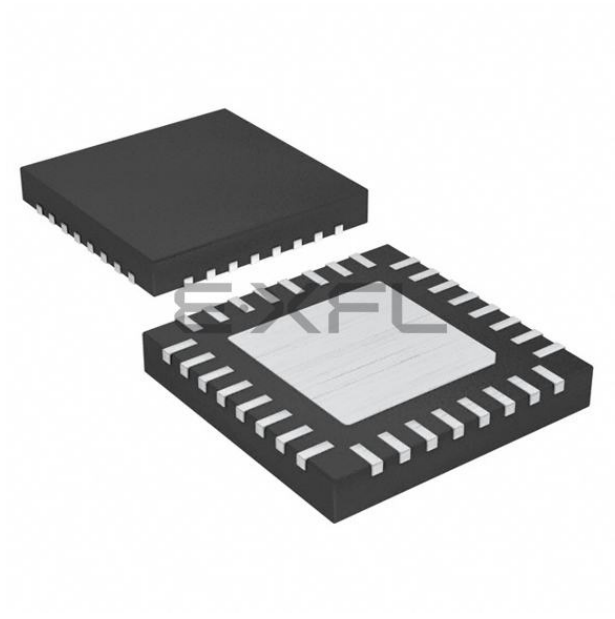


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

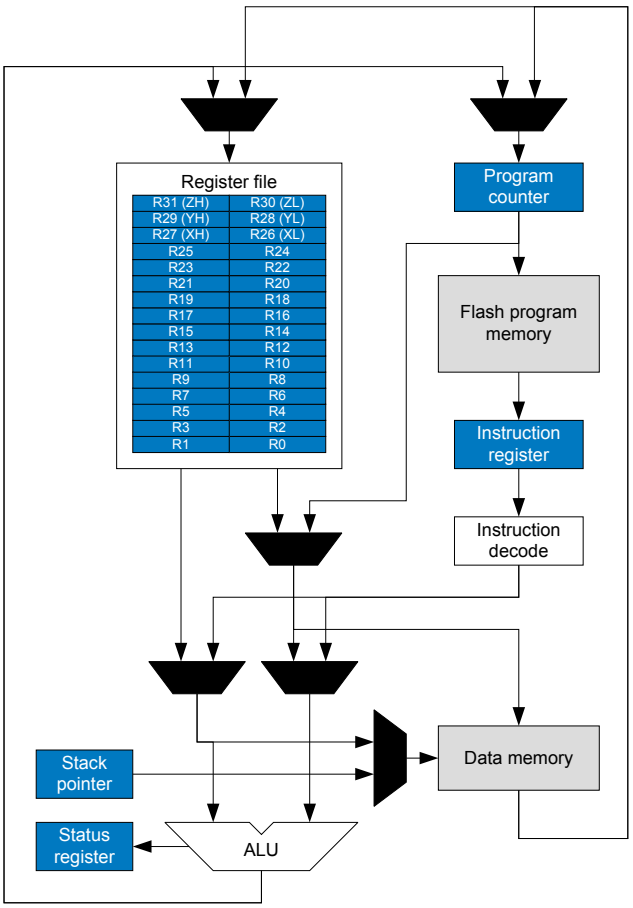
Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	27
Program Memory Size	4KB (2K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-VFQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega48pb-mn

12. AVR CPU Core

12.1. Overview

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

Figure 12-1. Block Diagram of the AVR Architecture

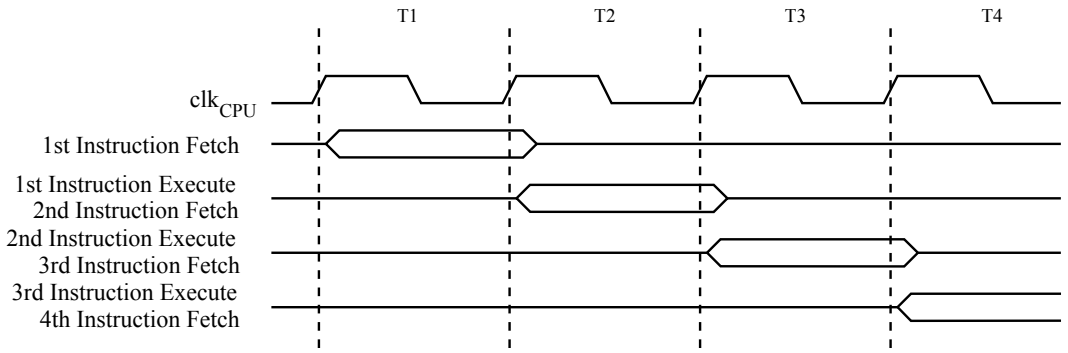


In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from 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.

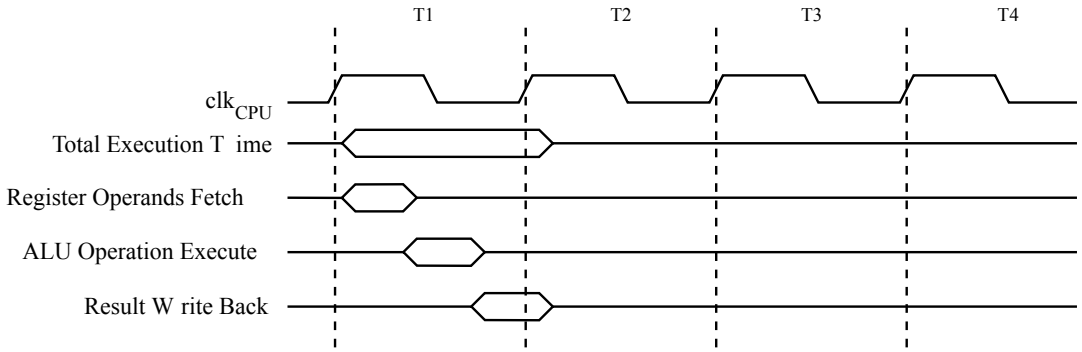
Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers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

Figure 12-4. The Parallel Instruction Fetches and Instruction Executions



The following Figure 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 12-5. Single Cycle ALU Operation



12.7. Reset and Interrupt Handling

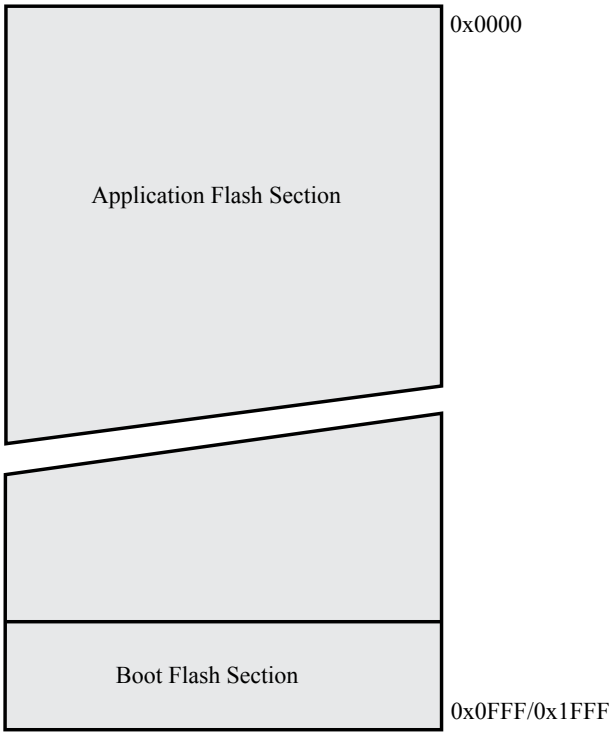
The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. 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 on MEMPROG- Memory Programming 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. 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 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see BTLDR - Boot Loader Support – Read-While-Write Self-Programming.

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

Figure 13-2. Program Memory Map ATmega88PB and ATmega168PB



Related Links

Memory Programming on page 374

Instruction Execution Timing on page 30

Boot Loader Support – Read-While-Write Self-Programming on page 356

Self-Programming the Flash on page 362

13.3. SRAM Data Memory

The following figure shows how the device SRAM Memory is organized.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 768/1280/1280 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover:

1. Direct
   - The direct addressing reaches the entire data space.
2. Indirect with Displacement
   - The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.
3. Indirect

- Brown-out Reset
- 2-wire Serial Interface address match
- Timer/Counter2 interrupt
- SPM/EEPROM ready interrupt
- External level interrupt on INT0 or INT1
- Pin change interrupt

Note: 1. Timer/Counter2 will only keep running in asynchronous mode.

Related Links

8-bit Timer/Counter2 with PWM and Asynchronous Operation on page 205

15.5. Power-Down Mode

When the SM[2:0] bits are written to '010', the SLEEP instruction makes the MCU enter Power-Down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the 2-wire Serial Interface address watch, and the Watchdog continue operating (if enabled).

Only one of these events can wake up the MCU:

- External Reset
- Watchdog System Reset
- Watchdog Interrupt
- Brown-out Reset
- 2-wire Serial Interface address match
- External level interrupt on INT0 or INT1
- Pin change interrupt

This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note: 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.

When waking up from Power-Down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period.

Related Links

Clock Sources on page 49
External Interrupts on page 95
System Clock and Clock Options on page 48

15.6. Power-save Mode

When the SM[2:0] 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.

Table 19-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2) PCINT7 (Pin Change Interrupt 7)
PB6	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1) PCINT6 (Pin Change Interrupt 6)
PB5	SCK0 (SPI0 Bus Master clock Input) XCK0 (USART0 External Clock Input/Output) PCINT5 (Pin Change Interrupt 5)
PB4	MISO0 (SPI0 Bus Master Input/Slave Output) RXD1 (USART1 Receive Pin) PCINT4 (Pin Change Interrupt 4)
PB3	MOSI0 (SPI Bus Master Output/Slave Input) TXD1 (USART1 Transmit Pin) OC2A (Timer/Counter2 Output Compare Match A Output) PCINT3 (Pin Change Interrupt 3)
PB2	SS0 (SPI0 Bus Master Slave select) OC1B (Timer/Counter1 Output Compare Match B Output) PCINT2 (Pin Change Interrupt 2)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output) PCINT1 (Pin Change Interrupt 1)
PB0	ICP1 (Timer/Counter1 Input Capture Input) CLKO (Divided System Clock Output) PCINT0 (Pin Change Interrupt 0)

Signal Name	PD3/OC2B/INT1/ PCINT19	PD2/INT0/ PCINT18	PD1/TXD0/ PCINT17	PD0/RXD0/ PCINT16
DI	PCINT19 INPUT / INT1 INPUT	PCINT18 INPUT / INT0 INPUT	PCINT17 INPUT	PCINT16 INPUT / RXD0
AIO	–	–	–	–

19.3.4. Alternate Functions of Port E

The Port E pins with alternate functions are shown in this table:

Table 19-12. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE3	ADC7 (ADC Input Channel 7) T3 (Timer/Counter 3 External Counter Input) PCINT27
PE2	ADC6 (ADC Input Channel 6) ICP3 (Timer/Counter3 Input Capture Input) SS1 (SPI1 Bus Master Slave select) PCINT26
PE1	T4 (Timer/Counter 4 External Counter Input) SCL1 (2-wire Serial1 Bus Clock Line) PCINT25
PE0	ACO (AC Output Channel 0) ICP4 (Timer/Counter4 Input Capture Input) SDA1 (2-wire Serial1 Bus Data Input/Output Line) PCINT24

The alternate pin configuration is as follows:

- ADC7/T3/MOSI1/PCINT27– Port E, Bit 3
  - PE3 can also be used as ADC input Channel 7.
  - T3: Timer/Counter3 counter source.
  - MOSI1: SPI1 Master Data output, Slave Data input for SPI1 channel. When the SPI1 is enabled as a Slave, this pin is configured as an input regardless of the setting of DDE3. When the SPI1 is enabled as a Master, the data direction of this pin is controlled by DDE3. When the pin is forced by the SPI1 to be an input, the pull-up can still be controlled by the PORTE3 bit.
  - PCINT27: Pin Change Interrupt source 27. The PE3 pin can serve as an external interrupt source.
- ADC6/ICP3/SS1/PCINT26 – Port E, Bit 2
  - PE2 can also be used as ADC input Channel 6.
  - ICP3: Input Capture Pin. The PE2 pin can act as an Input Capture Pin for Timer/Counter3.

19.4.10. Port D Input Pins Address

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: PIND
Offset: 0x29
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x09

Bit	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – PINDn: Port D Input Pins Address [n = 7:0]
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is down counting.

The setup of the OCnx should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCnx value is to use the Force Output Compare (FOCnx) strobe bits in Normal mode. The OCnx Registers keep their values even when changing between Waveform Generation modes.

Be aware that the COMnx[1:0] bits are not double buffered together with the compare value. Changing the COMnx[1:0] bits will take effect immediately.

20.6. Compare Match Output Unit

The Compare Output mode bits in the Timer/Counter Control Register A (TCCR1A.COM1x) have two functions:

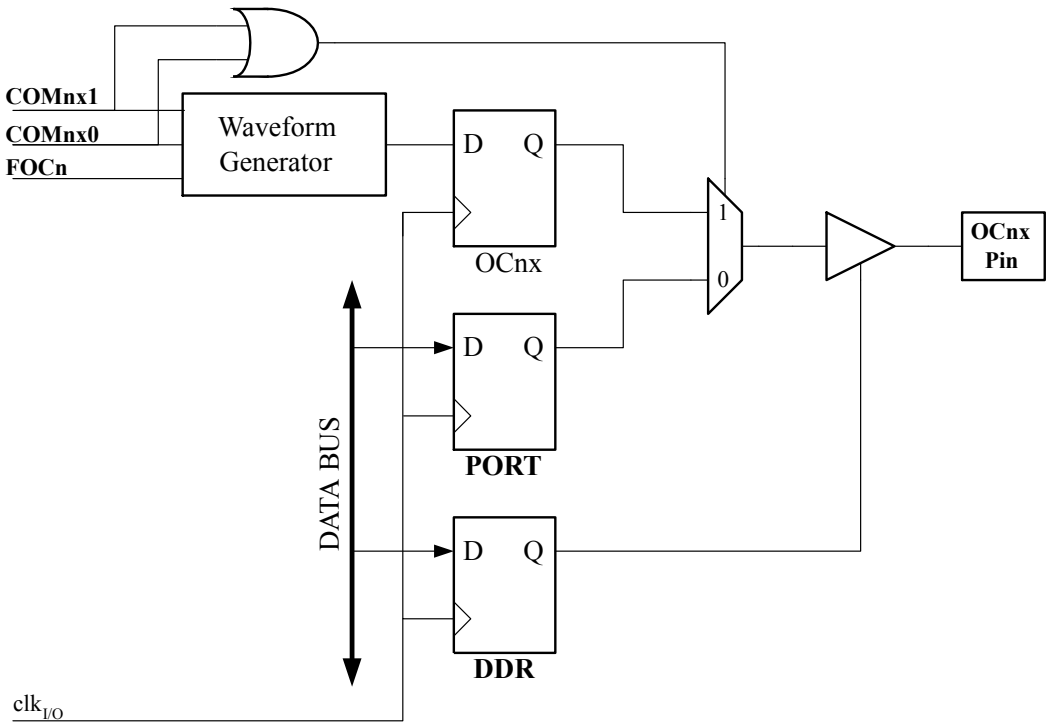
- The Waveform Generator uses the COM1x bits for defining the Output Compare (OC1x) register state at the next compare match.
- The COM1x bits control the OC1x pin output source

The figure below shows a simplified schematic of the logic affected by COM1x. 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 that are affected by the COM1x bits are shown, namely PORT and DDR.

On system reset the OC1x Register is reset to 0x00.

Note: 'OC1x state' is always referring to internal OC1x *registers*, not the OC1x *pin*.

Figure 20-4. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled

by the Data Direction Register (DDR) for the port pin. In the Data Direction Register, the bit for the OC1x pin (DDR.OC1x) must be set as output before the OC1x 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 OC1x register state before the output is enabled. Some TCCR1A.COM1x[1:0] bit settings are reserved for certain modes of operation.

The TCCR1A.COM1x[1:0] bits have no effect on the Input Capture unit.

Related Links

Modes of Operation on page 145

20.6.1. Compare Output Mode and Waveform Generation

The Waveform Generator uses the TCCR0A.COM0x[1:0] bits differently in Normal, CTC, and PWM modes. For all modes, setting the TCCR0A.COM0x[1:0]=0x0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCR0A.COM0x[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 TCCR0C.FOC0x strobe bits.

20.7. Modes of Operation

The mode of operation determines the behavior of the Timer/Counter and the Output Compare pins. It is defined by the combination of the Waveform Generation mode bits and Compare Output mode bits in the Timer/Counter control Registers A and B (TCCRnB.WGMn2, TCCRnA.WGMn1, TCCRnA.WGMn0, and TCCRnA.COMnx[1:0]). The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COMnx[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COMnx[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (See previous section *Compare Match Output Unit*).

For detailed timing information refer to the following section *Timer/Counter Timing Diagrams*.

20.7.1. Normal Mode

The simplest mode of operation is the Normal mode (WGMn[2:0] = 0x0). 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 mode operation, the Timer/Counter Overflow Flag (TOVn) will be set in the same clock cycle in which the TCNTn becomes zero. In this case, the TOVn Flag behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOVn Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal 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.

20.7.2. Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGMn[2:0]=0x2), the OCRnA Register is used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNTn) matches the OCRnA. The OCRnA defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the counting of external events.

20.9.4. General Timer/Counter Control Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GTCCR
Offset: 0x43
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x23

Bit	7	6	5	4	3	2	1	0
	TSM						PSRASY	PSRSYNC
Access	R/W						R/W	R/W
Reset	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 configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

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

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

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 TEMP register in the same clock cycle as the low byte is read, and must be read subsequently.

Note: To perform a 16-bit write operation, 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.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

16-bit Access

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using C, the compiler handles the 16-bit access.

Assembly Code Example(1)

```
    ...
; Set TCNT1 to 0x01FF
ldi     r17,0x01
ldi     r16,0xFF
out     TCNT1H,r17
out     TCNT1L,r16
; Read TCNT1 into r17:r16
in      r16,TCNT1L
in      r17,TCNT1H
    ...
```

The assembly code example returns the TCNT**1** value in the r17:r16 register pair.

C Code Example(1)

```
unsigned int i;
    ...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
    ...
```

Note:

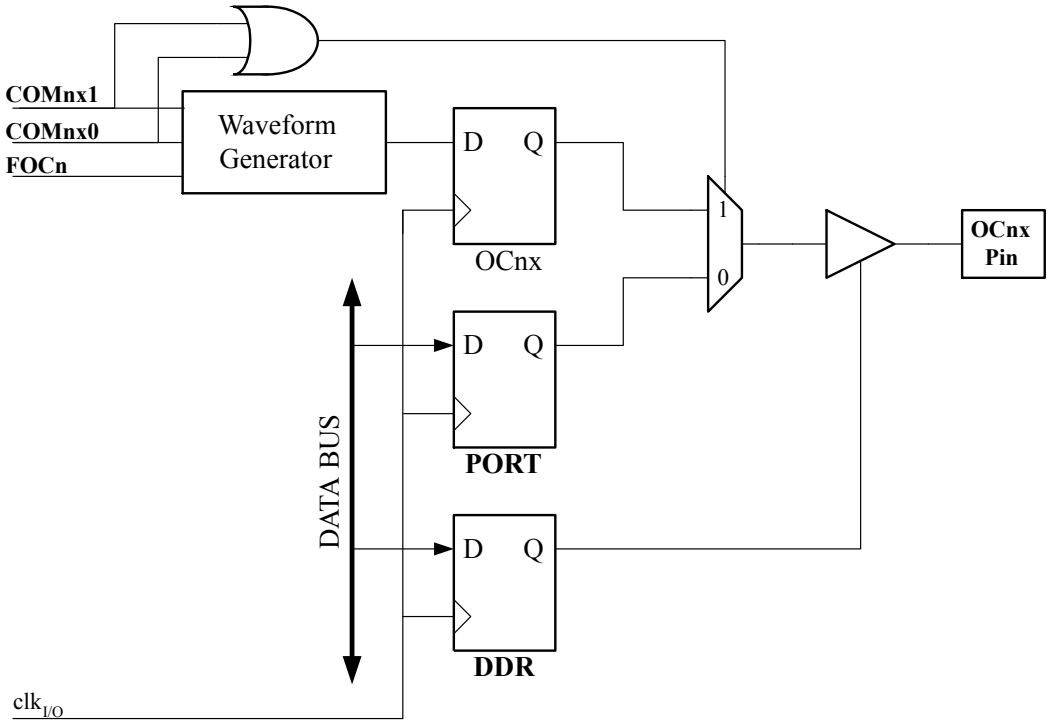
1. The example code assumes that the part specific header file is included. For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

Atomic Read

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to perform an atomic read of the TCNT1 Register contents. The OCR1A/B or ICR1 Registers can be ready by using the same principle.

Figure 21-5. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. In the Data Direction Register, the bit for the OC1x pin (DDR.OC1x) must be set as output before the OC1x 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 OC1x register state before the output is enabled. Some TCCR1A.COM1x[1:0] bit settings are reserved for certain modes of operation.

The TCCR1A.COM1x[1:0] bits have no effect on the Input Capture unit.

Related Links

Modes of Operation on page 145

21.9.1. Compare Output Mode and Waveform Generation

The Waveform Generator uses the TCCR1A.COM1x[1:0] bits differently in Normal, CTC, and PWM modes. For all modes, setting the TCCR1A.COM1x[1:0]=0x0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCR1A.COM1x[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 TCCR1C.FOC1x strobe bits.

21.10. 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 (WGM1[3:0]) and Compare Output mode

Table 23-1. Definitions

Constant	Description
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.

23.2.2. Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (shorten as 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, or asynchronously clocked from the TOSC1/2 pins, 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 he 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 Output Compare Unit 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.

23.3. 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/synchronous to the MCU clock, $clk_{I/O}$.

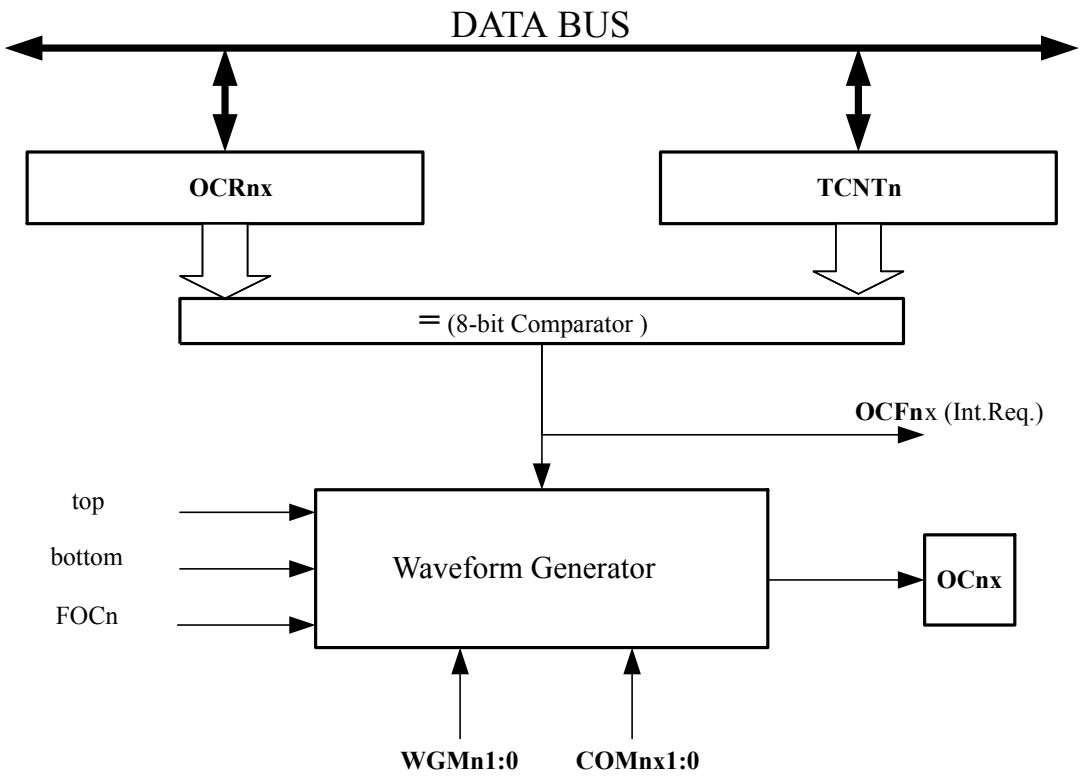
When the Asynchronous TC2 bit in the Asynchronous Status Register (ASSR.AS2) is written to '1', the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2.

For details on asynchronous operation, see the description of the ASSR. For details on clock sources and prescaler, see Timer/Counter Prescaler.

23.4. Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Below is the block diagram of the counter and its surroundings.

Figure 23-3. 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 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.

23.5.1. Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a '1' to the Force Output Compare (FOCnx) bit. Forcing compare match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real compare match had occurred (the COMnx[1:0] bits define whether the OCnx pin is set, cleared or toggled).

23.5.2. Compare Match Blocking by TCNTn Write

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

23.5.3. Using the Output Compare Unit

Since writing TCNTn in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNTn when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is down counting.

4. 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 TC2 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 interrupts 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.
5. If TC2 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: If re-entering sleep mode within the TOSC1 cycle, the interrupt will immediately occur and the device wake up again. The result is multiple interrupts and wake-ups within one TOSC1 cycle from the first interrupt. If the user is in doubt whether the time before re-entering Power-save or ADC Noise Reduction mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
    5.1. Write a value to TCCR2x, TCNT2, or OCR2x.
    5.2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
    5.3. Enter Power-save or ADC Noise Reduction mode.
6. When the asynchronous operation is selected, the 32.768kHz oscillator for TC2 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 TC2 after power-up or wake-up from Power-down or Standby mode. The contents of all TC2 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.
7. 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.
8. 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 Power-save 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:
    8.1. Wait for the corresponding Update Busy Flag to be cleared.
    8.2. 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.

26. USARTSPI - USART in SPI Mode

26.1. 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 ($f_{XCKmax} = f_{CK}/2$)
- Flexible Interrupt Generation

26.2. Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSELn[1:0] 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.

26.3. 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 table below contains the equations for calculating the baud rate or UBRRn setting for Synchronous Master Mode.

Table 26-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Synchronous Master mode	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} + -1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

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.

26.5.1. 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.

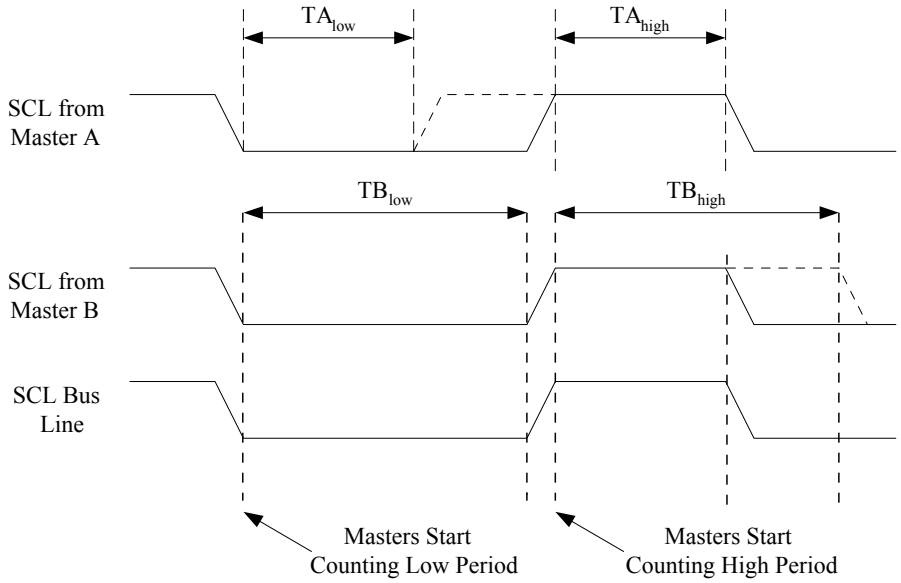
Assembly Code Example

```
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

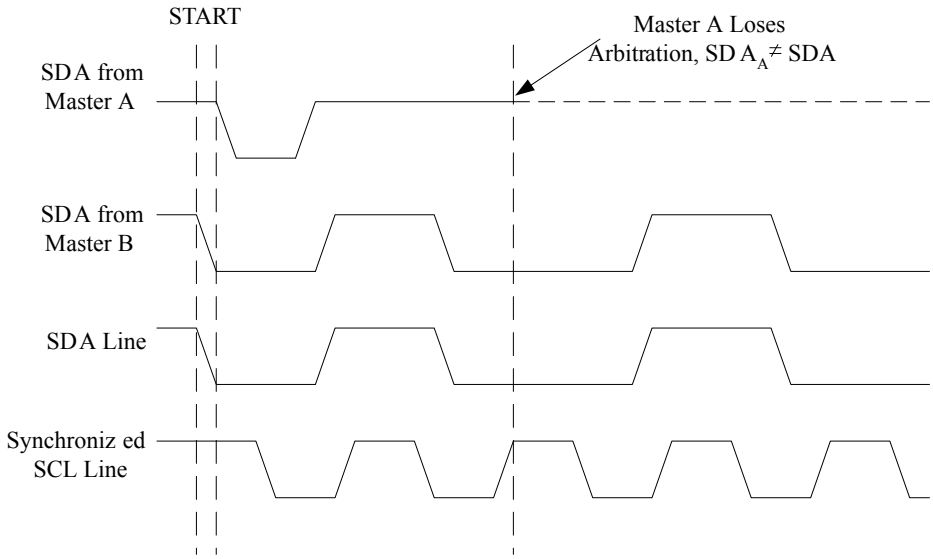
```
{
 UBRRn = 0;
 /* Setting the XCKn port pin as output, enables master mode. */
 XCKn_DDR |= (1<<XCKn);
 /* Set MSPI mode of operation and SPI data mode 0. */
 UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
 /* Enable receiver and transmitter. */
 UCSRnB = (1<<RXENn)|(1<<TXENn);
 /* Set baud rate. */
```

Figure 27-7. 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.

Figure 27-8. Arbitration Between Two Masters



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

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application SofTWARne Response					Next Action Taken by TWI Hardware
		To/from TWDRn	To TWCRn				
			STA	STO	TWINT	TWEA	
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte	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
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

start executing the application code. 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 32-4. Boot Reset Fuse

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset, as described by the Boot Loader Parameters

Note: '1' means unprogrammed, '0' means programmed.

32.7. Addressing the Flash During Self-Programming

The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages, 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 the following figure. The Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.