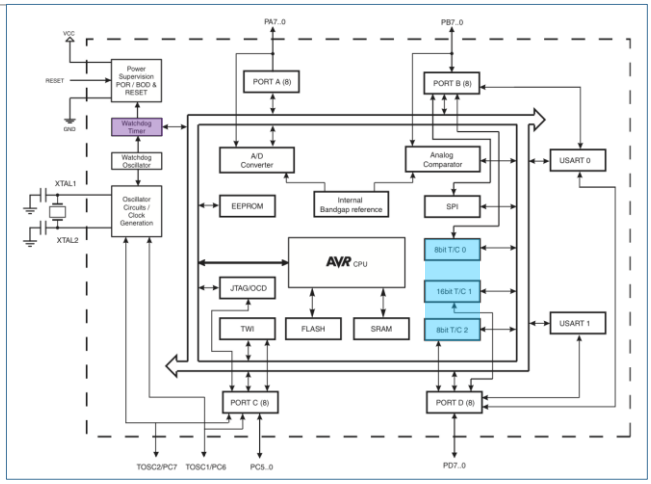
Sensors & Microsystem Electronics: microcontrollers

PART 2: MEMORY, TIMERS & INTERRUPTS

Timers

Timers

- 1 watchdog timer
- 3 timer/counters
  - Timer/counter 0 (8 bit)
  - Timer/counter 1 (16 bit)
  - Timer/counter 2 (8 bit)



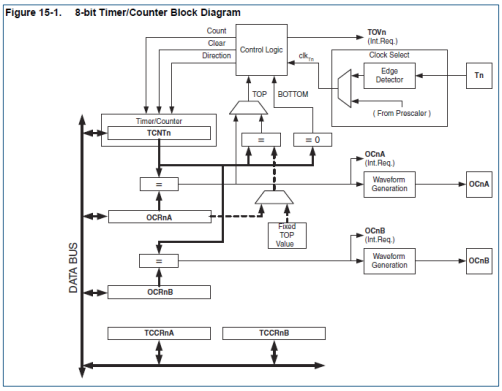
Watchdog timer

- Detects program malfunction and starts corrective actions
- Time-critical applications
- Applications where humans cannot access the device
- Principle of operation:
  - The program regularly resets the watchdog timer to prevent it from elapsing
  - Due to malfunction, the watchdog timer is not reset in time
  - A timeout signal is issued
  - The microcontroller resets (corrective action)

Timers 0, 1 and 2

- Timer 0,2: 8-bit
- Timer 1: 16-bit

- Modes:
  - Normal
  - Clear Timer on Compare (CTC)
  - PWM
  - (others)

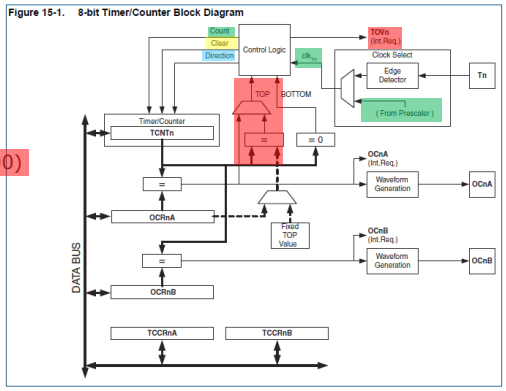


Timer 0 and 2 are 8 bit timers. The TCNT register only has 8 bits resulting in a maximum count of 255 before rolling over to 0.

Timer 1 is a 16 bit timers, meaning that the counting register is 16 bit (read: 2 registers of 8 bit) .

Timers 0, 1 and 2

- Normal mode
- clock @ output of prescaler
- count up by 1 every clock cycle
- TCNT (Timer/CouNTer) resets to 0x00(00)
- TOV (Timer OVerflow) flag when 0xFF(FF) → 0x00(00)

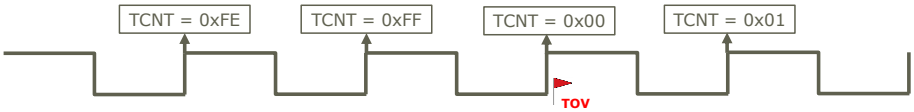


Timer 0 and 2 are 8 bit timers. The TCNT register only has 8 bits resulting in a maximum count of 255 before rolling over to 0.

Timer 1 is a 16-bit timers, meaning that the counting register is 16 bit (read: 2 registers of 8 bit) .

Timers 0, 1 and 2

- Normal mode
- clock @ output of prescaler
- count up by 1 every clock cycle
- TCNT (Timer/CouNTer) resets to 0x00(00)
- TOV (Timer OVerflow) flag when 0xFF(FF) → 0x00(00)



Configuring timer 0

- Normal mode (default)
- WGM02:0 = 0b000
- CS02:0 → Clock and prescaler select

15.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

15.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	0	0	

TCCR0A & B are not bitadressable!
Write full byte at once for configuration.

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
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	BOTTOM	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

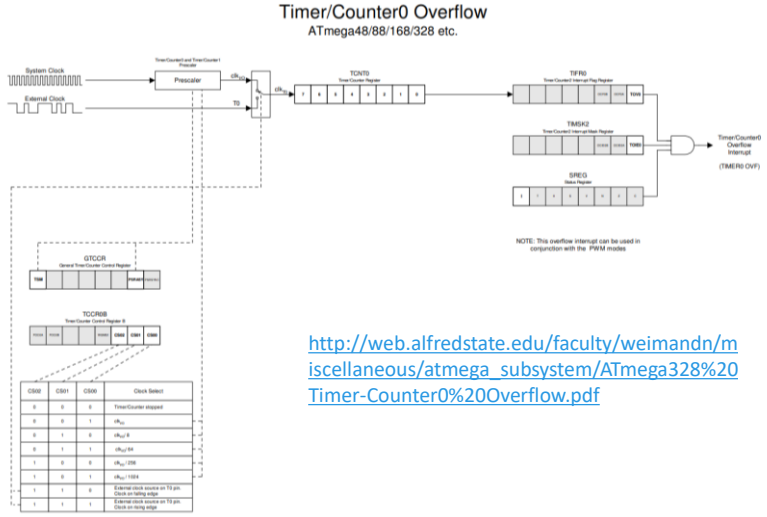
Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Table 15-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

The timer needs to be configured using the TCCRxA and TCCRxB registers (x denotes the number of the timer) by setting and clearing the correct bits according to the settings described in the datasheet.

Configuring timer 0

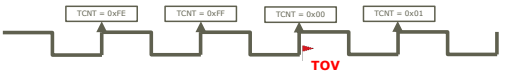


http://web.alfredstate.edu/faculty/weimandn/miscellaneous/atmega_subsystem/ATmega328%20Timer-Counter0%20Overflow.pdf

This PDF contains a schematic overview on how all registers and configuration bits relate to each other.

How to use a timer? Polling the overflow flag

- Start timer at $TCNT_{Initial\ value}$ → Wait for overflow: check TOV0 flag in a loop
- Period of timer = (Max value – $TCNT_{Initial\ value}$)*Count-rate
  - Max value: 2^[timer resolution] – 1
  - Count-rate: frequency after prescaler
- $t = \frac{1}{f_{clk}} * [0xFF - TCNT_{init}]$




```
; Start timer

Loop: SBIS TIFR0,TOV0 ; skip next line if TIFR0(0) = 1
      RJMP Loop

; Timer finished
```

15.9.7 TIFR0 – Timer/Counter 0 Interrupt Flag Register

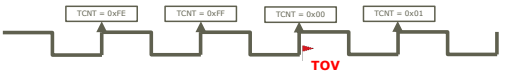
Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

There are multiple ways to operate a timer.
The first method is very inefficient and naïve.

The second method is by polling the overflow flag in your code to see if the timer has elapsed. Yet in some scenarios it can be usefull, for example to introduce a wait in your code.

How to use a timer? Interrupts

- Start timer at $TCNT_{Initial\ value}$ → let it run to its max. value → INTERRUPT
- Period of timer = (Max value – $TCNT_{Initial\ value}$)*Count-rate
  - Max value: 2^[timer resolution] – 1
  - Count-rate: frequency after prescaler
  - $t = \frac{1}{f_{clk}} * [0xFF - TCNT_{init}]$



Most used way!

15.9.7 TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The third method is the most frequently used one: Combining a timer and an interrupt.

You configure the timer to count at a specific rate, and when it overflows it generates an interrupt that will stop the code, handle what needs to be done when the timer is done counting, and then return to the main code.

Different timer interrupt also exist such as a compare and match, this timer operation will count and compare it to a value in a register on each increment. When it matches an interrupt is generated.

Choosing the prescaler

- Example: 1 ms period (1 kHz) for 8 bit timer
- $t_{timer} = \frac{1}{f_{clk}} * [0xFF - TCNT_{init}]$
- $TCNT_{init} = 0xFF - t_{timer} * f_{clk} = 0xFF - \frac{f_{clk}}{f_{timer}}$
- $f_{clk} = prescaled\ clock = \frac{System\ clock_{16Mhz}}{prescaler}$

f_{clk} = 16 MHz (no prescaler)

$$TCNT_{init} = 256 - \frac{16\,000\,000\ Hz}{1000\ Hz} = -15\,746$$

f_{clk} is to high!

f_{clk} = 250 kHz (prescaler 64)

$$TCNT_{init} = 256 - \frac{250\,000\ Hz}{1000\ Hz} = 6$$

Exact frequency!

f_{clk} = 15 625 Hz (prescaler 1024)

$$TCNT_{init} = 256 - \frac{15\,625\ Hz}{1000\ Hz} = 240{,}375 \cong 240$$

Rounding error -> little different frequency

The timer counts to 255 before it overflows. This overflow creates an interrupt if enabled. To calculate the period of the timer interrupt we do the following:

The incrementing frequency (Tinc) is the clock frequency (16MHz) divided by the prescaler.

Then for one period of the timer we need to have a certain number of counts at the incrementing frequency. Because we use the timer in overflow mode, we get the interrupt at count 256. So we need to count from a certain value to 256 in the previously obtained number of counts, so we subtract 256 by this number of counts to get the start value.

Clear Timer on Compare (CTC) mode

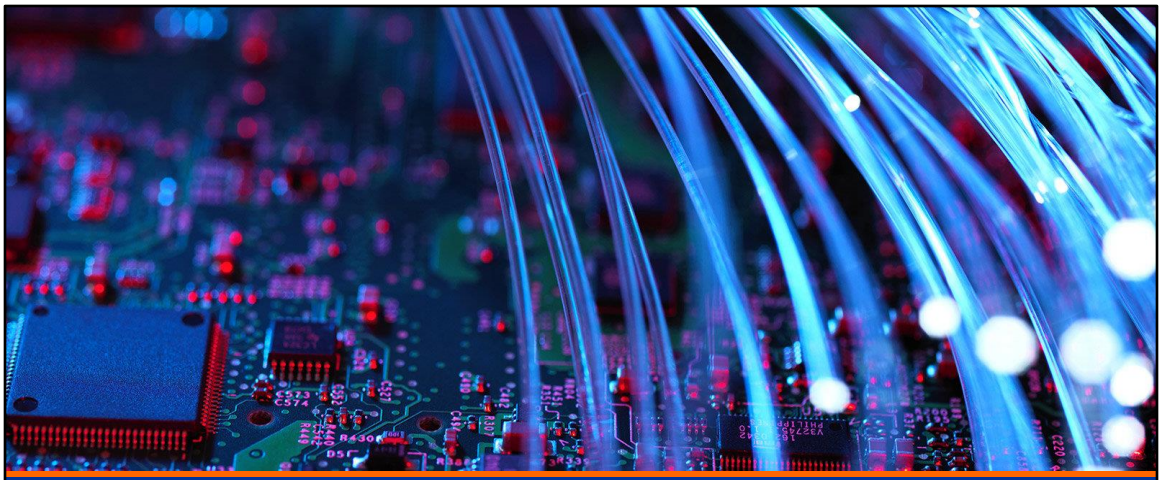
- Timer counts from 0
- Each count TCNT is compared to OCRA
- When TCNT = OCRA → OCF0A flag set (+interrupt)
- Timer cleared on match

- More involved to set up

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
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	BOTTOM	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



End of Part 2: Timers