AVR080: ATmega103 Replaced by ATmega128

Features
- ATmega103 Errata Corrected in ATmega128
- Improvements to Timers and Prescalers
- Oscillators and Selecting Start-up Delays
- Improvements to External Memory Interface
- Improvements to the ADC
- Improvements to SPI and UART
- Changes in Programming Interface
- Added JTAG Interface and On-chip Debug System
- Features not Available in ATmega103 Compatibility Mode

Introduction
The ATmega128 has two operating modes accessed as fuse options. This application note is a guide, which will help current ATmega103 users convert existing designs to the ATmega128 and gain access to new features contained in that device. The ATmega103 compatibility mode is selected with the M103C fuse setting. In this mode, the ATmega128 functions as an ATmega103. When leaving the M103C fuse unprogrammed, all new features described below are supported. Gaining access to these features may require some changes to the existing software. Additionally, the electrical characteristics of the ATmega128 are different including an increase in operating frequency because of a change in process technology. Check the datasheet for detailed information.

ATmega103 Erratas Corrected in ATmega128
The following items from the Errata Sheet are corrected:
Refer to ATmega103 Errata Sheet for a more detailed description of the Erratas.

Power Consumption During Slowly Rising Supply
ATmega128 power consumption is independent of power rising time.

UART Looses Synchronization if RXD Line is Low when UART Receive is Disabled
The UART is replaced with a USART, which does not have this problem. The starting edge of a reception is only accepted as valid if the Receive Enable bit in the USART control register is set.

Releasing Reset Condition Without Clock
ATmega128 has a new reset interface where an external reset pulse causes an internal reset even though the condition disappears before any valid clock is present.
<table>
<thead>
<tr>
<th>Issue Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake-up from Power-save</td>
<td>ATmega128 executes the Interrupt routine as the first instruction after wake-up from Power-save mode.</td>
</tr>
<tr>
<td>Executes Instructions Before Interrupt</td>
<td>If an enabled interrupt occurs while the ATmega128 is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles, executes the interrupt routine, and resumes execution from the instruction following SLEEP.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI Can Send Wrong Byte</td>
<td>In ATmega128, there is no need to wait for the previous transfer to complete before writing the next byte into the SPI Data Register when operating in Master mode.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Incorrect Clearing of XTRF in MCUSR</td>
<td>The POR and XTRF flag can be cleared individually in ATmega128.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset During EEPROM Write</td>
<td>If a Reset or Power-off occurs during an EEPROM write, the current location may be incorrect, but ATmega128 will not corrupt any other locations than the one being written.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI Interrupt Flag can be Undefined after Reset</td>
<td>ATmega128 resets the SPI interrupt flag to zero (0).</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Skip Instruction with Interrupts</td>
<td>ATmega128 interrupts always store the correct return address, also when interrupting a skip instruction skipping a two-word instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Back Value During EEPROM Polling</td>
<td>In ATmega128, the Read Back Value during EEPROM polling is always $FF.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MISO Output During In-System Programming</td>
<td>ATmega128 tri-states the MISO Output during In-System Programming. The In-System Programming interface is still using PE0 and PE1 for serial data in and serial data out, respectively.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The ADC has No Free-running Mode</td>
<td>The ATmega128 supports Free-running mode.</td>
</tr>
</tbody>
</table>
**Improvements to Timer/Counters and Prescalers**

For details about the improved and additional features, please refer to the datasheet. The following features have been added:

- The Prescalers in ATmega128 can be reset.
- Variable top value in PWM mode.
- Phase and Frequency correct PWM mode in addition to the Phase correct PWM mode.
- Overflow PWM mode.

**Differences Between ATmega128 and ATmega103**

Most of the improvements and changes apply to all the Timer/Counters and the description below is written in a general form. A lower case “x” replaces the output channel (A or B for Timer/Counter1, N/A for Timer/Counter0 and Timer/Counter2), while “n” replaces the Timer/Counter number (n = 0, 1, or 2).

**TCNT1 Cleared in PWM Mode**

In ATmega103 there are 3 different PWM resolutions - 8, 9, or 10 bits. Though only 8, 9, or 10 bits are compared, it is still possible to write values into the TCNT1 register that exceed the resolution. Thus, the Timer/Counter has to complete the count to 0xffff before the reduced resolution becomes effective (i.e, if 8-bit resolution is selected and the TCNT1 register contains 0x0100, the top value (0x00ff) will not be effective until the counter has counted up to 0xffff, turned, and counted down to 0x0000 again). In ATmega128 this has been changed so that the unused bits in TCNT1 are being cleared to zero to avoid this unintended counting up to 0xFFFF. In the ATmega128, the TCNT1 register never exceeds the selected resolution.

**ATmega128:**

The most significant bits in the TCNT1 register will be cleared at the first positive edge of the prescaled clock.

- 8-bit PWM: TCNT1H[7:0] = 0
- 9-bit PWM: TCNT1H[7:1] = 0
- 10-bit PWM: TCNT1H[7:2] = 0

**ATmega103:**

TCNT1H not cleared.

**OCR1xH Cleared in PWM Mode**

Clearing OCR1xH in PWM mode is slightly different from clearing TCNT1. The ATmega103 clears the most significant bits if 8, 9, or 10 bits PWM mode is selected, but only the 6 most significant bits. Thus, if 0xffff is written to OCR1x in PWM-mode and read OCR1x back, 0x03ff is read regardless of which PWM mode that is selected. In ATmega128 the number of cleared bits depends on the resolution.

**ATmega128:**

The most significant bits in OCR1AH and OCR1BH are cleared when they are updated at the TOP-value of the counter.

- 8-bit PWM: OCR1xH[7:0] = 0
- 9-bit PWM: OCR1xH[7:1] = 0
- 10-bit PWM: OCR1xH[7:2] = 0

**ATmega103:**

The six most significant bits in the OCR1AH and OCR1BH are cleared regardless of the resolution.
Clear Timer/Counter on Compare Match with Prescaler (Applies to all Timer/Counters)

The relation between a Clear on Compare match and the internal counting of the Timer/Counter has been changed. The Clear on Compare match in the ATmega103 clears the Timer/Counter after the first internal count matching the compare value, whereas the ATmega128 clears after the last internal count matching the compare value. See Figure 1 and Figure 2 for details on clearing, flag setting, and pin change.

Example: OCRnx = 0x02 when prescaler is enabled (divide clock by 8).

Figure 1. Setting Compare Flag/Pin for ATmega128(1)

<table>
<thead>
<tr>
<th>TCNTn</th>
<th>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 1 1 1 1 1 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin/Flag</td>
<td>h</td>
</tr>
</tbody>
</table>

Note: 1. “h” Indicates where the Compare flag/pin will be set.

Figure 2. Setting Compare Flag/Pin for ATmega103(1)

<table>
<thead>
<tr>
<th>TCNTn</th>
<th>0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 0 0 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin/Flag</td>
<td>h</td>
</tr>
</tbody>
</table>

Note: 1. “h” Indicates where the Compare flag/pin will be set.

Setting of Output Compare Pin/Flag with Prescaler Enabled (Applies to all Timer/Counters)

The relation between an Output Compare and the internal counting of the Timer/Counter has been changed. Output Compare in the ATmega103 sets the Output Compare pin/flag after the first internal count matching the compare value, whereas the ATmega128 sets the Output Compare pin/flag after the last internal count matching the compare value. See Figure 3 and Figure 4 for details on Output Compare flag setting and pin change.

Example: OCRnx = 0x02, prescaler enabled (divide clock by 8)

Figure 3. Setting Compare Flag/Pin for ATmega128(1)

<table>
<thead>
<tr>
<th>TCNTn</th>
<th>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin/Flag</td>
<td>h</td>
</tr>
</tbody>
</table>

Note: 1. “h” Indicates where the Compare flag/pin will be set.

Figure 4. Setting Compare Flag/Pin for ATmega103(1)

<table>
<thead>
<tr>
<th>TCNTn</th>
<th>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin/Flag</td>
<td>h</td>
</tr>
</tbody>
</table>

Note: 1. “h” Indicates where the Compare flag/pin will be set.
Write to OCR1x in PWM Mode, Change to Normal Mode Before OCR1x is Updated at the Top, Read OCR1x (Applies to 16-Bit Timer/Counter Only)

As described in the datasheet, the OCR1x registers are updated at the top value when written. Thus, when writing the OCR1x in PWM mode, the value is buffered in a temporary register which is latched until the Timer/Counter reaches the top. If PWM mode is left after the temporary register is written, but before the real Output Compare registers are updated, the behavior differs between ATmega128 and ATmega103.

**ATmega128:**

If the OCR1x register is read before the update is done, the actual compare value is read – not the OCR1x buffer.

**ATmega103:**

If the OCR1x register is read before the update is done, the value in the OCR1x buffer is read. For example, the value read is the one last written (to the OCR1x buffer), but since the Timer/Counter never reached the top value, it was not latched into the OCR1x register. Thus the value that is used for comparison is not necessarily the same as being read.

Note: This applies to 16 bits Timer/Counter only, for 8 bits Timer/Counter, the temporary register value is read for both devices.

Remember old OCx-pin Level

If COM1x1 and COM1x0 changes from "01" to "00" and a compare match occurs when the COM1x1:COM1x0 value is "00":

**ATmega128:**

The level of the OCx-pin before disabling the Output Compare mode is remembered. Re-enabling the Output Compare mode will cause the OCx-pin to resume operation from the state it had when it was disabled. All Output Compare pins are initialized to zero on Reset.

**ATmega103:**

OCx is cleared for 16 bits Timer/Counter. For 8 bits Timer/Counter, the state of the Output Compare pin is unknown when re-enabling the Output Compare.

Only the 8-bit Timer/Counters in the ATmega103 are initialized to zero on Reset.

OCR0 or OCR2 Equal Extreme Value in TCNT0 or TCNT2 Respectively (Applies to 8-bit Timer/Counter Only)

According to the values in COMn1 and COMn0, OCn will be cleared or set when changing the COMn bits. The response on the output differs from ATmega128 to ATmega103:

**ATmega128:**

When changing COMn bits setting, the output pin (OCn) changes accordingly to the COMn bits after a compare match has occurred.

**ATmega103:**

When changing COMn bits setting, the output pin (OCn) changes immediately.

For the 16 bits Timer/Counter1, neither ATmega128 nor ATmega103 change the output before a compare match occurs.

Oscillators and Selecting Start-up Delays

ATmega128 provides more oscillators and start-up time selections than ATmega103. In , the start-up delays from Power-down mode and Power-save mode depend on the CPU clock frequency. During Wake-up from Power-down mode and Power-save mode, the ATmega128 uses the CPU frequency to determine the Wake-up delay, while ATmega103 determines the delay from the WDT oscillator frequency (except SUT = 00).

Follow the guidelines from the section “System Clock and Clock Options” in the ATmega128 datasheet to find appropriate start-up values.
Special attention must be paid when changing the fuses in In-System Programming mode. In-System Programming is dependent on a system clock. If wrong oscillator setting is programmed, it may be impossible to re-enter In-System Programming mode due to missing system clock (parallel programming mode must then be used).

Improvements to External Memory Interface

The combined Address/Data port in ATmega128 outputs Data until a new address is set up. Refer to the ATmega128 Datasheet for details on the changed timing.

Improvements to ADC

- The ADC in ATmega128 supports Free-running mode.
- To improve accuracy, a single conversion now takes one additional cycle.
- ATmega128 supports both left adjusted and right adjusted 10-bit results.
- The ADC in ATmega128 supports differential and amplified measurements.

Improvements to SPI and USART

Both SPI and USART have new double-speed modes which allow higher communication speed.

The UART in ATmega103 has been replaced by a USART in ATmega128. The ATmega128 USART is compatible with the ATmega103 UART with one exception. The two-buffer receive register acts like a FIFO and the following must be kept in mind:

- The UDR must only be read once for each incoming data, if reading more than once, the next level of FIFO will be read.
- The error flags (FE and DOR) and the 9th data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR register is read. Otherwise, the error status will be lost.

Changes in EEPROM Write Timing

In ATmega103, the EEPROM write time is dependent on voltage, typically 2.5 ms @ $V_{CC} = 5\text{V}$ and 4 ms @ $V_{CC} = 2.7\text{V}$. In ATmega128, the EEPROM write time is 8.2 ms regardless of $V_{CC}$.

Programming Interface

Some changes have been done to the programming interface, especially in the In-System Programming interface. This has been done to support all the additional fuses in ATmega128. The timing requirements are unchanged. See the ATmega128 datasheet for details.

The parallel programming algorithm is changed. The most significant change is that the PAGEL pin on ATmega128 is located on PD7, while BS2 is located on PA0. On ATmega103 the opposite pin-mapping was chosen (PAGEL pin on PA0, while BS2 was mapped to PD7). This change has been done to make it possible to use the same programmer for all new AVR devices. In parallel mode, the ATmega128 supports page programming of the EEPROM. Note that the additional fuses and lock-bits also require a change in the fuse writing algorithm. The timing requirements for parallel programming have been changed. See the ATmega128 datasheet for details.

The STK500 supports both In-System Programming and parallel programming of the ATmega128.
### JTAG Interface and On-chip Debug System

The ATmega128 provides a JTAG interface, which can be used for programming, boundary-scan, and On-chip debug. Refer to datasheet for details. Note that the JTAG interface is also available in ATmega103 compatibility mode.

### Other Differences

<table>
<thead>
<tr>
<th><strong>Signature Byte</strong></th>
<th>The ATmega128 has a Signature Byte different from the one used in ATmega103.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Verifying the EEPROM at High Voltages During Programming</strong></td>
<td>There are no restrictions on the supply voltage or system frequency as long as operated inside the voltage and frequency range prescribed in the datasheet for the ATmega128.</td>
</tr>
<tr>
<td><strong>Verifying EEPROM In-System</strong></td>
<td>There are no restrictions on the supply voltage or system frequency as long as the device is operated inside the voltage and frequency range prescribed in the datasheet for the ATmega128.</td>
</tr>
<tr>
<td><strong>Serial Programming at Voltages Below 3.4V</strong></td>
<td>There are no restrictions on the supply voltage or system frequency as long as the device is operated inside the voltage and frequency range prescribed in the datasheet for the ATmega128.</td>
</tr>
</tbody>
</table>
The compatibility fuse makes the ATmega128 compatible to ATmega103. But with the compatibility fused programmed, some of the new features in ATmega128 become unavailable. The following features are not supported when the ATmega128 is used in the ATmega103 compatibility mode:

- **TWI - Two Wire Interface module.**
- **USART1 - The additional second USART.**
- **Prescaler Reset on Timer/Counters.**
- **Boot Loader Capabilities (SPM - Self Programming Memories).**
- **Advanced External Memory Control (more wait-states, configurable number of bits are assigned to address high byte, different wait-states settings for different pages of external memory).**
- **Additional Output Compare register (OCR1C) for Timer/Counter1.**
- **Timer/Counter3 (16-bit Timer/Counter identical to Timer/Counter1).**
- **PortC is general I/O (digital output only in ATmega103).**
- **PortF is general I/O (analog/digital input only in ATmega103).**
- **PortG (alternate functions only in ATmega103).**

If any of the features above are needed or wanted and the compatibility fuse is left unprogrammed, this introduces several differences between ATmega128 and ATmega103 which do not exist as long as the compatibility fuse is programmed:

- **Address space 0x0060-0x00FF is dedicated extended I/O, not internal SRAM.**
- **Address space 0x0100-0x10FF is dedicated internal SRAM (ATmega128 supports 4096 locations of internal SRAM compared to 4000 in ATmega103), thus the external memory starts at address 0x1100 (external memory on ATmega103 starts at address 0x1000).**
- **PortC is not initialized upon reset to drive 0x00, but it is tri-stated as all other ports.**
- **The ALE, RD, and WR pins (PG2:0) are not configured as output until the XRAM is enabled.**
- **The TOSC1 and TOSC2 (PG4:3) pins are configured as digital input pins after reset, not 32 kHz oscillator unless AS0 bit is written.**
- **A timed sequence must be followed to change Watchdog Timer prescaler settings by software.**
- **In the MCUCSR register, all RESET flags are present in the register, not only EXTRF and PORF as in ATmega103.**
- **The UART will have an extra input buffer which allows one more data byte to be received before the data overrun flag (DOR) is set.**