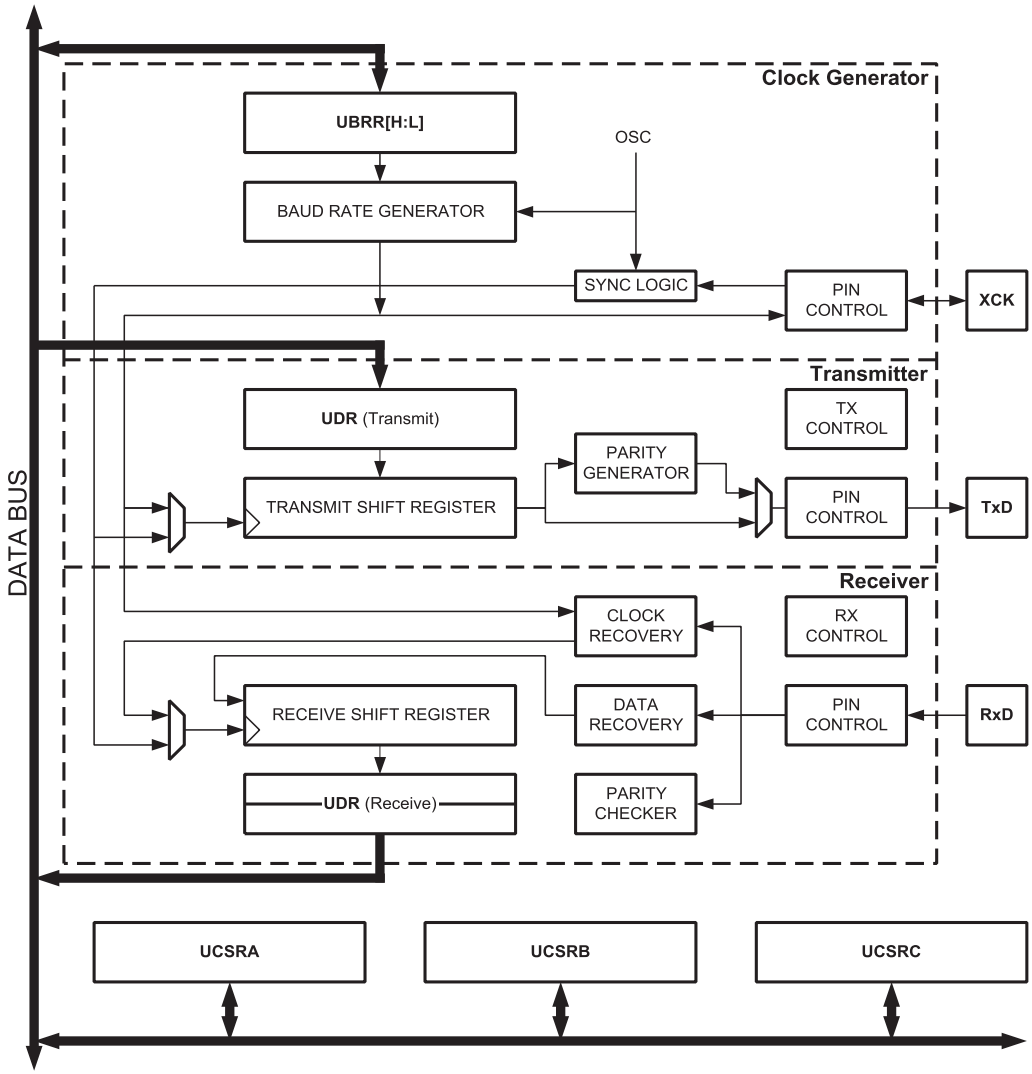
The ATmega128RFA1 has two USART's, USART0 and USART1. The functionality for all two USART's is described below. USART0 and USART1 have different I/O registers as shown in "Register Summary" on page 496.

A simplified block diagram of the USART transmitter is shown in Figure 23-1 on page 340 on page 340. CPU accessible I/O registers and I/O pins are shown in bold.

The Power Reduction USART0 bit, PRUSART0, in "PRR0 – Power Reduction Register0" on page 167 must be disabled by writing a logical zero to it. The Power Reduction USART1 bit, PRUSART1, in "PRR1 – Power Reduction Register 1" on page 168 must be disabled by writing a logical zero to it.

The dashed boxes in the block diagram Figure 23-1 on page 340 separate the three main parts of the USART (listed from the top): clock generator, transmitter and receiver. Control registers are shared by all units. The clock generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCK*n* (transfer clock) pin is only used by synchronous transfer mode. The transmitter consists of a single write buffer, a serial shift register, Parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the receiver includes a parity checker, control logic, a shift register and a two level receive buffer (UDR*n*). The receiver supports the same frame formats as the transmitter, and can detect frame, data overrun and parity errors.

Figure 23-1. USART Block Diagram[1]



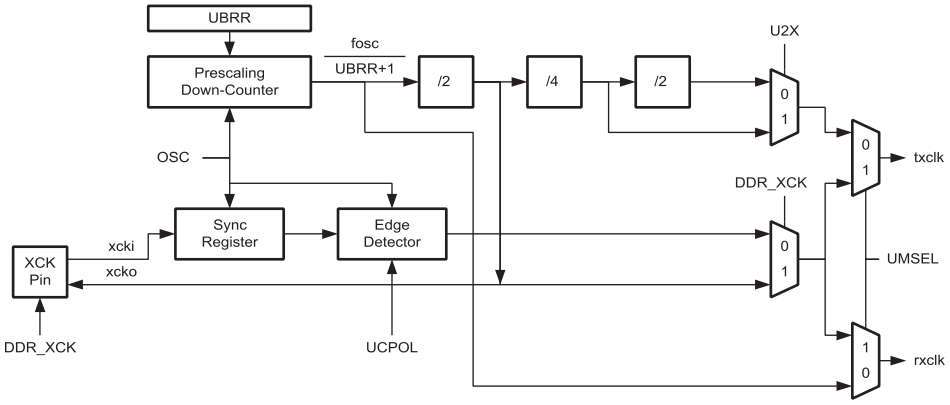
Note: 1. See "Figure 1-1" on page 2, Table 14-6 on page 195and Table 14-9 on page 197Table 14-9 on page 197for USART pin placement.

23.3 Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. The USART supports four modes of clock operation: Normal asynchronous, double speed asynchronous, master synchronous and slave synchronous mode. The UMSEL*n* bit in USART Control and Status Register C (UCSR*n*C) selects between asynchronous and synchronous operation. Double speed (asynchronous mode only) is controlled by the U2X*n* found in the UCSR*n*A register. When using synchronous mode (UMSEL*n* = 1), the data direction register for the XCK*n* pin (DDR_XCK*n*) controls whether the clock source is internal (master mode) or external (slave mode). The XCK*n* pin is only active when using synchronous mode.

Figure 22-2 on page 331 shows a block diagram of the clock generation logic.

Figure 23-2. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (internal signal).

rxclk Receiver base clock (internal signal).

xcki Input from XCK pin (internal signal). Used for synchronous slave operation.

xcko Clock output to XCK pin (internal signal). Used for synchronous master operation.

f_{osc} System clock frequency.

23.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 22-2 on page 331.

The USART Baud Rate Register (UBRR*n*) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRR*n* value each time the counter has counted down to zero or when the UBRRL*n* register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRR*n*+1)). The transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL*n*, U2X*n* and DDR_XCK*n* bits.

Table 23-1 below contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR*n* value for each mode of operation using an internally generated clock source.

Table 23-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X*n* = 0)	$BAUD = \frac{f_{osc}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{16\,BAUD} - 1$
Asynchronous Double Speed Mode (U2X*n* = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{8\,BAUD} - 1$

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Synchronous Master Mode	$BAUD = \frac{f_{osc}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{2\,BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{osc} System oscillator clock frequency

UBRR*n* Contents of the UBRRH*n* and UBRRL*n* registers, (0-4095)

Some examples of UBRR*n* values for some system clock frequencies are found in Table 23-14 on page 365.

23.3.2 Double Speed Operation (U2X*n*)

The transfer rate can be doubled by setting the U2X*n* bit in UCSR*n*A. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the transmitter, there are no downsides.

23.3.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 22-2 on page 331 for details.

External clock input from the XCK*n* pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the transmitter and receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK*n* clock frequency is limited by the following equation:

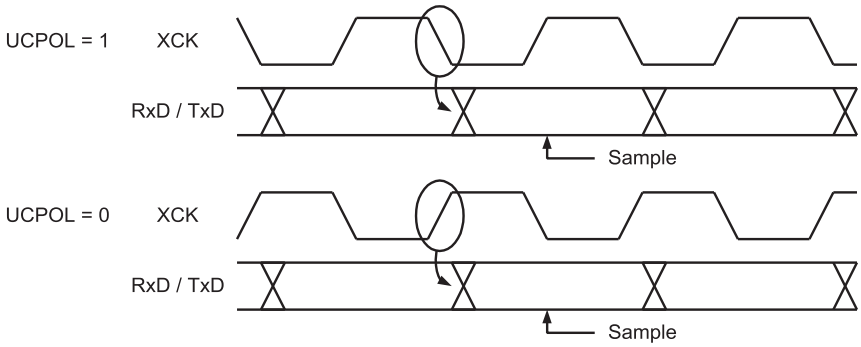
$$f_{XCK} < \frac{f_{osc}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

23.3.4 Synchronous Clock Operation

When synchronous mode is used (UMSEL*n* = 1), the XCK*n* pin will be used as either clock input (slave) or clock output (master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD*n*) is sampled at the opposite XCK*n* clock edge of the edge the data output (TxD*n*) is changed.

Figure 23-3. Synchronous Mode XCK*n* Timing



The UCPOL*n* bit UCRSC selects which XCK*n* clock edge is used for data sampling and which is used for data change. As Figure 22-3 on page 335 shows, when UCPOL*n* is zero the data will be changed at rising XCK*n* edge and sampled at falling XCK*n* edge. If UCPOLn is set, the data will be changed at falling XCK*n* edge and sampled at rising XCK*n* edge.

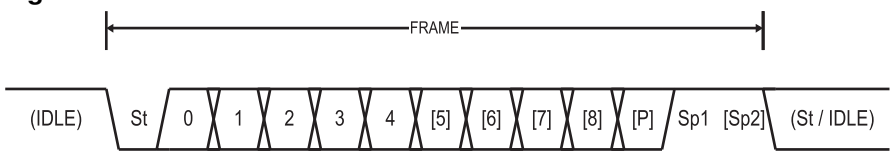
23.4 Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 23-4 below illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 23-4. Frame Formats



St Start bit, always low

(n) Data bits (0 to 8)

P Parity bit - can be odd or even

Sp Stop bit, always high

IDLE No transfers on the communication line (RxD*n* or TxD*n*). An IDLE line must be high

The frame format used by the USART is set by the UCSZ*n*2:0, UPM*n*1:0 and USBS*n* bits in UCSR*n*B and UCSR*n*C. The receiver and transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

The USART Character Size (UCSZ*n*2:0) bits select the number of data bits in the frame. The USART Parity Mode (UPM*n*1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS*n*) bit. The receiver ignores the second stop bit. A frame error will therefore only be detected in cases where the first stop bit is zero.

23.4.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The parity bit is located between the last data bit and first stop bit of a serial frame. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

$\mathbf{P_{even}}$	Parity bit using even parity
$\mathbf{P_{odd}}$	Parity bit using odd parity
$\mathbf{d_n}$	Data bit n of the character

23.5 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the transmitter or the receiver depending on the usage. For interrupt driven USART operation, the global interrupt flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC*n* flag can be used to check that the transmitter has completed all transfers, and the RXC flag can be used to check that there are no unread data in the receive buffer. Note that the TXC*n* flag must be cleared before each transmission (before UDR*n* is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

Assembly Code Example[1]

```
USART_Init:
 ; Set baud rate
 out UBRRnH, r17
 out UBRRnL, r16
 ; Enable receiver and transmitter
 ldi r16, (1<<RXENn)|(1<<TXENn)
 out UCSRnB,r16
 ; Set frame format: 8data, 2stop bit
 ldi r16, (1<<USBSn)|(3<<UCSZn0)
 out UCSRnC,r16
 ret
```

C Code Example[1]

```
#define FOSC 8000000// Clock Speed
#define BAUD 9600
#define (MYUBRR FOSC/16/BAUD-1)
void main( void )
{...
USART_Init ( MYUBRR );
...} // main
void USART_Init( unsigned int ubrr){
/* Set baud rate */
UBRRnH = (unsigned char)(ubrr>>8);
UBRRnL = (unsigned char) ubrr;
/* Enable receiver and transmitter */
UCSRnB = (1<<RXEN)|(1<<TXEN);
/* Set frame format: 8data, 2stop bit */
UCSRnC = (1<<USBS)|(3<<UCSZ0);
} // USART_Init
```

Note: 1. See "About Code Examples" on page 7

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

23.6 Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the Transmit Enable (TXEN) bit in the UCSR*n*B register. When the transmitter is enabled, the normal port operation of the TxD*n* pin is overridden by the USART and gives the function as the transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK*n* pin will be overridden and used as transmission clock.

23.6.1 Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR*n* I/O location. The buffered data in the transmit buffer will be moved to the shift register when the shift register is ready to send a new frame. The shift register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the shift register is loaded with new data, it will transfer one complete frame at the rate given by the baud rate register, U2Xn bit or by XCK*n* depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty Flag (UDRE*n*). When using frames with less than eight bits, the most significant bits written to the UDR*n* are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register r16.

Assembly Code Example[1]

```
USART_Transmit:
 ; Wait for empty transmit buffer
 sbis UCSRnA,UDREn rjmp USART_Transmit
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn)) );
 /* Put data into buffer, sends the data */
 UDRn = data;
}
```

Note: 1. See "About Code Examples" on page 7

The function simply waits for the transmit buffer to be empty by checking the UDRE*n* flag, before loading it with new data to be transmitted. If the data register empty interrupt is utilized, the interrupt routine writes the data into the buffer.

23.6.2 Sending Frames with 9 Data Bit

If 9 bit characters are used (UCSZ*n*2:0 = 7), the ninth bit must be written to the TXB8 bit in UCSR*n*B before the low byte of the character is written to UDR*n*. The following code examples show a transmit function that handles 9 bit characters. For the assembly code, the data to be sent is assumed to be stored in registers r17:r16.

Assembly Code Example[1][2]

```
USART_Transmit:
 ; Wait for empty transmit buffer
 sbis UCSRnA,UDREn
 rjmp USART_Transmit
 ; Copy 9th bit from r17 to TXB8
 cbi UCSRnB,TXB8
 sbrc r17,0
 sbi UCSRnB,TXB8
 ; Put LSB data (r16) into buffer, sends the data
 out UDRn,r16
 ret
```

C Code Example[1][2]

```
void USART_Transmit( unsigned int data )
{
 /* Wait for empty transmit buffer */
 while ( !( UCSRnA & (1<<UDREn))) );
 /* Copy 9th bit to TXB8 */
 UCSRnB &= ~(1<<TXB8);
 if ( data & 0x0100 )
   UCSRnB |= (1<<TXB8);
 /* Put data into buffer, sends the data */
 UDRn = data;
}
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the content of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB register is used after initialization.

2. See "About Code Examples" on page 7

The 9th bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

23.6.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE*n*) and Transmit Complete (TXC*n*). Both flags can be used for generating interrupts.

The Data Register Empty Flag (UDRE*n*) indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always write this bit to zero when writing the UCSR*n*A register.

When the USART Data Register Empty Interrupt Enable (UDRIE*n*) bit in UCSR*n*B is written to one, the USART data register empty interrupt will be executed as long as UDRE*n* is set (provided that global interrupts are enabled). UDRE*n* is cleared by writing UDR*n*. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to UDR*n* in order to clear UDRE*n* or disable the data register empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete Flag (TXC*n*) bit is set one when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXC*n* flag bit is automatically cleared when a transmission complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC*n* flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmission Complete Interrupt Enable (TXCIE*n*) bit in UCSR*n*B is set, the USART transmission complete interrupt will be executed when the TXC*n* flag becomes set (provided that global interrupts are enabled). When the transmission complete interrupt is used, the interrupt handling routine does not have to clear the TXC*n* flag. This is done automatically when the interrupt is executed.

23.6.4 Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM*n*1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

23.6.5 Disabling the Transmitter

The disabling of the transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. The transmitter will no longer override the TxD*n* pin when disabled.

23.7 Data Reception – The USART Receiver

The USART receiver is enabled by writing the Receive Enable (RXEN*n*) bit in the UCSR*n*B register to one. When the receiver is enabled, the normal pin operation of the RxD*n* pin is overridden by the USART and given the function as the receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK*n* pin will be used as transfer clock.

23.7.1 Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK*n* clock, and shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, i.e., a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR*n* I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete Flag (RXC*n*). When using frames with less than eight bits the most significant bits of the data read from the UDR*n* will be masked to zero. The USART has to be initialized before the function can be used. The function simply waits for data to be present in the receive buffer by checking the RXC*n* flag before reading the buffer and returning the value.

Assembly Code Example(1)

```
USART_Receive:
 ; Wait for data to be received
 sbis UCSRnA, RXCn
 rjmp USART_Receive
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

C Code Example(1)

```
unsigned char USART_Receive( void )
{
 /* Wait for data to be received */
 while ( !(UCSRnA & (1<<RXCn)) );
 /* Get and return received data from buffer */
 return UDRn;
}
```

Note: 1. See "About Code Examples" on page 7

23.7.2 Receiving Frames with 9 Data Bits

If 9 bit characters are used (UCSZ*n*2:0=7) the 9th bit must be read from the RXB8*n* bit in UCSR*n*B before reading the low bits from the UDR*n* register. This rule applies to the FE*n*, DOR*n* and UPE*n* status flags as well. Read status from UCSR*n*A, then data from UDR*n*. Reading the UDR*n* I/O location will change the state of the receive buffer FIFO and consequently the TXB8*n*, FE*n*, DOR*n* and UPE*n* bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example(1)

```
USART_Receive:
 ; Wait for data to be received
 sbis UCSRnA, RXCn
 rjmp USART_Receive
 ; Get status and 9th bit, then data from buffer
 in r18, UCSRnA
 in r17, UCSRnB
 in r16, UDRn
 ; If error, return -1
 andi r18,(1<<FEn)|(1<<DORn)|(1<<UPEn)
 breq USART_ReceiveNoError
 ldi r17, HIGH(-1)
 ldi r16, LOW(-1)
USART_ReceiveNoError:
 ; Filter the 9th bit, then return
 lsr r17
 andi r17, 0x01
 ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
 unsigned char status, resh, resl;
 /* Wait for data to be received */
 while ( !(UCSRnA & (1<<RXCn)) );
 /* Get status and 9th bit, then data */
 /* from buffer */
 status = UCSRnA;
 resh = UCSRnB;
 resl = UDRn;
 /* If error, return -1 */
 if ( status & (1<<FEn)|(1<<DORn)|(1<<UPEn) )
   return -1;
 /* Filter the 9th bit, then return */
 resh = (resh >> 1) & 0x01;
 return ((resh << 8) | resl);
}
```

Note: 1. See "About Code Examples" on page 7

The receive function example reads all the I/O registers into the register file before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

23.7.3 Receive Complete Flag and Interrupt

The USART receiver has one flag that indicates the receiver state.

The Receive Complete Flag (RXC*n*) indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled (RXEN*n* = 0), the receive buffer will be flushed and consequently the RXC*n* bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE*n*) in UCSR*n*B is set, the USART receive complete interrupt will be executed as long as the RXC*n* flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR*n* in order to clear the RXC*n* flag, otherwise a new interrupt will occur once the interrupt routine terminates.

23.7.4 Receiver Error Flags

The USART receiver has three error flags: Frame Error (FE*n*), Data OverRun (DOR*n*) and Parity Error (UPE*n*). All can be accessed by reading UCSR*n*A. Common for the error flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSR*n*A must be read before the receive buffer (UDR*n*), since reading the UDR*n* I/O location changes the buffer read location. The error flags cannot be altered by the application software doing a write to the flag location. However, all flags must be set to zero when the UCSR*n*A is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The Frame Error Flag (FE*n*) indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE*n* flag is zero when the stop bit was correctly

read (as one), and the FE*n* flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE*n* flag is not affected by the setting of the USBS*n* bit in UCSR*n*C since the receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSR*n*A.

The Data OverRun Flag (DOR*n*) indicates data loss due to a receiver buffer full condition. A data overrun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. If the DOR*n* flag is set there was one or more serial frame lost between the frame last read from UDR*n*, and the next frame read from UDR*n*. For compatibility with future devices, always write this bit to zero when writing to UCSR*n*A. The DOR*n* flag is cleared when the frame received was successfully moved from the shift register to the receive buffer.

The Parity Error Flag (UPE*n*) indicates that the next frame in the receive buffer had a parity error when received. If parity check is not enabled the UPE*n* bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSR*n*A. For more details see "Parity Bit Calculation" on page 344 and "Parity Checker" below.

23.7.5 Parity Checker

The parity checker is active when the high USART parity mode (UPM*n*1) bit is set. Type of parity check to be performed (odd or even) is selected by the UPM*n*0 bit. When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error Flag (UPE*n*) can then be read by software to check if the frame had a parity error.

The UPE*n* bit is set if the next character that can be read from the receive buffer had a parity error when received .The parity checking was enabled at that point (UPM*n*1 = 1). This bit is valid until the receive buffer (UDR*n*) is read.

23.7.6 Disabling the Receiver

In contrast to the transmitter, disabling of the receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXEN*n* is set to zero) the receiver will no longer override the normal function of the RxD*n* port pin. The receiver buffer FIFO will be flushed when the receiver is disabled. Remaining data in the buffer will be lost

23.7.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR*n* I/O location until the RXC*n* flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example[1]

```
USART_Flush:
 sbis UCSRnA, RXCn
 ret
 in r16, UDRn
 rjmp USART_Flush
```

C Code Example[1]

```
void USART_Flush( void )
{
 unsigned char dummy;
 while ( UCSRnA & (1<<RXCn) ) dummy = UDRn;
}
```

Note: 1. See "About Code Examples" on page 7

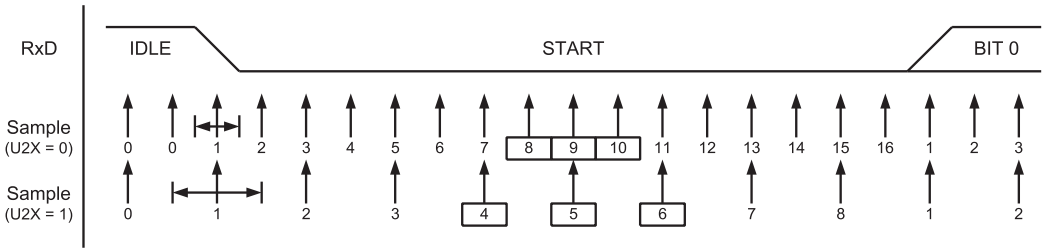
23.8 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD*n* pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

23.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 23-5 below illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and eight times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode (U2X*n* = 1) of operation. Samples denoted zero are samples done when the RxD*n* line is idle (i.e., no communication activity).

Figure 23-5. Start Bit Sampling



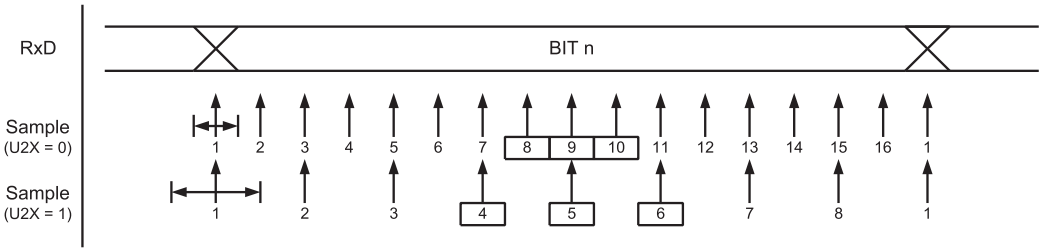
When the clock recovery logic detects a high (idle) to low (start) transition on the RxD*n* line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9 and 10 for Normal mode, and samples 4, 5 and 6 for double speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

23.8.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in double speed mode. Figure 23-6 on page 353

shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Figure 23-6. Sampling of Data and Parity Bit

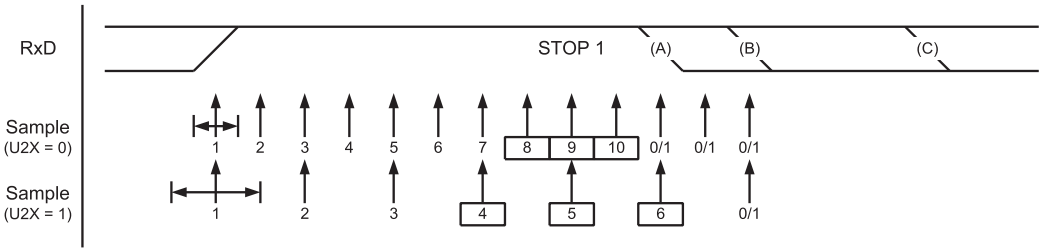


The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the centre of the received bit. The centre samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows:

If two or all three samples have high levels, the received bit is registered to be logic 1. If two or all three samples have low levels, the received bit is registered to be logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD*n* pin. The recovery process is then repeated until a complete frame is received including the first stop bit. Note that the receiver only uses the first stop bit of a frame.

Figure 23-7 below shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 23-7. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error Flag (FE*n*) will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For normal speed mode, the first low level sample can be at point marked (A) in Figure 23-7 above. For double speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the receiver.

23.8.3 Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the receiver does not have a similar (see Table 23-2 on page 354) base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_{MF}}$$

D Sum of character size and parity size (D = 5 to 10 bit)

S Samples per bit. S = 16 for normal speed mode and S = 8 for double speed mode.

S_F First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for double speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for double speed mode.

R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.

R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 23-2 below and Table 23-3 below list the maximum receiver baud rate error that can be tolerated. Note that normal speed mode has higher tolerance of baud rate variations.

Table 23-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2Xn = 0)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 23-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2Xn = 1)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

The recommendations of the maximum receiver baud rate error were made under the assumption that the receiver and transmitter equally divides the maximum total error.

There are two possible sources for the receiver baud rate error. The receiver's system clock will always have some minor instability over the supply voltage range and the temperature range. When using the radio transceiver crystal oscillator (XOSC) to generate the system clock, this is rarely a problem, but for the internal RC oscillator the system clock may differ more than 2% over the temperature range. The second source for the error is more controllable. The baud rate generator can not always do an exact

division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

23.9 Multi-processor Communication Mode

Setting the Multi-processor Communication Mode (MPCM*n*) bit in UCSR*n*A enables a filtering function of incoming frames received by the USART receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the MCU, in a system with multiple MCUs that communicate via the same serial bus. The transmitter is unaffected by the MPCM*n* setting, but has to be used differently when it is a part of a system utilizing the multi-processor communication mode.

If the receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the receiver is set up for frames with nine data bits, then the ninth bit (RXB8*n*) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The multi-processor communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

23.9.1 Using MPCMn

For an MCU to act as a master MCU, it can use a 9 bit character frame format (UCSZ*n*2:0 = 7). The 9th bit (TXB8*n*) must be set when an address frame (TXB8*n* = 1) or cleared when a data frame (TXB = 0) is being transmitted. The slave MCUs must in this case be set to use a 9 bit character frame format.

The following procedure should be used to exchange data in multi-processor communication mode:

1. All slave MCUs are in multi-processor communication mode (MPCM*n* in UCSR*n*A is set).
2. The master MCU sends an address frame, and all slaves receive and read this frame. In the slave MCUs, the RXC*n* flag in UCSRnA will be set as normal.
3. Each slave MCU reads the UDR*n* register and determines if it has been selected. If so, it clears the MPCM*n* bit in UCSR*n*A, otherwise it waits for the next address byte and keeps the MPCM*n* setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other slave MCUs, which still have the MPCM*n* bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM*n* bit and waits for a new address frame from master. The process then repeats from 2.

Using any of the 5 to 8 bit character frame formats is possible, but impractical since the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the transmitter and receiver uses the same character size setting. If 5 to 8 bit character frames are used, the transmitter must be set to use two stop bit (USBS*n* = 1) since the first stop bit is used for indicating the frame type.

Do not use read-modify-write instructions (SBI and CBI) to set or clear the MPCM*n* bit. The MPCM*n* bit shares the same I/O location as the TXC*n* flag and this might accidentally be cleared when using SBI or CBI instructions.

23.10 Register Description

23.10.1 UDR0 – USART0 I/O Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($C6)	UDR07:00								UDR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR0. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR0 Register location. Reading the UDR0 Register location will return the contents of the Receive Data Buffer Register (RXB). For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver. The transmit buffer can only be written when the UDRE0 Flag in the UCSR0A Register is set. Data written to UDR0 when the UDRE0 Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD0 pin. The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

- **Bit 7:0 – UDR07:00 - USART I/O Data Register**

23.10.2 UCSR0A – USART0 Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C0)	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/Write	R	RW	R	R	R	R	RW	RW	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC0 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

- **Bit 6 – TXC0 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

- **Bit 5 – UDRE0 - USART Data Register Empty**

The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE0 - Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FE0 bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

- **Bit 3 – DOR0 - Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register and a new start bit is detected. This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 2 – UPE0 - USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 1 – U2X0 - Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation. Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM0 - Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM0 bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM0 setting. For more detailed information see section "Multi-processor Communication Mode".

23.10.3 UCSR0B – USART0 Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C1)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	RW	RW	RW	RW	RW	RW	R	W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCIE0 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

- **Bit 6 – TXCIE0 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

- **Bit 5 – UDRIE0 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

- **Bit 4 – RXEN0 - Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD0 pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE0, DOR0 and UPE0 Flags.

- **Bit 3 – TXEN0 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXEN0 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

- **Bit 2 – UCSZ02 - Character Size**

The UCSZ02 bits combined with the UCSZ01:0 bit in UCSR0C sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

- **Bit 1 – RXB80 - Receive Data Bit 8**

RXB80 is the 9th data bit of the received character when operating with serial frames with nine data bits. The bit must be read before reading the lower 8 bits from UDR0.

- **Bit 0 – TXB80 - Transmit Data Bit 8**

TXB80 is the 9th data bit in the character to be transmitted when operating with serial frames with nine data bits. The bit must be written before writing the lower 8 bits to UDR0.

23.10.4 UCSR0C – USART0 Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($C2)	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	1	0	

- **Bit 7:6 – UMSEL01:00 - USART Mode Select**

These bits select the mode of operation of the USART0 as shown in the following table. See section "USART in SPI Mode" for a full description of the Master SPI Mode (MSPIM) operation.

Table 23-4 UMSEL0 Register Bits

Register Bits	Value	Description
UMSEL01:00	0x00	Asynchronous USART
	0x01	Synchronous USART
	0x02	Reserved
	0x03	Master SPI (MSPIM)

- **Bit 5:4 – UPM01:00 - Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and

compare it to the UPM0 setting. If a mismatch is detected, the UPE0 Flag in UCSR0A will be set.

Table 23-5 UPM0 Register Bits

Register Bits	Value	Description
UPM01:00	0x00	Disabled
	0x01	Reserved
	0x02	Enabled, Even Parity
	0x03	Enabled, Odd Parity

- **Bit 3 – USBS0 - Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 23-6 USBS0 Register Bits

Register Bits	Value	Description
USBS0	0x00	1-bit
	0x01	2-bit

- **Bit 2:1 – UCSZ01:00 - Character Size**

The UCSZ01:0 bits combined with the UCSZ02 bit in UCSR0B sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

Table 23-7 UCSZ0 Register Bits

Register Bits	Value	Description
UCSZ01:00	0	5-bit
	1	6-bit
	2	7-bit
	3	8-bit
	4	Reserved
	5	Reserved
	6	Reserved
	7	9-bit

- **Bit 0 – UCPOL0 - Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL0 bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK0).

Table 23-8 UCPOL0 Register Bits

Register Bits	Value	Description
UCPOL0	0	Rising XCKn Edge (Transmitted Data Changed), Falling XCKn Edge (Received Data Sampled)
	1	Falling XCKn Edge (Transmitted Data Changed), Rising XCKn Edge (Received Data Sampled)

23.10.5 UBRR0H – USART0 Baud Rate Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($C5)	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	UBRR0H
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR0 is a 12-bit register which contains the USART baud rate. The UBRR0H contains the four most significant bits, and the UBRR0L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR0L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:4 – Res3:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3:0 – UBRR11:8 - USART Baud Rate Register**

These bits represent bits [11:8] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.6 UBRR0L – USART0 Baud Rate Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($C4)	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRR0L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR0 is a 12-bit register which contains the USART baud rate. The UBRR0H contains the four most significant bits, and the UBRR0L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR0L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:0 – UBRR7:0 - USART Baud Rate Register**

These bits represent bits [7:0] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.7 UDR1 – USART1 I/O Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($CE)	UDR17:10								UDR1
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR1. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR1 Register location. Reading the UDR1 Register location will return the contents of the

Receive Data Buffer Register (RXB). For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver. The transmit buffer can only be written when the UDRE1 Flag in the UCSR1A Register is set. Data written to UDR1 when the UDRE1 Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD1 pin. The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

- **Bit 7:0 – UDR17:10 - USART I/O Data Register**

23.10.8 UCSR1A – USART1 Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C8)	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	UCSR1A
Read/Write	R	RW	R	R	R	R	RW	RW	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC1 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC1 bit will become zero. The RXC1 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE1 bit).

- **Bit 6 – TXC1 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR1). The TXC1 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC1 Flag can generate a Transmit Complete interrupt (see description of the TXCIE1 bit).

- **Bit 5 – UDRE1 - USART Data Register Empty**

The UDRE1 Flag indicates if the transmit buffer (UDR1) is ready to receive new data. If UDRE1 is one, the buffer is empty, and therefore ready to be written. The UDRE1 Flag can generate a Data Register Empty interrupt (see description of the UDRIE1 bit). UDRE1 is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE1 - Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR1) is read. The FE1 bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR1A.

- **Bit 3 – DOR1 - Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register and a new start bit is detected. This bit is valid until the receive buffer (UDR1) is read. Always set this bit to zero when writing to UCSR1A.

- **Bit 2 – UPE1 - USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM11 = 1). This bit is valid until the receive buffer (UDR1) is read. Always set this bit to zero when writing to UCSR1A.

- **Bit 1 – U2X1 - Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation. Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM1 - Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM1 bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM1 setting. For more detailed information see section "Multi-processor Communication Mode".

23.10.9 UCSR1B – USART1 Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C9)	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	UCSR1B
Read/Write	RW	RW	RW	RW	RW	RW	R	W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCIE1 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC1 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC1 bit in UCSR1A is set.

- **Bit 6 – TXCIE1 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC1 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC1 bit in UCSR1A is set.

- **Bit 5 – UDRIE1 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE1 Flag. A Data Register Empty interrupt will be generated only if the UDRIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE1 bit in UCSR1A is set.

- **Bit 4 – RXEN1 - Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD1 pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE1, DOR1 and UPE1 Flags.

- **Bit 3 – TXEN1 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD1 pin when enabled. The disabling of the Transmitter (writing TXEN1 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD1 port.

- **Bit 2 – UCSZ12 - Character Size**

The UCSZ12 bits combined with the UCSZ11:0 bit in UCSR1C sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

- **Bit 1 – RXB81 - Receive Data Bit 8**

RXB81 is the 9th data bit of the received character when operating with serial frames with nine data bits. The bit must be read before reading the lower 8 bits from UDR1.

- **Bit 0 – TXB81 - Transmit Data Bit 8**

TXB81 is the 9th data bit in the character to be transmitted when operating with serial frames with nine data bits. The bit must be written before writing the lower 8 bits to UDR1.

23.10.10 UCSR1C – USART1 Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($CA)	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	UCSR1C
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7:6 – UMSEL11:10 - USART Mode Select**

These bits select the mode of operation of the USART1 as shown in the following table. See section "USART in SPI Mode" for a full description of the Master SPI Mode (MSPIM) operation.

Table 23-9 UMSEL1 Register Bits

Register Bits	Value	Description
UMSEL11:10	0x00	Asynchronous USART
	0x01	Synchronous USART
	0x02	Reserved
	0x03	Master SPI (MSPIM)

- **Bit 5:4 – UPM11:10 - Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM1 setting. If a mismatch is detected, the UPE1 Flag in UCSR1A will be set.

Table 23-10 UPM1 Register Bits

Register Bits	Value	Description
UPM11:10	0x00	Disabled
	0x01	Reserved
	0x02	Enabled, Even Parity
	0x03	Enabled, Odd Parity

- **Bit 3 – USBS1 - Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 23-11 USBS1 Register Bits

Register Bits	Value	Description
USBS1	0x00	1-bit
	0x01	2-bit

- **Bit 2:1 – UCSZ11:10 - Character Size**

The UCSZ11:0 bits combined with the UCSZ12 bit in UCSR1B sets the number of data bits (Character Size) in the frame that the Receiver and Transmitter use.

Table 23-12 UCSZ1 Register Bits

Register Bits	Value	Description
UCSZ11:10	0	5-bit
	1	6-bit
	2	7-bit
	3	8-bit
	4	Reserved
	5	Reserved
	6	Reserved
	7	9-bit

- **Bit 0 – UCPOL1 - Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL1 bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK1).

Table 23-13 UCPOL1 Register Bits

Register Bits	Value	Description
UCPOL1	0	Rising XCKn Edge (Transmitted Data Changed), Falling XCKn Edge (Received Data Sampled)
	1	Falling XCKn Edge (Transmitted Data Changed), Rising XCKn Edge (Received Data Sampled)

23.10.11 UBRR1H – USART1 Baud Rate Register High Byte

Bit	7	6	5	4	3	2	1	0	
NA ($CD)	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	UBRR1H
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR1 is a 12-bit register which contains the USART baud rate. The UBRR1H contains the four most significant bits, and the UBRR1L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR1L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:4 – Res3:0 - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 3:0 – UBRR11:8 - USART Baud Rate Register**

These bits represent bits [11:8] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.10.12 UBRR1L – USART1 Baud Rate Register Low Byte

Bit	7	6	5	4	3	2	1	0	
NA ($CC)	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRR1L
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

UBRR1 is a 12-bit register which contains the USART baud rate. The UBRR1H contains the four most significant bits, and the UBRR1L contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRR1L will trigger an immediate update of the baud rate prescaler.

- **Bit 7:0 – UBRR7:0 - USART Baud Rate Register**

These bits represent bits [7:0] of the Baud Rate Register. Sample values for commonly used clock frequencies can be found in section "Examples of Baud Rate Setting".

23.11 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 23-14 below to Table 23-16 on page 367. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 353). The error values are calculated using the following equation:

$$Error[\%] = \left(\frac{BaudRate_{Closest\ Match}}{BaudRate} - 1 \right) \cdot 100\%$$

Table 23-14. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	fosc = 1.8432 MHz				fosc = 2.0000 MHz				fosc = 3.6864 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
4800	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
9600	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
14.4k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
19.2k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
28.8k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
38.4k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%

Baud Rate (bps)	fosc = 1.8432 MHz				fosc = 2.0000 MHz				fosc = 3.6864 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
57.6k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
76.8k	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%	2	0.0%	5	0.0%
115.2k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
230.4k	-	-	0	0.0%	-	-	-	-	0	0.0%	1	0.0%
250k	-	-	-	-	-	-	0	0.0%%	0	-7.8%	1	-7.8%
Max. (1)	115.2 kbps		230.4 kbps		125 kbps		250 kbps		230.4 kbps		460.8 kbps	

Notes: 1. UBRR = 0, Error = 0.0%

Table 23-15. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	fosc = 4.0000 MHz				fosc = 7.3728 MHz				fosc = 8.0000 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	103	0.2%	207	0.2%	191	0.0%	383	0.0%	207	0.2%	416	-0.1%
4800	51	0.2%	103	0.2%	95	0.0%	191	0.0%	103	0.2%	207	0.2%
9600	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
14.4k	16	2.1%	34	-0.8%	31	0.0%	63	0.0%	34	-0.8%	68	0.6%
19.2k	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
28.8k	8	-3.5%	16	2.1%	15	0.0%	31	0.0%	16	2.1%	34	-0.8%
38.4k	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
57.6k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
76.8k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
115.2k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
230.4k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
250k	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%	1	0.0%	3	0.0%
0.5M	-	-	0	0.0%	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%
1M	-	-	-	-	-	-	0	-7.8%	-	-	0	0.0%
Max. (1)	250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps		0.5 Mbps		1 Mbps	

Notes: 1. UBRR = 0, Error = 0.0%

Table 23-16. Examples of UBRR*n* Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	fosc = 11.0592 MHz				fosc = 14.7456 MHz				fosc = 16.0000 MHz			
	U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1		U2X*n* = 0		U2X*n* = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	287	0.0%	575	0.0%	383	0.0%	767	0.0%	416	-0.1%	832	0.0%
4800	143	0.0%	287	0.0%	191	0.0%	383	0.0%	207	0.2%	416	-0.1%
9600	71	0.0%	143	0.0%	95	0.0%	191	0.0%	103	0.2%	207	0.2%
14.4k	47	0.0%	95	0.0%	63	0.0%	127	0.0%	68	0.6%	138	-0.1%
19.2k	35	0.0%	71	0.0%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
28.8k	23	0.0%	47	0.0%	31	0.0%	63	0.0%	34	-0.8%	68	0.6%
38.4k	17	0.0%	35	0.0%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
57.6k	11	0.0%	23	0.0%	15	0.0%	31	0.0%	16	2.1%	34	-0.8%
76.8k	8	0.0%	17	0.0%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
115.2k	5	0.0%	11	0.0%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
230.4k	2	0.0%	5	0.0%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
250k	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%	3	0.0%	7	0.0%
0.5M	-	-	2	-7.8%	1	-7.8%	3	-7.8%	1	0.0%	3	0.0%
1M	-	-	-	-	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%
Max. (1)	691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps		1 Mbps		2 Mbps	

Notes: 1. UBRR = 0, Error = 0.0%

24 USART in SPI Mode

The Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation. The Master SPI Mode (MSPIM) has the following features:

- **Full duplex, three-wire synchronous data transfer**
- **Master operation**
- **Supports all four SPI modes of operation (mode 0, 1, 2, and 3)**
- **LSB first or MSB first data transfer (configurable data order)**
- **Queued operation (double buffered)**
- **High resolution baud rate generator**
- **High speed operation ($f_{XCK,MAX} = f_{CK}/2$)**
- **Flexible interrupt generation**

24.1 Overview

Setting both UMSELn1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

24.2 USART MSPIM vs. SPI

The ATmega128RFA1 USART in MSPIM mode is fully compatible with the ATmega128RFA1 SPI regarding:

- Master mode timing diagram.
- The UCPOLn bit functionality is identical to the SPI CPOL bit.
- The UCPHAn bit functionality is identical to the SPI CPHA bit.
- The UDORDn bit functionality is identical to the SPI DORD bit.

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer.
- The USART in MSPIM mode receiver includes an additional buffer level.
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode.
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly.
- Interrupt timing is not compatible.
- Pin control differs due to the master only operation of the USART in MSPIM mode.

A comparison of the USART in MSPIM mode and the SPI pins is shown in Table 24–3 on page 373.

24.2.1 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCKn pin (DDR_XCKn) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCKn should be set up before the USART in MSPIM is enabled (i.e. TXENn and RXENn bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRRn setting can therefore be calculated using the same equations, see Table 24-1 below:

Table 24-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Synchronous Master mode	$BAUD = \frac{f_{osc}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{2\,BAUD} - 1$

Note: The Baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System Oscillator clock frequency

UBRRn Contents of the UBRRHn and UBRRLn Registers, (0-4095)

24.3 SPI Data Modes and Timing

There are four combinations of XCKn (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHAn and UCPOLn. The data transfer timing diagrams are shown in Figure 24-1 below. Data bits are shifted out and latched in on opposite edges of the XCKn signal, ensuring sufficient time for data signals to stabilize. The UCPOLn and UCPHAn functionality is summarized in Table 24-2 below. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the receiver and transmitter.

Figure 24-1. UCPHAn and UCPOLn data transfer timing diagrams

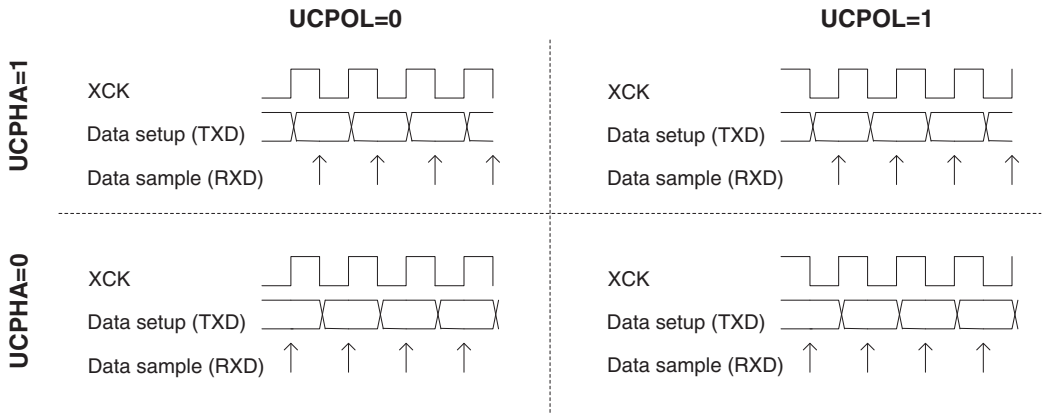


Table 24-2. UCPOLn and UCPHAn Functionality

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)

UCPOLn	UCPHAn	SPI Mode	Leading Edge	Trailing Edge
1	1	3	Setup (Falling)	Sample (Rising)

24.4 Frame Formats

A serial frame for the MSPIM is defined to be one character of 8 data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

24.4.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXCn Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example[1]

```
USART_Init:
 clr r18
 out UBRRnH,r18
 out UBRRnL,r18
 ; Setting the XCKn port pin as output, enables master mode.
 sbi XCKn_DDR, XCKn
 ; Set MSPI mode of operation and SPI data mode 0.
 ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
 out UCSRnC,r18
 ; Enable receiver and transmitter.
 ldi r18, (1<<RXENn)|(1<<TXENn)
 out UCSRnB,r18 ; Set baud rate.
 ; IMPORTANT:
 ; The Baud Rate must be set after the transmitter is enabled!
 out UBRRnH, r17
 out UBRRnL, r18
 ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  UBRRn = 0;
  /* Setting the XCKn port pin as output, enables master mode. */
  XCKn_DDR |= (1<<XCKn);
  /* Set MSPI mode of operation and SPI data mode 0. */
  UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
  /* Enable receiver and transmitter. */
  UCSRnB = (1<<RXENn)|(1<<TXENn);
  /* Set baud rate. */
  /* IMPORTANT:     */
  /* The Baud Rate must be set after the transmitter is enabled */
  UBRRn = baud;
}
```

Note: 1. See "About Code Examples" on page 7

24.5 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXENn bit in the UCSRnB register is set to one. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXENn bit in the UCSRnB register to one. When the receiver is enabled, the normal pin operation of the RxDn pin is overridden and given the function as the Receiver's serial input. The XCKn will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDRn I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to

UDRn is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDRn register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDRn is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDREn) Flag and the Receive Complete (RXCn) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register r16 and the data received will be available in the same register (r16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDREn Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

Assembly Code Example[1]

```
USART_MSPIM_Transfer:
 ; Wait for empty transmit buffer
 sbis UCSRnA, UDREn
 rjmp USART_MSPIM_Transfer
 ; Put data (r16) into buffer, sends the data
 out UDRn,r16
 ; Wait for data to be received
USART_MSPIM_Wait_RXCn:
 sbis UCSRnA, RXCn
 rjmp USART_MSPIM_Wait_RXCn
 ; Get and return received data from buffer
 in r16, UDRn
 ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRnA & (1<<UDREn)) );
  /* Put data into buffer, sends the data */
  UDRn = data;
  /* Wait for data to be received */
  while ( !(UCSRnA & (1<<RXCn)) );
  /* Get and return received data from buffer */
  return UDRn;
}
```

Notes: 1. See "About Code Examples" on page 7

24.5.1 Transmitter and Receiver Flags and Interrupts

The RXCn, TXCn, and UDREn flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and are always read as zero.

24.5.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

24.6 USART MSPIM Register Description

The following section describes the registers used for SPI operation using the USART.

24.6.1 UDRn – USART MSPIM I/O Data Register

The function and bit description of the USART data register (UDR*n*) in MSPI mode is identical to normal USART operation. See "UDR0 – USART0 I/O Data Register" on page 356.

24.6.2 UBRRnL and UBRRnH – USART MSPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode is identical to normal USART operation. See "UBRR0L – USART0 Baud Rate Register Low Byte" on page 360 and "UBRR0H – USART0 Baud Rate Register High Byte" on page 360.

Table 24–3. Comparison of USART in MSPIM mode and SPI pins

USART_MSPIM	SPI	Comment
TxDn	MOSI	Master Out only
RxDn	MISO	Master In only
XCKn	SCK	(Functional identical)
(N/A)	SS	Not supported by USART in MSPIM

24.6.3 UCSR0A – USART0 MSPIM Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C0)	RXC0	TXC0	UDRE0						UCSR0A
Read/Write	R	RW	R						
Initial Value	0	0	0						

- **Bit 7 – RXC0 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

- **Bit 6 – TXC0 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

- **Bit 5 – UDRE0 - USART Data Register Empty**

The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

24.6.4 UCSR0B – USART0 MSPIM Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C1)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0				UCSR0B
Read/Write	RW	RW	RW	RW	RW				
Initial Value	0	0	1	0	0				

- **Bit 7 – RXCIE0 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC0 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

- **Bit 6 – TXCIE0 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC0 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

- **Bit 5 – UDRIE0 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

- **Bit 4 – RXEN0 - Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxD0 pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXEN0=1 and TXEN0=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXEN0 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXEN0 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

24.6.5 UCSR0C – USART0 MSPIM Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($C2)						UDORD0	UCPHA0	UCPOL0	UCSR0C
Read/Write						RW	RW	RW	
Initial Value						1	1	0	

- **Bit 2 – UDORD0 - Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to section "Frame Formats" for details.

- **Bit 1 – UCPHA0 - Clock Phase**

The UCPHA0 bit setting determines if data is sampled on the leading (first) or tailing (last) edge of XCK0. Refer to the section "SPI Data Modes and Timing" for details.

- **Bit 0 – UCPOL0 - Clock Polarity**

The UCPOL0 bit sets the polarity of the XCK0 clock. The combination of the UCPOL0 and UCPHA0 bit settings determine the timing of the data transfer. Refer to the section "SPI Data Modes and Timing" for details.

24.6.6 UCSR1A – USART1 MSPIM Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($C8)	RXC1	TXC1	UDRE1						UCSR1A
Read/Write	R	RW	R						
Initial Value	0	0	0						

- **Bit 7 – RXC1 - USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC1 bit will become zero. The RXC1 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE1 bit).

- **Bit 6 – TXC1 - USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR1). The TXC1 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC1 Flag can generate a Transmit Complete interrupt (see description of the TXCIE1 bit).

- **Bit 5 – UDRE1 - USART Data Register Empty**

The UDRE1 Flag indicates if the transmit buffer (UDR1) is ready to receive new data. If UDRE1 is one, the buffer is empty, and therefore ready to be written. The UDRE1 Flag can generate a Data Register Empty interrupt (see description of the UDRIE1 bit). UDRE1 is set after a reset to indicate that the Transmitter is ready.

24.6.7 UCSR1B – USART1 MSPIM Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($C9)	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1				UCSR1B
Read/Write	RW	RW	RW	RW	RW				
Initial Value	0	0	1	0	0				

- **Bit 7 – RXCIE1 - RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC1 Flag. A USART Receive Complete interrupt will be generated only if the RXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC1 bit in UCSR1A is set.

- **Bit 6 – TXCIE1 - TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC1 Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC1 bit in UCSR1A is set.

- **Bit 5 – UDRIE1 - USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE1 Flag. A Data Register Empty interrupt will be generated only if the UDRIE1 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE1 bit in UCSR1A is set.

- **Bit 4 – RXEN1 - Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxD1 pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXEN1=1 and TXEN1=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXEN1 - Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD1 pin when enabled. The disabling of the Transmitter (writing TXEN1 to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD1 port.

24.6.8 UCSR1C – USART1 MSPIM Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($CA)						UDORD1	UCPHA1	UCPOL1	UCSR1C
Read/Write						RW	RW	RW	
Initial Value						1	1	0	

- **Bit 2 – UDORD1 - Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to section "Frame Formats" for details.

- **Bit 1 – UCPHA1 - Clock Phase**

The UCPHA1 bit setting determines if data is sampled on the leading (first) or tailing (last) edge of XCK1. Refer to the section "SPI Data Modes and Timing" for details.

- **Bit 0 – UCPOL1 - Clock Polarity**

The UCPOL1 bit sets the polarity of the XCK1 clock. The combination of the UCPOL1 and UCPHA1 bit settings determine the timing of the data transfer. Refer to the section "SPI Data Modes and Timing" for details.

25 2-wire Serial Interface

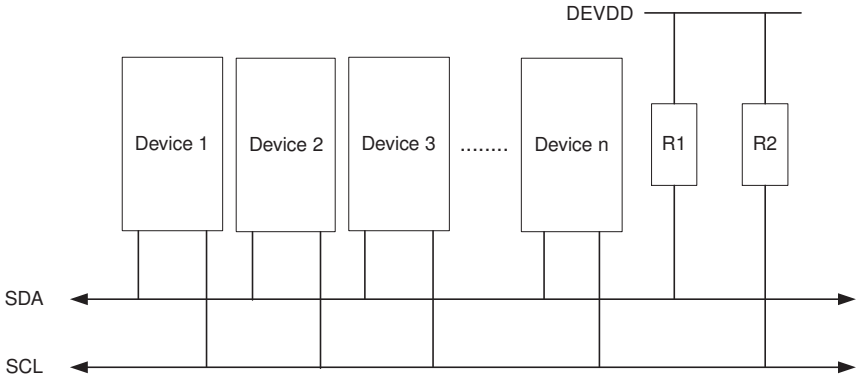
25.1 Features

- **Simple yet powerful and flexible communication interface, only two bus lines needed**
- **Both master and slave operation supported**
- **Device can operate as transmitter or receiver**
- **7-bit address space allows up to 128 different slave addresses**
- **Multi-master arbitration support**
- **Up to 400 kHz data transfer speed**
- **Slew-rate limited output drivers**
- **Noise suppression circuitry rejects spikes on bus lines**
- **Fully programmable slave address with general call support**
- **Address recognition causes wake-up when microcontroller is in sleep mode**

25.2 2-wire Serial Interface Bus Definition

The 2-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 25-1. TWI Bus Interconnection



25.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 25-1. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The Power Reduction TWI bit, PRTWI bit in "PRR0 – Power Reduction Register0" on page 167 must be written to zero to enable the 2-wire Serial Interface.

25.2.2 Electrical Interconnection

As depicted in Figure 25-1 on page 377, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices trim-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

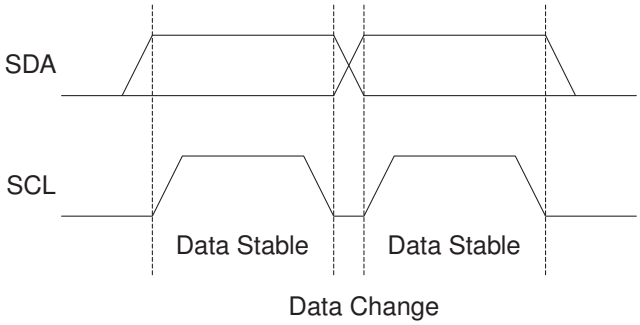
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400 pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in "2-wire Serial Interface Characteristics" on page 503. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

25.3 Data Transfer and Frame Format

25.3.1 Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

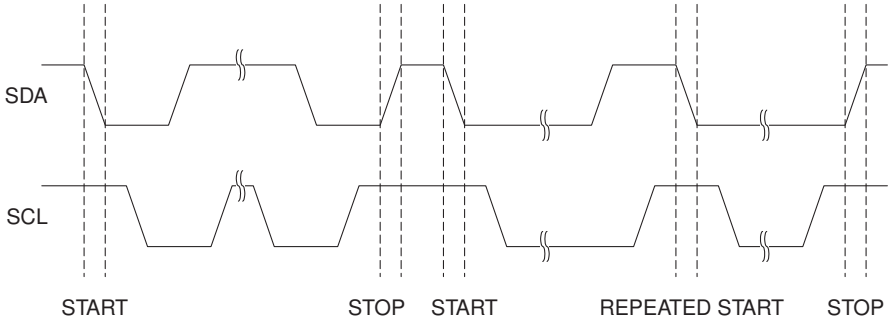
Figure 25-2. Data Validity



25.3.2 START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without relinquishing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.

Figure 25-3. START, REPEATED START and STOP conditions



25.3.3 Address Packet Format

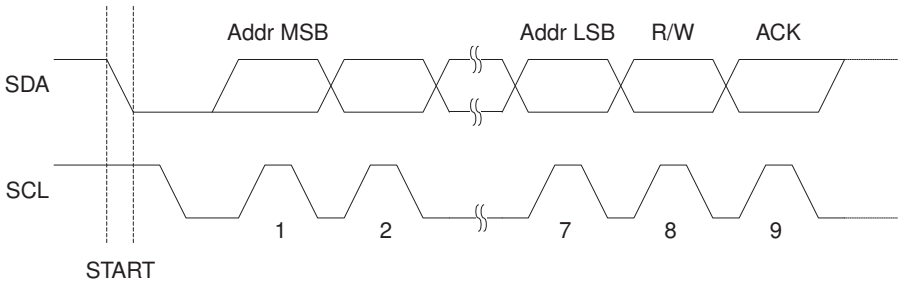
All address packets transmitted on the TWI bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 25-4. Address Packet Format

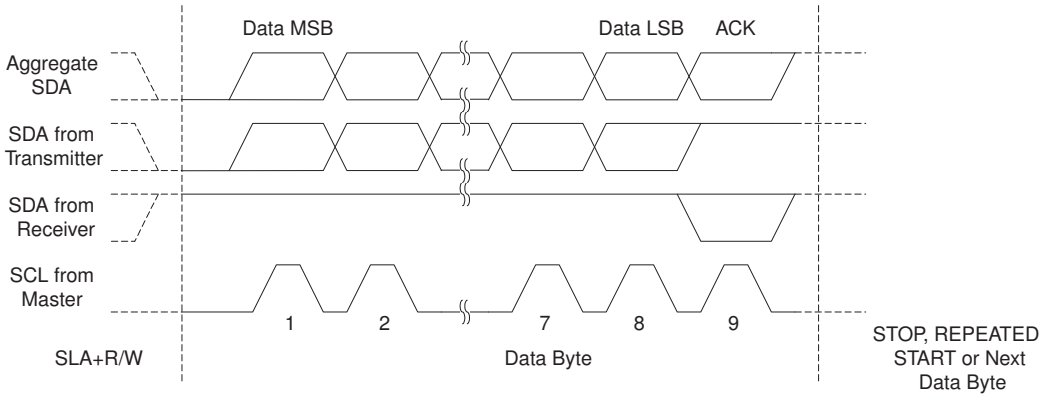


25.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signaled by the Receiver

pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signaled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 25-5. Data Packet Format

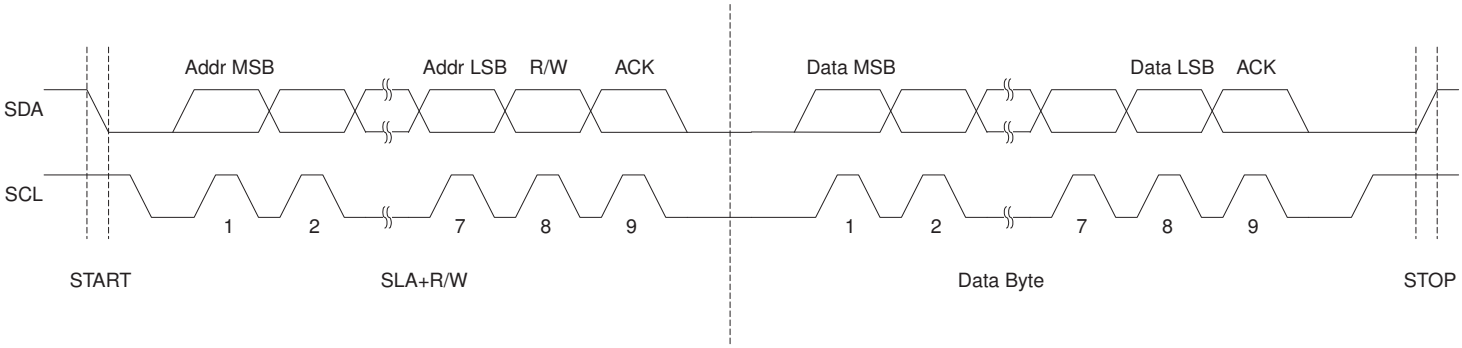


25.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the Wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 25-6 below shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 25-6. Typical Data Transmission



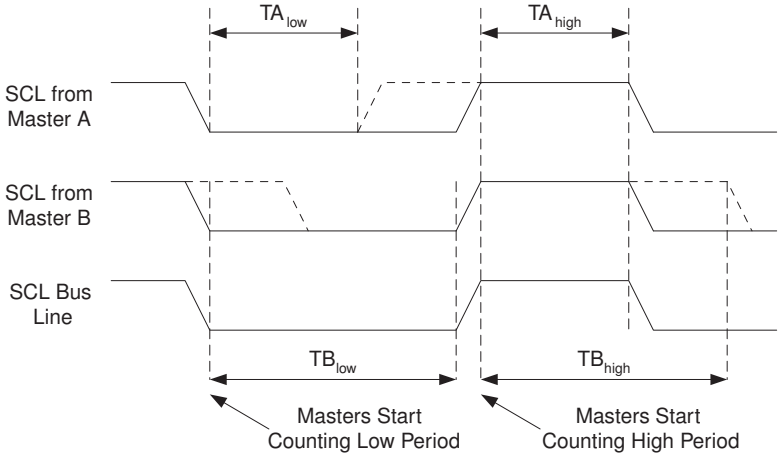
25.4 Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

Figure 25-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

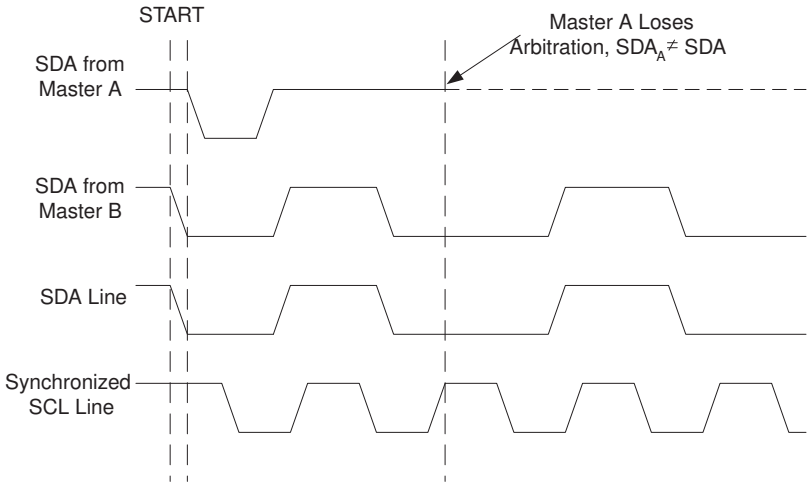
Note that arbitration is not allowed between:

- A REPEATED START condition and a data bit.
- A STOP condition and a data bit.
- A REPEATED START and a STOP condition.

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions

must contain the same number of data packets, otherwise the result of the arbitration is undefined.

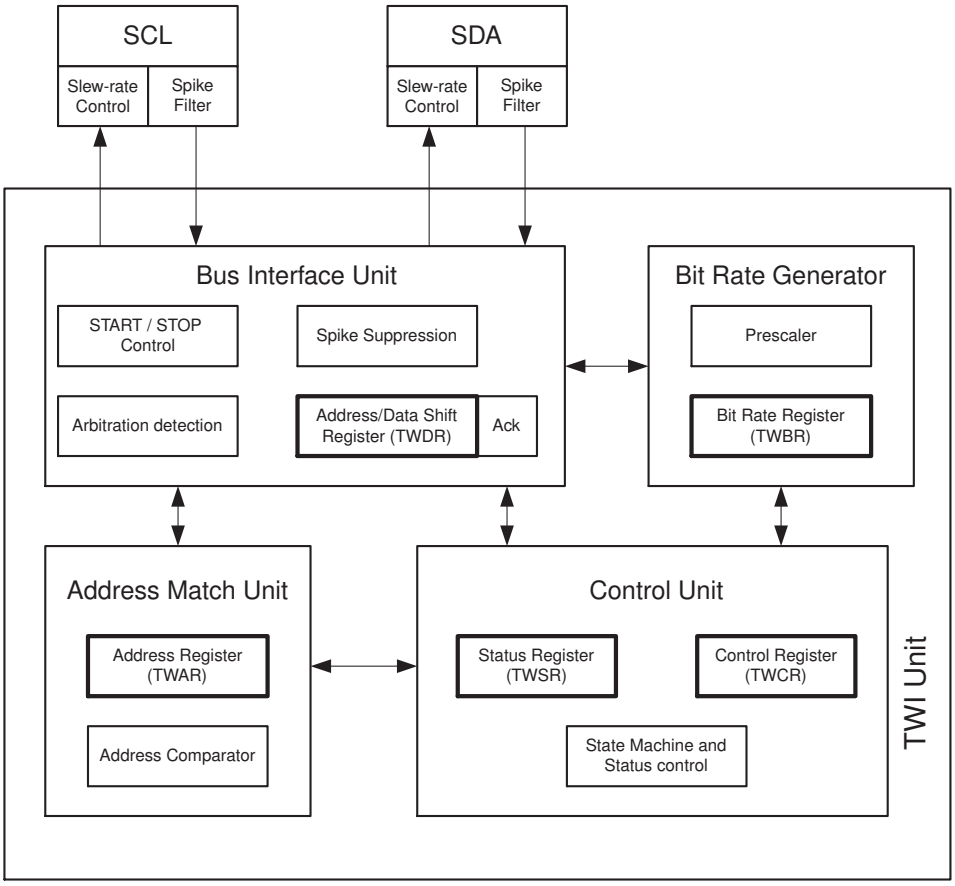
Figure 25-8. Arbitration Between Two Masters



25.5 Overview of the TWI Module

The TWI module is comprised of several sub-modules, as shown in Figure 25-9 below. All registers drawn in a thick line are accessible through the AVR data bus.

Figure 25-9. Overview of the TWI Module



25.5.1 SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

25.5.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$SCL\ frequency = \frac{CPU\ Clock\ frequency}{16 + 2(TWBR) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register.
- TWPS = Value of the prescaler bits in the TWI Status Register.

Note that pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See in "2-wire Serial Interface Characteristics" on page 503 for value of pull-up resistor.

25.5.3 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the AVR MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

25.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to

compare addresses even if the AVR MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master. If another interrupt (e.g., INT0) occurs during TWI Power-down address match and wakes up the CPU, the TWI aborts operation and return to it's idle state. If this cause any problems, ensure that TWI Address Match is the only enabled interrupt when entering Power-down.

25.5.5 Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition.
- After the TWI has transmitted SLA+R/W.
- After the TWI has transmitted an address byte.
- After the TWI has lost arbitration.
- After the TWI has been addressed by own slave address or general call.
- After the TWI has received a data byte.
- After a STOP or REPEATED START has been received while still addressed as a Slave.
- When a bus error has occurred due to an illegal START or STOP condition.

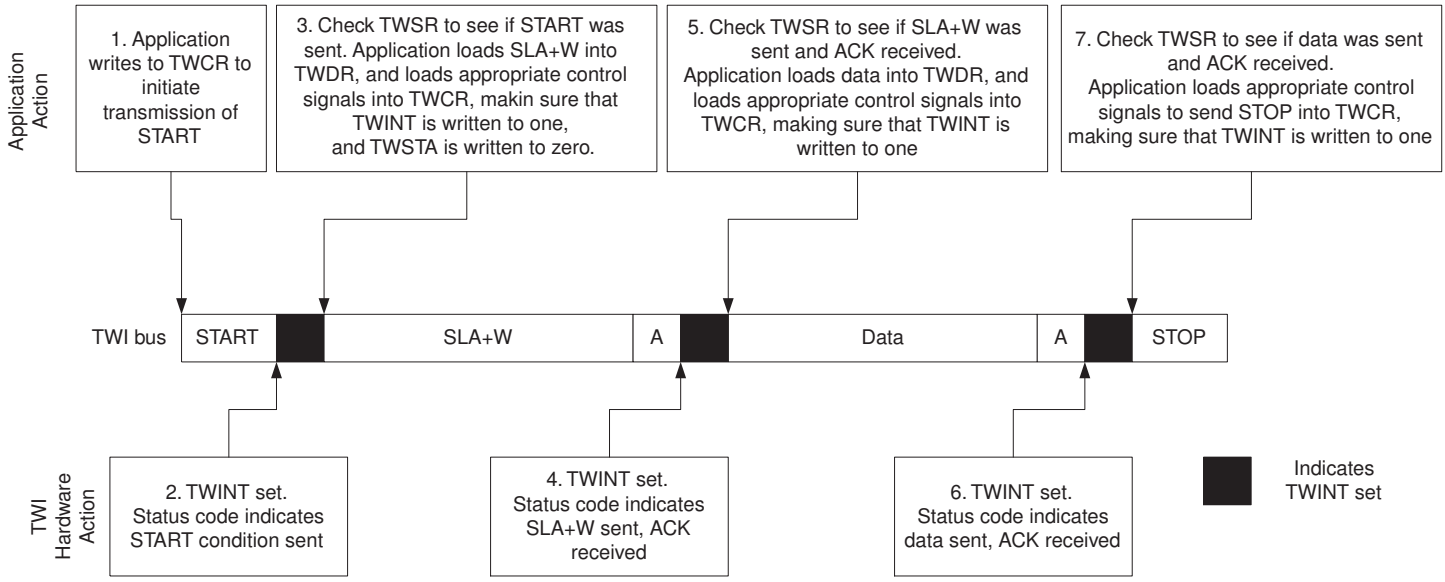
25.6 Using the TWI

The ATmega128RFA1 TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 25-10 on page 385 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 25-10. Interfacing the Application to the TWI in a Typical Transmission



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.
4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT

clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.

6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.

Table 25-2. Code example

	Assembly Code Example	C Example	Comments
1	`ldi r16,(1<<TWINT)|(1<<TWSTA)|(1<<TWEN)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)`	Send START condition

2	`wait1:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait1`
		`while (!(TWCR & (1<<TWINT)));`	Wait for TWINT Flag set. This indicates that the START condition has been transmitted

3	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, START`
	`brne ERROR`
		`if ((TWSR & 0xF8) != START)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR

	`ldi r16, SLA_W`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT)|(1<<TWEN)`
	`out TWCR, r16`
		`TWDR = SLA_W;`
		`TWCR = (1<<TWINT)|(1<<TWEN);`
			Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address

	Assembly Code Example	C Example	Comments
4	wait2: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait2	while (!(TWCR & (1<<TWINT))); 	Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.
5	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_SLA_ACK **brne** ERROR	if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
	ldi r16, DATA **out** TWDR, r16 **ldi** r16, (1<<TWINT)|(1<<TWEN) **out** TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN);	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data
6	wait3: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait3	while (!(TWCR & (1<<TWINT))); 	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
7	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_DATA_ACK **brne** ERROR	if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
	ldi r16,(1<<TWINT)|(1<<TWEN)| (1<<TWSTO) **out** TWCR, r16	TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);	Transmit STOP condition

25.7 Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S:	START condition	**Rs:**	REPEATED START condition
R:	Read bit (high level at SDA)	**W:**	Write bit (low level at SDA)
Data:	8-bit data byte	**P:**	STOP condition
SLA:	Slave Address	**A:**	Acknowledge bit (low level at SDA)
$\overline{A}$:	Not acknowledge bit (high level at SDA)		

In Figure 25-12 on page 389 to Figure 25-18 on page 399 circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

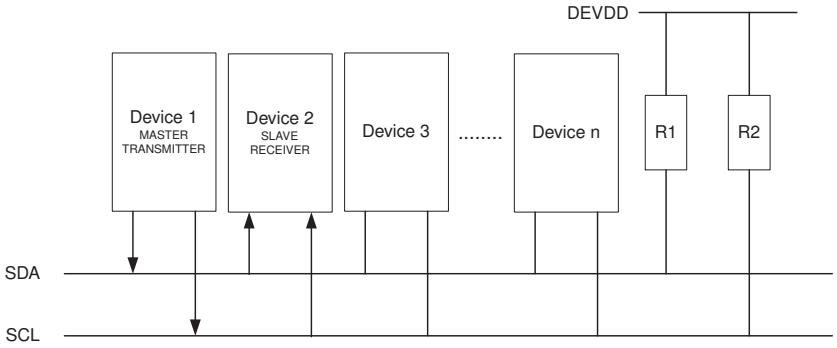
When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and

details of the following serial transfer are given in Table 25-3 on page 390 to Table 25-6 on page 398. Note that the prescaler bits are masked to zero in these tables.

25.7.1 Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 25-11 below). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-11. Data Transfer in Master Transmitter Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

TWEN must be set to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 25-3 on page 390). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in Table 25-3 on page 390.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	X	1	0	X	1	0	X	

After a REPEATED START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Figure 25-12. Formats and States in the Master Transmitter Mode

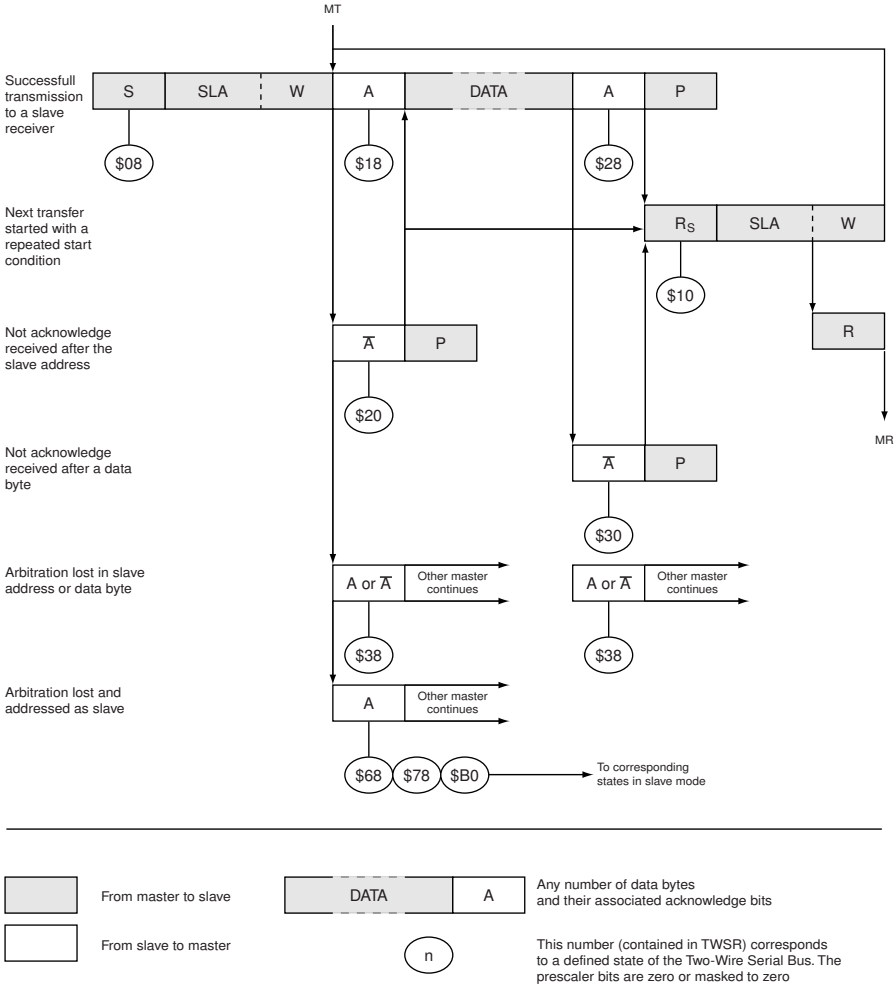


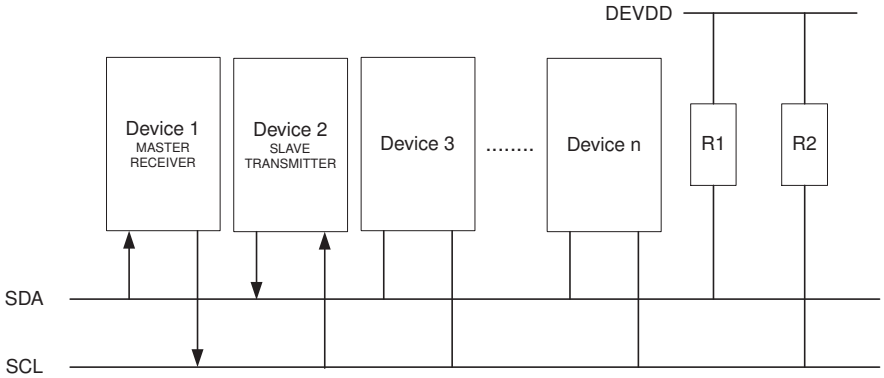
Table 25-3. Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte o	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be rese
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus be-comes free

25.7.2 Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (for Slave see Figure 25-13 on page 391). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-13. Data Transfer in Master Receiver Mode



A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

TWEN must be written to one to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 25-4 on page 392). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in Table 25-4 on page 392. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

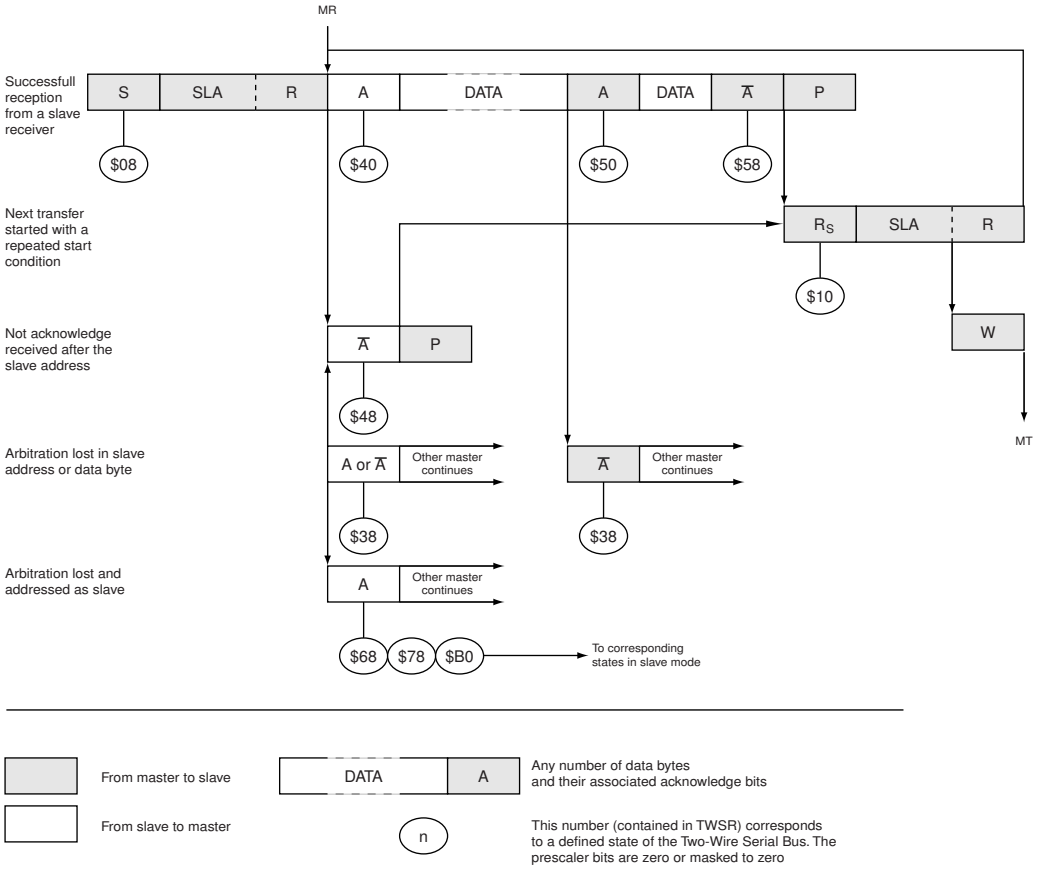
After a repeated START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated

START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 25-4. Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hard-ware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STD	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOTACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

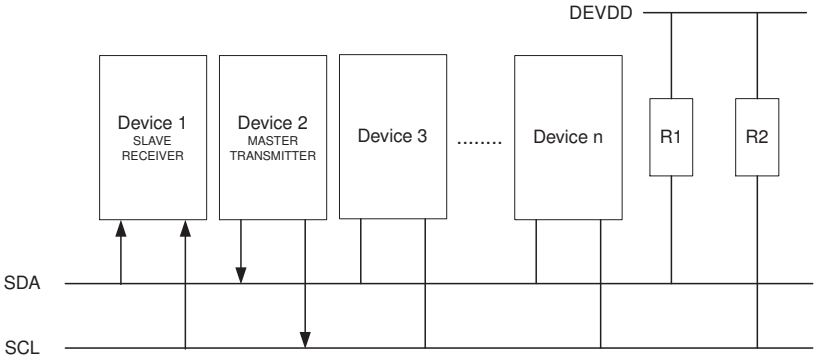
Figure 25-14. Formats and States in the Master Receiver Mode



25.7.3 Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 25-15 below). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-15. Data transfer in Slave Receiver mode



To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 25-5 below. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these Sleep modes.

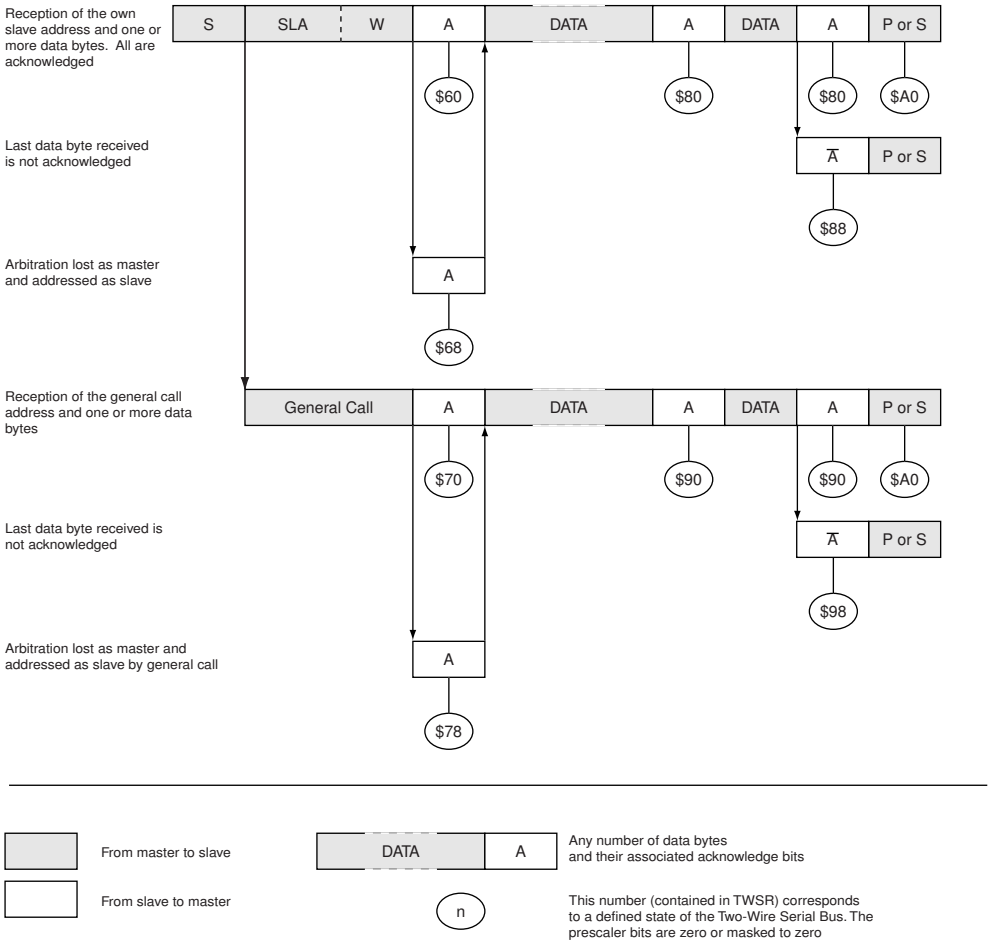
Table 25-5. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned

0x78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or No TWDR action	X X	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or Read data byte	X X	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte or Read data byte	0 0 1 1	0 0 0 0	1 1 1 1	0 1 0 1	Switched to the not addressed Slave mode; no recognition of own SLA or GCA Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0x90	Previously addressed with general call; data has been re-ceived; ACK has been returned	Read data byte or Read data byte	X X	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte or Read data byte	0 0 1 1	0 0 0 0	1 1 1 1	0 1 0 1	Switched to the not addressed Slave mode; no recognition of own SLA or GCA Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1” Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

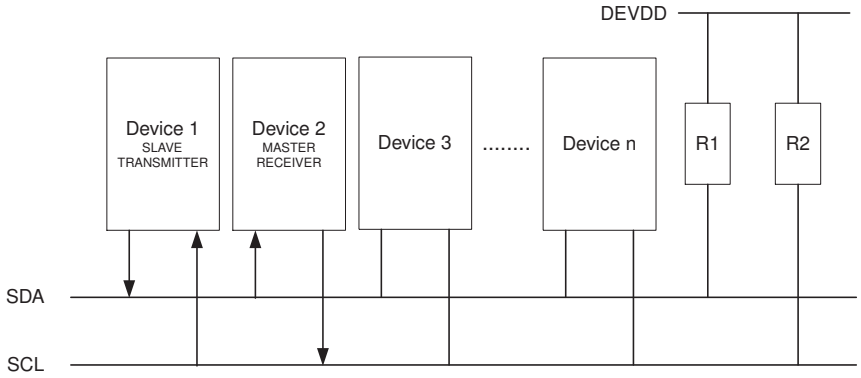
Figure 25-16. Formats and States in the Slave Receiver Mode



25.7.4 Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 25-17 on page 397). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 25-17. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

The upper seven bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 25-6 on page 398. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part

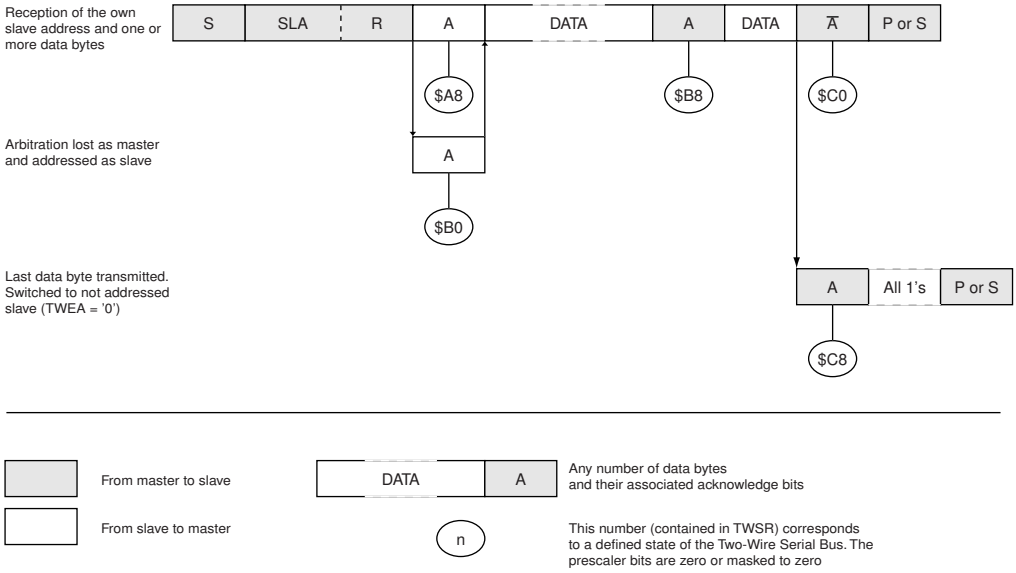
will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the AVR clocks running as normal. Observe that if the AVR is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

Table 25-6. Status Code for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STD	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xC8	Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 25-18. Formats and States in the Slave Transmitter Mode



25.7.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 25-7 below.

Status 0xF8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status 0x00 indicates that a bus error has occurred during a 2-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 25-7. Miscellaneous States

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hard-ware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xF8	No relevant state information available	TWDR action	No TWCR action				Wait or proceed current transfer
0x00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condi-tion is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

25.7.6 Combining Several TWI Modes

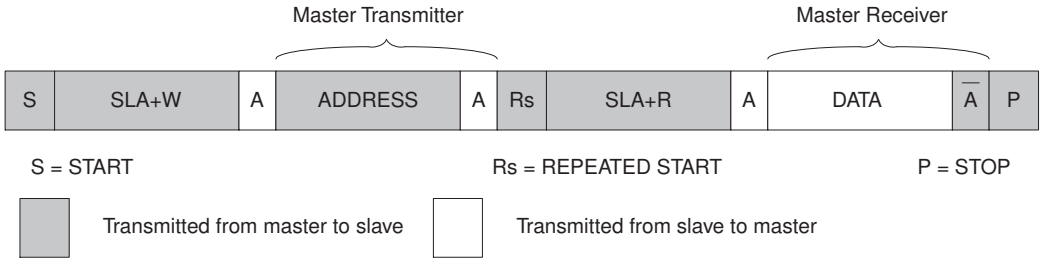
In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.

3. The reading must be performed.
4. The transfer must be finished.

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomic operation. If this principle is violated in a multi-master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

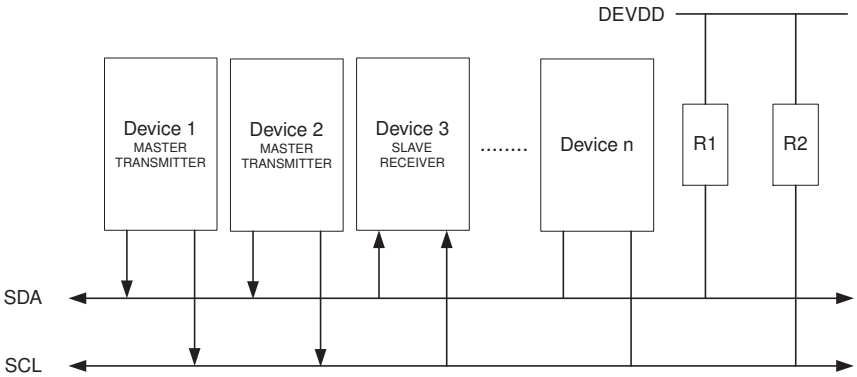
Figure 25-19. Combining Several TWI Modes to Access a Serial EEPROM



25.8 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 25-20. An Arbitration Example



Several different scenarios may arise during arbitration, as described below:

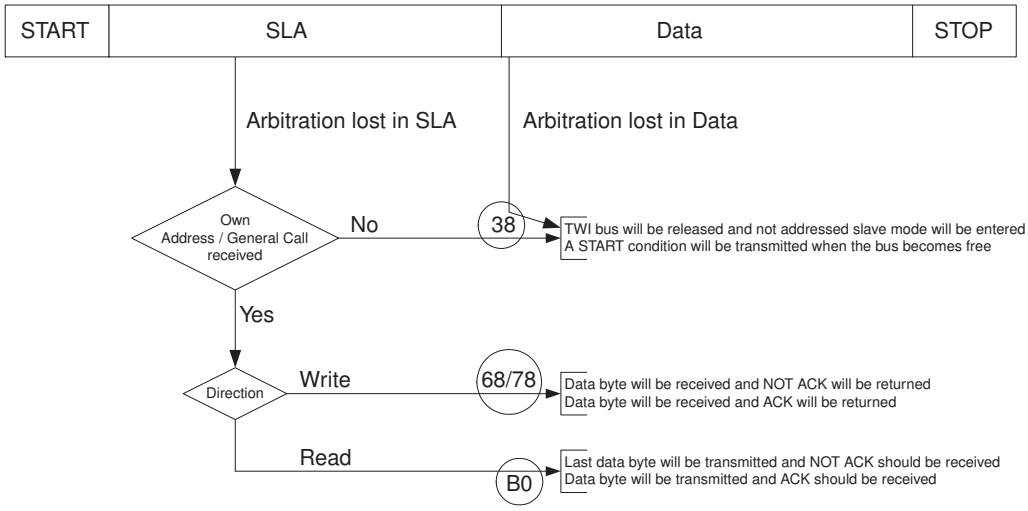
- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention.
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a one on SDA while another Master outputs a zero

will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

- Two or more masters are accessing different slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in Figure 25-21 below. Possible status values are given in circles.

Figure 25-21. Possible Status Codes Caused by Arbitration



25.9 Register Description

25.9.1 TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
NA ($B8)	TWBR7:0								TWBR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency.

- **Bit 7:0 – TWBR7:0 - TWI Bit Rate Register Value**

The TWBR register selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See section "Bit Rate Generator Unit" for calculating bit rates.

25.9.2 TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
NA ($BC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	Res	TWIE	TWCR
Read/Write	RW	RW	RW	RW	RW	RW	R	RW	
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are put into the TWDR. It also indicates a write collision if data writing to TWDR is attempted while the register is inaccessible.

- **Bit 7 – TWINT - TWI Interrupt Flag**

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI. So all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR) and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 6 – TWEA - TWI Enable Acknowledge Bit**

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met: 1. The devices own slave address has been received; 2. A general call has been received, while the TWGCE bit in the TWAR is set. 3. A data byte has been received in Master Receiver or Slave Receiver mode. By writing the TWEA bit to zero, the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

- **Bit 5 – TWSTA - TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a Master on the 2-wire Serial Bus. The TWI hardware checks if the bus is available and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO - TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined not-addressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC - TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN - TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O ports connected to the SCL and SDA

pins enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated regardless of any ongoing operation.

- **Bit 1 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 0 – TWIE - TWI Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

25.9.3 TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
NA ($B9)	TWS7	TWS6	TWS5	TWS4	TWS3	Res	TWPS1	TWPS0	TWSR
Read/Write	RW	RW	RW	RW	RW	R	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:3 – TWS4:0 - TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes for both transmitter and receiver mode are described in the following table. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

Table 25-8 TWS Register Bits

Register Bits	Value	Description
TWS4:0	0x00	Bus error due to illegal START or STOP condition.
	0x08	A START condition has been transmitted.
	0x10	A repeated START condition has been transmitted.
	0x18	SLA+W has been transmitted; ACK has been received.
	0x20	SLA+W has been transmitted; NOT ACK has been received.
	0x28	Data byte has been transmitted; ACK has been received.
	0x30	Data byte has been transmitted; NOT ACK has been received.
	0x38	Arbitration lost in SLA+W or data bytes (Transmitter); Arbitration lost in SLA+R or NOT ACK bit (Receiver).
	0x40	SLA+R has been transmitted; ACK has been received.
	0x48	SLA+R has been transmitted; NOT ACK has been received.
	0x50	Data byte has been received; ACK has been

Register Bits	Value	Description
		returned.
	0x58	Data byte has been received; NOT ACK has been returned.
	0x60	Own SLA+W has been received; ACK has been returned.
	0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned.
	0x70	General call address has been received; ACK has been returned.
	0x78	Arbitration lost in SLA+R/W as Master; general call address has been received; ACK has been returned.
	0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned.
	0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned.
	0x90	Previously addressed with general call; data has been received; ACK has been returned.
	0x98	Previously addressed with general call; data has been received; NOT ACK has been returned.
	0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave.
	0xA8	Own SLA+R has been received; ACK has been returned.
	0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned.
	0xB8	Data byte in TWDR has been transmitted; ACK has been received.
	0xC0	Data byte in TWDR has been transmitted; NO ACK has been received.
	0xC8	Last data byte in TWDR has been transmitted (TWEA = 0); ACK has been received.
	0xF8	No relevant state information available; TWINT = 0.

- **Bit 2 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 1:0 – TWPS1:0 - TWI Prescaler Bits**

These bits can be read and written and control the bit rate of the prescaler.

Table 25-9 TWPS Register Bits

Register Bits	Value	Description
TWPS1:0	0x00	1

Register Bits	Value	Description
	0x01	4
	0x02	16
	0x03	64

25.9.4 TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
NA ($BB)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is automatically controlled by the TWI logic. The CPU cannot access the ACK bit directly.

- **Bit 7:0 – TWD7:0 - TWI Data Register Byte**

25.9.5 TWAR – TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
NA ($BA)	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver. This register is not needed in the Master modes. In multi-master systems TWAR must be set in Masters which can be addressed as Slaves by other Masters. The LSB of TWAR is used to enable the recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bit 7:1 – TWA6:0 - TWI (Slave) Address**

These bits contain the TWI address operated as a Slave device.

- **Bit 0 – TWGCE - TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the 2-wire Serial Bus.

25.9.6 TWAMR – TWI (Slave) Address Mask Register

Bit	7	6	5	4	3	2	1	0	
NA ($BD)	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	Res	TWAMR
Read/Write	RW	RW	RW	RW	RW	RW	RW	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:1 – TWAM6:0 - TWI Address Mask**

The TWAMR can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWI Address Register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR.

- **Bit 0 – Res - Reserved Bit**

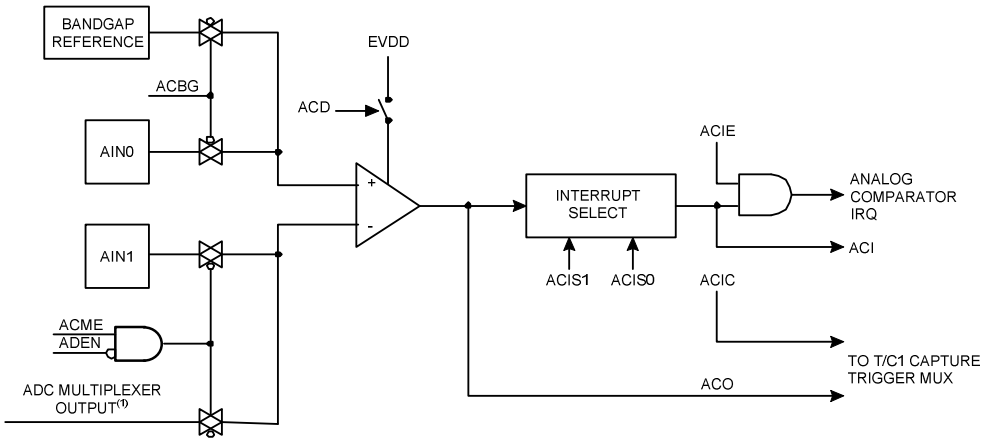
This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

26 AC – Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 26-1 below.

The Power Reduction ADC bit PRADC in PRR0 (see "PRR0 – Power Reduction Register0" on page 167) must be disabled by writing a logical zero to be able to use the ADC input multiplexer.

Figure 26-1. Analog Comparator Block Diagram



Note:
1. See Table 26-1 below.
2. Refer to Figure 1-1 on page 2 and Table 14-9 on page 197 for Analog Comparator pin placement.

26.1 Analog Comparator Multiplexed Input

It is possible to select any of the ADC7:0 pins as the negative input of the Analog Comparator. The ADC multiplexer is used to select this input and consequently the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX5 and MUX2:0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 26-1 below. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 26-1. Analog Comparator Multiplexed Input

ACME	ADEN	MUX5	MUX2:0	Analog Comparator Negative Input
0	x	x	xxx	AIN1
1	1	x	xxx	AIN1
1	0	0	000	ADC0
1	0	0	001	ADC1
1	0	0	010	ADC2
1	0	0	011	ADC3
1	0	0	100	ADC4

ACME	ADEN	MUX5	MUX2:0	Analog Comparator Negative Input
1	0	0	101	ADC5
1	0	0	110	ADC6
1	0	0	111	ADC7

26.2 Register Description

26.2.1 ACSR – Analog Comparator Control And Status Register

Bit	7	6	5	4	3	2	1	0	
$30 ($50)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	RW	RW	R	RW	RW	RW	RW	RW	
Initial Value	0	0	NA	0	0	0	0	0	

- **Bit 7 – ACD - Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG - Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage connects to the positive input of the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. When the bandgap reference is used as the input of the Analog Comparator, it will take a certain time for the voltage to stabilize. If not stabilized, the first comparison may give a wrong value. See section "Internal Voltage Reference" for details.

- **Bit 5 – ACO - Analog Compare Output**

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1-2 clock cycles.

- **Bit 4 – ACI - Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hard-ware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE - Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC - Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

- **Bit 1:0 – ACIS1:0 - Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in the following table. When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Table 26-2 ACIS Register Bits

Register Bits	Value	Description
ACIS1:0	0x00	Interrupt on Toggle
	0x01	Reserved
	0x02	Interrupt on Falling Edge
	0x03	Interrupt on Rising Edge

26.2.2 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($7B)		ACME							ADCSRB
Read/Write		RW							
Initial Value		0							

- **Bit 6 – ACME - Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer defines the negative input of the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see section "Analog Comparator Multiplexed Input".

26.2.3 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
NA ($7F)							AIN1D	AIN0D	DIDR1
Read/Write							RW	RW	
Initial Value							0	0	

- **Bit 1 – AIN1D - AIN1 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1 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 AIN1 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.

- **Bit 0 – AIN0D - AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN0 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 AIN0 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 ADC – Analog to Digital Converter

27.1 Features

- **10-bit Resolution**
- **Differential Non-Linearity is less than ± 0.5 LSB**
- **2 LSB Integral Non-Linearity**
- **3 - 240 µs Conversion Time**
- **Up to 330 kSPS (Up to 570 kSPS with 8-bit Resolution)**
- **8 Multiplexed Single Ended Input Channels**
- **11 Differential Input Channels**
- **2 Differential Input Channels with an Optional Gain of 10x and 200x**
- **Internal Linear Temperature Sensor**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{AVDD} ADC Input Voltage Range**
- **0 - V_{EVDD} Differential ADC Input Voltage Range**
- **Selectable 1.5V, 1.6V or V_{AVDD} ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceller**

The ATmega128RFA1 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows eight single-ended voltage inputs constructed from the pins of Port F. The single-ended voltage inputs refer to 0V (AVSS).

The device also supports multiple differential voltage input combinations. Two of the differential inputs (ADC1 & ADC0 and ADC3 & ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x) or 46 dB (200x) on the differential input voltage before the A/D conversion. The differential input channels are constructed of pairs out of the 8 single-ended inputs. They share a common negative terminal (ADC0, ADC1 or ADC2), while most of the other ADC inputs can be selected as the positive input terminal. If 1x or 10x gain is used, 8 bit resolution can be expected. If 200x gain is used, 6 bit resolution can be expected.

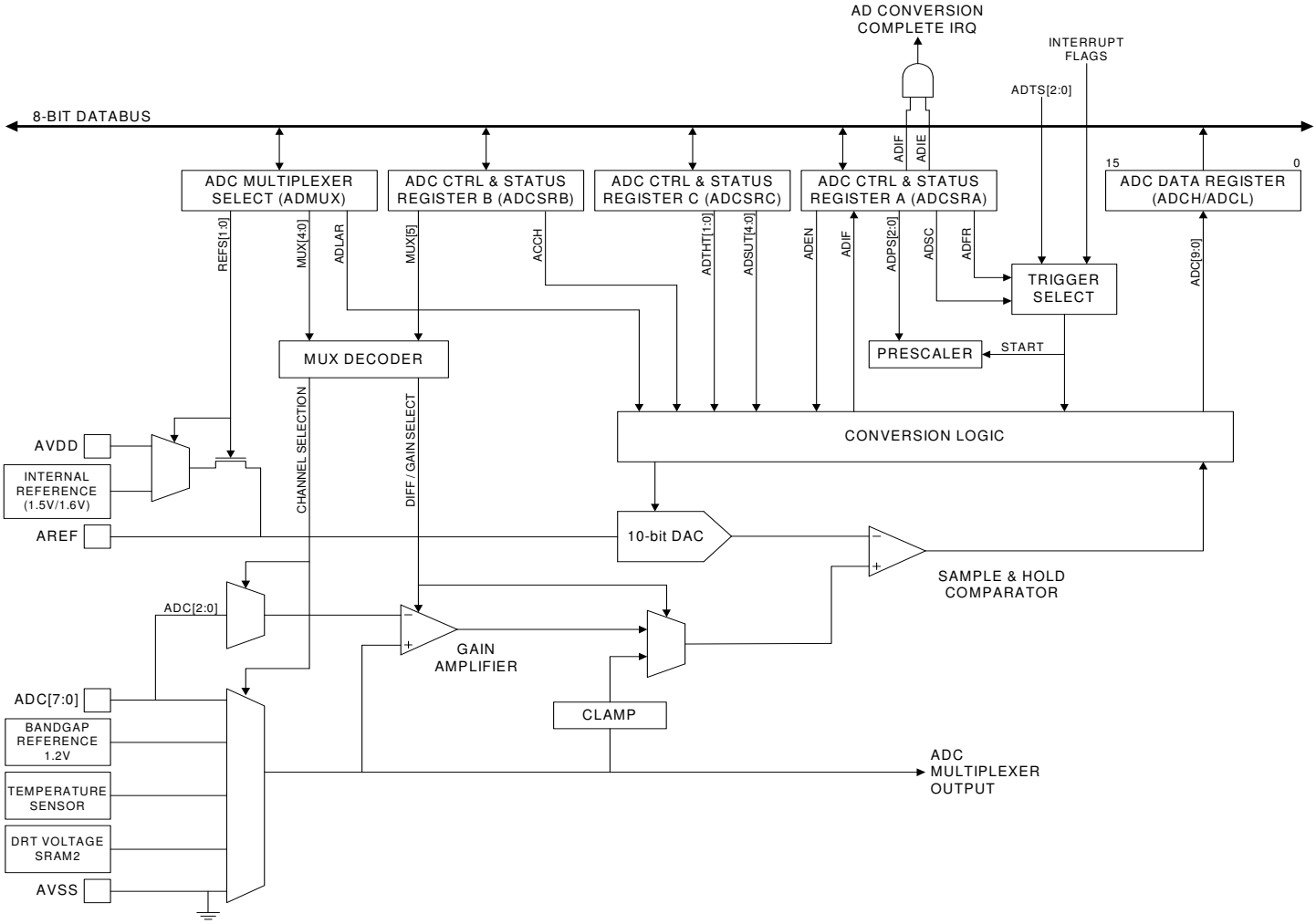
The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 27-1 on page 411.

The analog components of the ADC are supplied from the analog supply voltage AVDD. AVDD is generated from EVDD by an internal voltage generator. The logic part of the ADC is supplied from the digital supply voltage DVDD. DVDD is generated from DEVDD also by an internal voltage generator.

Internal reference voltages of nominally 1.5V, 1.6V or AVDD (1.8V) are provided on-chip. The 1.6V reference is calibrated to ± 1 LSB during manufacturing. The reference voltage can be monitored at the AREF pin. Additional de-coupling capacitance at AREF is not required. A high capacitive loading of AREF will de-stabilize the internal reference voltage generation. An external reference voltage in the range of $0 < V_{AREF,EXT} \leq V_{AVDD}$ may be used but must be supplied with a very low impedance.

The Power Reduction ADC bit, PRADC (see "PRR0 – Power Reduction Register0" on page 167) must be disabled by writing a logical zero to enable the ADC.

Figure 27-1. Analog to Digital Converter Block Schematic



27.2 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents 0V (conversion result 0x000) and the maximum value in single ended mode represents the reference voltage minus 1 LSB (conversion result 0x3FF). The reference voltage can be measured at the AREF pin. The internal, generated reference voltage can have the values 1.5V, 1.6V or AVDD where the 1.6V has the highest absolute accuracy. The reference voltage is selected by writing to the REFS*n* bits in the ADMUX Register. An external reference voltage can also be selected. Such an external voltage must be supplied with a very low impedance $\mathbf{R_{AREF,EXT}}$ (see "ADC Electrical Characteristics" on page 505). The load current $\mathbf{I_{L,AREF}}$ (see "ADC Electrical Characteristics" on page 505) seen by the external source is code dependent and changes in the course of the successive approximation process (load current steps). The internal voltage reference (except AVDD) must not be decoupled by an external capacitor. Adding unnecessary external capacitance at the AREF pin will

cause instable operation of the internal reference voltage buffer and will not improve noise immunity.

The analog input channel is selected by writing to the MUX bits in ADMUX and ADCSRB. Any of the ADC input pins, as well as AVSS and a fixed bandgap voltage reference can be selected as single ended inputs to the ADC. A choice of ADC input pins can be selected as positive and negative inputs to the differential amplifier. Furthermore the temperature sensor and the DRT voltages of SRAM2 can also be processed with the ADC.

If differential channels are selected, the amplified voltage difference between the selected input channel pair then becomes the input of the ADC. The respective pin voltages for a differential measurement can be in the range from 0V to EVDD. In this way it is possible to handle differential input voltages with a common mode value higher than AVDD e.g. process a 50mV differential signal with a 2.5V common mode voltage. If single ended channels are used, the gain amplifier is bypassed altogether. Any ADC input voltage (single-ended or amplified-differential) exceeding AVDD will be internally clamped to AVDD to avoid damaging the ADC circuitry. Note that the pin input current will not increase if the clamp circuit is active.

The ADC is enabled by setting ADEN bit in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared. It is required to disable the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

27.3 ADC Start-Up

After the ADC is enabled by setting ADEN, it will go through a start-up phase. The analog supply voltage AVDD is turned on. It takes time t_{AVREG} (see "Power Management Electrical Characteristics" on page 503) µs for AVDD to stabilize. A stable AVDD voltage is indicated by the AVDDOK bit in ADCSRB. After this the ADC and, for differential input channels also the gain amplifier, is powered up. The duration of this phase depends on the ADC clock period and the configuration of the Start-Up and Track-And–Hold Time bits, ADSUT4:0 and ADTHT1:0 in ADCSRC. For details about the start-up timing refer to section "Pre-scaling and Conversion Timing" on page 413.

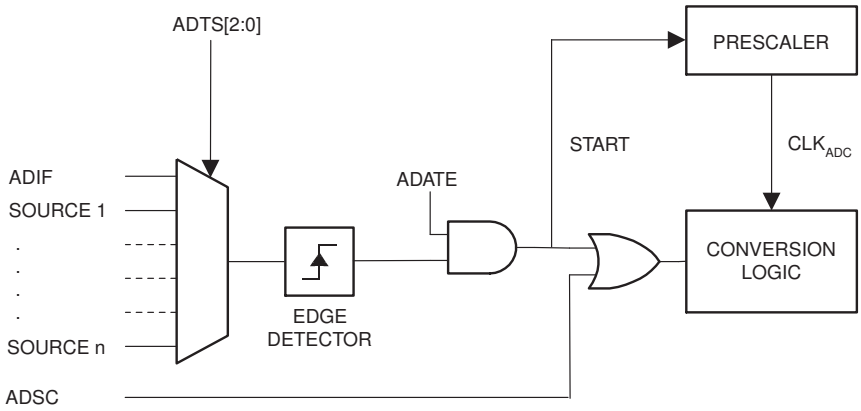
During the ADC start-up phase a conversion start can already be requested by writing a logical one to the ADC Start Conversion bit, ADSC in ADCSRA. In this case a conversion is started directly after the start-up phase. During the start-up phase it is still possible to change the analog input channel until the AVDDOK bit changes to logic one or, if the AVDDOK bit is one, until the ADSC bit is set.

27.4 Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 27-2. ADC Auto Trigger Logic



Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

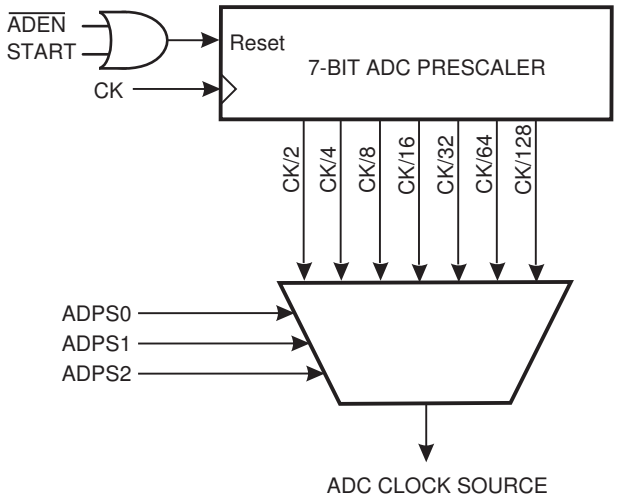
27.5 Pre-scaling and Conversion Timing

27.5.1 Prescaler

By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 4 MHz. If a lower resolution than 10 bits is needed, the input clock

frequency to the ADC can be as high as 8 MHz to get a higher sample rate. For differential input channels the ADC clock speed is restricted to a maximum of 2 MHz.

Figure 27-3. ADC Prescaler



The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100 kHz. The pre-scaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment when the ADC is enabled. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

27.5.2 Start-Up Timing

The ADC is enabled by setting the ADEN bit in ADCSRA. First the analog voltage regulator is powered up which takes t_{AVREG} (see "Power Management Electrical Characteristics" on page 503). A stable AVDD is indicated by the AVDDOK bit in ADCSRB.

After AVDD has stabilized, the ADC is started. The ADC start-up time has a length of t_{ADSU} and can be adjusted by the Start-Up time bits ADSUT4:0 in ADCSRC. If differential input channels are used, then an additional initialization period t_{AINIT} is required by the gain amplifier. This period is configured by the Track-And-Hold Time bits, ADTHT1:0 in ADCSRC. ADSUT4:0 and ADTHT1:0 are fixed numbers of ADC clock cycles and can be setup for different ADC clock speeds.

The minimum required ADC start-up time is 20 µs. Note that for the maximum ADC speed of 8 MHz the start-up time can not be set higher than 16 µs in ADSUT4:0. Under this condition the user has either to ensure that a conversion is not started earlier than 20 µs after the ADC is enabled or the first conversion result should be discarded.

For a summary of start-up times and sequences see Table 27-1 below, Table 27-2 below, Figure 27-4 on page 415 and Figure 27-5 on page 415.

Table 27-1. Start-Up Time, Single Ended Channels

Parameter	Duration in ADC Clock Cycles
ADC Start-Up Time t_{ADSU}	4(ADSUT+1), minimum 20 µs

Table 27-2. Start-Up Time, Differential Channels

Parameter	Duration in ADC Clock Cycles
ADC Start-Up Time t_{ADSU}	4(ADSUT+1), minimum 20 µs

Parameter	Duration in ADC Clock Cycles
Gain Amplifier Initialization Time t_{AINIT}	2(ADTHT+2)

Figure 27-4. ADC Timing Diagram, Start-Up for Single Ended Channels

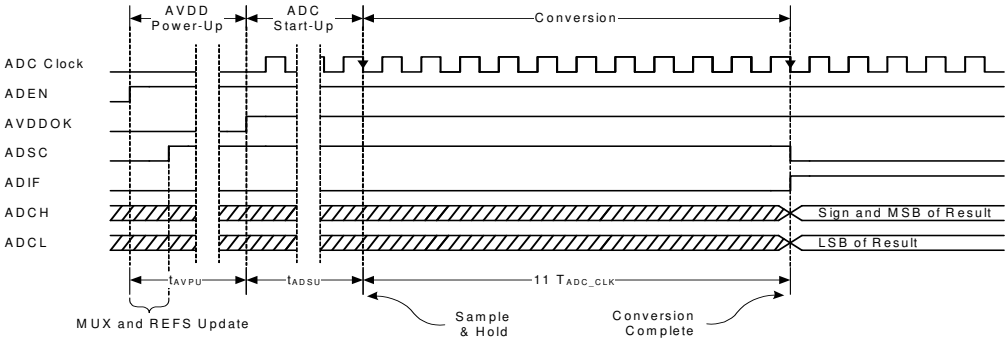
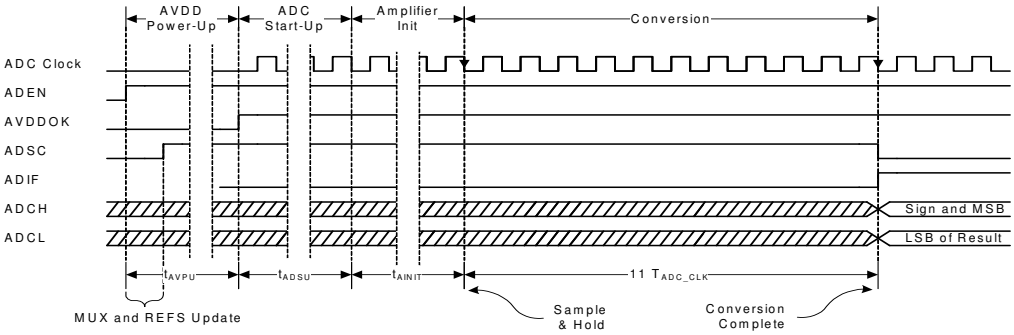


Figure 27-5. ADC Timing Diagram, Start-Up for Differential Channels



27.5.3 Conversion Timing

The delay from requesting a conversion start by setting the ADSC bit in ADCSRA to the moment where the sample-and-hold takes place is fixed. The same fixed delay also applies for auto triggered conversions. In this case three additional CPU clock cycles are used for the trigger event synchronization logic. The delay depends on the prescaler configuration ADPS and if single-ended or differential channels are used. A summary is given in Table 27-3 on page 416. All conversions take 11 ADC clock cycles.

When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated at the earliest after the following tracking phase. The tracking phase is required after each conversion. Its duration can be adjusted according to the ADC clock speed by the ADTHT bits in ADCSRC and is different for single-ended and differential channels. For details see Table 27-4 on page 416.

In Free Running mode, a new conversion will be started immediately after the tracking phase of the previous conversion while ADSC remains high. The calculation of the resulting sample rate is given in Table 27-5 on page 416.

For timing diagrams of single and auto triggered and free running conversions see Figure 27-6 on page 416 to Figure 27-8 on page 417.

Table 27-3. Conversion Start Delay

Channel	ADPS	Delay from Conversion Start Request to Sample & Hold t_{SCSMP}
Single-Ended	0, 1	2 CPU clock cycles
	2	4 CPU clock cycles
	3	0 CPU clock cycles
	4…7	0 CPU clock cycles
Differential	0…7	2 ADC clock cycles

Table 27-4. Tracking Time

Channel	Tracking Phase Duration t_{TRCK} in ADC Clock Cycles
Single-Ended	ADTHT+1, minimum 500 ns
Differential	2ADTHT+3

Table 27-5. Sample Rate in Free Running Mode

Channel	Sample Rate in ADC Clock Cycles
Single-Ended	ADTHT+12
Differential	2ADTHT+14

Figure 27-6. ADC Timing Diagram, Single Conversion

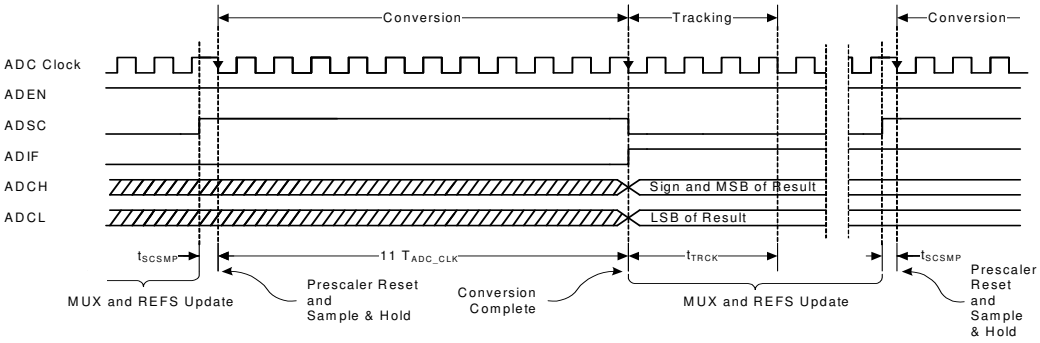


Figure 27-7. ADC Timing Diagram, Auto Triggered Conversion

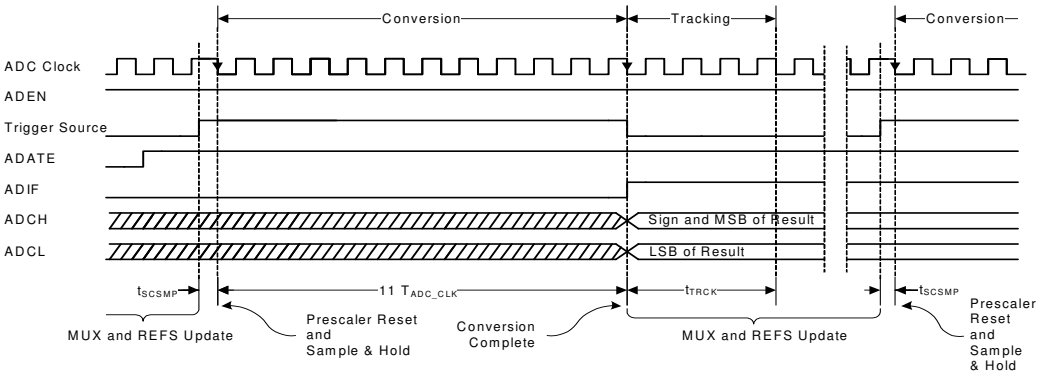
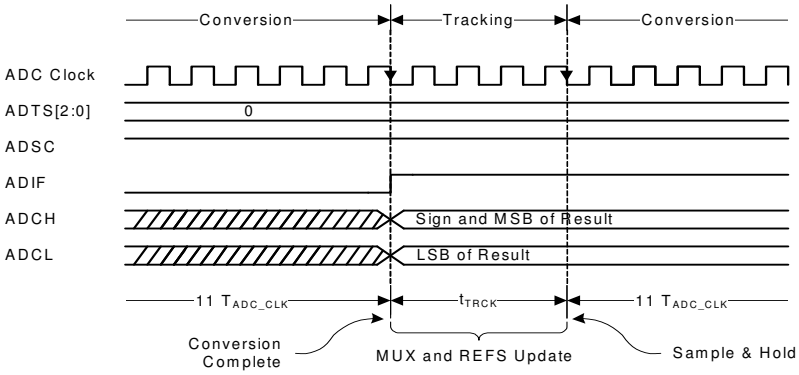


Figure 27-8. ADC Timing Diagram, Free Running Conversion



27.6 Changing Channel or Reference Selection

The MUX*n* and REFS*n* bits in the ADMUX and ADCSRB Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated either during the AVDD power-up phase or until a conversion is started by setting ADSC. After this the channel and reference selection is locked to ensure a sufficient initialization and sampling time for the ADC. Continuous updating of the channel selection resumes after the conversion has completed (ADIF in ADCSRA is set). The reference selection can only be updated if the ADC is disabled and enabled again.

If Auto Triggering is used, the exact time of the triggering event can be undetermined. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN in the ADSCRA Register are written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
2. During a conversion
3. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next A/D conversion.

After the channel or reference voltage selection is updated a settling time is required for the ADC and the gain amplifier or the reference voltage to stabilize. When changing the channel selection while the ADC is enabled the required settling phase is automatically inserted by the ADC interface, see section "ADC Input Channels" on page 418. For consideration on changing the reference voltage selection please refer to section"ADC Voltage Reference" on page 419.

27.6.1 Accessing the ADMUX Register

The channel selection bits MUX4:0 and MUX5 are located in two different register, the ADMUX and the ADCSRB register. To ensure that changes go only into effect after both register have been changed they are internally buffered (see Figure 27-9 on page 419 and Figure 27-10 on page 419). The MUX5 bit has to written first followed by a write access to the MUX4:0 bits which triggers the update of the internal buffer. If only

the MUX4:0 bits need to be modified then a write access to the MUX4:0 bits is sufficient.

27.6.2 ADC Input Channels

The ADC input channels can be changed while the ADC is running under the condition that the previous channel was a single-ended one. Changing between differential channels however requires that the ADC is disabled and enabled again to make the ADC go through the initial start-up phase.

If changing from single-ended to single-ended or from single-ended to differential input channels a settling phase is automatically inserted by the ADC interface logic after the input channel is modified. The settling phase is required by the ADC and the gain amplifier to stabilize. If a conversions start is requested during this settling phase, by setting ADSC or by a trigger event in Auto Triggered mode then the conversion is started only after the settling phase has completed.

In case the MUX*n* bits are altered during an ongoing conversion, the ADC input channel is changed after the conversion has completed. MUX*n* changes occurring during the tracking phase, which follows a conversion, will stop the tracking phase and the ADC settling phase will be entered.

In Free Running mode MUX*n* can also be modified. In this case the ADC input channel is changed after the conversion end or from the subsequent tracking phase. As a consequence the time from one conversion to the next is extended by the duration of the ADC settling phase.

The ADC settling time t_{ASET} depends on the previous and the new channel and on the configuration of the ADSUT4:0 and ADTHT1:0 bits as shown in Table 27-6 below. Additionally a synchronization delay t_{CHDLY} from 2 CPU to 2 ADC Clock cycles is required between changing the ADC input channel selection and the beginning of the settling phase. For details see the timing diagrams Figure 27-9 on page 419 and Figure 27-10 on page 419.

If the analog input signal encounters large variations it can be useful to manually reset the ADC and the gain amplifier before starting a new conversion. To achieve this, the settling phase can be forced without modifying MUX*n* by writing a logic one to the Analog Channel Change bit ACCH in ADCSRB. Using the ACCH bit is only recommended for single-ended input channels. For differential input channels the ADC and the gain amplifier can be reset if the ADC is disabled and enabled again.

Table 27-6. Settling Time after Channel Changes

Channel Transition	Settling Time t_{ASET} in ADC Clock Cycles
Single-Ended or Differential to Single-Ended	ADTHT+2
Single-Ended to Differential	4(ADSUT+1) + 2(ADTHT+2)
Differential to Differential	Requires the ADC to be disabled and enabled again.

Figure 27-9. ADC Timing Diagram, Changing MUX*n* after a Conversion

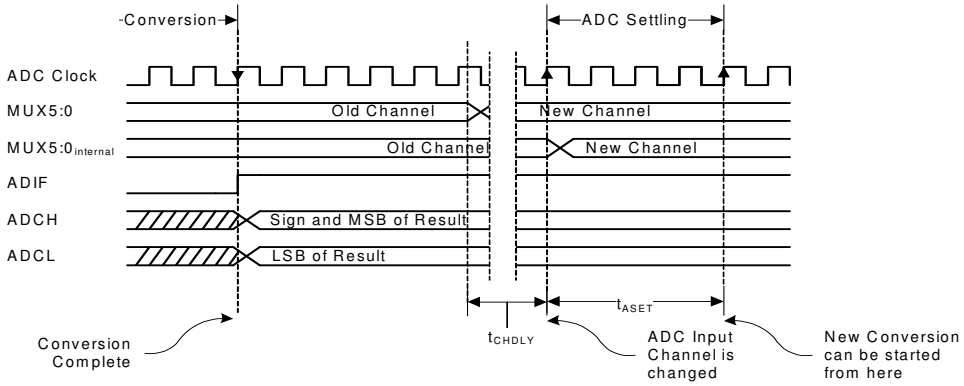
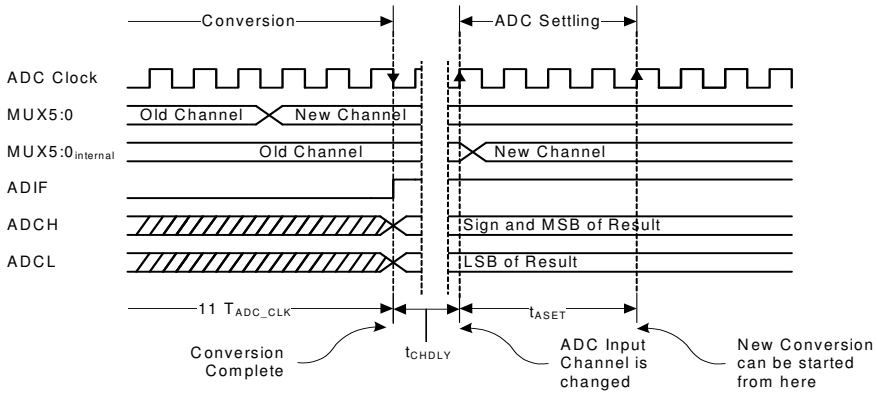


Figure 27-10. ADC Timing Diagram, Changing MUX*n* during a Conversion



27.6.3 ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result 0x3FF. V_{REF} can be selected by the REFS*n* bits in the ADMUX register as either AVDD (1.8V), internal 1.5V or 1.6V reference or an external voltage at the AREF pin.

AVDD is connected to the ADC through a passive switch. The internal 1.5V and 1.6V references are generated from a bandgap reference (VBG) through an amplifier. In either case, the external AREF pin is directly connected to the ADC and the reference voltage can be measured at the AREF pin with a high impedance voltmeter. When using the internal 1.5V or 1.6V references no external de-coupling capacitor must be connected to AREF. High capacitive loading will de-stabilize the internal voltage amplifier. The 1.6V reference voltage is calibrated to an absolute accuracy of 1 LSB during the manufacturing process.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. An external reference voltage must be supplied with a very low impedance $\mathbf{R_{AREF,EXT}}$ (see "ADC Electrical Characteristics" on page 505). The load current $\mathbf{I_{L,AREF}}$ (see "ADC Electrical Characteristics" on page 505) seen by the external source is code dependent and changes (current steps) in the course of the successive approximation process. If no external voltage is applied to the AREF pin, the user may switch between AVDD, 1.5V and 1.6V as reference selection.

Changes of the reference selection bits REFS*n* will only take effect until the first conversion start is requested by setting ADSC in ADCSRA. After this the ADC has to be

disabled and enabled again for new reference selections. For internal references a stable voltage is indicated by the REFOK bit in ADCSRB.

27.7 ADC Noise Canceller

The ADC features a noise canceller that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceller can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC Conversion Complete interrupt must be enabled.
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the A/D conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the A/D conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the A/D conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

27.7.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 27-11 on page 421. An analog source applied to ADC*n* is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

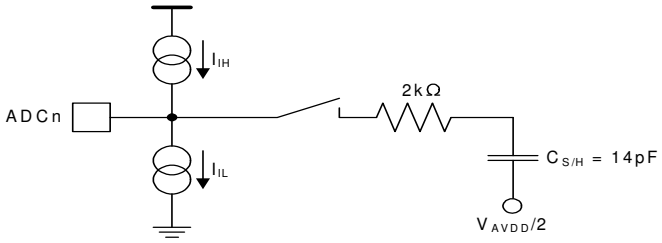
The ADC is optimized for analog signals having output impedance Z_{OUT} of approximately 3 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the correct sampling time will depend on how much time is needed to charge the S/H capacitor, which can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor. The required tracking time (input sampling switch closed) t_{DTRCK} to settle to within 1 LSB can be estimated to

$$t_{DTRCK} = (Z_{OUT} / k\Omega + 2000) \cdot 0.097ns$$

for Z_{OUT} > 3kΩ (worst case: maximum input step). A minimum tracking time of 500ns is guaranteed by the conversion logic. Based on the ADC clock frequency the bits ADTHT[1:0] of register ADCSRC allow the adjustment of the tracking time to the user's requirements.

Tracking time requirements should also be considered for the differential mode. The input signal is sampled by the gain amplifier. The value of the input capacitance $C_{S/H}$ depends on the selected gain (~7pF for 200x gain, <1pF otherwise). The tracking is equal to 50% of the clock period of CK_{ADC2}. Hence in differential mode a slower clock frequency is required for input sources with high impedance.

Figure 27-11. Analog Input Circuitry



Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

27.7.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the ground plane, and keep them well away from high-speed switching digital tracks.
2. Use the ADC noise canceller function to reduce induced noise from the CPU.
3. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

27.7.3 Offset Compensation Schemes

The differential amplifier has a built-in offset cancellation circuitry that nulls the offset of differential measurements as much as possible. The remaining offset in the analog path can be measured directly by selecting the same channel for both differential inputs. This offset residue can then be subtracted in software from the measurement results. The offset on any channel can be reduced below one LSB using this kind of software based offset correction.

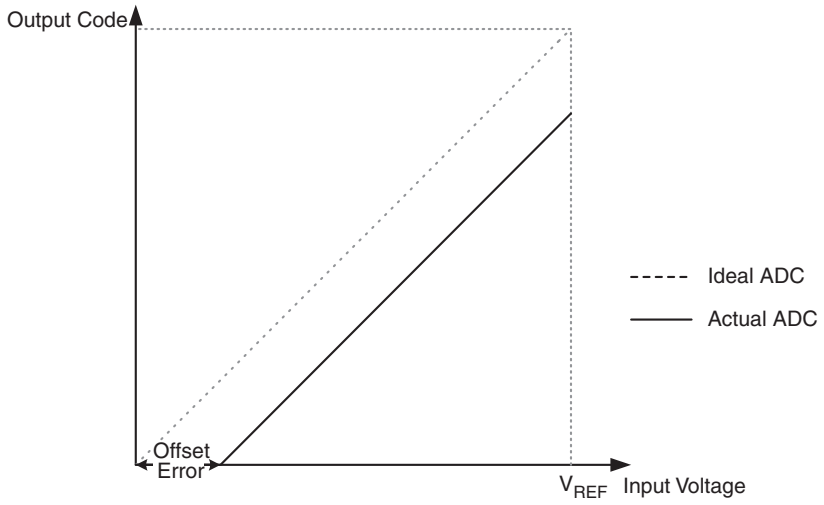
27.7.4 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between 0V and V_{REF} in 2^n steps (LSB's). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

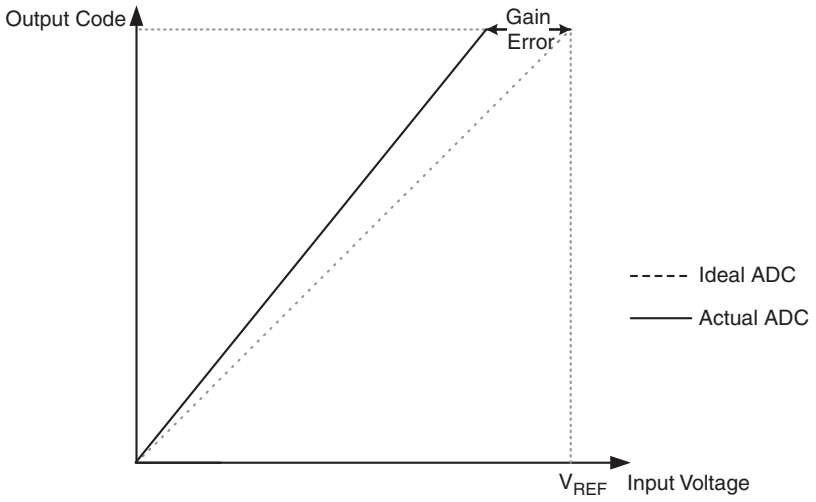
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 27-12. Offset Error



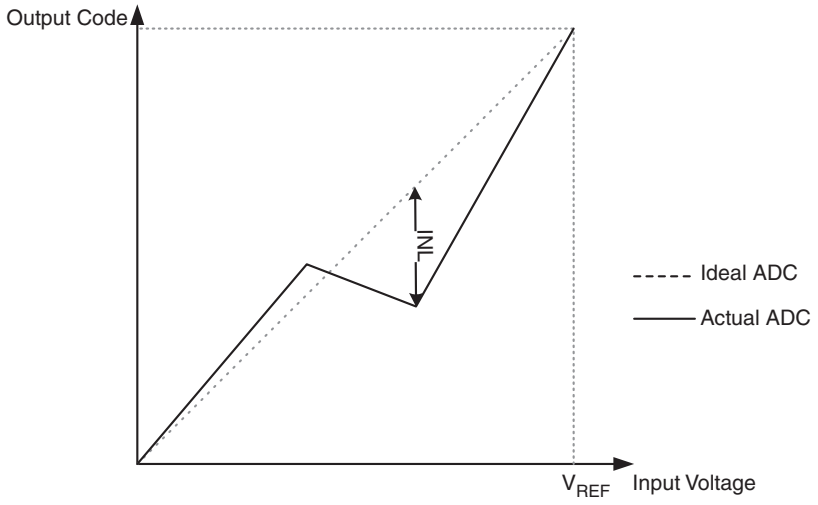
- Gain Error: After adjusting for offset, the Gain Error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

Figure 27-13. Gain Error



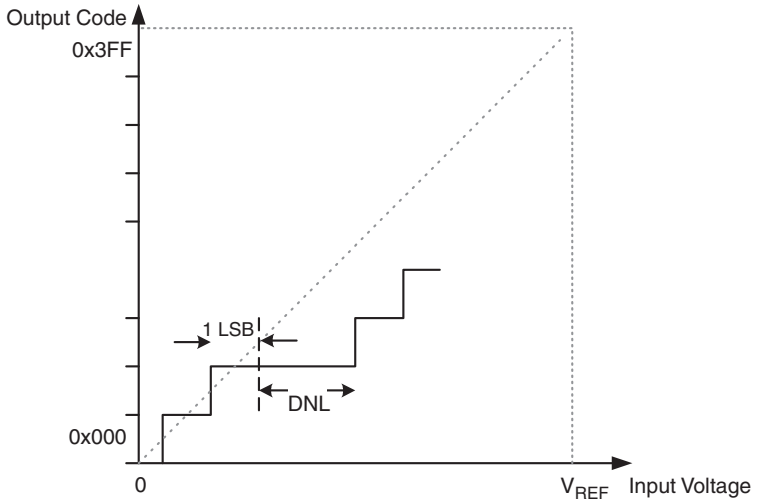
- Integral Non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 27-14. Integral Non-linearity (INL)



- Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 27-15. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. It is always ±0.5 LSB.
- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5 LSB.

27.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see "Table 27-10" on page 427 and "Table 27-11" on page 428). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the result, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive. Figure 27-16 below shows the decoding of the differential input range.

Table 27-7 on page 425 shows the resulting output codes if the differential input channel pair (ADCn - ADCm) is selected with a gain of GAIN and a reference voltage of V_{REF}.

Figure 27-16. Differential Measurement Range

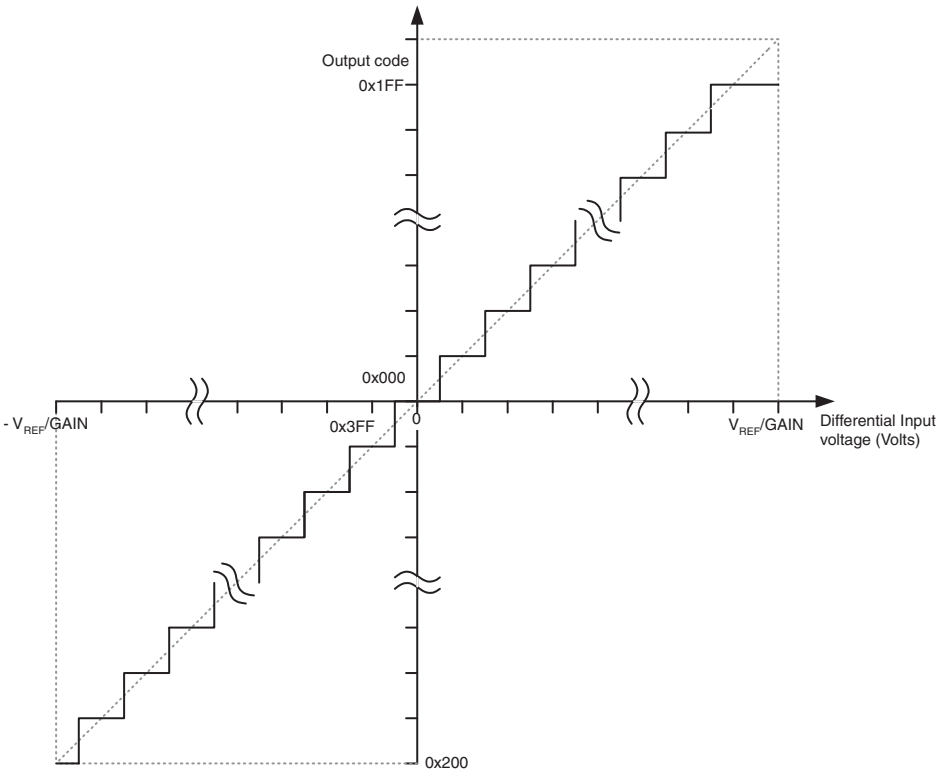


Table 27-7. Correlation Between Input Voltage and Output Codes

V_{ADCn}	Read Code	Corresponding Decimal Value
V_{ADCm} + V_{REF} / GAIN	0x1FF	511
V_{ADCm} + 0.999 V_{REF} / GAIN	0x1FF	511
V_{ADCm} + 0.998 V_{REF} / GAIN	0x1FE	510
…	…	…
V_{ADCm} + 0.001 V_{REF} / GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 0.001 V_{REF} / GAIN	0x3FF	-1
…	…	…
V_{ADCm} - 0.999 V_{REF} / GAIN	0x201	-511
V_{ADCm} - V_{REF} / GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10x gain, 1.6V reference, left adjusted result)

The voltage on ADC3 is 300 mV; the voltage on ADC2 is 425 mV.

ADCR = 512 * 10 * (300 - 425) / 1600 = -400 = 0x270.

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

27.9 Internal Temperature Measurement

The on-chip temperature can be measured using a special setup of the A/D converter inputs. The integrated temperature sensor provides a linear, medium-accurate voltage proportional to the absolute temperature (in Kelvin). This voltage is first amplified with the programmable gain amplifier and then processed with the A/D converter. A low frequency of the conversion clock must be selected due to the nature of the input signal. The absolute accuracy of the temperature measurement is limited by manufacturing tolerances, noise from supply and ground voltages and the exactness of the reference voltage. The following table summarizes the preferred setup of the temperature measurement:

Table 27-8. Recommended ADC Setup for Temperature Measurement

Parameter	Register	Recommended Setup
ADC Channel	ADMUX, ADCSRB	Select the Temperature Sensor, MUX4:0 = 01001; MUX5 = 1;
ADC Clock	ADCSRA	Select a clock frequency of 500kHz or lower;
V_{REF}	ADMUX	Select the internal 1.6V reference voltage;
Start-up time	ADCSRC	Standard requirement of 20µs is sufficient;
Tracking time	ADCSRC	Setting ADTHT = 0 is sufficient;

The A/D conversion result ADC_{TEMP} will always be a positive number. The ideal result can be calculated when using the internal 1.6V reference voltage according to the following equation:

$$ADC_{TEMP} = 241.4 + 0.885 \cdot \theta / °C$$

Similar the Celsius-temperature θ can be extracted from the A/D conversion result with this formula:

$$\theta / °C = 1.13 \cdot ADC_{TEMP} - 272.8$$

Note that the above equations are only valid in the allowed operating temperature range. The translation of the A/D measurement result to a Celsius-temperature value can be easily achieved with a look-up table in software. The accuracy of the temperature reading can be improved by averaging of multiple A/D conversion results. In this way the impact of noise is reduced. The temperature sensor is connected to a differential input channel with a gain of 10. The offset error of the channel can be corrected to the first order by using an appropriate channel (e.g. MUX4:0=01000, MUX5=0, see Table 27-11 on page 428). The in that manner measured error of the differential signal processing is then subtracted from the temperature sensor ADC reading.

Note that changing between the temperature sensor channel and the channel for the offset error correction can lead to a large difference of the analog input voltage. Therefore it is recommended to disable the ADC, select the new channel and then enable the ADC again.

27.10 SRAM DRT Voltage Measurement

The decrease of the supply voltage of SRAM block 2 for the leakage current reduction can also be measured using a special setup of the A/D converter inputs. The details of the SRAM leakage current reduction are described in section "SRAM with Data Retention" on page 163. The supply voltage of a disabled SRAM block can be reduced to save leakage power while maintaining data retention. This feature applies to all four SRAM blocks however only the voltage of SRAM block 2 can be verified using the A/D converter.

The default factory setting for the data retention (DRT) voltage normally guarantees the best leakage performances. Other values are nevertheless possible and can be selected by the application software. The true value of the supply voltage reduction is depending on the manufacturing process and environmental conditions like temperature. The A/D converter allows determining the value of the DRT voltage of SRAM block 2. The same voltage setting results for all practical purposes in the same supply voltage for all other SRAM blocks.

Care must be taken when verifying the DRT voltage of SRAM block 2 with the A/D converter because it will be put into sleep mode and hence it is not available for the application program. Addressing the disabled SRAM will return invalid data (all data read zero). The voltage measurement is split into two parts. One setting allows measuring the voltage drop from DVDD. The other setting allows verifying the voltage shift from DVSS. Both measurements are differential and use the programmable gain amplifier. A low frequency of the conversion clock must be selected due to the high-impedance nature of the input signal. Accurate and stable voltage readings may just be available after a long waiting time of up to 100 ms. This limitation is the consequence of the small leakage currents that discharge the internal de-coupling capacitances before the supply voltage settles to the DRT value. The following table summarizes the preferred setup of the DRT voltage measurement:

Table 27-9. Recommended ADC Setup for DRT Voltage Measurements

Parameter	Register	Recommended Setup
SRAM DRT on	DRTRAM2	Set bits DISPC and ENDRT to 1;
ADC Channel	ADMUX,	Select MUX4:0 = 10100 to measure V_{DRTBBP}; Select MUX4:0 = 11101 to measure V_{DRTBBN};

Parameter	Register	Recommended Setup
	ADCSRB	MUX5 = 1;
ADC Clock	ADCSRA	Select a clock frequency of 500kHz or lower;
V_{REF}	ADMUX	Select the internal 1.6V reference voltage;
Start-up time	ADCSRC	Standard requirement of 20µs is sufficient;
Tracking time	ADCSRC	Setting ADTHT = 0 is sufficient;

The A/D conversion result will always be a positive number for both V_{DRTBBP} and V_{DRTBBN}. The SRAM supply voltage is easily calculated according to the following equation (see chapter "SRAM with Data Retention" on page 163):

$$V_{DD,SRAM,DRT} = V_{DD} - (V_{DRTBBP} + V_{DRTBBN})$$

The conversion result is coded as described in "ADC Conversion Result" on page 423 with a GAIN of 0.5. It is not possible to read both V_{DRTBBP} and V_{DRTBBN} at the same time. However the time required for the A/D conversion is short compared to the time constant of a DRT voltage change.

27.11 Register Description

27.11.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
NA ($7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in the following table. Changes of these bits will only take effect until the first conversion start is requested by setting ADSC. After this the ADC has to be disabled and enabled again for new reference selections. The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 27-10. Reference Voltage Selections for ADC

REFS1	REFS0	Reference Voltage Selection
0	0	AREF, Internal V_{REF} turned off
0	1	AVDD (1.8V)
1	0	Internal 1.5V Voltage Reference (no external capacitor at AREF pin)
1	1	Internal 1.6V Voltage Reference (no external capacitor at AREF pin)

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the A/D conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 432.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs is connected to the ADC. See Table 27-11 on page 428 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in

ADCSRA is set). Note that the MUX5 bit is located in the ADCSRB register. A write access to the MUX4:0 bits triggers the update of the internally buffered MUX5 bit, see "Accessing the ADMUX Register" on page 417 .

27.11.2 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
NA ($7B)	AVDDOK	ACME	REFOK	ACCH	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	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 – AVDDOK: AVDD Supply Voltage OK**

The analog functions of the ADC are powered from the AVDD domain. AVDD is supplied from an internal voltage regulator. Setting the ADEN bit in register ADCSRA will power-up the AVDD domain if not already requested by another functional group of the device. The bit allows the user to monitor (poll) the status of the AVDD domain. A status of 1 indicates that AVDD has been powered-up.

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

This bit is used for the Analog Comparator only. See "ADCSRB – ADC Control and Status Register B" on page 409 for details.

- **Bit 5 – REFOK: Reference Voltage OK**

The status of the internal generated reference voltage can be monitored through this bit. Setting the ADEN bit in register ADCSRA will enable the reference voltage for the ADC according to the REFS*n* bits in the ADMUX register. The reference voltage will be available after a start-up delay. A REFOK value of 1 indicates that the internal generated reference voltage is approaching final levels.

- **Bit 4 – ACCH: Analog Channel Change**

The user can force a reset of the analog blocks by setting this bit to 1 without requesting a different channel. The analog blocks of the ADC will be reset to handle possible new voltage ranges. Such a reset phase is especially important for the gain amplifier. It could be temporarily disabled by a large step of its input common voltage leading to erroneous A/D conversion results. ACCH will read as one until the reset phase of the analog blocks can be entered.

- **Bit 3 – MUX5: Analog Channel and Gain Selection Bit**

This bit is used together with MUX4:0 in ADMUX to select the analog input signals connected to the ADC. See the following table for details. If this bit is changed during a conversion, the change will not go in effect until this conversion is complete. Note that the MUX5 bit is internally buffered and a write access to the MUX4:0 bits is required to trigger the update of the MUX5 bit, see "Accessing the ADMUX Register" on page 417 .

Table 27-11. Input Channel Selections

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000110	ADC6			
000111	ADC7			
001000	N/A	ADC0	ADC0	10x
001001		ADC1	ADC0	10x
001010		ADC0	ADC0	200x
001011		ADC1	ADC0	200x
001100		ADC2	ADC2	10x
001101		ADC3	ADC2	10x
001110		ADC2	ADC2	200x
001111		ADC3	ADC2	200x
010000	N/A	ADC0	ADC1	1x
010001		ADC1	ADC1	1x
010010		ADC2	ADC1	1x
010011		ADC3	ADC1	1x
010100		ADC4	ADC1	1x
010101		ADC5	ADC1	1x
010110		ADC6	ADC1	1x
010111		ADC7	ADC1	1x
011000	N/A	ADC0	ADC2	1x
011001		ADC1	ADC2	1x
011010		ADC2	ADC2	1x
011011		ADC3	ADC2	1x
011100		ADC4	ADC2	1x
011101		ADC5	ADC2	1x
011110	1.2V (V_{BG})	N/A		
011111	0V (AVSS)			
100000	Reserved	N/A		
100001	Reserved			
100010	Reserved			
100011	Reserved			
100100	Reserved			
100101	Reserved			
100110	Reserved			
100111	Reserved			
101000	N/A	Reserved		
101001		Temperature Sensor		
101010		Reserved		
101011		Reserved		
101100		Reserved		
101101		Reserved		
101110		Reserved		

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
101111		Reserved		
110000	N/A	Reserved		
110001		Reserved		
110010		Reserved		
110011		Reserved		
110100		SRAM Back-bias Voltage V_{DRTBBP}		
110101		Reserved		
110110		Reserved		
110111		Reserved		
111000	N/A	Reserved		
111001		Reserved		
111010		Reserved		
111011		Reserved		
111100		Reserved		
111101		SRAM Back-bias Voltage V_{DRTBBN}		
111110	Reserved	N/A		
111111	Reserved			

- **Bits 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an A/D conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared, to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS2:0=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 27-12. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

27.11.3 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
NA ($7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. The AVDD supply voltage will also be enabled if not already available. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will include a start-up time to initialize the analog blocks of the ADC. The start-up time is defined by the ADSUT bits of register ADCSRC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an A/D conversion is completed and the Data Register are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the CPU frequency and the input clock to the ADC.

Table 27-13. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

27.11.4 ADCSRC – ADC Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
NA ($77)	ADTHT1	ADTHT0	Res0	ADSUT4	ADSUT3	ADSUT2	ADSUT1	ADSUT0	ADCSRC
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	1	0	1	0	1	0	0	

This register defines the track-and-hold time for sampling the analog input voltage of the ADC and it defines the start-up time for the analog blocks based on a number of ADC clock cycles. The ADC clock is generated from the system clock with the ADC prescaler. The bits ADPS2:0 of register ADCSRA set the prescaler ratio. Correct start-up and track-and-hold times are important for precise conversion results.

- **Bits 7:6 – ADTHT1:0: ADC Track-and-Hold Time**

These bits define the number of ADC clock cycles for the sampling time of the analog input voltage. For a complete description of this bit, see "Pre-scaling and Conversion Timing" on page 413.

- **Bit 5 – Res0: Reserved**
- **Bits 4:0 – ADSUT4:0: ADC Start-up Time**

These bits define the number of ADC clock cycles for the start-up time of the analog blocks. For a complete description of this bit, see "Pre-scaling and Conversion Timing" on page 413.

27.11.5 ADCL and ADCH – The ADC Data Register

27.11.5.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
NA ($79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
NA ($78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	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	

27.11.5.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
NA ($79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
NA ($78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	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 an A/D conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision (7 bit + sign bit for differential input channels) is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUX*n* bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: A/D Conversion Result**

These bits represent the result from the conversion as detailed in "ADC Conversion Result" on page 423.

27.11.6 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
NA ($7E)	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – ADC7D:ADC0D: Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7:0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

27.11.7 DIDR2 – Digital Input Disable Register 2

Bit	7	6	5	4	3	2	1	0	
NA ($7D)	ADC15D	ADC14D	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	DIDR2
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

Reserved for future use.

- **Bit 7:0 – ADC15D:ADC8D - Reserved Bits**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero.

27.11.8 BGCR – Reference Voltage Calibration Register

Bit	7	6	5	4	
NA ($67)	Res	BGCAL_FINE3	BGCAL_FINE2	BGCAL_FINE1	BGCR
Read/Write	R	RW	RW	RW	
Initial Value	0	0	0	0	
Bit	3	2	1	0	
NA ($67)	BGCAL_FINE0	BGCAL2	BGCAL1	BGCAL0	BGCR
Read/Write	RW	RW	RW	RW	
Initial Value	0	0	0	0	

This register contains the calibration values of the reference voltage of the ADC. The values are loaded from the fuse memory after power-up. They can be corrected by the application software e.g. to compensate for temperature changes. The internal 1.6V reference voltage is calibrated and has therefore the highest accuracy compared to the 1.5V or AVDD reference.

- **Bit 7 – Res - Reserved Bit**

This bit is reserved for future use. A read access always will return zero. A write access does not modify the content.

- **Bit 6:3 – BGCAL_FINE3:0 - Fine Calibration Bits**

These bits allow the calibration of the AREF voltage with a resolution of 2mV.

Table 27-14 BGCAL_FINE Register Bits

Register Bits	Value	Description
BGCAL_FINE3:0	0	Center value
	1	Voltage step up
	8	Voltage step down
	7	Setting for highest voltage
	15	Setting for lowest voltage

- **Bit 2:0 – BGCAL2:0 - Coarse Calibration Bits**

These bits allow the calibration of the AREF voltage with a resolution of 10mV.

Table 27-15 BGCAL Register Bits

Register Bits	Value	Description
BGCAL2:0	4	Center value
	3	Voltage step up
	5	Voltage step down
	0	Setting for highest voltage
	7	Setting for lowest voltage

28 JTAG Interface and On-chip Debug System

28.1 Features

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard**
- **Debugger Access to:**
  - **All Internal Peripheral Units**
  - **Internal and External RAM**
  - **The Internal Register File–Program Counter**
  - **EEPROM and Flash Memories**
- **Extensive on-chip debug Support for Break Conditions, Including**
  - **AVR Break Instruction**
  - **Break on Change of Program Memory Flow**
  - **Single Step Break**
  - **Program Memory Breakpoints on Single Address or Address Range**
  - **Data Memory Breakpoints on Single Address or Address Range**
- **Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface**
- **On-chip debugging Supported by AVR Studio®**

28.2 Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

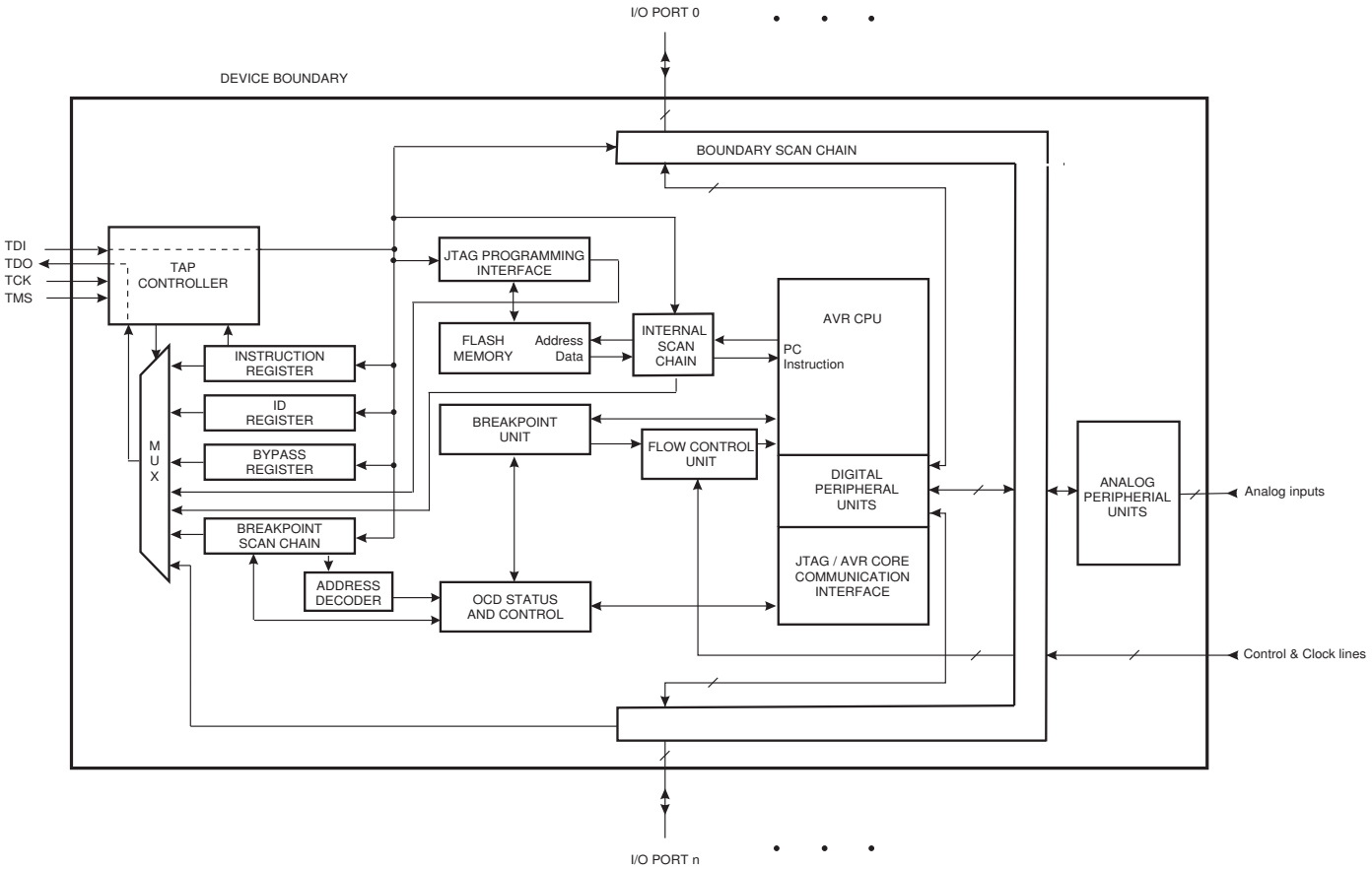
- Testing PCBs by using the JTAG Boundary-scan capability
- Programming the non-volatile memories, Fuses and Lock bits
- On-chip debugging

A brief description is given in the following sections. Detailed descriptions for Programming via the JTAG interface, and using the Boundary-scan Chain can be found in the sections "Programming via the JTAG Interface" on page 481 and "Programming via the JTAG Interface" on page 481, respectively. The on-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only.

Figure 28-1 on page 436 shows a block diagram of the JTAG interface and the on-chip debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (Shift Register) between the TDI – input and TDO – output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

The ID-Register, Bypass Register, and the Boundary-scan Chain are the Data Registers used for board-level testing. The JTAG Programming Interface (actually consisting of several physical and virtual Data Registers) is used for serial programming via the JTAG interface. The internal scan-chain and breakpoint scan-chain are used for on-chip debugging only.

Figure 28-1. Block Diagram



28.3 TAP - Test Access Port

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port – TAP. These pins are:

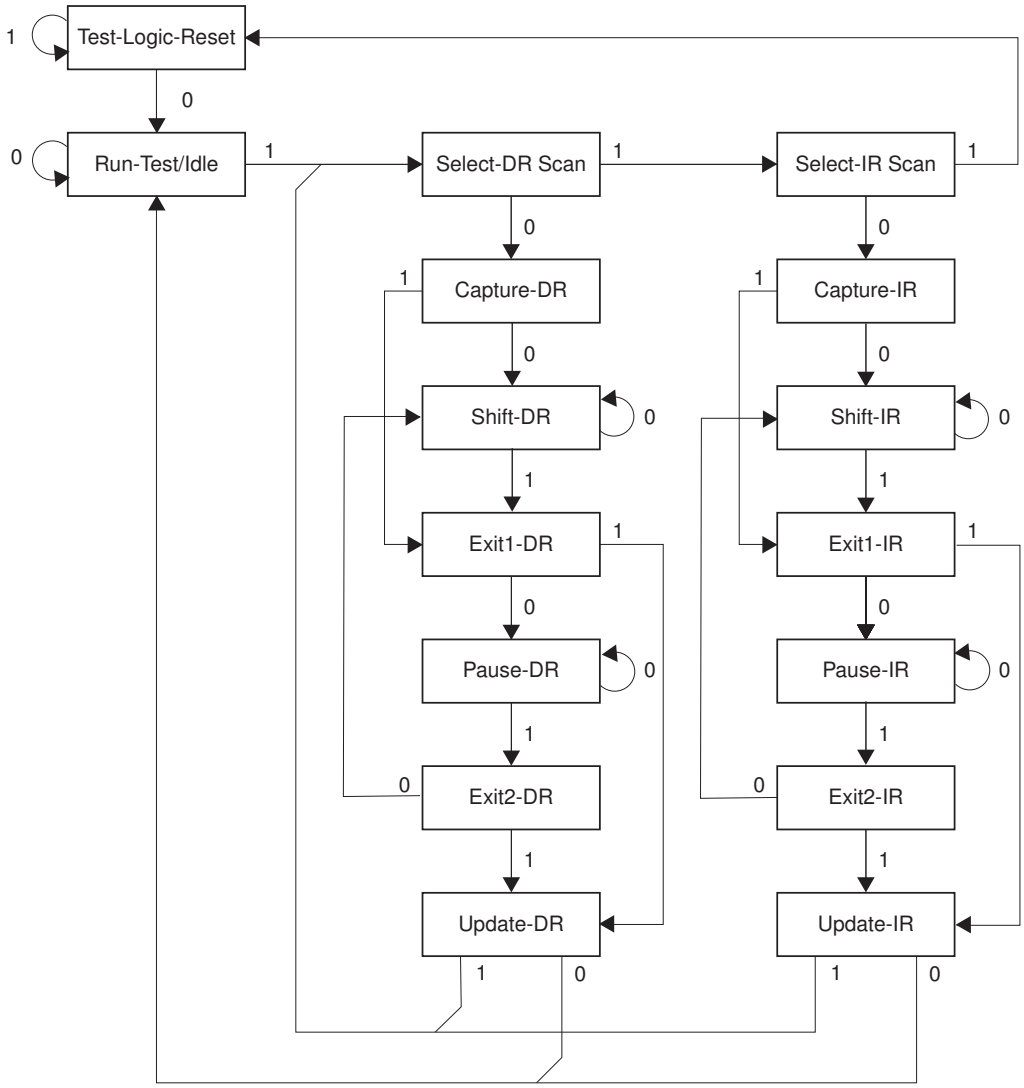
- TMS: Test mode select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK.
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains).
- TDO: Test Data Out. Serial output data from Instruction Register or Data Register.

The IEEE std. 1149.1 also specifies an optional TAP signal; TRST – Test ReSeT – which is not provided.

When the JTAGEN Fuse is un-programmed, these four TAP pins are normal port pins, and the TAP controller is in reset. When programmed the input TAP signals are internally pulled high and the JTAG is enabled for Boundary-scan and programming. The device is shipped with this fuse programmed.

For the on-chip debug system, in addition to the JTAG interface pins, the RESET pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the RESET pin low to reset the whole system, assuming only open collectors on the reset line are used in the application.

Figure 28-2. TAP Controller State Diagram



28.4 TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or on-chip debug system. The state transitions depicted in Figure 28-2 above depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-on Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out on the TDO pin. The JTAG Instruction

selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register – Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note that independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods. For detailed information on the JTAG specification, refer to the literature listed in "Bibliography" on page 440.

28.5 Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 441.

28.6 Using the On-chip Debug System

As shown in Figure 28-1, the hardware support for on-chip debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units.
- Breakpoint unit.
- Communication interface between the CPU and JTAG system.

All read or modify/write operations needed for implementing the debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Breakpoint Unit implements *Break on Change of Program Flow*, *Single Step Break*, two program memory breakpoints and two combined breakpoints. Together, the four breakpoints can be configured as either:

- 4 single program memory breakpoints;
- 3 single program memory breakpoint + 1 single data memory breakpoint;
- 2 single program memory breakpoints + 2 single data memory breakpoints;

- 2 single program memory breakpoints + 1 program memory breakpoint with mask (“range breakpoint”).
- 2 single program memory breakpoints + 1 data memory breakpoint with mask (“range breakpoint”).

A debugger, like the AVR Studio, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the on-chip debug specific JTAG instructions is given in "On-chip Debug Specific JTAG Instructions" below.

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the on-chip debug system to work. As a security feature, the on-chip debug system is disabled when either of the LB1 or LB2 Lock-bits are set. Otherwise, the on-chip debug system would have provided a back-door into a secured device.

The AVR Studio enables the user to fully control execution of programs on an AVR device with on-chip debug capability, AVR In-Circuit Emulator, or the built-in AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly programs assembled with Atmel Corporation’s AVR Assembler and C programs compiled with third party vendors’ compilers. For a full description of the AVR Studio, please refer to the AVR Studio User Guide. Only highlights are presented in this document.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code breakpoints (using the BREAK instruction) and up to two data memory Breakpoints, alternatively combined as a mask (range) breakpoint.

28.7 On-chip Debug Specific JTAG Instructions

The on-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction operation codes are listed for reference.

28.7.1 PRIVATE0; 0x8

Private JTAG instruction for accessing on-chip debug system;

28.7.2 PRIVATE1; 0x9

Private JTAG instruction for accessing on-chip debug system;

28.7.3 PRIVATE2; 0xA

Private JTAG instruction for accessing on-chip debug system;

28.7.4 PRIVATE3; 0xB

Private JTAG instruction for accessing on-chip debug system;

28.8 Using the JTAG Programming Capabilities

Programming of the ATmega128RFA1 via JTAG is performed via the 4-pin JTAG port, TCK, TMS, TDI, and TDO. These are the only pins that need to be controlled and observed to perform JTAG programming (in addition to power pins). The JTAGEN Fuse must be programmed and the JTD bit in the MCUCR Register must be cleared to enable the JTAG Test Access Port.

The JTAG programming capability supports:

- Flash programming and verifying.
- EEPROM programming and verifying.
- Fuse programming and verifying.
- Lock bit programming and verifying.

The Lock bit security is exactly as in parallel programming mode. If the Lock bits LB1 or LB2 are programmed, the OCDEN Fuse cannot be programmed unless first doing a chip erase. This is a security feature that ensures no back-door exists for reading out the content of a secured device.

The details on programming through the JTAG interface and programming specific JTAG instructions are given in the section "Programming via the JTAG Interface" on page 481.

28.9 Bibliography

For more information about general Boundary-scan, the following literature can be consulted:

- IEEE: IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993.
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992.

28.10 On-chip Debug Related Register in I/O Memory

28.10.1 OCDR – On-Chip Debug Register

Bit	7	6	5	4	3	2	1	0	
$31 ($51)	OCDR7:0								OCDR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The OCDR Register provides a communication channel from the running program in the microcontroller to the debugger. The CPU can transfer a byte to the debugger by writing to this location. At the same time, an internal flag; I/O Debug Register Dirty IDRD is set to indicate to the debugger that the register has been written. When the CPU reads the OCDR Register the 7 LSB will be from the OCDR Register, while the MSB is the IDRD bit. The debugger clears the IDRD bit when it has read the information. In some AVR devices, this register is shared with a standard I/O location. In this case, the OCDR Register can only be accessed if the OCDEN Fuse is programmed, and the debugger enables access to the OCDR Register. In all other cases, the standard I/O location is accessed.

- **Bit 7:0 – OCDR7:0 - On-Chip Debug Register Data**

Table 28-16 OCDR Register Bits

Register Bits	Value	Description
OCDR7:0	0	Refer to the debugger documentation for further information on how to use this register.

29 IEEE 1149.1 (JTAG) Boundary-scan

29.1 Features

- **JTAG (IEEE std. 1149.1 compliant) Interface**
- **Boundary-scan Capabilities According to the JTAG Standard**
- **Full Scan of all Port Functions as well as Analog Circuitry having Off-chip Connections**
- **Supports the Optional IDCODE Instruction**
- **Additional Public AVR_RESET Instruction to Reset the ATmega128RFA1**

29.2 System Overview

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long Shift Register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the four TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the Data Register path will show the ID-Code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an undetermined state when exiting the test mode. Entering reset, the outputs of any port pin will instantly enter the high impedance state, making the HIGHZ instruction redundant. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The device can be set in the reset state either by pulling the external RESET pin low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-Register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the external pins during normal operation of the part.

The JTAGEN Fuse must be programmed and the JTD bit in the I/O Register MCUCR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

29.3 Data Registers

The Data Registers relevant for Boundary-scan operations are:

- Bypass Register
- Device Identification Register

- Reset Register
- Boundary-scan Chain

29.3.1 Bypass Register

The Bypass Register consists of a single Shift Register stage. When the Bypass Register is selected as path between TDI and TDO, the register is reset to 0 when leaving the Capture-DR controller state. The Bypass Register can be used to shorten the scan chain on a system when the other devices are to be tested.

29.3.2 Device Identification Register

Figure 29-1. The Format of the Device Identification Register

	MSB			LSB
Bit	31 28	27 12	11 1	0
Device ID	**Version**	**Part Number**	**Manufacturer ID**	**1**
	4 bits	16 bits	11 bits	1 bit

29.3.2.1 Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on.

29.3.2.2 Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega128RFA1 is listed in Table 31-6 on page 467.

29.3.2.3 Manufacturer ID

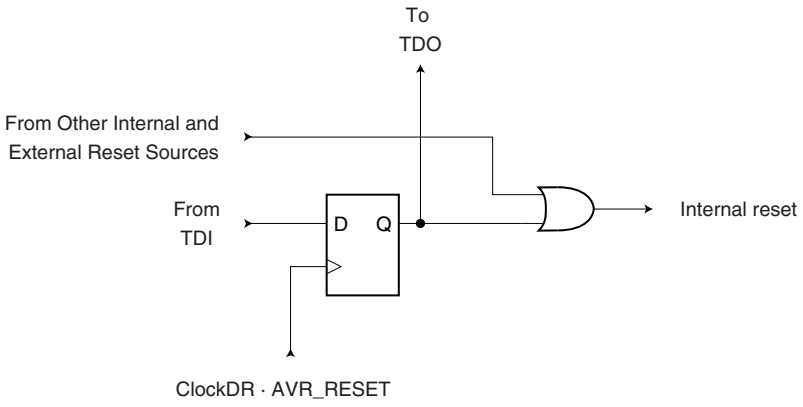
The Manufacturer ID is a 11-bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 31-6 on page 467.

29.3.3 Reset Register

The Reset Register is a test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

A high value in the Reset Register corresponds to pulling the external Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the fuse settings for the clock options, the part will remain reset for a reset time-out period (see "Clock Sources" on page 148) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 29-2 on page 443.

Figure 29-2. Reset Register



29.3.4 Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections.

See "Boundary-scan Chain" on page 444 for a complete description.

29.4 Boundary-scan Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedance state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

29.4.1 EXTEST; 0x0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-Register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

29.4.2 IDCODE; 0x1

Optional JTAG instruction selecting the 32 bit ID-Register as Data Register. The ID-Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE Register is sampled into the Boundary-scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

29.4.3 SAMPLE_PRELOAD; 0x2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan chain is applied to the output latches. However, the output latches are not connected to the pins.

29.4.4 AVR_RESET; 0xC

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

29.4.5 BYPASS; 0xF

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic “0” into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

29.5 Boundary-scan Chain

The Boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connection.

29.5.1 Scanning the Digital Port Pins

Figure 29-3 on page 445 shows the Boundary-scan Cell for a bi-directional port pin. The pull-up function is disabled during Boundary-scan when the JTAG IC contains EXTEST or SAMPLE_PRELOAD. The cell consists of a bi-directional pin cell that combines the three signals Output Control - OCxn, Output Data - ODxn, and Input Data - IDxn, into only a two-stage Shift Register. The port and pin indexes are not used in the following description.

The Boundary-scan logic is not included in the figures in the datasheet. Figure 29-4 on page 446 shows a simple digital port pin as described in the section "I/O-Ports" on page 186. The Boundary-scan details from Figure 29-3 on page 445 replaces the dashed box in Figure 29-4 on page 446.

When no alternate port function is present, the Input Data - ID - corresponds to the PINxn Register value (but ID has no synchronizer), Output Data corresponds to the PORT Register, Output Control corresponds to the Data Direction - DD Register, and the Pull-up Enable - PUExn – corresponds to logic expression:

$$\overline{PUD} \cdot \overline{DDxn} \cdot PORTxn$$

Digital alternate port functions are connected outside the dotted box Figure 29-4 on page 446 to make the scan chain read the actual pin value. For analog function, there is a direct connection from the external pin to the analog circuit. There is no scan chain on the interface between the digital and the analog circuitry, but some digital control signal to analog circuitry are turned off to avoid driving contention on the pads.

When JTAG IR contains EXTEST or SAMPLE_PRELOAD the clock is not sent out on the port pins even if the CKOUT fuse is programmed. Even though the clock is output when the JTAG IR contains SAMPLE_PRELOAD, the clock is not sampled by the boundary scan.

Figure 29-3. Boundary-scan Cell for Bi-directional Port Pin with Pull-up Function

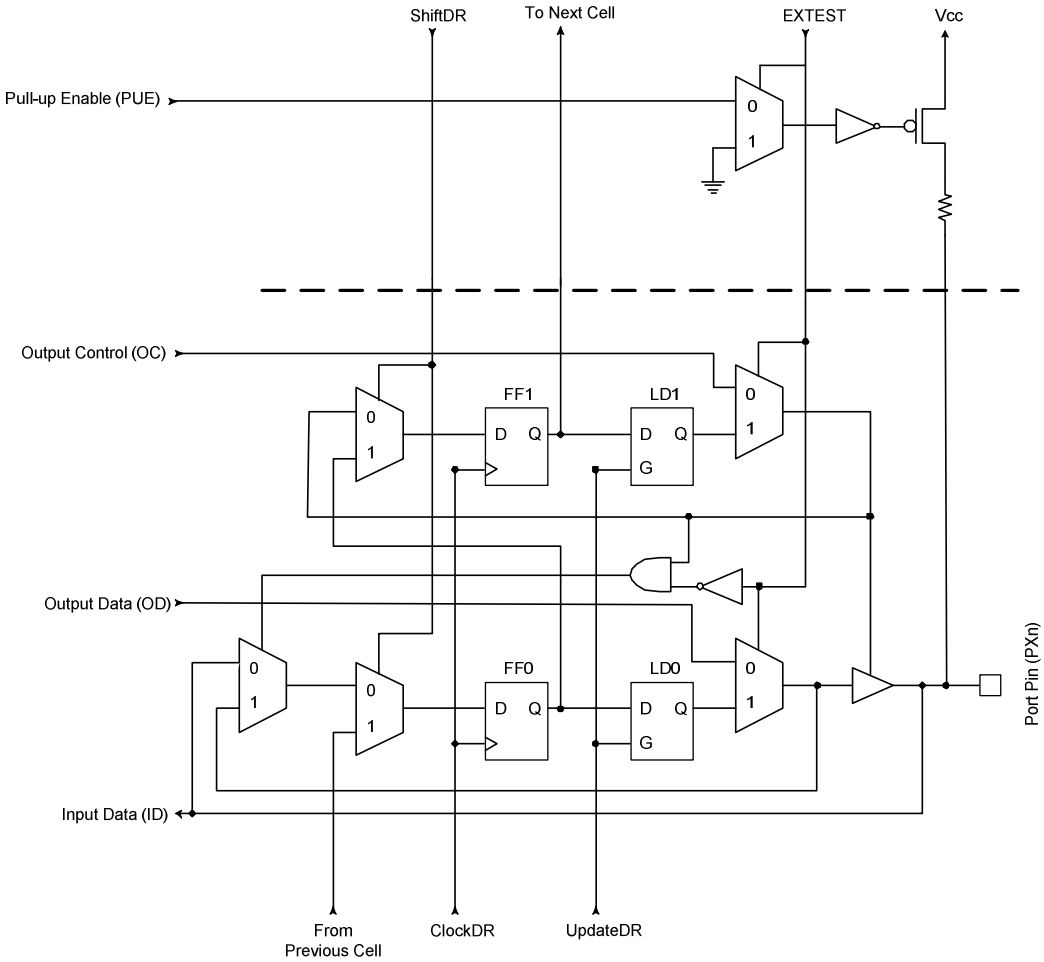
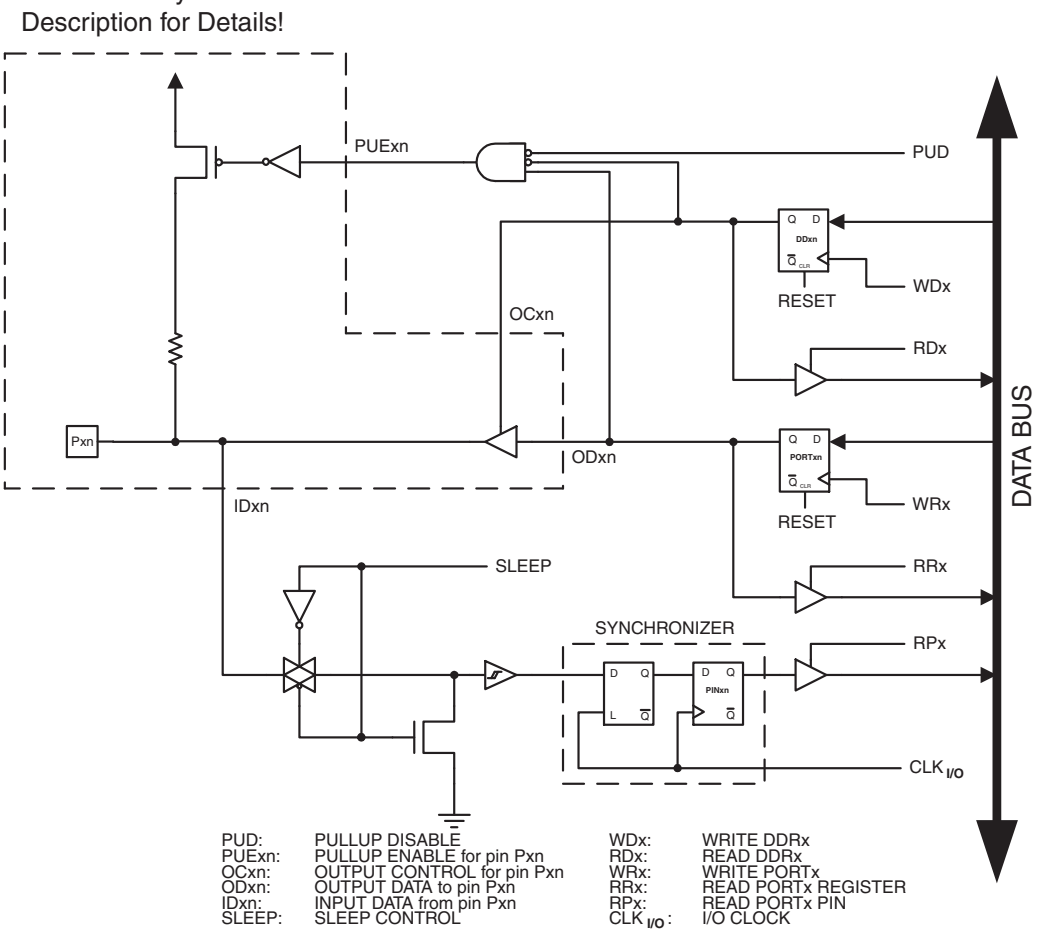


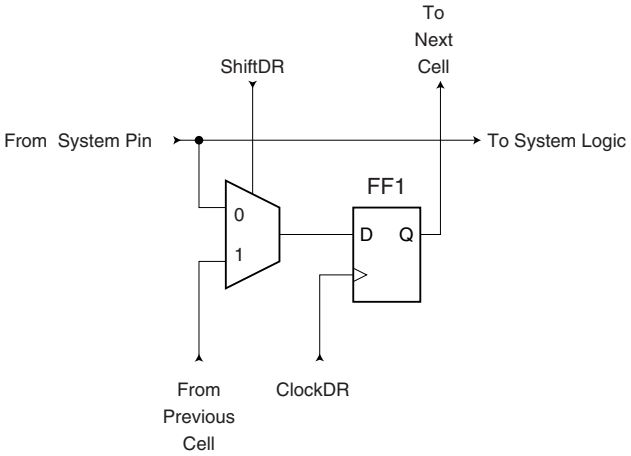
Figure 29-4. General Port Pin Schematic Diagram



29.5.2 Scanning the RSTN, CLKI and TST Pin

An observe-only cell as shown in Figure 29-5 below is inserted for the active low reset signal RSTN, for the active high programming and test mode enable signal TSTN and for the clock input CLKI.

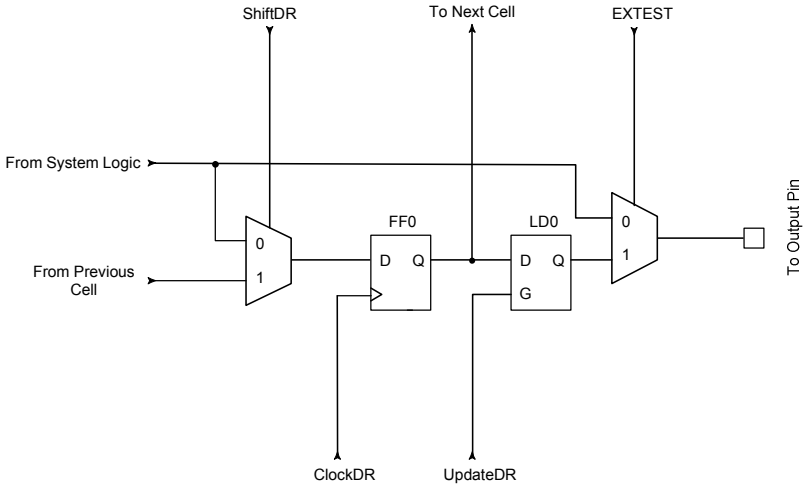
Figure 29-5. Observe-only Cell



29.5.3 Scanning the RSTON Pin

For the low-active reset output pin RSTON a boundary-scan cell as shown in Figure 29-6 below is inserted.

Figure 29-6. Boundary-scan Cell for Output Pins without Pull-up Function



29.6 Boundary-scan Related Register in I/O Memory

29.6.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	JTD								MCUCR
Read/Write	RW								
Initial Value	0								

The MCU Control Register contains control bits for general Microcontroller Unit functions.

- **Bit 7 – JTD - JTAG Interface Disable**

When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

29.6.2 MCUSR – MCU Status Register

Bit	7	6	5	4	3	2	1	0	
$34 ($54)				JTRF					MCUSR
Read/Write				RW					
Initial Value				0					

The MCU Status Register provides information on which reset source caused an MCU reset.

- **Bit 4 – JTRF - JTAG Reset Flag**

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

29.7 Boundary-scan Description Language Files

Boundary-scan Description Language (BSDL) files describe Boundary-scan capable devices in a standard format used by automated test-generation software. The order and function of bits in the Boundary-scan Data Register are included in this description. BSDL files are available for ATmega128RFA1.

29.8 ATmega128RFA1 Boundary-scan Order

Table 29-1 on page 449 shows the Scan order between TDI and TDO when the Boundary-scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order. In Figure 29-3 on page 445, PXn. Data corresponds to FF0, PXn. Control corresponds to FF1, PXn. Bit 4, 5, 6 and 7 of Port F is not in the scan chain, since these pins constitute the TAP pins when the JTAG is enabled.

Table 29-1. ATmega128RFA1 Boundary-Scan Order

Bit Number	Signal Name	Module
0	PF1.Control	Port F
1	PF1.Data	
2	PF0.Control	
3	PF0.Data	
4	PE7.Control	Port E
5	PE7.Data	
6	PE6.Control	
7	PE6.Data	
8	PE5.Control	
9	PE5.Data	
10	PE4.Control	
11	PE4.Data	
12	PE3.Control	
13	PE3.Data	
14	PE2.Control	
15	PE2.Data	
16	PE1.Control	
17	PE1.Data	
18	PE0.Control	
19	PE0.Data	
20	PB7.Control	Port B
21	PB7.Data	
22	PB6.Control	
23	PB6.Data	
24	PB5.Control	
25	PB5.Data	
26	PB4.Control	
27	PB4.Data	
28	PB3.Control	
29	PB3.Data	
30	PB2.Control	
31	PB2.Data	
32	PB1.Control	
33	PB1.Data	
34	PB0.Control	
35	PB0.Data	

Bit Number	Signal Name	Module
36	CLKI.Data	Clock Input (Input Only)
37	PD7.Control	Port D
38	PD7.Data	
39	PD6.Control	
40	PD6.Data	
41	PD5.Control	
42	PD5.Data	
43	PD4.Control	
44	PD4.Data	
45	PD3.Control	
46	PD3.Data	
47	PD2.Control	
48	PD2.Data	
49	PD1.Control	
50	PD1.Data	
51	PD0.Control	
52	PD0.Data	
53	PG5.Control	Port G
54	PG5.Data	
55	PG4.Control	
56	PG4.Data	
57	PG3.Control	
58	PG3.Data	
59	PG2.Control	
60	PG2.Data	
61	PG1.Control	
62	PG1.Data	
63	PG0.Control	
64	PG0.Data	
65	RSTON.Data	Reset Logic Output (Output Only without Pull-up)
66	RSTT.Data	Reset Logic (Observe Only)
67	TST.Data	Test and Programming Mode Enable (Observe Only)
68	PF3.Control	Port F
69	PF3.Data	
70	PF2.Control	
71	PF2.Data	

30 Boot Loader Support – Read-While-Write Self-Programming

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 that (program) 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 (including erasing) 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.

30.1 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 "Table 31-7" on page 467) used during programming. The page organization does not affect normal operation.

30.2 Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections: the Application section and the Boot Loader section (see Figure 30-2 on page 452). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 30-7 on page 461 and Figure 30-2 on page 452. These two sections can have different level of protection since they have different sets of Lock bits.

30.2.1 Application Section

The Application section is the region 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, BLB0), see Table 31-2 on page 464. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

30.2.2 BLS – Boot Loader Section

While the Application section is used for storing the application code, the Boot Loader software must be located in the BLS. The SPM instruction can only initiate programming when executed from the BLS. 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, BLB1), see Table 31-2 on page 464.

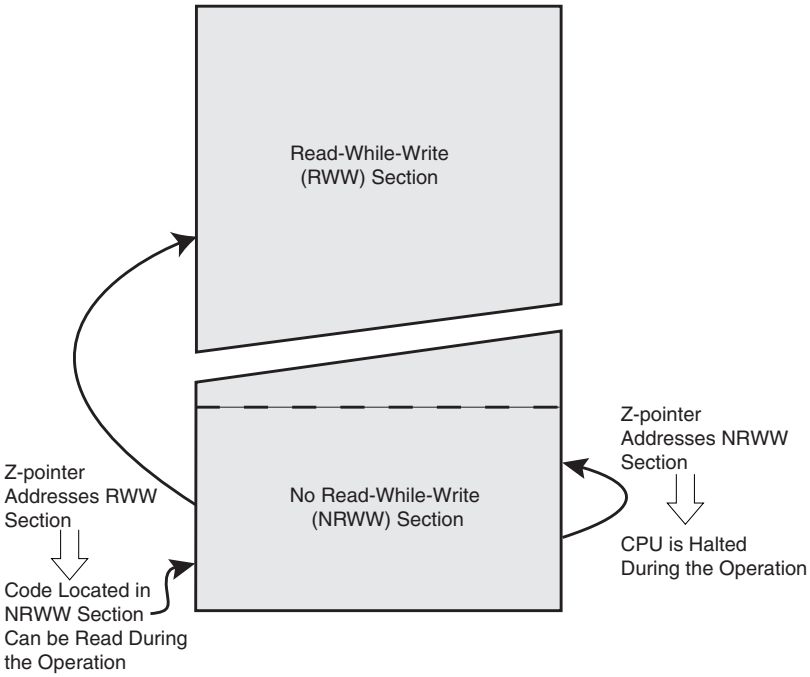
30.3 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on the 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 Table 30-1 on page 452 and Figure 30-1 below. The main differences between the two sections are:

- 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 the section that is being programmed (erased or written) and not to the section that actually is being read during a Boot Loader software update.

Figure 30-1. Read-While-Write vs. No Read-While-Write



30.3.1 RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by load program memory, call, or jump instructions or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a

programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See "SPMCSR – Store Program Memory Control Register" on page 462 for details on how to clear RWWSB.

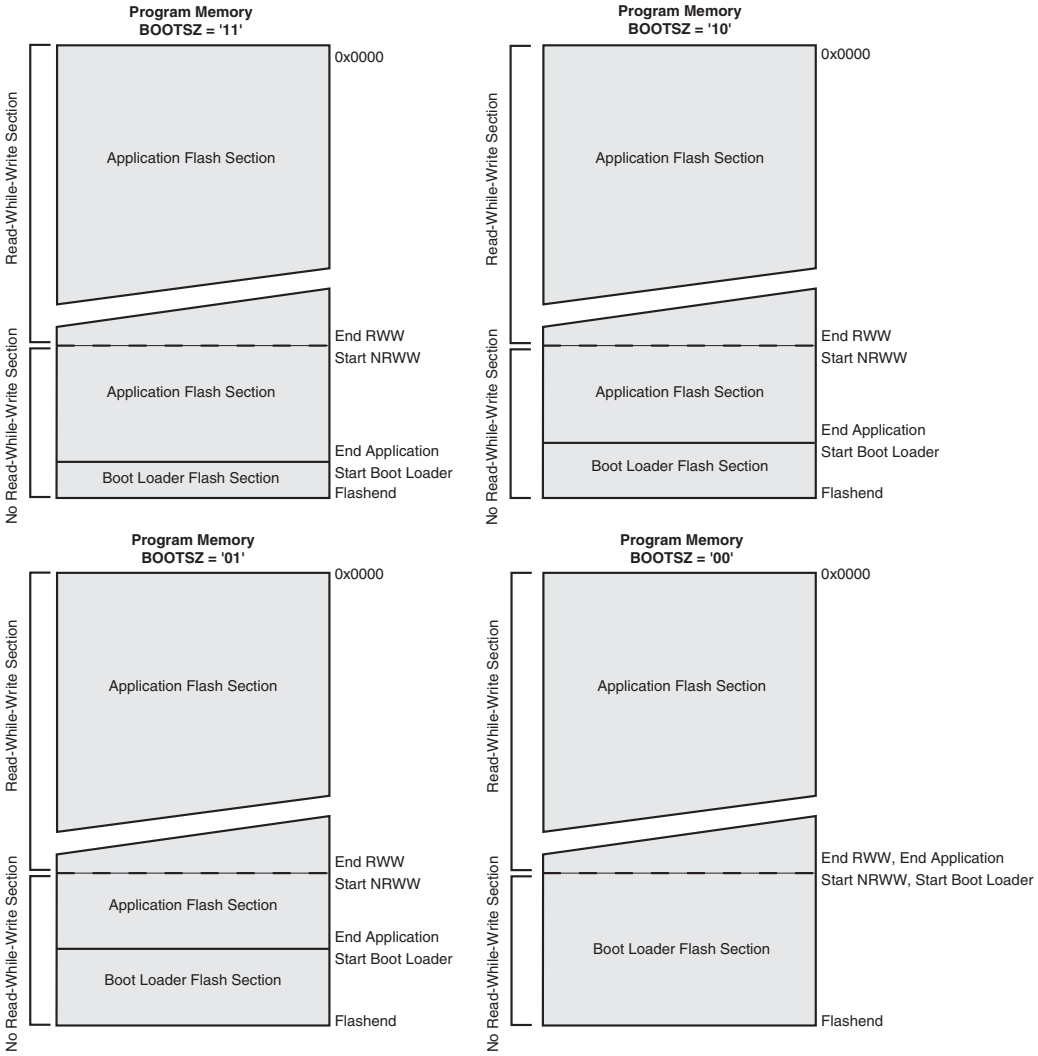
30.3.2 NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire Page Erase or Page Write operation.

Table 30-1. 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 30-2. Memory Sections



Note: 1. The parameters in the figure above are given in Table 30-7 on page 461.

30.4 Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU.
- To protect only the Boot Loader Flash section from a software update by the MCU.
- To protect only the Application Flash section from a software update by the MCU.
- Allow software update in the entire Flash.

See Table 31-2 on page 464 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 (E)LPM/SPM, if it is attempted.

30.4.1 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 USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

Table 30-2. Boot Reset Fuse(1)

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 30-7 on page 461)

Note: 1. “1” means unprogrammed, “0” means programmed

30.5 Addressing the Flash During Self-Programming

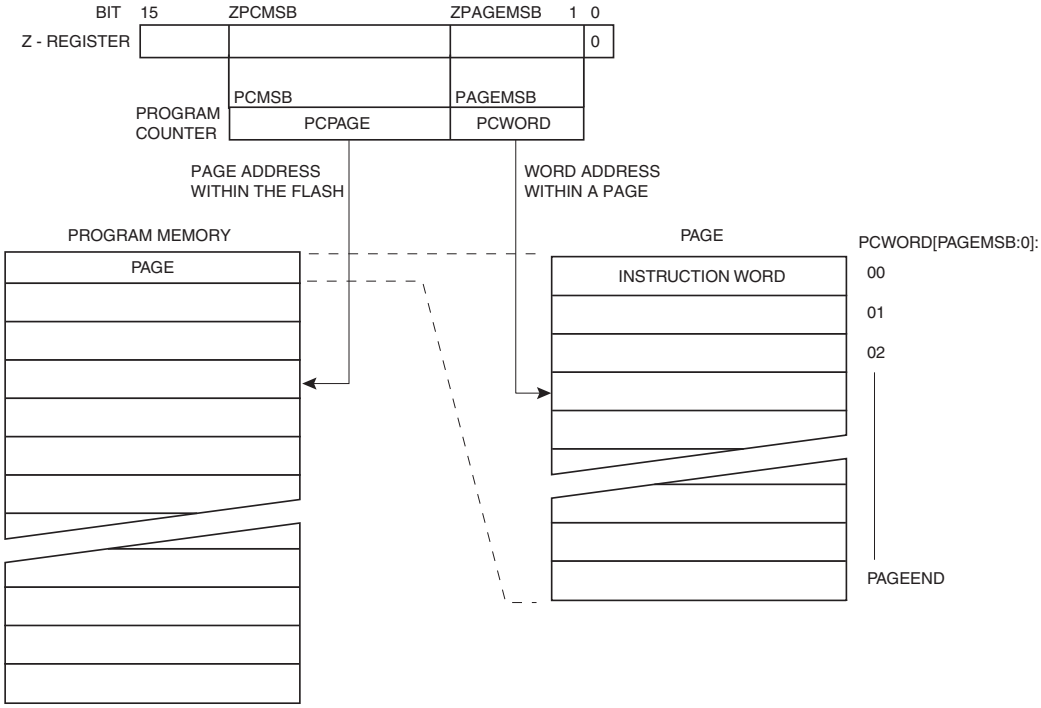
The Z-pointer is used to address the SPM commands. The Z pointer consists of the Z-registers ZL and ZH in the register file, and RAMPZ in the I/O space. The number of bits actually used is implementation dependent. Note that the RAMPZ register is only implemented when the program space is larger than 64K bytes.

Bit	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8
RAMPZ							**RAMPZ1**	**RAMPZ0**
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see "Table 31-7" on page 467), 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 30-3 below. 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 (E)LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also bit Z0 of the Z-pointer is used.

Figure 30-3. Addressing the Flash during SPM



Note: 1. The different variables used in Figure 30-3 above are listed in Table 30-6 on page 461.

30.6 Self-Programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

Alternative 1, fill the buffer before a Page Erase

- Fill temporary page buffer,
- Perform a Page Erase,
- Perform a Page Write;

Alternative 2, fill the buffer after Page Erase

- Perform a Page Erase,

- Fill temporary page buffer,
- Perform a Page Write;

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the Boot Loader provides an effective Read-Modify-Write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page. For an assembly code example see "Simple Assembly Code Example for a Boot Loader" on page 458.

30.6.1 Performing Page Erase by SPM

To execute Page Erase, set up the address in the Z-pointer, write “X0000011” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- Page Erase to the RWW section: The NRWW section can be read during the Page Erase.
- Page Erase to the NRWW section: The CPU is halted during the operation.

30.6.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will be auto-erased after a Page Write operation or by writing the RWWSRE bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded is still buffered.

30.6.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “X0000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- Page Write to the RWW section: The NRWW section can be read during the Page Write.
- Page Write to the NRWW section: The CPU is halted during the operation.

30.6.4 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 "Interrupts" on page 211.

30.6.5 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 un-programmed. 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.

30.6.6 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 211, or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See "Simple Assembly Code Example for a Boot Loader" on page 458 for an example.

30.6.7 Setting the Boot Loader Lock Bits by SPM

To set the Boot Loader Lock bits and general Lock bits, 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 Table 31-2 on page 464 for how the different settings of the Boot Loader 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 BLBSET and SPMEN are set in SPMCSR. The Z-pointer is don’t care during this operation, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the Lock 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.

30.6.8 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Signature Row, Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

30.6.9 Reading the Fuse and Lock Bits from Software

It is possible to read both the Fuse and Lock bits from software. To read the Lock bits, load the Z-pointer with 0x0001 and set the BLBSET and SPMEN bits in SPMCSR. When an (E)LPM instruction is executed within three CPU cycles after the BLBSET and SPMEN bits are set in SPMCSR, the value of the Lock bits will be loaded in the destination register. The BLBSET and SPMEN bits will auto-clear upon completion of reading the Lock bits or if no (E)LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When BLBSET and SPMEN are cleared, (E)LPM will work as described in the Instruction Set Manual.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	BLB12	BLB11	BLB02	BLB01	LB2	LB1

The algorithm for reading the Fuse Low byte is similar to the one described above for reading the Lock bits. To read the Fuse Low byte, load the Z-pointer with 0x0000 and set the BLBSET and SPMEN bits in SPMCSR. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Fuse Low byte (FLB) will be loaded in the destination register as shown on the next page. Refer to (see "Table 31-5" on page 466) 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, load 0x0003 in the Z-pointer for reading the Fuse High byte. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Fuse High byte (FHB) will be loaded in the destination register as shown below. Refer to "Table 31-4" on page 465 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

Load 0x0002 in the Z-pointer for reading the Extended Fuse byte. When an (E)LPM instruction is executed within three cycles after the BLBSET and SPMEN bits are set in the SPMCSR, the value of the Extended Fuse byte (EFB) will be loaded in the destination register as shown below. Refer to Table 31-3 on page 465 for detailed description and mapping of the Extended Fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd	-	-	-	-	-	EFB2	EFB1	EFB0

Fuse and Lock bits that are programmed will be read as zero. Fuse and Lock bits that are un-programmed will be read as one.

30.6.10 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 30-3 below and set the SIGRD and SPMEN bits in SPMCSR. When a LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the Signature Row or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction Set Manual. The Signature Row cannot be read during an EEPROM write/erase operation.

Table 30-3. Signature Row Addressing

Signature Byte	Z-Pointer Address
Device Signature Byte 1	0x0000
Device Signature Byte 2	0x0002
Device Signature Byte 3	0x0004
RC Oscillator Calibration Byte	0x0001

Note: 2. All other addresses are reserved for future use.

30.6.11 Preventing Flash Corruption

During periods of V_{DEVDD}<1.8V, the Flash program can be corrupted because the supply voltage is too low for the CPU and the Flash to operate properly. These issues are the same as for board level systems using Flash, and the same design solutions should be applied.

A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
2. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{DEVDD} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed under the condition that the power supply voltage is sufficient.
3. Keep the AVR core in Power-down sleep mode during periods of low V_{DEVDD}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

30.6.12 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 30-4 below shows the typical programming time for Flash accesses from the CPU.

Table 30-4. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms
Flash write (Page Erase)	7.3 ms	8.9 ms

30.6.13 Simple Assembly Code Example for a Boot Loader

Assembly Code Example[1]

```
;-the routine writes one page of data from RAM to Flash
; the first data location in RAM is pointed to by the Y pointer
; the first data location in Flash is pointed to by the Z-pointer
;-error handling is not included
;-the routine must be placed inside the Boot space
; (at least the Do_spm sub routine). Only code inside NRWW section
; can be read during Self-Programming (Page Erase and Page Write).
;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
; loophi (r25), spmcrval (r20)
; storing and restoring of registers is not included in the routine
; register usage can be optimized at the expense of code size
;-It is assumed that either the interrupt table is moved to the
; Boot loader section or that the interrupts are disabled.
```

Assembly Code Example[1]

```
.equ PAGESIZEB=PAGESIZE*2 ;PAGESIZEB is page 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:
  elpm 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
```

Assembly Code Example[1]

```
    ; If RWWSB is set, the RWW section is not ready yet
    sbrs temp1, RWWSB
    ret
    ; re-enable the RWW section
    ldi spmcrval, (1<<RWWSRE) | (1<<SPMEN)
    call Do_spm
    rjmp Return
Do_spm:
    ; check for previous SPM complete
Wait_spm:
    in temp1, SPMCSR
    sbrc temp1, SPMEN
    rjmp Wait_spm
    ; input: spmcrval determines SPM action
    ; disable interrupts if enabled, store status
    in temp2, SREG
    cli
    ; check that no EEPROM write access is present
Wait_ee:
    sbic EECR, EEPE
    rjmp Wait_ee
    ; SPM timed sequence
    out SPMCSR, spmcrval
    spm
    ; restore SREG (to enable interrupts if originally enabled)
    out SREG, temp2
    ret
```

Notes: 1. See "About Code Examples" on page 7.

30.6.14 Boot Loader Parameters for 128kByte of Flash Memory

In Table 30-7 on page 461 through Table 30-6 on page 461, the parameters used in the description of the Self-Programming are given.

Table 30-5. Read-While-Write Limit with 128kByte of Flash Memory

Section[1]	Pages	Address
Read-While-Write section (RWW)	480	0x0000 – 0xEFFF
No Read-While-Write section (NRWW)	32	0xF000 – 0xFFFF

Note: 1. For details about these two sections see "NRWW – No Read-While-Write Section" on page 452 .

Table 30-6. Explanation of different variables used in Figure 30-3 on page 454 and the mapping to the Z-pointer for 128kByte of Flash Memory

Variable	Value	Corresponding Z-value[2]	Description[1]
PCMSB	15		Most significant bit in the Program Counter. (The Program Counter is 16 bits PC[15:0])
PAGEMSB	6		Most significant bit which is used to address the words within one page (128 words in a page requires seven bits PC [6:0]).
ZPCMSB		Z16[3]	Bit in Z-pointer that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z7	Bit in Z-pointer that is mapped to PCMSB. Because Z0 is not used; the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[15:7]	Z16[3]:Z8	Program Counter page address: Page select, for Page Erase and Page Write.
PCWORD	PC[6:0]	Z7:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

Notes:
1. Z0: should be zero for all SPM commands, byte select for the (E)LPM instruction.
2. See "Addressing the Flash During Self-Programming" on page 453 for details about the use of Z-pointer during Self-Programming.
3. The Z-register is only 16 bits wide. Bit 16 is located in the RAMPZ register in the I/O map.

Table 30-7. Boot Size Configuration with 128kByte of Flash Memory[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	512 words	4	0x0000 – 0xFDFF	0xFE00 – 0xFFFF	0xFDFF	0xFE00
1	0	1024 words	8	0x0000 – 0xFBFF	0xFC00 – 0xFFFF	0xFBFF	0xFC00
0	1	2048 words	16	0x0000 – 0xF7FF	0xF800 – 0xFFFF	0xF7FF	0xF800
0	0	4096 words	32	0x0000 – 0xEFFF	0xF000 – 0xFFFF	0xEFFF	0xF000

Note:
1. The different BOOTSZ Fuse configurations are shown in Figure 30-2 on page 452.

30.7 Register Description

30.7.1 SPMCSR – Store Program Memory Control Register

Bit	7	6	5	4	3	2	1	0	
$37 ($57)	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	RW	R	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

The Store Program Memory Control Register contains the control bits needed to control the Boot Loader operations. Note: Only one SPM instruction should be active at any time.

- **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 SPMCR 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. A SPM instruction within four cycles after SIGRD and SPMEN are set, will have no effect. This operation is reserved for future use and should not be used.

- **Bit 4 – RWWSRE - Read While Write Section Read Enable**

When programming (page erase or page write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the Flash is busy with a page erase or a page write (SPMEN is set). If the RWWSRE bit is written while the Flash is being loaded, the Flash load operation will abort and the data loaded will be lost.

- **Bit 3 – BLBSET - Boot Lock Bit Set**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets Boot Lock bits, according to the data in R0. The data in R1 and the address in the Z pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the lock bit set, or if no SPM instruction is executed within four clock cycles. A LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCR register, will read either the Lock-bits or the Fuse bits (depending on Z0 in the Z pointer) into the destination register.

- **Bit 2 – PGWRT - Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM

instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 1 – PGERS - Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page erase. The page address is taken from the high part of the Z pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 0 – SPMEN - Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLB-SET, PGWRT or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z pointer. The LSB of the Z pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remain high until the operation is completed. Writing any other combination than "10001", "01001", "00101", "00011" or "00001" in the lower five bits will have no effect.

30.7.2 NEMCR – Flash Extended-Mode Control-Register

Bit	7	6	5	4	3	2	1	0	
NA ($75)	Resx7	ENEAM	AEAM1	AEAM0	Resx3	Resx2	Resx1	Resx0	NEMCR
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	1	0	1	0	

The Flash Extended-Mode Control-Register handles the extended address-mode of the extra rows.

- **Bit 7 – Resx7 - Reserved**
- **Bit 6 – ENEAM - Enable Extended Address Mode for Extra Rows**

When active high, the extended address mode of the extra rows is enabled. The address is decoded from bits AEAM1:0 of this register.

- **Bit 5:4 – AEAM1:0 - Address for Extended Address Mode of Extra Rows**

These bits are only used when bit ENEAM of this register is set high. Then AEAM1:0 are used to decode the addresses of the extra rows. A value of 0 decodes the default factory row that is also accessible when the extended address mode is deactivated.

Table 30-8 AEAM Register Bits

Register Bits	Value	Description
AEAM1:0	0	Factory Row
	1	User Row 1
	2	User Row 2
	3	User Row 3

- **Bit 3:0 – Resx3:0 - Reserved**

31 Memory Programming

31.1 Program And Data Memory Lock Bits

The ATmega128RFA1 provides six Lock bits which can be left un-programmed ("1") or can be programmed ("0") to obtain the additional features listed in Table 31-2 below. The Lock bits can only be erased to "1" with the Chip Erase command.

Table 31-1. Lock Bit Byte [1]

Lock Bit Byte	Bit No	Description	Default Value
–	7	–	1 (un-programmed)
–	6	–	1 (un-programmed)
BLB12	5	Boot Lock bit	1 (un-programmed)
BLB11	4	Boot Lock bit	1 (un-programmed)
BLB02	3	Boot Lock bit	1 (un-programmed)
BLB01	2	Boot Lock bit	1 (un-programmed)
LB2	1	Lock bit	1 (un-programmed)
LB1	0	Lock bit	1 (un-programmed)

Note: 1. "1" means un-programmed, "0" means programmed.

Table 31-2. 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	**BL02**	**BL01**	
1	1	1	No restrictions for SPM or (E)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 (E)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	(E)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.

Memory Lock Bits			Protection Type
BLB1 Mode	BL12	BL11	
1	1	1	No restrictions for SPM or (E)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 (E)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	(E)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 un-programmed, "0" means programmed.

31.2 Fuse Bits

The ATmega128RFA1 has three Fuse bytes. Table 31-3 below – Table 31-5 on page 466 describe briefly the functionality of all the fuses and how they are mapped into the Fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 31-3. Extended Fuse Byte

Fuse Low Byte	Bit No	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	3	–	1
BODLEVEL2(1)	2	Brown-out Detector trigger level	1 (un-programmed)
BODLEVEL1(1)	1	Brown-out Detector trigger level	1 (un-programmed)
BODLEVEL0(1)	0	Brown-out Detector trigger level	1 (un-programmed)

Notes: 1. See Table 34-23 on page 503 for BODLEVEL Fuse decoding.

Table 31-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
OCDEN(4)	7	Enable On-chip debugging (OCD)	1 (un-programmed, OCD disabled)
JTAGEN	6	Enable JTAG interface	0 (programmed, JTAG enabled)
SPIEN(1)	5	Enable Serial Program and Data Downloading (SPI)	0 (programmed, SPI programming enabled)
WDTON(3)	4	Watchdog Timer always on	1 (un-programmed)

Fuse High Byte	Bit No	Description	Default Value
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (un-programmed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 30-7 on page 461 for details)	0 (programmed)(2)
BOOTSZ0	1	Select Boot Size (see Table 30-7 on page 461for details)	0 (programmed) (2)
BOOTRST	0	Select Reset Vector	1 (un-programmed)

Notes:
1. The SPIEN Fuse is not accessible in serial programming mode.
2. The default value of BOOTSZ1:0 results in maximum Boot Size. See Table 30-7 on page 461 for details.
3. See "WDTCSR – Watchdog Timer Control Register" on page 183 for details.
4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.

Table 31-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8(4)	7	Divide clock by 8	0 (programmed)
CKOUT(3)	6	Clock output	1 (un-programmed)
SUT1	5	Select start-up time	1 (un-programmed)(1)
SUT0	4	Select start-up time	0 (programmed) (1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed) (2)
CKSEL1	1	Select Clock source	1 (un-programmed) (2)
CKSEL0	0	Select Clock source	0 (programmed) (2)

Notes:
1. The default value of SUT1:0 results in maximum start-up time for the default clock source. See "System Control and Reset" on page 176 for details.
2. The default setting of CKSEL3:0 results in internal RC Oscillator @ 8 MHz. See "Table 11-1" on page 148 for details.
3. The CKOUT Fuse allows the system clock to be output on PORTE7. See "Clock Output Buffer" on page 152 for details.
4. See "System Clock Prescaler" on page 152 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

31.2.1 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.

31.3 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked.

The three bytes reside in a separate address space. For the ATmega128RFA1 the signature bytes are given in Table 31-6 below. Accessing the signature bytes from software is described in section "Reading the Signature Row from Software" on page 457.

Table 31-6. Device and JTAG ID

Part	Signature Byte Number			JTAG	
	0	1	2	Part Number	Manufacturer ID
ATmega128RFA1	0x1E	0xA7	0x01	0xA701	0x1F

31.4 Calibration Byte

The ATmega128RFA1 has a byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

31.5 Page Size

Table 31-7. Number of Words in a Page and Number of Pages in the Flash

Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
64k words (128k bytes)	128 words	PC[6:0]	512	PC[15:7]	15

Table 31-8. Number of Bytes in a Page and Number of Pages in the EEPROM

EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
4k bytes	8 bytes	EEA[2:0]	512	EEA[11:3]	11

31.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega128RFA1.

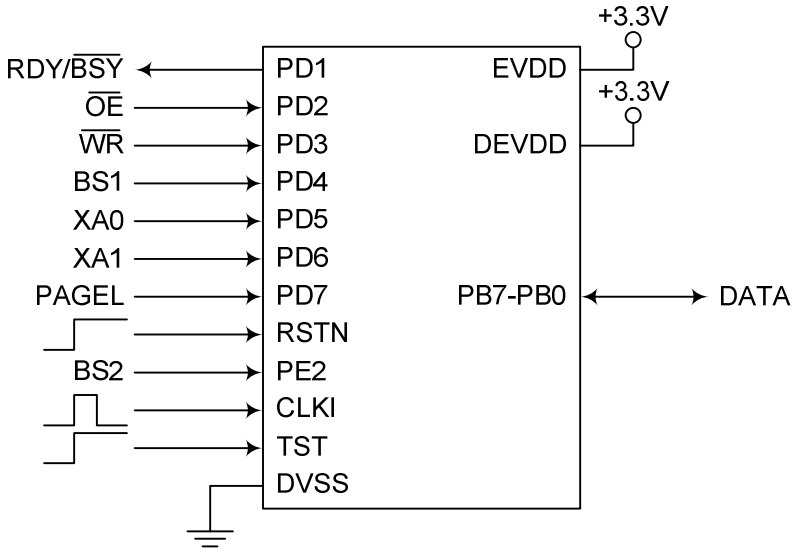
31.6.1 Signal Names

In this section, some pins of the ATmega128RFA1 are referenced by signal names describing their functionality during parallel programming; see Figure 31-1 on page 468 and Table 31-9 on page 468. Pins not described in this table are referenced by their default pin names.

The XA1/XA0 pins determine the action executed when the CLKI pin is given a positive pulse. The bit coding is shown in Table 31-12 on page 469.

When pulsing $\overline{WR}$ or $\overline{OE}$ or, the command loaded determines the action executed. The different commands are shown in Table 31-13 on page 469.

Figure 31-1. Parallel Programming [1]



Note: 1. Unused Pins should be left floating.

Table 31-9. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{\text{BSY}}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command.
$\overline{\text{OE}}$	PD2	I	Output Enable (Active low).
$\overline{\text{WR}}$	PD3	I	Write Pulse (Active low).
BS1	PD4	I	Byte Select 1.
XA0	PD5	I	XTAL Action Bit 0.
XA1	PD6	I	XTAL Action Bit 1.
PAGEL	PD7	I	Program Memory and EEPROM data Page Load.
BS2	PE2	I	Byte Select 2.
DATA	PB7-0	I/O	Bi-directional Data bus (Output when $\overline{\text{OE}}$ is low).

Table 31-10. BS2 and BS1 Encoding

BS2	BS1	Flash / EEPROM Address	Flash Data Loading / Reading	Fuse Programming	Reading Fuse and Lock Bits
0	0	Low Byte	Low Byte	Low Byte	Fuse Low Byte
0	1	High Byte	High Byte	High Byte	Lock Bits
1	0	Extended High Byte	Reserved	Extended Byte	Extended Fuse Byte
1	1	Reserved	Reserved	Reserved	Fuse High Byte

Table 31-11. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Table 31-12. XA1 and XA0 Encoding

XA1	XA0	Action when CLKI is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS2 and BS1).
0	1	Load Data (High or Low data byte for Flash determined by BS1).
1	0	Load Command.
1	1	No Action, Idle.

Table 31-13. Command Byte Bit Encoding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature bytes and Calibration byte
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

31.7 Parallel Programming

Pulses of CLKI and in the following command sequences are assumed to be at least 250 ns wide unless otherwise noted.

31.7.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 3.3V between DEVDD and DVSS.
2. Set RSTN to 0 and TST to 0.
3. Set the Prog_enable pins listed in Table 31-11 above to “0000” and wait at least 100ns.
4. Set TST to 1. TST can be set high any time before but not after the rising edge of RSTN (t_{TSTRNH}).
5. Set RSTN to 1. Any activity on Prog_enable pins within 100 ns after RSTN is set to 1 will cause the device to fail entering programming mode.
6. Wait at least 50 µs before sending a command.

31.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

31.7.3 Chip Erase

The Chip Erase will erase the Flash and EEPROM [(1)] memories plus Lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

Note: 1. The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give CLKI a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/ $\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

31.7.4 Programming the Flash

The Flash is organized in pages; see Table 31-7 on page 467. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give CLKI a positive pulse. This loads the command.

B. Load Address Low byte (Address bits 7:0)

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS2, BS1 to “00”. This selects the address low byte.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give CLKI a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (see Figure 31-3 on page 472 for signal waveforms).

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 31-5 on page 472. Note that if less than eight bits are required to address words in the page (page size < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

G. Load Address High byte (Address bits15:8)

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS2, BS1 to “01”. This selects the address high byte.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give CLKI a positive pulse. This loads the address high byte.

H. Program Page

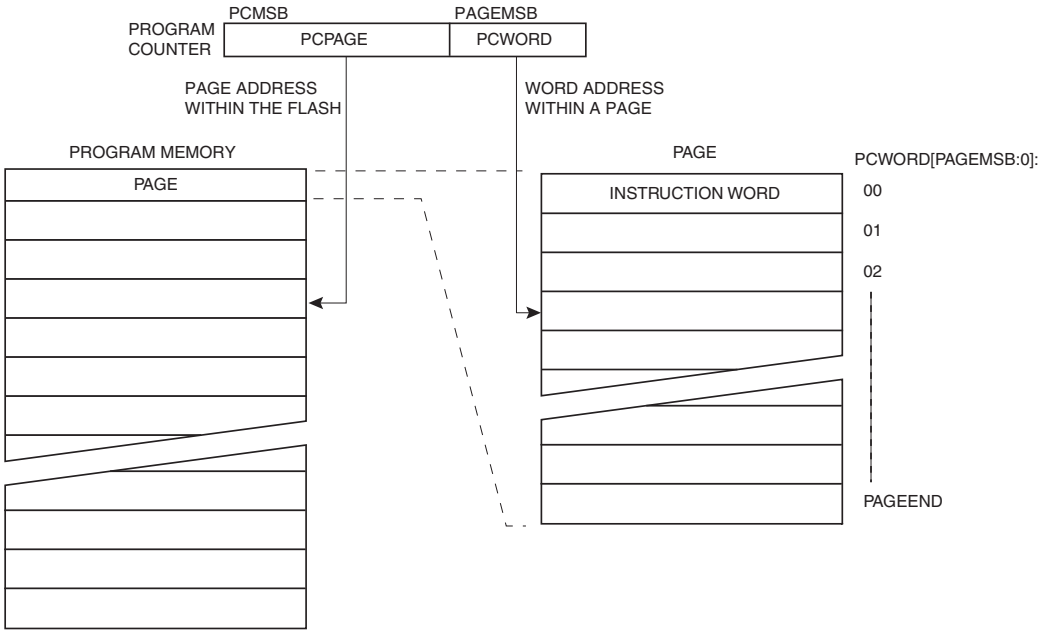
1. Set BS2, BS1 to “00”
2. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high (See Figure 31-3 on page 472 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

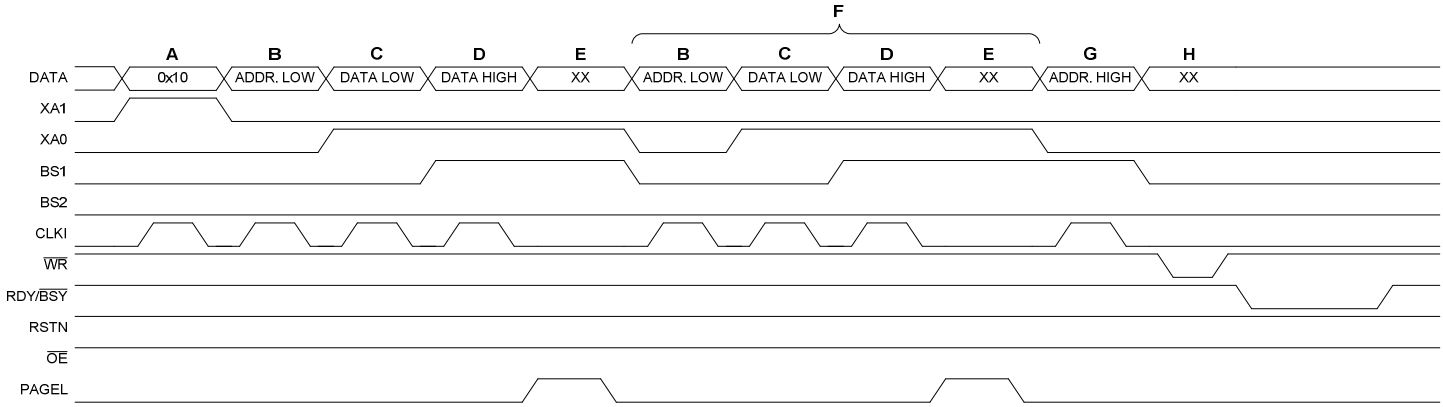
1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give CLKI a positive pulse. This loads the command, and the internal write signals are reset.

Figure 31-5. Addressing the Flash which is Organized in Pages [1]



Note: 1. PCPAGE and PCWORD are listed in Table 31-7 on page 467.

Figure 31-3. Programming the Flash Waveforms [1]



Note: 1. “XX” is don’t care. The letters refer to the programming description above.

31.7.5 Programming the EEPROM

The EEPROM is organized in pages; see Table 31-8 on page 467. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" on page 470 for details on Command, Address and Data loading):

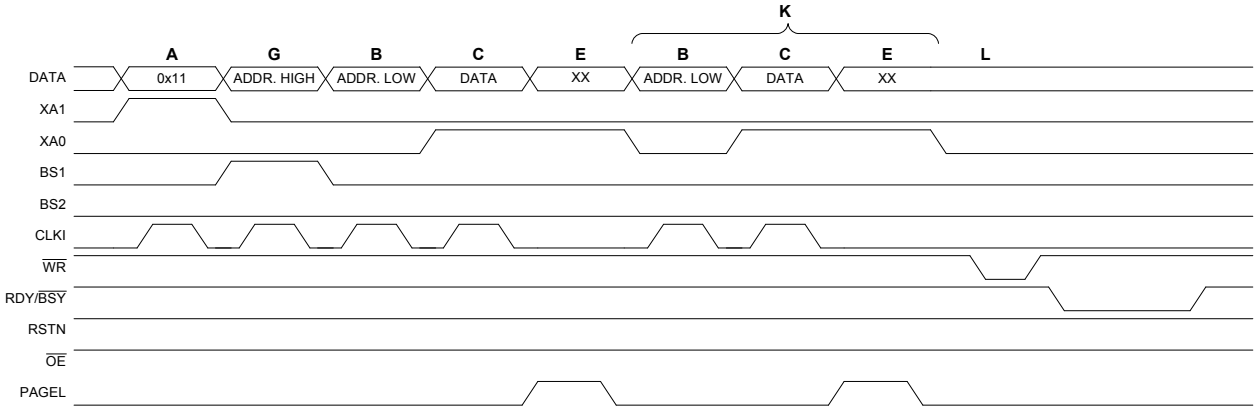
1. A: Load Command “0001 0001”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. C: Load Data (0x00 - 0xFF).
5. E: Latch data (give PAGEL a positive pulse).

K: Repeat 3 through 5 until the entire buffer is filled.

L: Program EEPROM page

1. Set BS2, BS1 to “00”.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 31-7 below for signal waveforms).

Figure 31-7. Programming the EEPROM Waveforms



31.7.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" on page 470 for details on Command and Address loading):

1. A: Load Command “0000 0010”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
5. Set BS1 to “1”. The Flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

31.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" on page 470 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

31.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0100 0000”.

2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

31.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

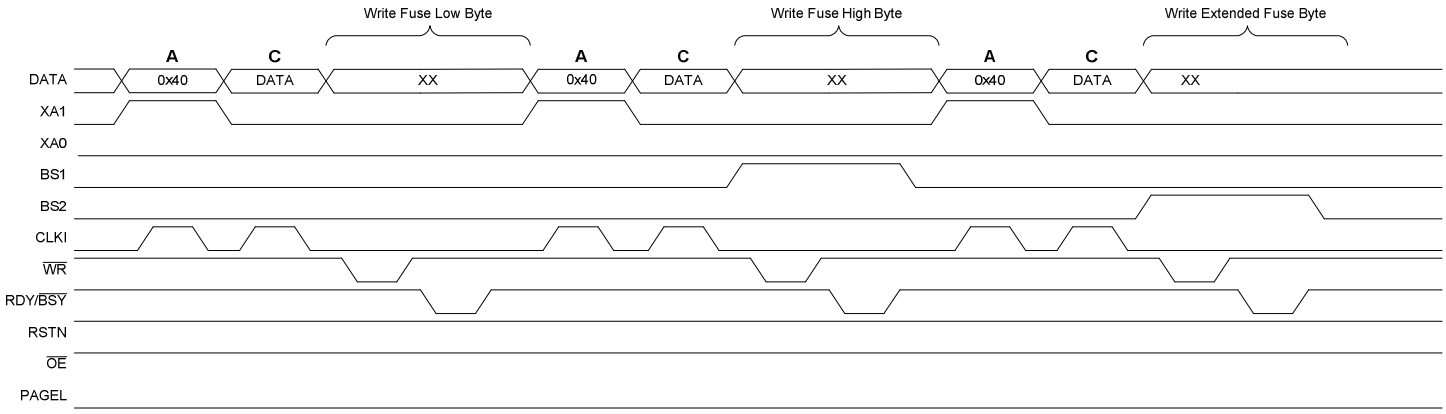
1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS2, BS1 to “01”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2, BS1 to “00”. This selects low data byte.

31.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS2, BS1 to “10”. This selects extended data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2, BS1 to “00”. This selects low data byte.

Figure 31-8. Programming the Fuses Waveforms



31.7.11 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is active (LB1 and LB2 are programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

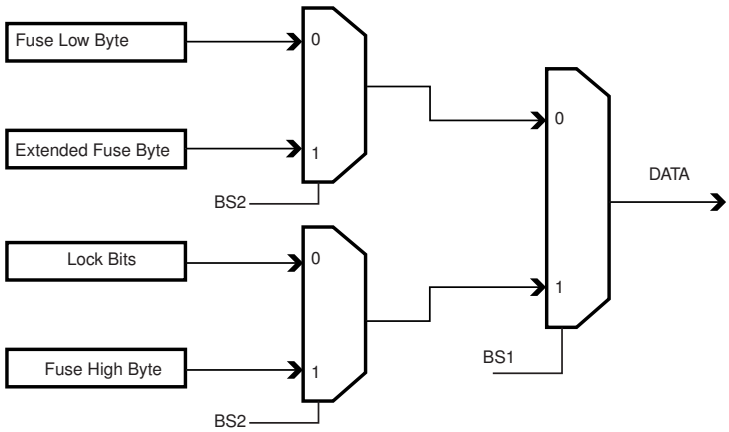
The Lock bits can only be cleared by executing Chip Erase.

31.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, and BS2, BS1 to “00”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, and BS2, BS1 to “11”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set $\overline{OE}$ to “0”, and BS2, BS1 to “10”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, and BS2, BS1 to “01”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 31-9. Mapping between BS1, BS2 and the Fuse and Lock Bits during Read



31.7.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0” and BS to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

31.7.14 Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to "Programming the Flash" on page 470 for details on Command and Data loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0” and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

31.7.15 Parallel Programming Characteristics

Figure 31-10. Parallel programming timing including some general timing requirements

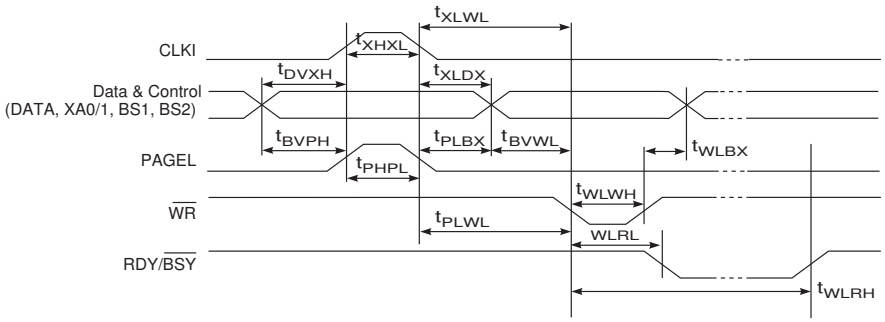
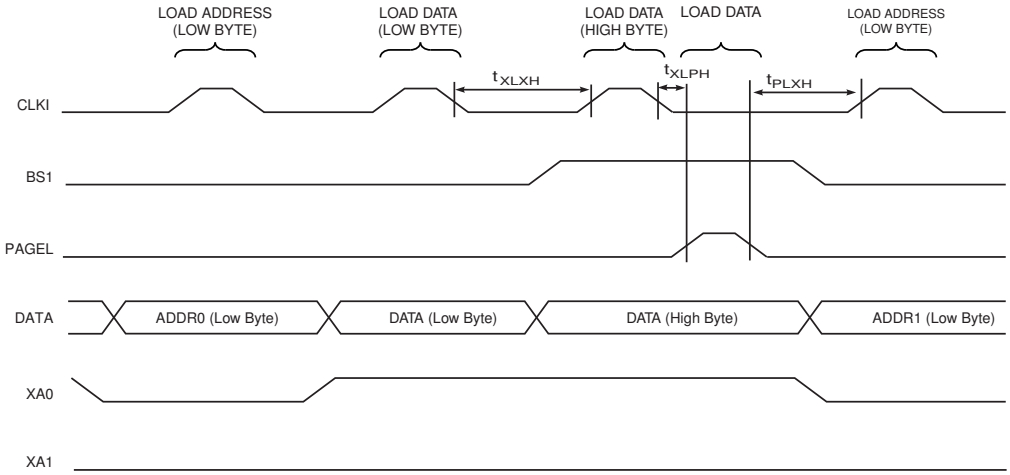
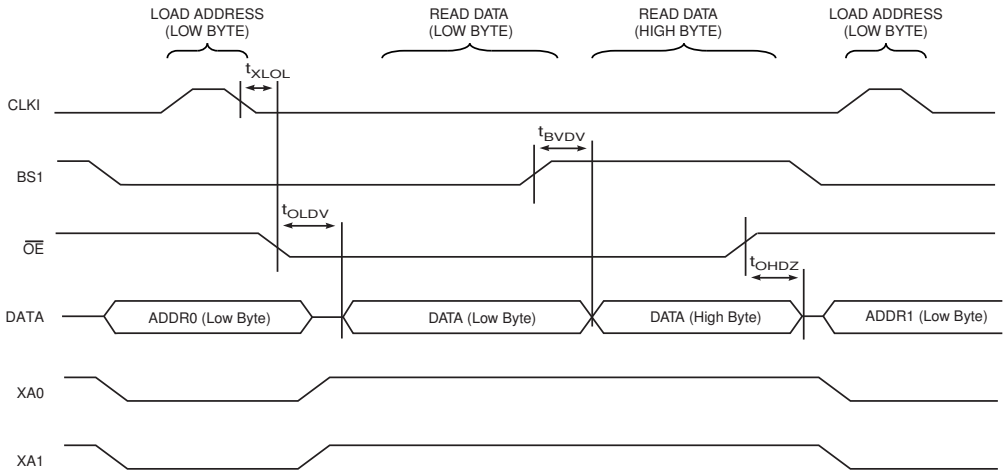


Figure 31-11. Parallel programming loading sequence with timing requirements (1)



Note: 1. The timing requirements shown in Figure 31-10 above (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 31-12. Parallel programming reading sequence (within the same page) with timing requirements (1)



Note: 1. The timing requirements shown in Figure 31-10 above (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 31-14. Parallel Programming Characteristics, V_{DEVDD} = 3.3V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
t_{TSTRNH}	Delay TST High before RSTN High	0			ns
t_{DVXH}	Data and Control Valid before CLKI High	67			ns
t_{XLXH}	CLKI Low to CLKI High	200			ns
t_{XHXL}	CLKI Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after CLKI Low	67			ns
t_{XLWL}	CLKI Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	CLKI Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to CLKI high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS2/1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1)	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2)	12		14.5	ms
t_{XLOL}	CLKI Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

31.8 Serial Downloading

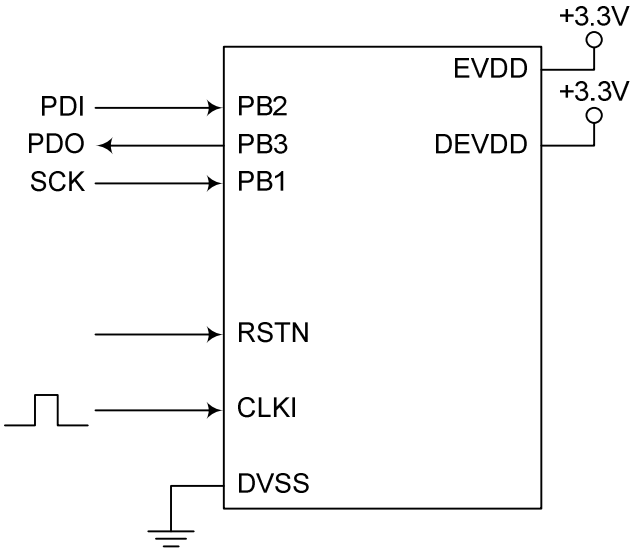
Both the Flash and EEPROM memory arrays can be programmed using a serial programming bus while RSTN is pulled to DVSS. The serial programming interface consists of pins SCK, PDI (input) and PDO (output). After RSTN is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 31-15 below, the pin mapping for serial programming is listed.

31.8.1 Serial Programming Pin Mapping

Table 31-15. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
PDI	PB2	I	Serial Data In
PDO	PB3	O	Serial Data Out
SCK	PB1	I	Serial Clock

Figure 31-13. Serial Programming and Verify [(1)(2)]



Notes: 1. If the device is clocked by the internal Oscillator, it is not required to connect a clock source to the CLKI pin.

2. V_{DEVDD}-0.3V < V_{EVDD} < V_{DEVDD}+0.3V, both V_{EVDD} and V_{DEVDD} must stay in valid supply voltage limits.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz;

High: > 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz;

31.8.2 Serial Programming Algorithm

When writing serial data to the ATmega128RFA1, data is clocked on the rising edge of SCK.

When reading data from the ATmega128RFA1, data is clocked on the falling edge of SCK. See Figure 31-15 on page 481 for timing details.

To program and verify the ATmega128RFA1 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 31-17 on page 479):

1. Power-up sequence: Apply power between DEVDD and DVSS while RSTN 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, RSTN 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 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin PDI.
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 RSTN a positive pulse and issue a new Programming Enable command.

4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 7 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 address lines 15:8. If polling (RDY/BSY) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page (see Table 31-16 below). Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte (see Table 31-16 below). In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output PDO.
7. At the end of the programming session, RSTN can be set high to commence normal operation.
8. Power-off sequence (if needed): Set RESET to “1”. Turn DEVDD power off.

Table 31-16. Minimum Wait Delay before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9 ms
$t_{WD_CHIPERASE}$	14.5 ms

31.8.3 Serial Programming Instruction Set

Table 31-17 below and Figure 31-14 on page 480 describe the Instruction set.

Table 31-17. Serial Programming Instruction Set [(4)(5)]

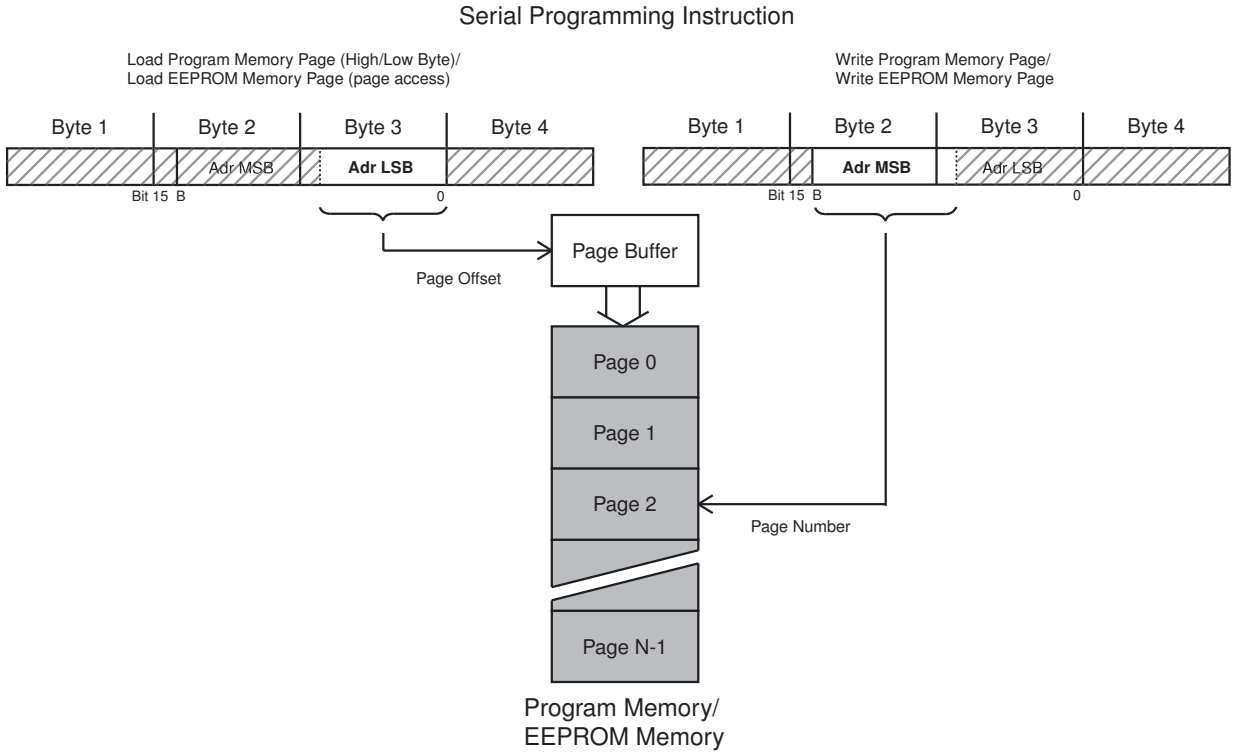
Instruction/Operation	Instruction Format [(1)]			
	Byte1	Byte2	Byte3	Byte4
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/BSY	$F0	$00	$00	data byte out
Load Instruction				
Load Program Memory Page, High Byte	$48	$00	addr. LSB	high data byte in
Load Program Memory Page, Low Byte	$40	$00	addr. LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	0000 000aa	data byte in
Read Instruction				
Read Program Memory, High byte	$28	addr. MSB	addr. LSB	high data byte out
Read Program Memory, Low byte	$20	addr. MSB	addr. LSB	low data byte out
Read EEPROM Memory	$A0	0000 aaaa	aaaa aaaa	data byte out
Read Lock Bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	0000 000aa	data byte out

	Instruction Format [1]			
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 [2][3]				
Write Program Memory Page	$4C	addr. MSB	addr. LSB	$00
Write EEPROM Memory	$C0	0000 aaaa	aaaa aaaa	data byte in
Write EEPROM Memory Page (page access)	$C2	0000 aaaa	aaaa 00	$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. a = address.
2. Bits are programmed ‘0’, un-programmed ‘1’.
3. To ensure future compatibility, unused Fuses and Lock bits should be un-programmed (‘1’).
4. Refer to the corresponding section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
5. See http://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 31-14 below.

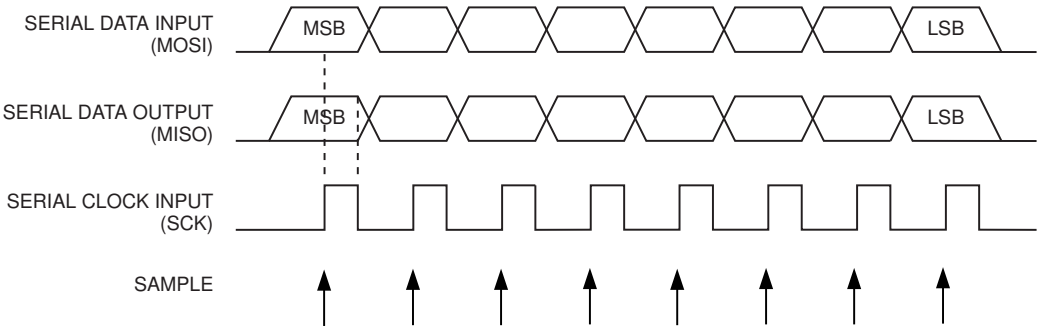
Figure 31-14. Serial Programming Instruction Example



31.8.4 Serial Programming Characteristics

For characteristics of the Serial Programming module see "SPI Timing Characteristics" on page 504.

Figure 31-15. Serial Programming Waveforms



31.9 Programming via the JTAG Interface

Programming through the JTAG interface requires control of the four JTAG specific pins: TCK, TMS, TDI, and TDO. Control of the reset and clock pins is not required.

To be able to use the JTAG interface, the JTAGEN Fuse must be programmed. The device is default shipped with the fuse programmed. In addition, the JTD bit in MCUCR must be cleared. Alternatively, if the JTD bit is set, the external reset can be forced low. Then, the JTD bit will be cleared after two chip clocks, and the JTAG pins are available for programming. This provides a means of using the JTAG pins as normal port pins in running mode while still allowing In-System Programming via the JTAG interface. Note that this technique can not be used when using the JTAG pins for Boundary-scan or On-chip Debug. In these cases the JTAG pins must be dedicated for this purpose.

During programming the clock frequency of the TCK Input must be less than the maximum frequency of the chip. The System Clock Prescaler can not be used to divide the TCK Clock Input into a sufficiently low frequency.

As a definition in this datasheet, the LSB is shifted in and out first of all Shift Registers.

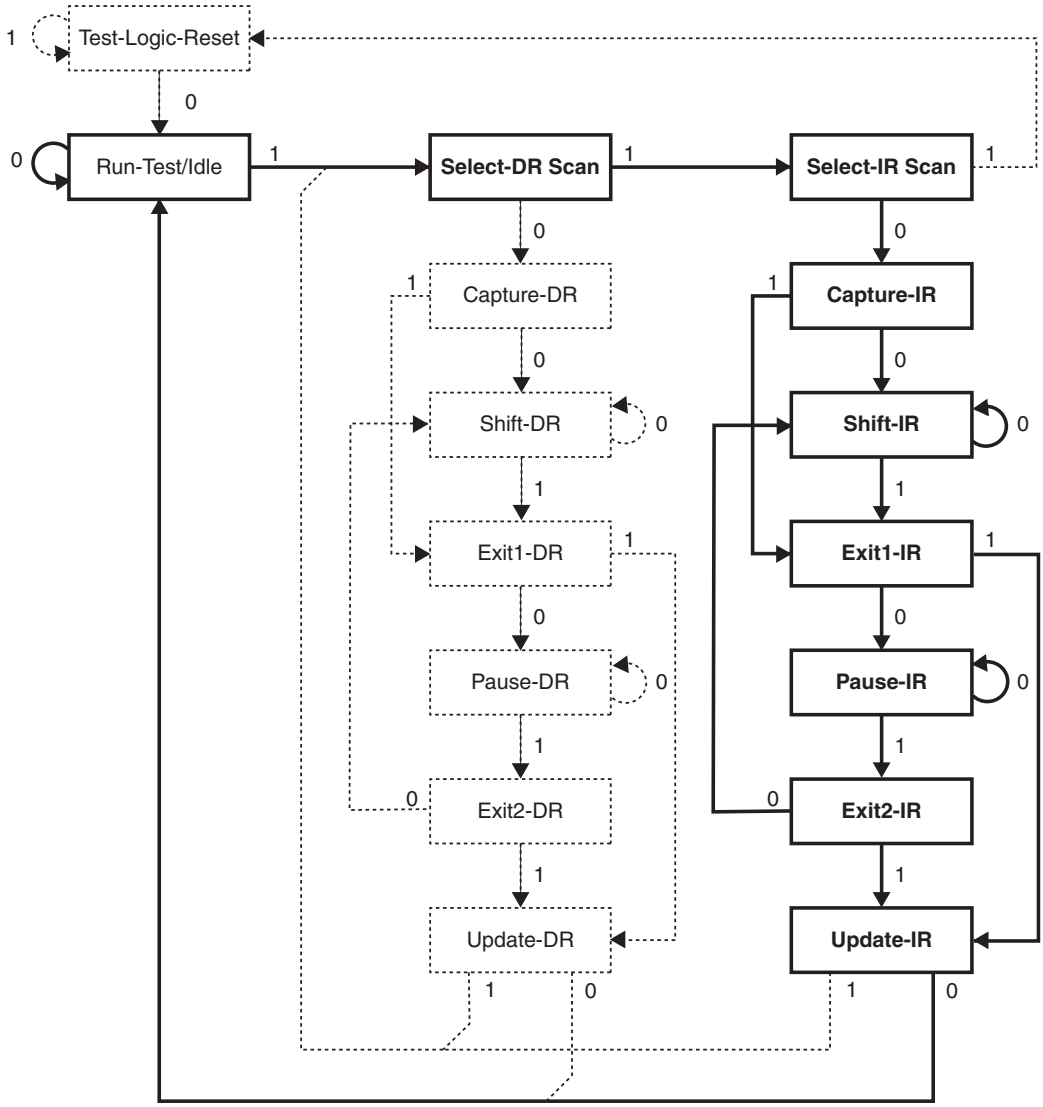
31.9.1 Programming Specific JTAG Instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. The JTAG instructions useful for programming are listed below.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

The Run-Test/Idle state of the TAP-controller is used to generate internal clocks. It can also be used as an idle state between JTAG sequences. The state machine sequence for changing the instruction word is shown in Figure 31-16 on page 482.

Figure 31-16. State Machine Sequence for Changing the Instruction Word



31.9.2 AVR_RESET (0xC)

The AVR specific public JTAG instruction is used for setting the AVR device in the Reset mode or taking the device out from the Reset mode. The TAP-controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

31.9.3 PROG_ENABLE (0x4)

The AVR specific public JTAG instruction enables programming via the JTAG port. The 16-bit Programming Enable Register is selected as Data Register. The active states are the following:

- Shift-DR: The programming enable signature is shifted into the Data Register.

- Update-DR: The programming enable signature is compared to the correct value, and Programming mode is entered if the signature is valid.

31.9.4 PROG_COMMANDS (0x5)

The AVR specific public JTAG instruction is used for entering programming commands via the JTAG port. The 15-bit Programming Command Register is selected as Data Register. The active states are the following:

- Capture-DR: The result of the previous command is loaded into the Data Register.
- Shift-DR: The Data Register is shifted by the TCK input, shifting out the result of the previous command and shifting in the new command.
- Update-DR: The programming command is applied to the Flash inputs.
- Run-Test/Idle: One clock cycle is generated, executing the applied command.

31.9.5 PROG_PAGELOAD (0x6)

The AVR specific public JTAG instruction directly loads the Flash data page via the JTAG port. An 8-bit Flash Data Byte Register is selected as the Data Register. This is physically the 8 LSB's of the Programming Command Register. The active states are the following:

- Shift-DR: The Flash Data Byte Register is shifted by the TCK input.
- Update-DR: The content of the Flash Data Byte Register is copied into a temporary register. A write sequence is initiated that within 11 TCK cycles loads the content of the temporary register into the Flash page buffer. The AVR automatically alternates between writing the low and the high byte for each new Update-DR state, starting with the low byte for the first Update-DR encountered after entering the PROG_PAGELOAD command. The Program Counter is pre-incremented before writing the low byte, except for the first written byte. This ensures that the first data is written to the address set up by PROG_COMMANDS, and loading the last location in the page buffer does not make the program counter increment into the next page.

31.9.6 PROG_PAGEREAD (0x7)

The AVR specific public JTAG instruction directly captures the Flash content via the JTAG port. An 8-bit Flash Data Byte Register is selected as the Data Register. This is physically the 8 LSB's of the Programming Command Register. The active states are the following:

- Capture-DR: The content of the selected Flash byte is captured into the Flash Data Byte Register. The AVR automatically alternates between reading the low and the high byte for each new Capture-DR state, starting with the low byte for the first Capture-DR encountered after entering the PROG_PAGEREAD command. The Program Counter is post-incremented after reading each high byte, including the first read byte. This ensures that the first data is captured from the first address set up by PROG_COMMANDS, and reading the last location in the page makes the program counter increment into the next page.
- Shift-DR: The Flash Data Byte Register is shifted by the TCK input.

31.9.7 Data Registers

The Data Registers are selected by the JTAG instruction registers described in section "Programming Specific JTAG Instructions" on page 481. The Data Registers relevant for programming operations are:

- Reset Register

- Programming Enable Register
- Programming Command Register
- Flash Data Byte Register

31.9.8 Reset Register

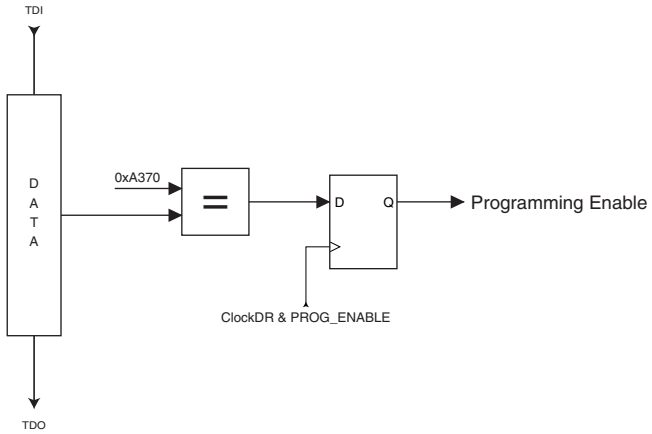
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering Programming mode.

A high value in the Reset Register corresponds to pulling the external reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-out period (refer to "Clock Sources" on page 148) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in "Figure 29-2" on page 443.

31.9.9 Programming Enable Register

The Programming Enable Register is a 16-bit register. The content of this register is compared to the programming enable signature, binary code 1010_0011_0111_0000. When the content of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving Programming mode.

Figure 31-17. Programming Enable Register



31.9.10 Programming Command Register

The Programming Command Register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 31-18 on page 485. The state sequence when shifting in the programming commands is illustrated in Figure 31-19 on page 488.

Figure 31-18. Programming Command Register

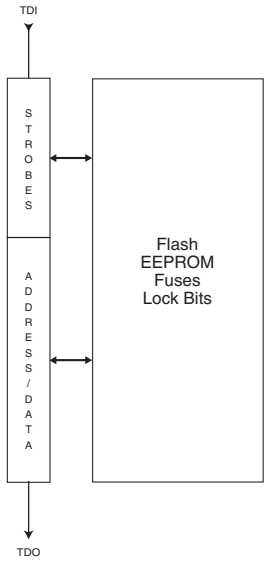


Table 31-18. JTAG Programming Instruction (set **a** = address high bits, **b** = address low bits, **c** = address extended bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care)

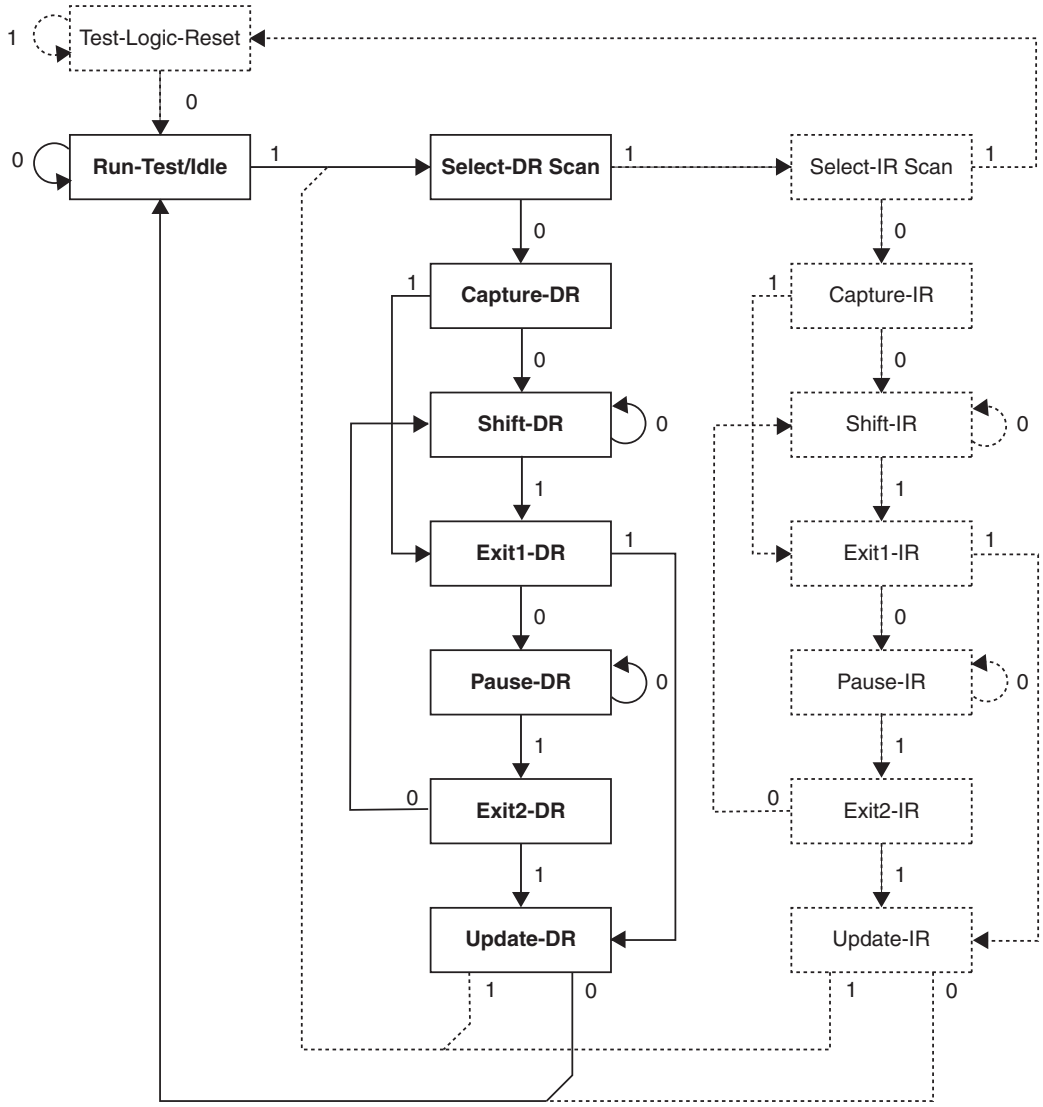
Instruction	TDI Sequence	TDO Sequence	Notes
1a. Chip Erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for Chip Erase Complete	0110011_10000000	xxxxx**o**x_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
2c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
2d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
2e. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2f. Load Data High Byte	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2g. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2i. Poll for Page Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address Extended High Byte	0001011_**cccccccc**	xxxxxxx_xxxxxxxx	(10)
3c. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	
3d. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
3e. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	 Low byte High byte

Instruction	TDI Sequence	TDO Sequence	Notes
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(10)
4c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write Complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(10)
5c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_**bbbbbbbb** 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(6)
6c. Write Fuse Extended Byte	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6d. Poll for Fuse Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6e. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(7)
6f. Write Fuse High Byte	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Poll for Fuse Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6h. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)(8)
6i. Write Fuse Low Byte	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6j. Poll for Fuse Write Complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
7a. Enter Lock Bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte	0010011_11**iiiiii**	xxxxxxx_xxxxxxxx	(4)(9)
7c. Write Lock Bits	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
7d. Poll for Lock Bit Write complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)

Instruction	TDI Sequence	TDO Sequence	Notes
8a. Enter Fuse/Lock Bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Extended Fuse Byte	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(6)(5)
8c. Read Fuse High Byte	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(7)(5)
8d. Read Fuse Low Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	(8)(5)
8e. Read Lock Bits	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xx**oooooo**	(9)(5)
8f. Read Fuses and Lock Bits	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	(5) Fuse Ext. byte Fuse High byte Fuse Low byte Lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Notes:
1. This command sequence is not required if the seven MSB's are correctly set by the previous command sequence (which is normally the case).
2. Repeat until o = "1".
3. Set bits to "0" to program the corresponding Fuse, "1" to un-program the Fuse.
4. Set bits to "0" to program the corresponding Lock bit, "1" to leave the Lock bit unchanged.
5. "0" = programmed, "1" = un-programmed.
6. The bit mapping for Fuses Extended byte is listed in Table 31-3 on page 465.
7. The bit mapping for Fuses High byte is listed in Table 31-4 on page 465.
8. The bit mapping for Fuses Low byte is listed in Table 31-5 on page 466
9. The bit mapping for Lock bits byte is listed in Table 31-1 on page 464.
10. Address bits exceeding PCMSB and EEAMSB (Table 31-7 on page 467 and Table 31-8 on page 467) are don't care.
11. All TDI and TDO sequences are represented by binary digits.

Figure 31-19. State Machine Sequence for Changing/Reading the Data Word



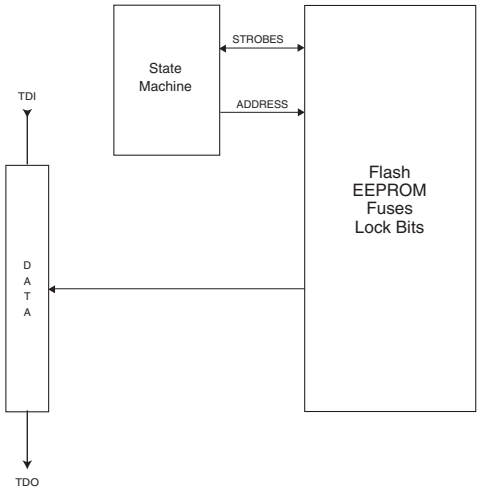
31.9.11 Flash Data Byte Register

The Flash Data Byte Register provides an efficient way to load the entire Flash page buffer before executing Page Write, or to read out/verify the content of the Flash. A state machine sets up the control signals to the Flash and senses the strobe signals from the Flash, thus only the data words need to be shifted in/out.

The Flash Data Byte Register actually consists of the 8-bit scan chain and an 8-bit temporary register. During page load, the Update-DR state copies the content of the scan chain over to the temporary register and initiates a write sequence that within 11 TCK cycles loads the content of the temporary register into the Flash page buffer. The AVR automatically alternates between writing the low and the high byte for each new Update-DR state, starting with the low byte for the first Update-DR encountered after entering the PROG_PAGELOAD command. The Program Counter is pre-incremented before writing the low byte, except for the first written byte. This ensures that the first data is written to the address set up by PROG_COMMANDS, and loading the last location in the page buffer does not make the Program Counter increment into the next page.

During Page Read, the content of the selected Flash byte is captured into the Flash Data Byte Register during the Capture-DR state. The AVR automatically alternates between reading the low and the high byte for each new Capture-DR state, starting with the low byte for the first Capture-DR encountered after entering the PROG_PAGEREAD command. The Program Counter is post-incremented after reading each high byte, including the first read byte. This ensures that the first data is captured from the first address set up by PROG_COMMANDS, and reading the last location in the page makes the program counter increment into the next page.

Figure 31-20. Flash Data Byte Register



The state machine controlling the Flash Data Byte Register is clocked by TCK. During normal operation in which eight bits are shifted for each Flash byte, the clock cycles needed to navigate through the TAP-controller automatically feeds the state machine for the Flash Data Byte Register with sufficient number of clock pulses to complete its operation transparently for the user. However, if too few bits are shifted between each Update-DR state during page load, the TAP-controller should stay in the Run-Test/Idle state for some TCK cycles to ensure that there are at least 11 TCK cycles between each Update-DR state.

31.9.12 Programming Algorithm

All references below of type “1a”, “1b”, and so on, refer to Table 31-18 on page 485.

31.9.13 Entering Programming Mode

1. Enter JTAG instruction AVR_RESET and shift 1 in the Reset Register.
2. Enter instruction PROG_ENABLE and shift 0b1010_0011_0111_0000 in the Programming Enable Register.

31.9.14 Leaving Programming Mode

1. Enter JTAG instruction PROG_COMMANDS.
2. Disable all programming instructions by using no operation instruction 11a.
3. Enter instruction PROG_ENABLE and shift 0b0000_0000_0000_0000 in the programming Enable Register.
4. Enter JTAG instruction AVR_RESET and shift 0 in the Reset Register.

31.9.15 Performing Chip Erase

1. Enter JTAG instruction PROG_COMMANDS.
2. Start Chip Erase using programming instruction 1a.
3. Poll for Chip Erase complete using programming instruction 1b, or wait for t_{WLRH_CE} (refer to Table 31-14 on page 477).

31.9.16 Programming the Flash

Before programming the Flash a Chip Erase must be performed, see section "Performing Chip Erase" above.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load High byte of address using programming instruction 2c.
4. Load Low byte of address using programming instruction 2d.
5. Load data using programming instructions 2e, 2f and 2g.
6. Repeat steps 5 and 6 for all instruction words in the page.
7. Write the page using programming instruction 2h.
8. Poll for Flash write complete using programming instruction 2i, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

A more efficient data transfer can be achieved using the PROG_PAGELOAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash write using programming instruction 2a.
3. Load the page address using programming instructions 2c and 2d. PCWORD (refer to Table 31-7 on page 467) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGELOAD.
5. Load the entire page by shifting in all instruction words in the page byte-by-byte, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. Use Update-DR to copy the contents of the Flash Data Byte Register into the Flash page location and to auto-increment the Program Counter before each new word.
6. Enter JTAG instruction PROG_COMMANDS.
7. Write the page using programming instruction 2h.
8. Poll for Flash write complete using programming instruction 2i, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

31.9.17 Reading the Flash

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load address using programming instructions 3c and 3d.
4. Read data using programming instruction 3e.
5. Repeat steps 3 and 4 until all data have been read.

A more efficient data transfer can be achieved using the PROG_PAGEREAD instruction:

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Flash read using programming instruction 3a.
3. Load the page address using programming instructions 3c and 3d. PCWORD (refer to Table 31-7 on page 467) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page (or Flash) by shifting out all instruction words in the page (or Flash), starting with the LSB of the first instruction in the page (Flash) and ending with the MSB of the last instruction in the page (Flash). The Capture-DR state both captures the data from the Flash, and also auto-increments the program counter after each word is read. Note that Capture-DR comes before the shift-DR state. Hence, the first byte which is shifted out contains valid data.
6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

31.9.18 Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed, see "Performing Chip Erase" on page 490.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load High byte of address using programming instruction 4b.
4. Load Low byte of address using programming instruction 4c.
5. Load data using programming instructions 4d and 4e.
6. Repeat steps 4 and 5 for all data bytes in the page.
7. Write the data using programming instruction 4f.
8. Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM.

31.9.19 Reading the EEPROM

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM read using programming instruction 5a.
3. Load address using programming instructions 5b and 5c.
4. Read data using programming instruction 5d.
5. Repeat steps 3 and 4 until all data have been read.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM.

31.9.20 Programming the Fuses

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data high byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse; a “1” will un-program the fuse.
4. Write Fuse High byte using programming instruction 6c.

5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 31-14 on page 477).
6. Load data low byte using programming instructions 6e. A “0” will program the fuse, a “1” will un-program the fuse.
7. Write Fuse low byte using programming instruction 6f.
8. Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 31-14 on page 477).

31.9.21 Programming the Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 31-14 on page 477).

31.9.22 Reading the Fuses and Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse/Lock bit read using programming instruction 8a.
3. To read all Fuses and Lock bits, use programming instruction 8e.

   To only read Fuse High byte, use programming instruction 8b.

   To only read Fuse Low byte, use programming instruction 8c.

   To only read Lock bits, use programming instruction 8d.

31.9.23 Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address 0x00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address 0x01 and address 0x02 to read the second and third signature bytes, respectively.

31.9.24 Reading the Calibration Byte

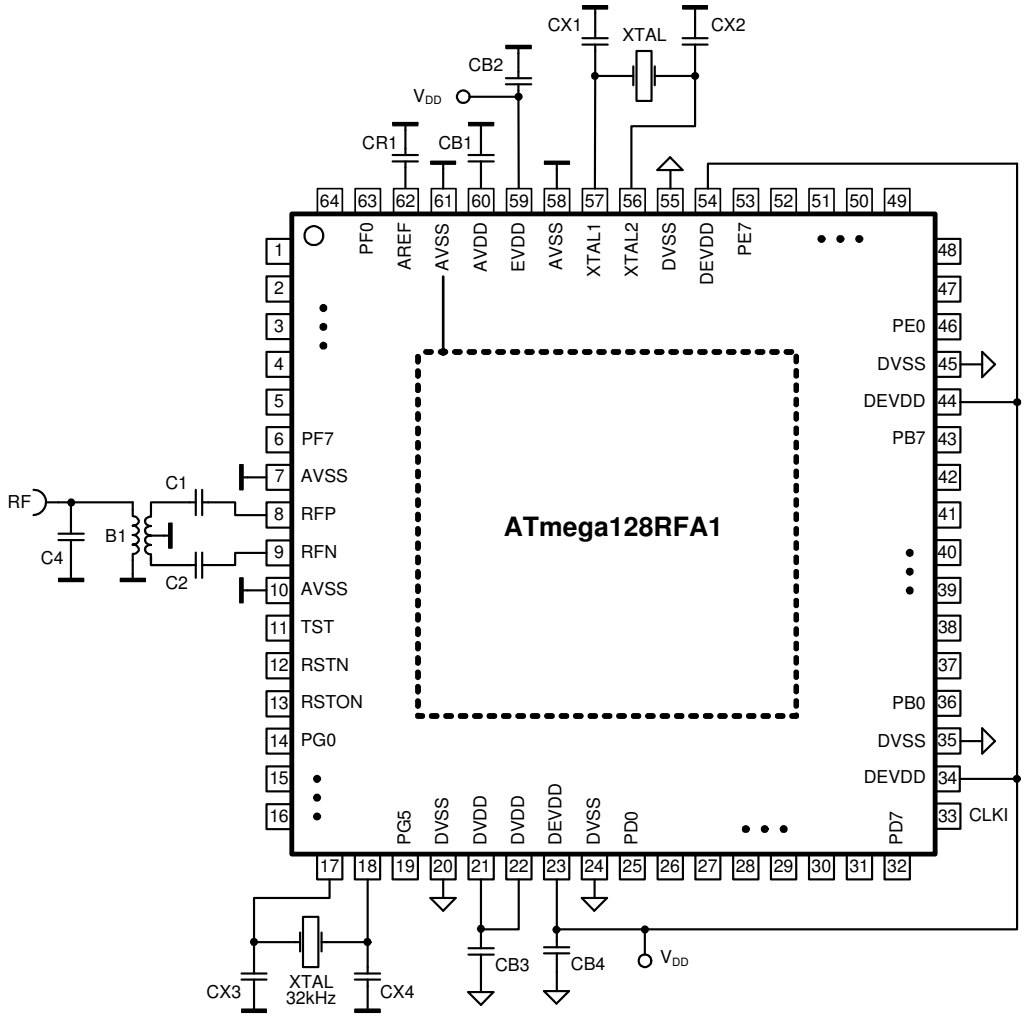
1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address 0x00 using programming instruction 10b.
4. Read the calibration byte using programming instruction 10c.

32 Application Circuits

32.1 Basic Application Schematic

A basic application schematic of the ATmega128RFA1 with a single-ended RF connector is shown in Figure 32-1 below. The 50Ω single-ended RF input is transformed to the 100Ω differential RF port impedance using Balun B1. The capacitors C1 and C2 provide AC coupling of the RF input to the RF port, capacitor C4 improves matching.

Figure 32-1. Basic Application schematic (Table 32-1 on page 494)



The power supply bypass capacitors (CB2, CB4) are connected to the external analog supply pin (EVDD, pin 59) and external digital supply pin (DEVDD, pin 23). Pins 34, 44 and 54 supply the digital port pins.

Capacitors CB1 and CB3 are bypass capacitors for the integrated analog and digital voltage regulators to ensure stable operation and to improve noise immunity. Capacitors should be placed as close as possible to the pins and should have a low-resistance and low-inductance connection to ground to achieve the best performance.

The crystal (XTAL), the two load capacitors (CX1, CX2), and the internal circuitry connected to pins XTAL1 and XTAL2 form the 16MHz crystal oscillator for the 2.4GHz transceiver. To achieve the best accuracy and stability of the reference frequency, large parasitic capacitances must be avoided. Crystal lines should be routed as short as possible and not in proximity of digital I/O signals. This is especially required for the High Data Rate Modes.

The 32.768 kHz crystal connected to the internal low power (sub 1µA) crystal oscillator provides a stable time reference for all low power modes including 32 Bit IEEE 802.15.4 Symbol Counter ("MAC Symbol Counter" on page 133) and real time clock application using the asynchronous timer T/C2 ("8-bit Timer/Counter2 with PWM and Asynchronous Operation" on page 309). Total capacitance including CX3, CX4 should not exceed 15pF on both pins. The very low supply current of the oscillator requires careful layout of the PCB and any leakage path must be avoided.

Crosstalk and radiation from switching digital signals to the crystal pins or the RF pins can degrade the system performance. The programming of minimum drive strength settings for the digital output signal is recommended (see "DPDS0 – Port Driver Strength Register 0" on page 174).

Table 32-1. Example Bill of Materials (BoM) for "Basic Application Schematic" on page 493

Designator	Description	Value	Manufacturer	Part Number	Comment
B1	SMD balun SMD balun / filter	2.4 GHz	Wuerth Johanson Technology	748421245 2450FB15L0001	Filter included
CB1 CB3	LDO VREG bypass capacitor	1 µF (100nF minimum)	AVX Murata	0603YD105KAT2A GRM188R61C105KA12D	X5R 10% 16V (0603)
CB2 CB4	Power supply bypass capacitor	1 µF (100nF minimum)			
CX1, CX2	Crystal load capacitor	12 pF	AVX Murata	06035A120JA GRP1886C1H120JA01	COG 5% 50V (0603)
C1, C2	RF coupling capacitor	22 pF	Epcos Epcos AVX	B37930 B37920 06035A220JAT2A	C0G 5% 50V (0402 or 0603)
C4 (optional)	RF matching	0.47 pF	Johnstech		
R1	CLKM low-pass filter resistor	680Ω			Designed for f_{CLKM} = 1 MHz
XTAL	Crystal	CX-4025 16 MHz SX-4025 16 MHz	ACAL Taitjen Siward	XWBBPL-F-1 A207-011	
XTAL 32kHz	Crystal				Rs=100 kOhm

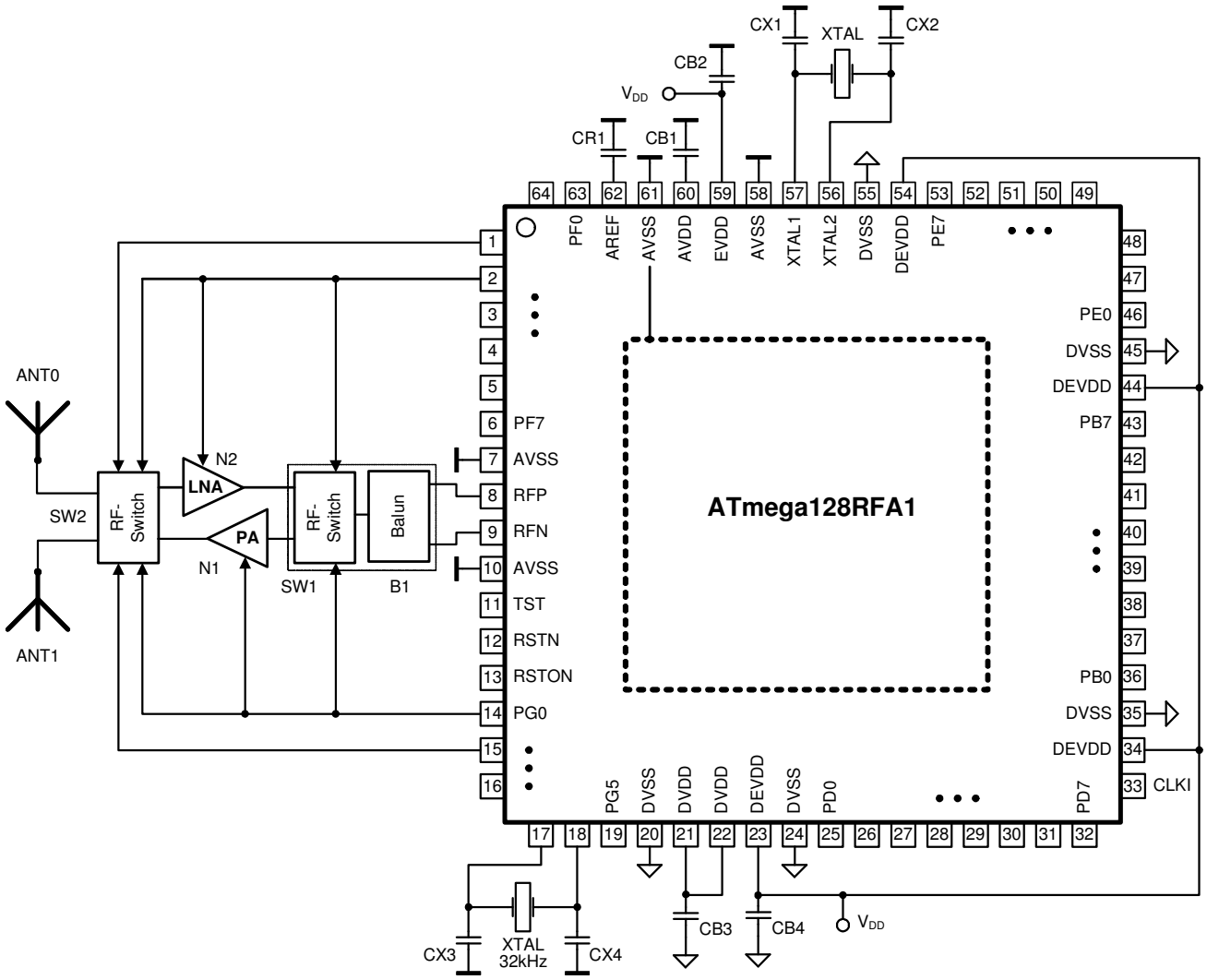
32.2 Extended Feature Set Application Schematic

The ATmega128RFA1 supports additional features like:

- Security Module (AES)
- High Data Rate Mode up to 2MBits/s
- Antenna Diversity using alternate pin function DIG1/2 at Port G and F
- RX/TX Indicator using alternate pin function DIG3/4 at Port G and F

An extended feature set application schematic illustrating the use of the ATmega128RFA1 Extended Feature Set, is shown in Figure 32-2 on page 495.

Figure 32-2. Extended Feature Application schematic



Although this example shows all additional hardware features combined, it is possible to use all features separately or in various combinations.

33 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x1FF)	TRXFBEND	TRXFBEND7	TRXFBEND6	TRXFBEND5	TRXFBEND4	TRXFBEND3	TRXFBEND2	TRXFBEND1	TRXFBEND0	132
...										
(0x180)	TRXFBST	TRXFBST7	TRXFBST6	TRXFBST5	TRXFBST4	TRXFBST3	TRXFBST2	TRXFBST1	TRXFBST0	132
(0x17F)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17E)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17D)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17C)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x17B)	TST_RX_LENGTH	RX_LENGTH7	RX_LENGTH6	RX_LENGTH5	RX_LENGTH4	RX_LENGTH3	RX_LENGTH2	RX_LENGTH1	RX_LENGTH0	131
(0x17A)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x179)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x178)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x177)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x176)	TST_CTRL_DIGI	Res7	Res6	Res5	Res4	TST_CTRL_DIG3	TST_CTRL_DIG2	TST_CTRL_DIG1	TST_CTRL_DIG0	131
(0x175)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...										
(0x173)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x172)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x171)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x170)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x16F)	CSMA_BE	MAX_BE3	MAX_BE2	MAX_BE1	MAX_BE0	MIN_BE3	MIN_BE2	MIN_BE1	MIN_BE0	130
(0x16E)	CSMA_SEED_1	AACK_FVN_MODE1	AACK_FVN_MODE0	AACK_SET_PD	AACK_DIS_ACK	AACK_I_AM_COORD	CSMA_SEED_12	CSMA_SEED_11	CSMA_SEED_10	129
(0x16D)	CSMA_SEED_0	CSMA_SEED_07	CSMA_SEED_06	CSMA_SEED_05	CSMA_SEED_04	CSMA_SEED_03	CSMA_SEED_02	CSMA_SEED_01	CSMA_SEED_00	128
(0x16C)	XAH_CTRL_0	MAX_FRAME_RETRES3	MAX_FRAME_RETRES2	MAX_FRAME_RETRES1	MAX_FRAME_RETRES0	MAX_CSMA_RETRES2	MAX_CSMA_RETRES1	MAX_CSMA_RETRES0	SLOTTED_OPERATION	127
(0x16B)	IEEE_ADDR_7	IEEE_ADDR_77	IEEE_ADDR_76	IEEE_ADDR_75	IEEE_ADDR_74	IEEE_ADDR_73	IEEE_ADDR_72	IEEE_ADDR_71	IEEE_ADDR_70	126
(0x16A)	IEEE_ADDR_6	IEEE_ADDR_67	IEEE_ADDR_66	IEEE_ADDR_65	IEEE_ADDR_64	IEEE_ADDR_63	IEEE_ADDR_62	IEEE_ADDR_61	IEEE_ADDR_60	126
(0x169)	IEEE_ADDR_5	IEEE_ADDR_57	IEEE_ADDR_56	IEEE_ADDR_55	IEEE_ADDR_54	IEEE_ADDR_53	IEEE_ADDR_52	IEEE_ADDR_51	IEEE_ADDR_50	126
(0x168)	IEEE_ADDR_4	IEEE_ADDR_47	IEEE_ADDR_46	IEEE_ADDR_45	IEEE_ADDR_44	IEEE_ADDR_43	IEEE_ADDR_42	IEEE_ADDR_41	IEEE_ADDR_40	126
(0x167)	IEEE_ADDR_3	IEEE_ADDR_37	IEEE_ADDR_36	IEEE_ADDR_35	IEEE_ADDR_34	IEEE_ADDR_33	IEEE_ADDR_32	IEEE_ADDR_31	IEEE_ADDR_30	125
(0x166)	IEEE_ADDR_2	IEEE_ADDR_27	IEEE_ADDR_26	IEEE_ADDR_25	IEEE_ADDR_24	IEEE_ADDR_23	IEEE_ADDR_22	IEEE_ADDR_21	IEEE_ADDR_20	125
(0x165)	IEEE_ADDR_1	IEEE_ADDR_17	IEEE_ADDR_16	IEEE_ADDR_15	IEEE_ADDR_14	IEEE_ADDR_13	IEEE_ADDR_12	IEEE_ADDR_11	IEEE_ADDR_10	125
(0x164)	IEEE_ADDR_0	IEEE_ADDR_07	IEEE_ADDR_06	IEEE_ADDR_05	IEEE_ADDR_04	IEEE_ADDR_03	IEEE_ADDR_02	IEEE_ADDR_01	IEEE_ADDR_00	124
(0x163)	PAN_ID_1	PAN_ID_17	PAN_ID_16	PAN_ID_15	PAN_ID_14	PAN_ID_13	PAN_ID_12	PAN_ID_11	PAN_ID_10	124
(0x162)	PAN_ID_0	PAN_ID_07	PAN_ID_06	PAN_ID_05	PAN_ID_04	PAN_ID_03	PAN_ID_02	PAN_ID_01	PAN_ID_00	124
(0x161)	SHORT_ADDR_1	SHORT_ADDR_17	SHORT_ADDR_16	SHORT_ADDR_15	SHORT_ADDR_14	SHORT_ADDR_13	SHORT_ADDR_12	SHORT_ADDR_11	SHORT_ADDR_10	124
(0x160)	SHORT_ADDR_0	SHORT_ADDR_07	SHORT_ADDR_06	SHORT_ADDR_05	SHORT_ADDR_04	SHORT_ADDR_03	SHORT_ADDR_02	SHORT_ADDR_01	SHORT_ADDR_00	123
(0x15F)	MAN_ID_1	MAN_ID_17	MAN_ID_16	MAN_ID_15	MAN_ID_14	MAN_ID_13	MAN_ID_12	MAN_ID_11	MAN_ID_10	123
(0x15E)	MAN_ID_0	MAN_ID_07	MAN_ID_06	MAN_ID_05	MAN_ID_04	MAN_ID_03	MAN_ID_02	MAN_ID_01	MAN_ID_00	123
(0x15D)	VERSION_NUM	VERSION_NUM7	VERSION_NUM6	VERSION_NUM5	VERSION_NUM4	VERSION_NUM3	VERSION_NUM2	VERSION_NUM1	VERSION_NUM0	122
(0x15C)	PART_NUM	PART_NUM7	PART_NUM6	PART_NUM5	PART_NUM4	PART_NUM3	PART_NUM2	PART_NUM1	PART_NUM0	122
(0x15B)	PLL_DCU	PLL_DCU_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	121
(0x15A)	PLL_CF	PLL_CF_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	121
(0x159)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x158)	FTN_CTRL	FTN_START	Res6	Res5	Res4	Res3	Res2	Res1	Res0	120
(0x157)	XAH_CTRL_1	Res1	Res0	AACK_FLTR_RES_FT	AACK_UPLD_RES_FT	Res	AACK_ACK_TIME	AACK_PROM_MODE	Res	119
...	Reserved									
(0x155)	RX_SYN	RX_PDT_DIS	Res2	Res1	Res0	RX_PDT_LEVEL3	RX_PDT_LEVEL2	RX_PDT_LEVEL1	RX_PDT_LEVEL0	118
...	Reserved									
(0x152)	XOSC_CTRL	XTAL_MODE3	XTAL_MODE2	XTAL_MODE1	XTAL_MODE0	XTAL_TRIM3	XTAL_TRIM2	XTAL_TRIM1	XTAL_TRIM0	117
(0x151)	BATMON	BAT_LOW	BAT_LOW_EN	BATMON_OK	BATMON_HR	BATMON_VTH3	BATMON_VTH2	BATMON_VTH1	BATMON_VTH0	116
(0x150)	VREG_CTRL	AVREG_EXT	AVDD_OK	Res5	Res4	Res3	DVDD_OK	Res1	Res0	115
(0x14F)	IRQ_STATUS	AWAKE	TX_END	AMI	CCA_ED_DONE	RX_END	RX_START	PLL_UNLOCK	PLL_LOCK	114
(0x14E)	IRQ_MASK	AWAKE_EN	TX_END_EN	AMI_EN	CCA_ED_DONE_EN	RX_END_EN	RX_START_EN	PLL_UNLOCK_EN	PLL_LOCK_EN	114
(0x14D)	ANT_DIV	ANT_SEL	Res2	Res1	Res0	ANT_DIV_EN	ANT_EXT_SW_EN	ANT_CTRL1	ANT_CTRL0	112
(0x14C)	TRX_CTRL_2	RX_SAFE_MODE	Res4	Res3	Res2	Res1	Res0	OQPSK_DATA_RATE1	OQPSK_DATA_RATE0	112
(0x14B)	SFD_VALUE	SFD_VALUE7	SFD_VALUE6	SFD_VALUE5	SFD_VALUE4	SFD_VALUE3	SFD_VALUE2	SFD_VALUE1	SFD_VALUE0	111
(0x14A)	RX_CTRL	Res7	Res6	Res5	Res4	PDT_THRES3	PDT_THRES2	PDT_THRES1	PDT_THRES0	111
(0x149)	CCA_THRES	CCA_CS_THRES3	CCA_CS_THRES2	CCA_CS_THRES1	CCA_CS_THRES0	CCA_ED_THRES3	CCA_ED_THRES2	CCA_ED_THRES1	CCA_ED_THRES0	110
(0x148)	PHY_CC_CCA	CCA_REQUEST	CCA_MODE1	CCA_MODE0	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0	109
(0x147)	PHY_ED_LEVEL	ED_LEVEL7	ED_LEVEL6	ED_LEVEL5	ED_LEVEL4	ED_LEVEL3	ED_LEVEL2	ED_LEVEL1	ED_LEVEL0	108
(0x146)	PHY_RSSI	RX_CRC_VALID	RND_VALUE1	RND_VALUE0	RSSI4	RSSI3	RSSI2	RSSI1	RSSI0	107
(0x145)	PHY_TX_PWR	PA_BUF_LT1	PA_BUF_LT0	PA_LT1	PA_LT0	TX_PWR3	TX_PWR2	TX_PWR1	TX_PWR0	106

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x144)	TRX_CTRL_1	PA_EXT_EN	IRQ_2_EXT_EN	TX_AUTO_CRC_ON	Res4	Res3	Res2	Res1	Res0	105
(0x143)	TRX_CTRL_0	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	105
(0x142)	TRX_STATE	TRAC_STATUS2	TRAC_STATUS1	TRAC_STATUS0	TRX_CMD4	TRX_CMD3	TRX_CMD2	TRX_CMD1	TRX_CMD0	104
(0x141)	TRX_STATUS	CCA_DONE	CCA_STATUS	TST_STATUS	TRX_STATUS4	TRX_STATUS3	TRX_STATUS2	TRX_STATUS1	TRX_STATUS0	102
...	Reserved									
(0x13F)	AES_KEY	AES_KEY7	AES_KEY6	AES_KEY5	AES_KEY4	AES_KEY3	AES_KEY2	AES_KEY1	AES_KEY0	102
(0x13E)	AES_STATE	AES_STATE7	AES_STATE6	AES_STATE5	AES_STATE4	AES_STATE3	AES_STATE2	AES_STATE1	AES_STATE0	102
(0x13D)	AES_STATUS	AES_ER	Res5	Res4	Res3	Res2	Res1	Res0	AES_DONE	101
(0x13C)	AES_CTRL	AES_REQUEST	Res	AES_MODE	Res	AES_DIR	AES_IM	Res1	Res0	100
(0x13B)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...	Reserved									
(0x139)	TRXPR	Res3	Res2	Res1	Res0	Res3	Res2	SLPTR	TRXRST	169
...	Reserved									
(0x137)	DPDS1	Res5	Res4	Res3	Res2	Res1	Res0	PGDRV1	PGDRV0	175
(0x136)	DPDS0	PFDRV1	PFDRV0	PEDRV1	PEDRV0	PDDRV1	PDDRV0	PBDRV1	PBDRV0	174
(0x135)	DRTRAM0	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	170
(0x134)	DRTRAM1	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	170
(0x133)	DRTRAM2	Res7	Res	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	171
(0x132)	DRTRAM3	Res1	Res0	DRTSWOK	ENDRT	Res3	Res2	Res1	Res0	171
(0x131)	LLDRH	Res2	Res1	Res0	LLDRH4	LLDRH3	LLDRH2	LLDRH1	LLDRH0	173
(0x130)	LLDRL	Res3	Res2	Res1	Res0	LLDRL3	LLDRL2	LLDRL1	LLDRL0	173
(0x12F)	LLCR	Res1	Res0	LLDONE	LLCOMP	LLCAL	LLTCO	LLSHORT	LLENCAL	172
...	Reserved									
(0x12D)	OCR5CH	OCR5CH7	OCR5CH6	OCR5CH5	OCR5CH4	OCR5CH3	OCR5CH2	OCR5CH1	OCR5CH0	300
(0x12C)	OCR5CL	OCR5CL7	OCR5CL6	OCR5CL5	OCR5CL4	OCR5CL3	OCR5CL2	OCR5CL1	OCR5CL0	301
(0x12B)	OCR5BH	OCR5BH7	OCR5BH6	OCR5BH5	OCR5BH4	OCR5BH3	OCR5BH2	OCR5BH1	OCR5BH0	299
(0x12A)	OCR5BL	OCR5BL7	OCR5BL6	OCR5BL5	OCR5BL4	OCR5BL3	OCR5BL2	OCR5BL1	OCR5BL0	300
(0x129)	OCR5AH	OCR5AH7	OCR5AH6	OCR5AH5	OCR5AH4	OCR5AH3	OCR5AH2	OCR5AH1	OCR5AH0	299
(0x128)	OCR5AL	OCR5AL7	OCR5AL6	OCR5AL5	OCR5AL4	OCR5AL3	OCR5AL2	OCR5AL1	OCR5AL0	299
(0x127)	ICR5H	ICR5H7	ICR5H6	ICR5H5	ICR5H4	ICR5H3	ICR5H2	ICR5H1	ICR5H0	301
(0x126)	ICR5L	ICR5L7	ICR5L6	ICR5L5	ICR5L4	ICR5L3	ICR5L2	ICR5L1	ICR5L0	301
(0x125)	TCNT5H	TCNT5H7	TCNT5H6	TCNT5H5	TCNT5H4	TCNT5H3	TCNT5H2	TCNT5H1	TCNT5H0	298
(0x124)	TCNT5L	TCNT5L7	TCNT5L6	TCNT5L5	TCNT5L4	TCNT5L3	TCNT5L2	TCNT5L1	TCNT5L0	298
...	Reserved									
(0x122)	TCCR5C	FOC5A	FOC5B	FOC5C	Res4	Res3	Res2	Res1	Res0	297
(0x121)	TCCR5B	ICNC5	ICES5	Res	WGM53	WGM52	CS52	CS51	CS50	296
(0x120)	TCCR5A	COM5A1	COM5A0	COM5B1	COM5B0	COM5C1	COM5C0	WGM51	WGM50	294
...	Reserved									
(0xF8)	SCOCR1HH	SCOCR1HH7	SCOCR1HH6	SCOCR1HH5	SCOCR1HH4	SCOCR1HH3	SCOCR1HH2	SCOCR1HH1	SCOCR1HH0	140
(0xF7)	SCOCR1HL	SCOCR1HL7	SCOCR1HL6	SCOCR1HL5	SCOCR1HL4	SCOCR1HL3	SCOCR1HL2	SCOCR1HL1	SCOCR1HL0	141
(0xF6)	SCOCR1LH	SCOCR1LH7	SCOCR1LH6	SCOCR1LH5	SCOCR1LH4	SCOCR1LH3	SCOCR1LH2	SCOCR1LH1	SCOCR1LH0	141
(0xF5)	SCOCR1LL	SCOCR1LL7	SCOCR1LL6	SCOCR1LL5	SCOCR1LL4	SCOCR1LL3	SCOCR1LL2	SCOCR1LL1	SCOCR1LL0	141
(0xF4)	SCOCR2HH	SCOCR2HH7	SCOCR2HH6	SCOCR2HH5	SCOCR2HH4	SCOCR2HH3	SCOCR2HH2	SCOCR2HH1	SCOCR2HH0	141
(0xF3)	SCOCR2HL	SCOCR2HL7	SCOCR2HL6	SCOCR2HL5	SCOCR2HL4	SCOCR2HL3	SCOCR2HL2	SCOCR2HL1	SCOCR2HL0	142
(0xF2)	SCOCR2LH	SCOCR2LH7	SCOCR2LH6	SCOCR2LH5	SCOCR2LH4	SCOCR2LH3	SCOCR2LH2	SCOCR2LH1	SCOCR2LH0	142
(0xF1)	SCOCR2LL	SCOCR2LL7	SCOCR2LL6	SCOCR2LL5	SCOCR2LL4	SCOCR2LL3	SCOCR2LL2	SCOCR2LL1	SCOCR2LL0	142
(0xF0)	SCOCR3HH	SCOCR3HH7	SCOCR3HH6	SCOCR3HH5	SCOCR3HH4	SCOCR3HH3	SCOCR3HH2	SCOCR3HH1	SCOCR3HH0	142
(0xEF)	SCOCR3HL	SCOCR3HL7	SCOCR3HL6	SCOCR3HL5	SCOCR3HL4	SCOCR3HL3	SCOCR3HL2	SCOCR3HL1	SCOCR3HL0	143
(0xEE)	SCOCR3LH	SCOCR3LH7	SCOCR3LH6	SCOCR3LH5	SCOCR3LH4	SCOCR3LH3	SCOCR3LH2	SCOCR3LH1	SCOCR3LH0	143
(0xED)	SCOCR3LL	SCOCR3LL7	SCOCR3LL6	SCOCR3LL5	SCOCR3LL4	SCOCR3LL3	SCOCR3LL2	SCOCR3LL1	SCOCR3LL0	143
(0xEC)	SCTSRHH	SCTSRHH7	SCTSRHH6	SCTSRHH5	SCTSRHH4	SCTSRHH3	SCTSRHH2	SCTSRHH1	SCTSRHH0	138
(0xEB)	SCTSRHL	SCTSRHL7	SCTSRHL6	SCTSRHL5	SCTSRHL4	SCTSRHL3	SCTSRHL2	SCTSRHL1	SCTSRHL0	139
(0xEA)	SCTSRLH	SCTSRLH7	SCTSRLH6	SCTSRLH5	SCTSRLH4	SCTSRLH3	SCTSRLH2	SCTSRLH1	SCTSRLH0	139
(0xE9)	SCTSRLL	SCTSRLL7	SCTSRLL6	SCTSRLL5	SCTSRLL4	SCTSRLL3	SCTSRLL2	SCTSRLL1	SCTSRLL0	139
(0xE8)	SCBTSRHH	SCBTSRHH7	SCBTSRHH6	SCBTSRHH5	SCBTSRHH4	SCBTSRHH3	SCBTSRHH2	SCBTSRHH1	SCBTSRHH0	139
(0xE7)	SCBTSRHL	SCBTSRHL7	SCBTSRHL6	SCBTSRHL5	SCBTSRHL4	SCBTSRHL3	SCBTSRHL2	SCBTSRHL1	SCBTSRHL0	140
(0xE6)	SCBTSRLH	SCBTSRLH7	SCBTSRLH6	SCBTSRLH5	SCBTSRLH4	SCBTSRLH3	SCBTSRLH2	SCBTSRLH1	SCBTSRLH0	140
(0xE5)	SCBTSRLL	SCBTSRLL7	SCBTSRLL6	SCBTSRLL5	SCBTSRLL4	SCBTSRLL3	SCBTSRLL2	SCBTSRLL1	SCBTSRLL0	140
(0xE4)	SCCNTHH	SCCNTHH7	SCCNTHH6	SCCNTHH5	SCCNTHH4	SCCNTHH3	SCCNTHH2	SCCNTHH1	SCCNTHH0	137
(0xE3)	SCCNTHL	SCCNTHL7	SCCNTHL6	SCCNTHL5	SCCNTHL4	SCCNTHL3	SCCNTHL2	SCCNTHL1	SCCNTHL0	138
(0xE2)	SCCNTLH	SCCNTLH7	SCCNTLH6	SCCNTLH5	SCCNTLH4	SCCNTLH3	SCCNTLH2	SCCNTLH1	SCCNTLH0	138
(0xE1)	SCCNTLL	SCCNTLL7	SCCNTLL6	SCCNTLL5	SCCNTLL4	SCCNTLL3	SCCNTLL2	SCCNTLL1	SCCNTLL0	138
(0xE0)	SCIRQS	Res2	Res1	Res0	IRQSBO	IRQSOF	IRQSCP3	IRQSCP2	IRQSCP1	145
(0xDF)	SCIRQM	Res2	Res1	Res0	IRQMBO	IRQMOF	IRQMCP3	IRQMCP2	IRQMCP1	146

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xDE)	SCSR	Res6	Res5	Res4	Res3	Res2	Res1	Res0	SCBSY	145
(0xDD)	SCCR1	Res6	Res5	Res4	Res4	Res3	Res2	Res1	SCENBO	144
(0xDC)	SCCR0	SCRES	SCMBTS	SCEN	SCCKSEL	SCTSE	SCCMP3	SCCMP2	SCCMP1	143
...	Reserved									
(0xD1)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0xD0)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
...	Reserved									
(0xCE)	UDR1	UDR17	UDR16	UDR15	UDR14	UDR13	UDR12	UDR11	UDR10	360
(0xCD)	UBRR1H	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	364
(0xCC)	UBRR1L	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	365
...	Reserved									
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UDORD1	UCPHA1	UCPOL1	376
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	375
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	375
...	Reserved									
(0xC6)	UDR0	UDR07	UDR06	UDR05	UDR04	UDR03	UDR02	UDR01	UDR00	356
(0xC5)	UBRR0H	Res3	Res2	Res1	Res0	UBRR11	UBRR10	UBRR9	UBRR8	360
(0xC4)	UBRR0L	UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	360
...	Reserved									
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UDORD0	UCPHA0	UCPOL0	374
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	374
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	373
...	Reserved									
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	Res	406
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	Res	TWIE	402
(0xBB)	TWDR	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	405
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	405
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	Res	TWPS1	TWPS0	403
(0xB8)	TWBR	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	401
...	Reserved									
(0xB6)	ASSR	EXCLKAMR	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	328
...	Reserved									
(0xB4)	OCR2B	OCR2B7	OCR2B6	OCR2B5	OCR2B4	OCR2B3	OCR2B2	OCR2B1	OCR2B0	327
(0xB3)	OCR2A	OCR2A7	OCR2A6	OCR2A5	OCR2A4	OCR2A3	OCR2A2	OCR2A1	OCR2A0	327
(0xB2)	TCNT2	TCNT27	TCNT26	TCNT25	TCNT24	TCNT23	TCNT22	TCNT21	TCNT20	327
(0xB1)	TCCR2B	FOC2A	FOC2B	Res1	Res0	WGM22	CS22	CS21	CS20	326
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	Res1	Res0	WGM21	WGM20	325
...	Reserved									
(0xAD)	OCR4CH	OCR4CH7	OCR4CH6	OCR4CH5	OCR4CH4	OCR4CH3	OCR4CH2	OCR4CH1	OCR4CH0	291
(0xAC)	OCR4CL	OCR4CL7	OCR4CL6	OCR4CL5	OCR4CL4	OCR4CL3	OCR4CL2	OCR4CL1	OCR4CL0	292
(0xAB)	OCR4BH	OCR4BH7	OCR4BH6	OCR4BH5	OCR4BH4	OCR4BH3	OCR4BH2	OCR4BH1	OCR4BH0	291
(0xAA)	OCR4BL	OCR4BL7	OCR4BL6	OCR4BL5	OCR4BL4	OCR4BL3	OCR4BL2	OCR4BL1	OCR4BL0	291
(0xA9)	OCR4AH	OCR4AH7	OCR4AH6	OCR4AH5	OCR4AH4	OCR4AH3	OCR4AH2	OCR4AH1	OCR4AH0	290
(0xA8)	OCR4AL	OCR4AL7	OCR4AL6	OCR4AL5	OCR4AL4	OCR4AL3	OCR4AL2	OCR4AL1	OCR4AL0	290
(0xA7)	ICR4H	ICR4H7	ICR4H6	ICR4H5	ICR4H4	ICR4H3	ICR4H2	ICR4H1	ICR4H0	292
(0xA6)	ICR4L	ICR4L7	ICR4L6	ICR4L5	ICR4L4	ICR4L3	ICR4L2	ICR4L1	ICR4L0	292
(0xA5)	TCNT4H	TCNT4H7	TCNT4H6	TCNT4H5	TCNT4H4	TCNT4H3	TCNT4H2	TCNT4H1	TCNT4H0	289
(0xA4)	TCNT4L	TCNT4L7	TCNT4L6	TCNT4L5	TCNT4L4	TCNT4L3	TCNT4L2	TCNT4L1	TCNT4L0	289
...	Reserved									
(0xA2)	TCCR4C	FOC4A	FOC4B	FOC4C	Res4	Res3	Res2	Res1	Res0	288
(0xA1)	TCCR4B	ICNC4	ICES4	Res	WGM43	WGM42	CS42	CS41	CS40	287
(0xA0)	TCCR4A	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	285
...	Reserved									
(0x9D)	OCR3CH	OCR3CH7	OCR3CH6	OCR3CH5	OCR3CH4	OCR3CH3	OCR3CH2	OCR3CH1	OCR3CH0	282
(0x9C)	OCR3CL	OCR3CL7	OCR3CL6	OCR3CL5	OCR3CL4	OCR3CL3	OCR3CL2	OCR3CL1	OCR3CL0	283
(0x9B)	OCR3BH	OCR3BH7	OCR3BH6	OCR3BH5	OCR3BH4	OCR3BH3	OCR3BH2	OCR3BH1	OCR3BH0	282
(0x9A)	OCR3BL	OCR3BL7	OCR3BL6	OCR3BL5	OCR3BL4	OCR3BL3	OCR3BL2	OCR3BL1	OCR3BL0	282
(0x99)	OCR3AH	OCR3AH7	OCR3AH6	OCR3AH5	OCR3AH4	OCR3AH3	OCR3AH2	OCR3AH1	OCR3AH0	281
(0x98)	OCR3AL	OCR3AL7	OCR3AL6	OCR3AL5	OCR3AL4	OCR3AL3	OCR3AL2	OCR3AL1	OCR3AL0	281
(0x97)	ICR3H	ICR3H7	ICR3H6	ICR3H5	ICR3H4	ICR3H3	ICR3H2	ICR3H1	ICR3H0	283
(0x96)	ICR3L	ICR3L7	ICR3L6	ICR3L5	ICR3L4	ICR3L3	ICR3L2	ICR3L1	ICR3L0	283
(0x95)	TCNT3H	TCNT3H7	TCNT3H6	TCNT3H5	TCNT3H4	TCNT3H3	TCNT3H2	TCNT3H1	TCNT3H0	280
(0x94)	TCNT3L	TCNT3L7	TCNT3L6	TCNT3L5	TCNT3L4	TCNT3L3	TCNT3L2	TCNT3L1	TCNT3L0	280
...	Reserved									

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x92)	TCCR3C	FOC3A	FOC3B	FOC3C	Res4	Res3	Res2	Res1	Res0	279
(0x91)	TCCR3B	ICNC3	ICES3	Res	WGM33	WGM32	CS32	CS31	CS30	278
(0x90)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	276
...	Reserved									
(0x8D)	OCR1CH	OCR1CH7	OCR1CH6	OCR1CH5	OCR1CH4	OCR1CH3	OCR1CH2	OCR1CH1	OCR1CH0	273
(0x8C)	OCR1CL	OCR1CL7	OCR1CL6	OCR1CL5	OCR1CL4	OCR1CL3	OCR1CL2	OCR1CL1	OCR1CL0	273
(0x8B)	OCR1BH	OCR1BH7	OCR1BH6	OCR1BH5	OCR1BH4	OCR1BH3	OCR1BH2	OCR1BH1	OCR1BH0	272
(0x8A)	OCR1BL	OCR1BL7	OCR1BL6	OCR1BL5	OCR1BL4	OCR1BL3	OCR1BL2	OCR1BL1	OCR1BL0	272
(0x89)	OCR1AH	OCR1AH7	OCR1AH6	OCR1AH5	OCR1AH4	OCR1AH3	OCR1AH2	OCR1AH1	OCR1AH0	271
(0x88)	OCR1AL	OCR1AL7	OCR1AL6	OCR1AL5	OCR1AL4	OCR1AL3	OCR1AL2	OCR1AL1	OCR1AL0	271
(0x87)	ICR1H	ICR1H7	ICR1H6	ICR1H5	ICR1H4	ICR1H3	ICR1H2	ICR1H1	ICR1H0	273
(0x86)	ICR1L	ICR1L7	ICR1L6	ICR1L5	ICR1L4	ICR1L3	ICR1L2	ICR1L1	ICR1L0	274
(0x85)	TCNT1H	TCNT1H7	TCNT1H6	TCNT1H5	TCNT1H4	TCNT1H3	TCNT1H2	TCNT1H1	TCNT1H0	270
(0x84)	TCNT1L	TCNT1L7	TCNT1L6	TCNT1L5	TCNT1L4	TCNT1L3	TCNT1L2	TCNT1L1	TCNT1L0	271
...	Reserved									
(0x82)	TCCR1C	FOC1A	FOC1B	FOC1C	Res4	Res3	Res2	Res1	Res0	270
(0x81)	TCCR1B	ICNC1	ICES1	Res	WGM13	WGM12	CS12	CS11	CS10	268
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	266
(0x7F)	DIDR1							AIN1D	AIN0D	409
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	433
(0x7D)	DIDR2	ADC15D	ADC14D	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	433
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	427
(0x7B)	ADCSRB	AVDDOK	ACME	REFOK	ACCH	MUX5	ADTS2	ADTS1	ADTS0	428
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	430
(0x79)	ADCH	ADCH7	ADCH6	ADCH5	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	432
(0x78)	ADCL	ADCL7	ADCL6	ADCL5	ADCL4	ADCL3	ADCL2	ADCL1	ADCL0	432
(0x77)	ADCSRC	ADTHT1	ADTHT0	Res0	ADSUT4	ADSUT3	ADSUT2	ADSUT1	ADSUT0	431
...	Reserved									
(0x75)	NEMCR	Res7	ENEAM	AEAM1	AEAM0	Res3	Res2	Res1	Res0	463
(0x74)	Reserved	Res7	Res6	Res5	Res4	Res3	Res2	Res1	Res0	
(0x73)	TIMSK5	Res1	Res0	ICIE5	Res	OCIE5C	OCIE5B	OCIE5A	TOIE5	302
(0x72)	TIMSK4	Res1	Res0	ICIE4	Res	OCIE4C	OCIE4B	OCIE4A	TOIE4	293
(0x71)	TIMSK3	Res1	Res0	ICIE3	Res	OCIE3C	OCIE3B	OCIE3A	TOIE3	284
(0x70)	TIMSK2	Res4	Res3	Res2	Res1	Res0	OCIE2B	OCIE2A	TOIE2	323
(0x6F)	TIMSK1	Res1	Res0	ICIE1	Res	OCIE1C	OCIE1B	OCIE1A	TOIE1	274
(0x6E)	TIMSK0	Res4	Res3	Res2	Res1	Res0	OCIE0B	OCIE0A	TOIE0	242
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	224
(0x6C)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	224
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	225
(0x6A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	220
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	219
(0x68)	PCICR	Res4	Res3	Res2	Res1	Res0	PCIE2	PCIE1	PCIE0	223
(0x67)	BGCR	Res	BGCAL_FINE3	BGCAL_FINE2	BGCAL_FINE1	BGCAL_FINE0	BGCAL2	BGCAL1	BGCAL0	433
(0x66)	OSCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	153
(0x65)	PRR1	Res	PRTRX24	PRTIM5	PRTIM4	PRTIM3			PRUSART1	168
(0x64)	PRR0	PRTWI	PRTIM2	PRTIM0	PRPGA	PRTIM1	PRSPI	PRUSART0	PRADC	167
(0x63)	PRR2	Res3	Res2	Res1	Res0	PRRAM3	PRRAM2	PRRAM1	PRRAM0	168
...	Reserved									
(0x61)	CLKPR	CLKPCE	Res2	Res1	Res0	CLKPS3	CLKPS2	CLKPS1	CLKPS0	154
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	183
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	14
...	Reserved									
0x3B (0x5B)	RAMPZ	Res5	Res4	Res3	Res2	Res1	Res0	RAMPZ1	RAMPZ0	14
...	Reserved									
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	462
...	Reserved									
0x35 (0x55)	MCUCR	JTD	Res1	Res0	PUD	Res1	Res0	IVSEL	IVCE	204
0x34 (0x54)	MCUSR	Res2	Res1	Res0	JTRF	WDRF	BORF	EXTRF	PORF	183
0x33 (0x53)	SMCR	Res3	Res2	Res1	Res0	SM2	SM1	SM0	SE	166
...	Reserved									
0x31 (0x51)	OCDR	OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	440
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	408

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
...	Reserved									
0x2E (0x4E)	SPDR	SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0	338
0x2D (0x4D)	SPSR	SPIF	WCOL	Res4	Res3	Res2	Res1	Res0	SPI2X	337
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	336
0x2B (0x4B)	GPIOR2	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	26
0x2A (0x4A)	GPIOR1	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	26
...	Reserved									
0x28 (0x48)	OCR0B	OCR0B_7	OCR0B_6	OCR0B_5	OCR0B_4	OCR0B_3	OCR0B_2	OCR0B_1	OCR0B_0	242
0x27 (0x47)	OCR0A	OCR0A_7	OCR0A_6	OCR0A_5	OCR0A_4	OCR0A_3	OCR0A_2	OCR0A_1	OCR0A_0	241
0x26 (0x46)	TCNT0	TCNT0_7	TCNT0_6	TCNT0_5	TCNT0_4	TCNT0_3	TCNT0_2	TCNT0_1	TCNT0_0	241
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	Res1	Res0	WGM02	CS02	CS01	CS00	240
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	Res1	Res0	WGM01	WGM00	238
0x23 (0x43)	GTCCR	TSM	Res4	Res3	Res2	Res1	Res0	PSRASY	PSRSYNC	329
0x22 (0x42)	EEARH	Res3	Res2	Res1	Res0	EEAR11	EEAR10	EEAR9	EEAR8	23
0x21 (0x41)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	23
0x20 (0x40)	EEDR	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	23
0x1F (0x3F)	EECR	Res1	Res0	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	24
0x1E (0x3E)	GPIOR0	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	26
0x1D (0x3D)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	222
0x1C (0x3C)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	222
0x1B (0x3B)	PCIFR	Res4	Res3	Res2	Res1	Res0	PCIF2	PCIF1	PCIF0	223
0x1A (0x3A)	TIFR5	Res1	Res0	ICF5	Res	OCF5C	OCF5B	OCF5A	TOV5	302
0x19 (0x39)	TIFR4	Res1	Res0	ICF4	Res	OCF4C	OCF4B	OCF4A	TOV4	293
0x18 (0x38)	TIFR3	Res1	Res0	ICF3	Res	OCF3C	OCF3B	OCF3A	TOV3	284
0x17 (0x37)	TIFR2	Res4	Res3	Res2	Res1	Res0	OCF2B	OCF2A	TOV2	324
0x16 (0x36)	TIFR1	Res1	Res0	ICF1	Res	OCF1C	OCF1B	OCF1A	TOV1	275
0x15 (0x35)	TIFR0	Res4	Res3	Res2	Res1	Res0	OCF0B	OCF0A	TOV0	243
0x14 (0x34)	PORTG	Res1	Res0	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	209
0x13 (0x33)	DDRG	Res1	Res0	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	209
0x12 (0x32)	PING	Res1	Res0	PING5	PING4	PING3	PING2	PING1	PING0	210
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	208
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	208
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	209
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	207
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	207
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	208
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	206
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	207
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	207
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	27
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	28
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	28
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	205
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	206
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	206
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	27
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	27
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	27

Notes:
1. Reserved registers, bits and I/O memory addresses (marked as Res*) may not be modified.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that 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 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 device is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Op-code for the IN and OUT instructions. For the Extended I/O space from 0x60 – 0x1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

34 Electrical Characteristics

34.1 Absolute Maximum Ratings

Note that 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 are not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
T_{STOR}	Storage temperature		-50		150	°C
T_{LEAD}	Lead temperature	T = 10s, (soldering profile compliant with IPC/JEDEC J-STD-020B)			260	°C
V_{ESD}	ESD robustness	Compl. to [3], Compl. to [4]	4 750			kV V
P_{RF}	Input RF level				+14	dBm
V_{DIG}	Voltage on all pins (except pins 4, 5, 13, 14, 29)		-0.3		V_{DD}+0.3	V
V_{ANA}	Voltage on pins 4, 5, 13, 14, 29		-0.3		2.0	V

34.1.1 Recommended Operating Range

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
T_{OP}	Operating temperature range		-40		+85	°C
V_{DD}	Supply voltage	Voltage on pins 15,28(2)	1.8	3.0	3.6	V
$V_{DD1.8}$	Supply voltage (on pins 13, 14, 29)	External voltage supply(1)	1.7	1.8	1.9	V

Notes:
1. Register VREG_CTRL needs to be programmed to disable internal voltage regulators and supply blocks by an external 1.8V supply, refer to section "Voltage Regulators (AVREG, DVREG)" on page 163.
2. Even if an implementation uses the external 1.8V voltage supply $V_{DD1.8}$ it is required to connect V_{DD}.

34.1.2 Digital Pin Characteristics

Test Conditions: T_{OP} = 25°C (unless otherwise stated)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IH}	High level input voltage(1)		V_{DD} – 0.4			V
V_{IL}	Low level input voltage(1)				0.4	V
V_{OH}	High level output voltage(1)	For all output driver strengths defined in DPDS0, DPDS1	V_{DD} – 0.4			V
V_{OL}	Low level output voltage(1)	For all output driver strengths defined in DPDS0, DPDS1			0.4	V

Note:
5. The capacitive load should not be larger than 50 pF for all I/Os when using the default driver strength settings, refer to section "DPDS0 – Port Driver Strength Register 0" on page 174 and "DPDS1 – Port Driver Strength Register 1" on page 175. Generally, large load capacitances increase the overall current consumption.

34.2 Clock Characteristics

34.2.1 Calibrated Internal RC Oscillator Accuracy

Table 34-2. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{DEVDD}	Temperature	Calibration Accuracy
Factory Calibration	16 MHz	3.0V	25°C	± TBD %
User Calibration	TBD	1.8V – 3.6V	-TBD°C - TBD°C	± TBD %

34.2.2 External Clock Drive

Figure 34-1 External Clock Drive Waveforms

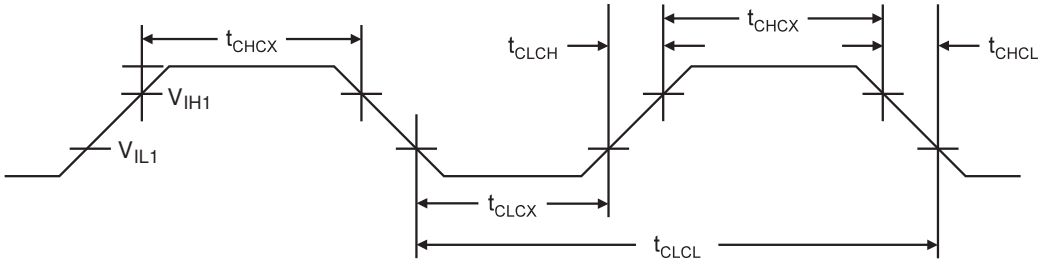


Table 34-3. External Clock Drive

Symbol	Parameter	Min.	Max.	Units
$1/t_{CLCL}$	Oscillator Frequency		16	MHz
t_{CLCL}	Clock Period	62.5		ns
t_{CHCX}	High Time	25		ns
t_{CLCX}	Low Time	25		ns
t_{CLCH}	Rise Time		0.1	µs
t_{CHCL}	Fall Time		0.1	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		1	%

34.3 System and Reset Characteristics

Table 34-4. Reset, Brown-out and Internal Voltage Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			TBD		V
	Power-on Reset Threshold Voltage (falling)[1]			0.3		V
V_{PSR}	Power-on slope rate			TBD		V/ms
V_{RST}	RSTN Pin Threshold Voltage		$0.2V_{DEVDD}$		$0.9V_{DEVDD}$	V
t_{RST}	Minimum pulse width on RSTN Pin			TBD		ns
V_{HYS}	Brown-out Detector Hysteresis			50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset			100		ns
V_{BG}	Bandgap reference voltage	V_{DEVDD} = 3.0V, T_A = 25°C		1.2		V

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

Table 34-23. BODLEVEL Fuse Coding

BODLEVEL2:0 Fuses	Min V_{BOD}	Typ V_{BOD}	Max V_{BOD}	Units
111	BOD Disabled			
110		1.8		V
101		1.9		V
100		2.0		V
011		2.1		V
010		2.2		V
001		2.3		V
000		2.4		V

Note: V_{BOT} may be below nominal minimum operating voltage. The device is operated down to V_{DEVDD} = V_{BOT} during the production test. This guarantees that a Brown-Out Reset will occur before V_{DEVDD} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 110 for 16 MHz operation of the ATmega128RFA1.

34.4 Power Management Electrical Characteristics

34.4.1 Power Switches

Table 34-6. Timing Characteristics of the Power Switches

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{POR}	Power-on reset time	Applies if the device is powered up		TBD		µs
t_{BG}	Bandgap startup time			7		µs
t_{DRT_ON}	DRT switch switch-on time			2		µs
t_{PWRSW_ON}	Power switch switch-on time			2		µs

34.4.2 Voltage Regulators

Table 34-7. Timing Characteristics of the Voltage regulators

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{AVREG}	Power up time AVREG	external capacity on pin AVDD TBD		TBD		µs
t_{DVREG}	Power up time DVREG	Startup after wakeup, Startup after deep sleep, external capacity on pin DVDD TBD TRX24 and all SRAM modules enabled)		30		µs
t_{BG}	Power up time bandgap			TBD		µs

34.5 2-wire Serial Interface Characteristics

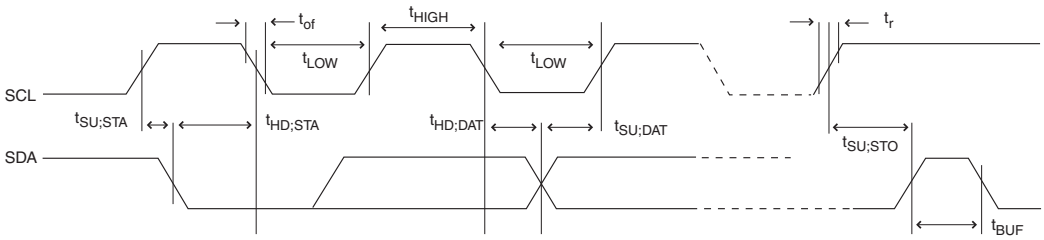
The timing characteristics refer to Table 34-8.

Table 34-8 2-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min.	Max.	Units
V_{IL}	Input Low-voltage		-0.5	$0.3V_{DEVDD}$	V

Symbol	Parameter	Condition	Min.	Max.	Units
V_{IH}	Input High-voltage		$0.7V_{DEVDD}$	V_{DEVDD} +0.5	V
V_{hys}	Hysteresis of Schmitt Trigger Inputs				V
V_{OL}	Output Low-voltage	3mA sink current	0	0.4	V
t_r	Rise Time for both SDA and SCL			300	ns
t_{of}	Output Fall Time from V_{IHmin} to V_{ILmax}			250	ns
t_{SP}	Spikes suppressed by the input filter			50	ns

Figure 34-2



34.6 SPI Timing Characteristics

See Figure 34-3and Figure 34-4 for details.

Table 34-9. SPI Timing Parameters

Description	Mode	Min	Typ	Max	Units
SCK period	Master		See "SPCR – SPI Control Register" on page 336.		ns
SCK high/low	Master		50% duty cycle		
Rise/fall time	Master		TBD		
Setup	Master		TBD		
Hold	Master		TBD		
Out to SCK	Master		0.5 t_{SCK}		
SCK to out	Master		TBD		
SCK to out high	Master		TBD		
$\overline{SS}$ low to out	Slave		TBD		
SCK period	Slave	4 t_{CK}			
SCK high/low[1]	Slave	2 t_{CK}			
Rise/fall time	Slave			TBD	
Setup	Slave	TBD			
Hold	Slave	t_{CK}			
SCK to out	Slave		TBD		
SCK to $\overline{SS}$ high	Slave	TBD			
$\overline{SS}$ high to tri-state	Slave		TBD		
$\overline{SS}$ low to SCK	Slave	TBD			

Note: 1. In SPI Programming mode the minimum SCK high/low period is 2 t_{CLCL} for f_{CK} < 12 MHz and 3 t_{CLCL} for f_{CK} > 12 MHz.

Figure 34-3. SPI timing Requirements (Master Mode)

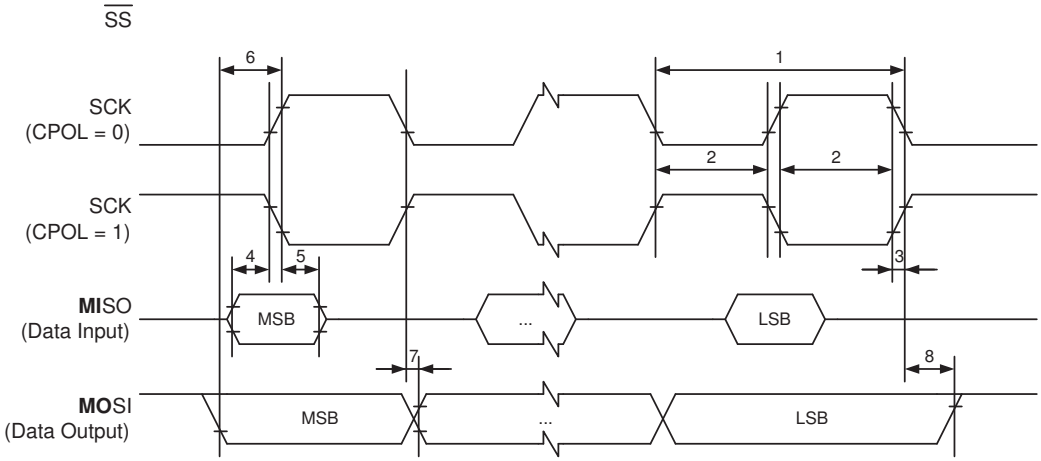
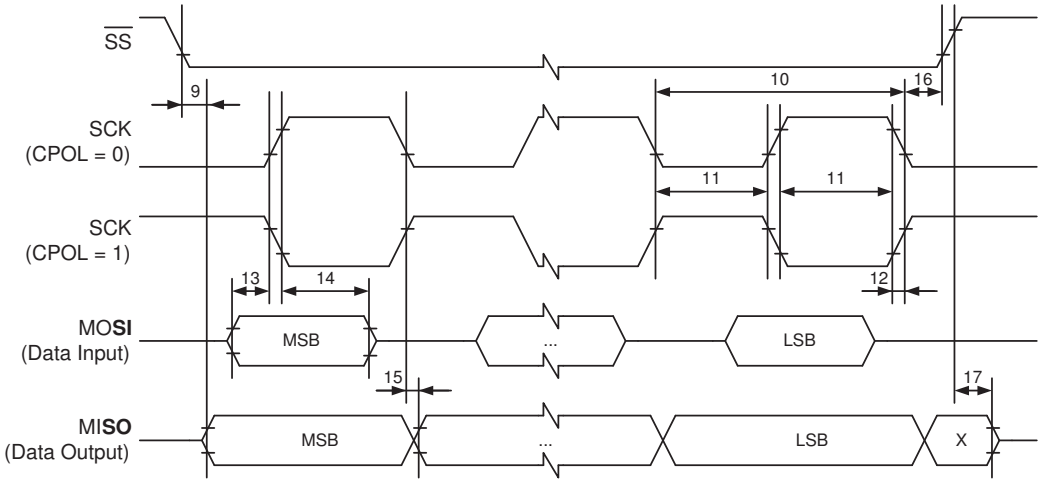


Figure 34-4. SPI timing Requirements (Slave Mode)



34.7 ADC Characteristics

Table 34-10. ADC Electrical Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{INT1}	Internal Voltage Reference			1.5		V
V_{INT2}	Internal Voltage Reference			1.6		V
V_{INT3}	Internal Voltage Reference			AVDD		V
$R_{AREF,EXT}$	External Voltage Impedance					Ω
$I_{L,AREF}$	Load Current					A

Table 34-11. ADC Characteristics, Single Ended Channels

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution	Single Ended Conversion CLKADC <= 4 MHz		10		Bits
		Single Ended Conversion CLKADC > 8 MHz		8		Bits
	Absolute accuracy (Including	Single Ended Conversion		TBD		LSB

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	INL, DNL, quantization error, gain and offset error)	VREF = 1.6V CLKADC = 200kHz				
		Single Ended Conversion VREF = 1.6V CLKADC = 1MHz		TBD		LSB
		Single Ended Conversion VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		TBD		LSB
	Integral Non-Linearity (INL)	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		0.8		LSB
	Differential Non-Linearity (DNL)	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz	-0.5			LSB
	Gain Error	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		1		LSB
	Offset Error	Single Ended Conversion VREF = 1.6V CLKADC = 4MHz		1.5		LSB
	Conversion Time	Free Running Conversion	3		240	µs
	Clock Frequency	Single Ended Conversion			8000	kHz
V_{REF}	Reference Voltage		1.5		AVDD	V
V_{IN}	Input Voltage		0		AVDD	V
	Input Bandwith		20			kHz
R_{REF}	Reference Input Resistance			TBD		kΩ
R_{AIN}	Analog Input Resistance			2		kΩ

Note: 1. Values are guidelines only
2. All values are valid for EVDD = 3.0V

Table 34-12. PGA and ADC Characteristics, Differential Channels

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution	Gain = 1x		10		Bits
		Gain = 10x		10		Bits
		Gain = 200x		10		Bits
	Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Gain = 1x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz		TBD		LSB
	Integral Non-Linearity (INL)	Gain = 1x VREF = 1.6V CLKADC = 2MHz			3	LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz			5	LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz			10	LSB
	Differential Non-Linearity (DNL)	Gain = 1x VREF = 1.6V CLKADC = 2MHz	-0.5			LSB

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
		Gain = 10x VREF = 1.6V CLKADC = 2MHz	-0.75			LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz	TBD			LSB
	Gain Error	Gain = 1x		1		LSB
		Gain = 10x		1.5		
		Gain = 200x		10		
	Offset Error	Gain = 1x VREF = 1.6V CLKADC = 2MHz		0.7		LSB
		Gain = 10x VREF = 1.6V CLKADC = 2MHz		0.75		LSB
		Gain = 200x VREF = 1.6V CLKADC = 2MHz		13		LSB
	Conversion Time	Free Running Conversion	100			µs
	Clock Frequency	Single Ended Conversion			2000	kHz
V_{REF}	Reference Voltage		1.5		AVDD	V
V_{IN}	Input Voltage		0		AVDD	V
V_{DIFF}	Input Differential Voltage		-AVDD		AVDD	V
	ADC Conversion Output		-512		511	LSB
	Input Bandwith		20			kHz
R_{REF}	Reference Input Resistance			TBD		kΩ
R_{AIN}	Analog Input Resistance			2		kΩ

Note: 1. Values are guidelines only
2. All values are valid for EVDD = 3.0V

34.8 Transceiver Electrical Characteristics

34.8.1 Digital Interface Timing Characteristics

Test Conditions: T_{OP} = 25°C, V_{DD} = 3.0V, C_L = 50 pF (unless otherwise stated)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{12}	AES core cycle time			24		µs
t_{IRQ}	Interrupt event latency	Relative to the event to be indicated		9		µs

34.8.2 General RF Specifications

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_{RF}	Frequency range	As specified in [1],[2]	2405		2480	MHz
f_{CH}	Channel spacing	As specified in [1],[2]		5		MHz
f_{HDR}	Header bit rate (SHR, PHR)	As specified in [1],[2]		250		kb/s

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_{PSDU}	PSDU bit rate	As specified in [1],[2] OQPSK_DATA_RATE = 1 OQPSK_DATA_RATE = 2 OQPSK_DATA_RATE = 3		250 500 1000 2000		kb/s kb/s kb/s kb/s
f_{CHIP}	Chip rate	As specified in [1],[2]		2000		kchip/s
f_{CLK}	Crystal oscillator frequency	Reference oscillator		16		MHz
t_{XTAL}	Reference oscillator settling time	Leaving SLEEP state to clock available at pin 17 (CLKM)		215	1000	µs
	Symbol rate deviation Reference frequency accuracy for correct functionality	PSDU bit rate 250 kb/s 500 kb/s 1000 kb/s 2000 kb/s	-60(1) -40 -40 -30		+60 +40 +40 +30	ppm ppm ppm ppm
B_{20dB}	20 dB bandwidth			2.8		MHz

Note: 6. A reference frequency accuracy of ±40 ppm is required by [1], [2].

34.8.3 Transmitter Characteristics

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
P_{TX}	TX Output power	Maximum configurable TX output power value Register bit TX_PWR = 0	0	+3.5	+6	dBm
P_{RANGE}	Output power range	16 steps, configurable in register 0x05 (PHY_TX_PWR)		20		dB
P_{ACC}	Output power tolerance				±3	dB
	TX Return loss	100Ω differential impedance, P_{TX} = +3.5 dBm		10		dB
	EVM			8		%rms
P_{HARM}	Harmonics 2nd harmonic 3rd harmonic			 -38 -45		 dBm dBm
P_{SPUR}	Spurious Emissions 30 – ≤ 1000 MHz >1 – 12.75 GHz 1.8 – 1.9 GHz 5.15 – 5.3 GHz	Complies with EN 300 328/440, FCC-CFR-47 part 15, ARIB STD-66, RSS-210		 -36 -30 -47 -47		 dBm dBm dBm dBm

34.8.4 Receiver Characteristics

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, PSDU bit rate = 250 kb/s, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
P_{SENS}	Receiver sensitivity 250 kb/s 500 kb/s 1000 kb/s 2000 kb/s	AWGN channel, PER ≤ 1%, PSDU length 20 octets High Data Rate Modes: PSDU length 20 octets		 -100 -96 -94 -86		 dBm dBm dBm dBm
	Antenna Diversity	250 kb/s, PSDU 20 octets		-99		dBm
RL	Return loss	100Ω differential impedance		10		dB
NF	Noise figure			6		dB
P_{RXMAX}	Maximum RX input level	PER ≤ 1%, PSDU length of 20 octets		10		dBm
P_{ACRN}	Adjacent channel rejection: -5 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		34		dB
P_{ACRP}	Adjacent channel rejection: +5 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		38		dB
P_{AACRN}	Alternate channel rejection: -10 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		54		dB
P_{AACRP}	Alternate channel rejection: +10 MHz	PER ≤ 1%, PSDU length of 20 octets, P_{RF} = -82 dBm		54		dB
P_{SPUR}	Spurious emissions: LO leakage 30 – ≤ 1000 MHz >1 – 12.75 GHz			 -71	 -57 -47	 dBm dBm dBm
$f_{RXTXOFFS}$	TX/RX carrier frequency offset	Sensitivity loss < 2 dB	-300(1)		+300	kHz
IIP3	3rd – order intercept point	At maximum gain Offset freq. interf. 1 = 5 MHz Offset freq. interf. 2 = 10 MHz		-14		dBm
IIP2	2nd – order intercept point	At maximum gain Offset freq. interf. 1 = 60 MHz Offset freq. interf. 2 = 62 MHz		17		dBm
	RSSI tolerance	Tolerance within gain step			±5	dB
	RSSI dynamic range			81		dB
	RSSI resolution			3		dB
	RSSI sensitivity	Defined as RSSI_BASE_VAL		-90		dBm
	Minimum RSSI value	P_{RF} ≤ RSSI_BASE_VAL		0		
	Maximum RSSI value	P_{RF} > RSSI_BASE_VAL + 81 dB		28		

Note: 1. Offset equals ±120 ppm

34.8.5 Current Consumption Specifications

Test Conditions (unless otherwise stated):

V_{DD} = 3.0V, f_{RF} = 2.45 GHz, T_{OP} = 25°C, Measurement setup see Figure 32-1 on page 493.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{BUSY_TX}	Supply current transmit state	P_{TX} = 3 dBm P_{TX} = 1 dBm P_{TX} = -3 dBm P_{TX} = -17 dBm (current consumption is reduced at V_{DD} = 1.8V for each output power level)		14.5 10 9 8		mA mA mA mA
I_{RX_ON}	Supply current RX_ON state	RX_ON state		12.5		mA
$I_{RX_ON_P}$	Supply current RX_ON state	RX_ON state, with register setting RX_PDT_LEVEL > 0[(1)]		12.0		mA
I_{PLL_ON}	Supply current PLL_ON state	PLL_ON state		5.7		mA
I_{TRX_OFF}	Supply current TRX_OFF state	TRX_OFF state		0.4		mA
I_{SLEEP}	Supply current SLEEP state	SLEEP state		0.02		µA

Note: 1. Refer to section "Figure 32-1" on page 493

34.8.6 Crystal Parameter Requirements

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
f_0	Crystal frequency			16		MHz
C_L	Load capacitance		8		14	pF
C_0	Static capacitance				7	pF
R_1	Series resistance				100	Ω

35 Typical Characteristics

35.1 Internal Oscillator Speed

t.b.d.

36 Ordering Information

ATmega128RFA1

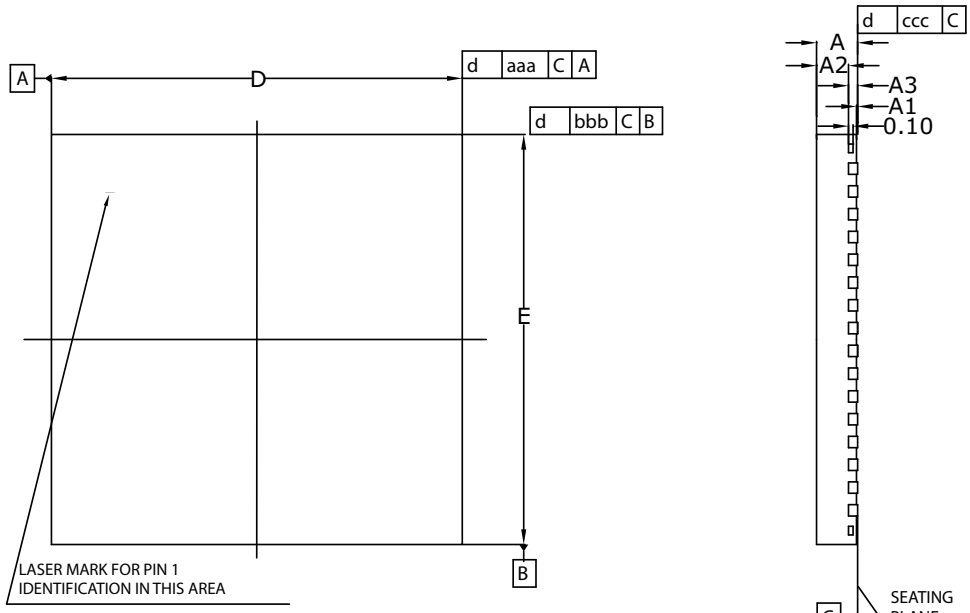
Speed (MHz)	Power Supply	Ordering Code	Package	Packing	Operation Range
16	1.8 – 3.6V	ATmega128RFA1-ZU	PI	Tray	Industrial (-40ºC to 85ºC)
16	1.8 – 3.6V	ATmega128RFA1-ZUR	PI	Tape & Reel	Industrial (-40ºC to 85ºC)

Notes: 1. Pb-free packaging, complies to European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
PI	64-lead, 9 x 9 x 0.9 mm Body, Quad Flat No-lead Package (QFN)

37 Packaging Information

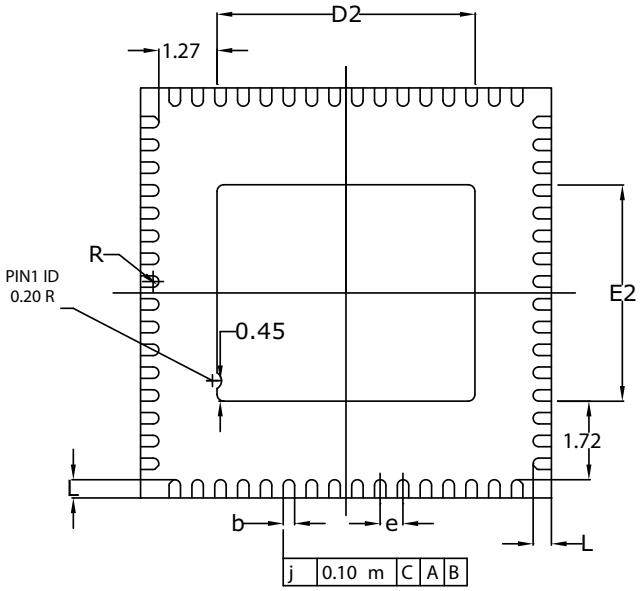
PI



TOP VIEW

DRAWINGS NOT SCALED

SIDE VIEW



BOTTOM VIEW

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	---	---	0.90	---	---	0.035
A1	---	---	0.05	---	---	0.001
A2	---	0.65	0.70	---	0.026	0.028
A3	0.20 REF.			0.008 REF.		
b	0.18	0.25	0.30	0.007	0.010	0.012
D	9.00 bsc			0.354 bsc		
D2	5.55	5.65	5.75	0.219	0.222	0.226
E	9.00 bsc			0.354 bsc		
E2	4.65	4.75	4.85	0.183	0.187	0.191
L	0.35	0.40	0.45	0.014	0.016	0.018
e	0.50 bsc			0.020 bsc		
R	0.09	---	---	0.004	---	---
TOLERANCES OF FORM AND POSITION						
aaa	0.10			0.004		
bbb	0.10			0.004		
ccc	0.05			0.002		

ALL DIMENSIONS ARE IN MILLIMETERS.

PACKAGE WARPAGE MAX 0.08 mm.

02/12/2008

Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	TITLE PI - 64 leads - 9.0 x 9.0 mm - pitch 0.5mm Quad Flat No Lead Package QFN	DRAWING No. PI	REV. A

38 Errata

38.1 ATmega128RFA1 revision D (1.2)

- **Power-Chain turns off when power supply drops below 1.6V**
- **JTAG interface reads wrong data**
- **CSMA back-off calculation has reduced degree of randomness**
- **Update of internal temporary registers for CSMA_SEED register may fail**
- **Interrupt TRX24_CCA_ED_DONE may occur twice**

38.2 ATmega128RFA1 revision C (1.1)

- **Power-Chain turns off when power supply drops below 1.6V**
- **JTAG interface reads wrong data**
- **CSMA back-off calculation has reduced degree of randomness**
- **Update of internal temporary registers for CSMA_SEED register may fail**
- **Interrupt TRX24_CCA_ED_DONE may occur twice**
- **DVREG_EXT bit is not write-protected**
- **ENDRT bits have wrong reset value**

38.3 ATmega128RFA1 revision AB (1.0)

Not sampled.

38.4 Compiler package WinAVR-20090313

In the compiler package WinAVR-20090313 the SRAM start address has a wrong value of 0x100. In this case the variables are randomly allocated across the extended I/O space 0x100 to 0x1FF. It causes an unpredictable behavior by random overwrite of registers (see also "JTAG interface reads wrong data" on page 514 and "DVREG_EXT bit is not write-protected" on page 514).

Problem Fix/Workaround

Use the linker option -Wl,--section-start=.data=0x800200

38.5 Detailed errata description

38.5.1 Power-Chain turns off when power supply drops below 1.6V

If the voltage of the pins DEVDD drops below 1.6V, the internal power chain turns off. Some hardware settings (e.g. clock source) can alter their state unintentionally. Raising the supply voltage above 1.8V again does not bring the circuit back to normal operation. This condition can happen either by lowering the power supply voltage below 1.6V or turn-off the supply source while other external devices are feeding DEVDD by the internal ESD diodes of the IO stages (e.g. hardware debugger attached to the JTAG interface) (2606).

If the power supply drops below 1.6V while being in Deep Sleep mode, the internal power chain is not affected.

Problem Fix/Workaround

Turn on the Brown-Out Detector at any voltage level. The supply current in Deep Sleep does not increase.

38.5.2 JTAG interface reads wrong data

If the Power Reduction Register bits associated with the SRAM's (PRRAM3…0 in PRR2) and the 2.4GHz Transceiver (PRTRX24 in PRR1) are set, the JTAG interface reads wrong data. (2613).

Problem Fix/Workaround

Do not use PRRAM3…0 in PRR2 and PRTRX24 in PRR1. Force pin RSTN=0 and the JTAG interface can erase the program memory.

38.5.3 CSMA back-off calculation has reduced degree of randomness

The CSMA back-off calculation in the transceiver extended operating modes has a reduced degree of randomness (e.g. transceiver is in the state TX_ARET_ON) (2665).

Problem Fix/Workaround

Initialize CSMA_SEED registers with a random value.

38.5.4 Update of internal temporary registers for CSMA_SEED register may fail

The update of the internal temporary registers of the CSMA_SEED registers may fail. Read/write operation to the CSMA_SEED registers itself works as expected (2646).

Problem Fix/Workaround

A sleep cycle of the transceiver updates the internal temporary registers.

38.5.5 Interrupt TRX24_CCA_ED_DONE may occur twice

When requesting a manually initiated CCA measurement in BUSY_RX state and during an internal ED measurement, a TRX24_CCA_ED_DONE interrupt could be issued immediately after the request. In this case the register bit CCA_DONE is equal to 0 and an additional TRX24_CCA_ED_DONE interrupt is issued after finishing the CCA measurement and register bit CCA_DONE is set to 1 (2000).

Problem Fix/Workaround

Prevent a frame reception during manually initiated CCA measurement

- make sure that TRX_STATUS is not in RX_BUSY (i.e. start from state PLL_ON)
- set bit RX_PDT_DIS=1
- switch TRX_STATE to RX_ON
- perform CCA measurement
- set bit RX_PDT_DIS=0

38.5.6 DVREG_EXT bit is not write-protected

The external mode of the DVDD voltage regulator is not write-protected. If it is enabled (DVREG_EXT=1 in the register VREG_CTRL) with no external power supply for DVDD, the device leaves normal operation and can't be recovered by the Watchdog (2658).

Problem Fix/Workaround

Do not write the bit DVREG_EXT in the register VREG_CTRL.

38.5.7 ENDRT bits have wrong reset value

The ENDRT bits in the registers DRTRAM3…0 have the wrong reset value. The data retention of the associated SRAM in DEEP_SLEEP is disabled (2495).

Problem Fix/Workaround

Set ENDRT=1 in DRTRAM3…0 at the beginning of the firmware program.

39 Revision history

Please note that the referring page numbers in this section are referring to this document. The referring revision in this section are referring to the document revision

Rev. 8266A-MCU Wireless-12/09

1. Initial release

Table of Contents

1 Pin Configurations ***2***

2 Disclaimer ***2***

3 Overview ***3***

3.1 Block Diagram 3

3.2 Pin Descriptions 5

3.3 Compatibility to ATmega1281/2561 6

4 Resources ***7***

5 About Code Examples ***7***

6 Data Retention ***7***

7 AVR CPU Core ***9***

7.1 Introduction 9

7.2 Architectural Overview 9

7.3 ALU – Arithmetic Logic Unit 10

7.4 Status Register 11

7.5 General Purpose Register File 12

7.6 Stack Pointer 13

7.7 Instruction Execution Timing 15

7.8 Reset and Interrupt Handling 15

8 AVR Memories ***18***

8.1 In-System Reprogrammable Flash Program Memory 18

8.2 SRAM Data Memory 18

8.3 EEPROM Data Memory 20

8.4 EEPROM Register Description 23

8.5 I/O Memory 25

8.6 General Purpose I/O Registers 26

8.7 Other Port Registers 26

9 Low-Power 2.4 GHz Transceiver ***29***

9.1 Features 29

9.2 General Circuit Description 30

9.3 Transceiver to Microcontroller Interface 31

9.4 Operating Modes 35

9.5 Functional Description 61

9.6 Module Description 74

9.7 Radio Transceiver Usage 83

9.8 Radio Transceiver Extended Feature Set 85

9.9 Continuous Transmission Test Mode 96
9.10 Abbreviations 98
9.11 Reference Documents 100
9.12 Register Description 100
10 MAC Symbol Counter 133
10.1 Main Features 133
10.2 Clock source selection and Sleep/Active mode operation 133
10.3 32 bit Register Access (Atomic Read/Write) 133
10.4 Symbol Counter (32 bit, SCCNT) 134
10.5 Symbol Counter SFD Timestamp Register (32 bit, SCTSR, Read Only) 134
10.6 Symbol Counter Beacon Timestamp Register (32 bit, SCBTSR) 134
10.7 Compare Unit (3x 32 bit, SCOCR1, SCOCR2, SCOCR3) 135
10.8 Interrupt Control Registers 135
10.9 Backoff Slot Counter 135
10.10 Symbol Counter Usage 135
10.11 Register Description 137
11 System Clock and Clock Options 147
11.1 Overview 147
11.2 Clock Systems and their Distribution 147
11.3 Clock Sources 148
11.4 Calibrated Internal RC Oscillator 149
11.5 128 kHz Internal Oscillator 150
11.6 External Clock 150
11.7 Transceiver Crystal Oscillator 151
11.8 Clock Output Buffer 152
11.9 Timer/Counter Oscillator 152
11.10 System Clock Prescaler 152
11.11 Register Description 153
12 Power Management and Sleep Modes 156
12.1 Deep-Sleep Mode 156
12.2 AVR Microcontroller Sleep Modes 156
12.3 Power Reduction Register 159
12.4 Minimizing Power Consumption 159
12.5 Supply Voltage and Leakage Control 161
12.6 Register Description 166
13 System Control and Reset 176

13.1 Resetting the AVR 176
13.2 Reset Sources 176
13.3 Internal Voltage Reference 179
13.4 Watchdog Timer 180
13.5 Register Description 183
14 I/O-Ports 186
14.1 Introduction 186
14.2 Ports as General Digital I/O 187
14.3 Alternate Port Functions 191
14.4 Register Description 204
15 Interrupts 211
15.1 Interrupt Vectors in ATmega128RFA1 211
15.2 Reset and Interrupt Vector Placement 213
15.3 Moving Interrupts Between Application and Boot Section 216
15.4 Register Description 216
16 External Interrupts 218
16.1 Pin Change Interrupt Timing 218
16.2 Register Description 219
17 8-bit Timer/Counter0 with PWM 226
17.1 Features 226
17.2 Overview 226
17.3 Timer/Counter Clock Sources 227
17.4 Counter Unit 227
17.5 Output Compare Unit 228
17.6 Compare Match Output Unit 230
17.7 Modes of Operation 232
17.8 Timer/Counter Timing Diagrams 236
17.9 Register Description 238
18 16-bit Timer/Counter (Timer/Counter 1, 3, 4, and 5) 244
18.1 Features 244
18.2 Overview 244
18.3 Accessing 16-bit Registers 246
18.4 Timer/Counter Clock Sources 249
18.5 Counter Unit 249
18.6 Input Capture Unit 250
18.7 Output Compare Units 252

18.8 Compare Match Output Unit 254
18.9 Modes of Operation 256
18.10 Timer/Counter Timing Diagrams 264
18.11 Register Description 266
19 Timer/Counter 0, 1, 3, 4, and 5 Prescaler 304
19.1 Internal Clock Source 304
19.2 Prescaler Reset 304
19.3 External Clock Source 304
19.4 Register Description 305
20 Output Compare Modulator (OCM1C0A) 307
20.1 Overview 307
20.2 Description 307
20.3 Timing Example 308
21 8-bit Timer/Counter2 with PWM and Asynchronous Operation 309
21.1 Features 309
21.2 Overview 309
21.3 Timer/Counter Clock Sources 310
21.4 Counter Unit 311
21.5 Modes of Operation 311
21.6 Output Compare Unit 316
21.7 Compare Match Output Unit 317
21.8 Timer/Counter Timing Diagrams 319
21.9 Asynchronous Operation of Timer/Counter2 320
21.10 Timer/Counter Prescaler 322
21.11 Register Description 323
22 SPI- Serial Peripheral Interface 330
22.1 Features 330
22.2 Functional Description 330
22.3 $\overline{SS}$ Pin Functionality 334
22.4 Register Description 336
23 USART 339
23.1 Features 339
23.2 Overview 339
23.3 Clock Generation 340
23.4 Frame Formats 343
23.5 USART Initialization 344

23.6 Data Transmission – The USART Transmitter 345
23.7 Data Reception – The USART Receiver 348
23.8 Asynchronous Data Reception 352
23.9 Multi-processor Communication Mode 355
23.10 Register Description 356
23.11 Examples of Baud Rate Setting 365
24 USART in SPI Mode 368
24.1 Overview 368
24.2 USART MSPIM vs. SPI 368
24.3 SPI Data Modes and Timing 369
24.4 Frame Formats 370
24.5 Data Transfer 371
24.6 USART MSPIM Register Description 373
25 2-wire Serial Interface 377
25.1 Features 377
25.2 2-wire Serial Interface Bus Definition 377
25.3 Data Transfer and Frame Format 378
25.4 Multi-master Bus Systems, Arbitration and Synchronization 380
25.5 Overview of the TWI Module 382
25.6 Using the TWI 384
25.7 Transmission Modes 387
25.8 Multi-master Systems and Arbitration 400
25.9 Register Description 401
26 AC – Analog Comparator 407
26.1 Analog Comparator Multiplexed Input 407
26.2 Register Description 408
27 ADC – Analog to Digital Converter 410
27.1 Features 410
27.2 Operation 411
27.3 ADC Start-Up 412
27.4 Starting a Conversion 413
27.5 Pre-scaling and Conversion Timing 413
27.6 Changing Channel or Reference Selection 417
27.7 ADC Noise Canceller 420
27.8 ADC Conversion Result 423
27.9 Internal Temperature Measurement 425

27.10 SRAM DRT Voltage Measurement 426
27.11 Register Description 427
28 JTAG Interface and On-chip Debug System 435
28.1 Features 435
28.2 Overview 435
28.3 TAP - Test Access Port 436
28.4 TAP Controller 437
28.5 Using the Boundary-scan Chain 438
28.6 Using the On-chip Debug System 438
28.7 On-chip Debug Specific JTAG Instructions 439
28.8 Using the JTAG Programming Capabilities 439
28.9 Bibliography 440
28.10 On-chip Debug Related Register in I/O Memory 440
29 IEEE 1149.1 (JTAG) Boundary-scan 441
29.1 Features 441
29.2 System Overview 441
29.3 Data Registers 441
29.4 Boundary-scan Specific JTAG Instructions 443
29.5 Boundary-scan Chain 444
29.6 Boundary-scan Related Register in I/O Memory 447
29.7 Boundary-scan Description Language Files 448
29.8 ATmega128RFA1 Boundary-scan Order 448
30 Boot Loader Support – Read-While-Write Self-Programming ... 450
30.1 Features 450
30.2 Application and Boot Loader Flash Sections 450
30.3 Read-While-Write and No Read-While-Write Flash Sections 451
30.4 Boot Loader Lock Bits 453
30.5 Addressing the Flash During Self-Programming 453
30.6 Self-Programming the Flash 454
30.7 Register Description 462
31 Memory Programming 464
31.1 Program And Data Memory Lock Bits 464
31.2 Fuse Bits 465
31.3 Signature Bytes 466
31.4 Calibration Byte 467
31.5 Page Size 467

31.6 Parallel Programming Parameters, Pin Mapping, and Commands 467
31.7 Parallel Programming 469
31.8 Serial Downloading 477
31.9 Programming via the JTAG Interface 481
32 Application Circuits 493
32.1 Basic Application Schematic 493
32.2 Extended Feature Set Application Schematic 494
33 Register Summary 496
34 Electrical Characteristics 501
34.1 Absolute Maximum Ratings 501
34.2 Clock Characteristics 502
34.3 System and Reset Characteristics 502
34.4 Power Management Electrical Characteristics 503
34.5 2-wire Serial Interface Characteristics 503
34.6 SPI Timing Characteristics 504
34.7 ADC Characteristics 505
34.8 Transceiver Electrical Characteristics 507
35 Typical Characteristics 510
35.1 Internal Oscillator Speed 510
36 Ordering Information 511
37 Packaging Information 512
38 Errata 513
38.1 ATmega128RFA1 revision D (1.2) 513
38.2 ATmega128RFA1 revision C (1.1) 513
38.3 ATmega128RFA1 revision AB (1.0) 513
38.4 Compiler package WinAVR-20090313 513
38.5 Detailed errata description 513
39 Revision history 516
Table of Contents 517

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

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 ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, 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 OF 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 product 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's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR®, and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.