

Timer/Counter Subsystem

- Timer/Counter0 – 8-bit timer
- Timer/Counter1 – 16-bit timer.
- Timer/Counter2 – 8-bit timer.
- Timer/Counter is useful for:
 - Delay loops.
 - Timed interrupts.
 - Signal with variable on time, a.k.a., pulse width modulation.

This will detail Timer/Counter0, but the others are similar.

Overview

- We have an 8-bit timer/counter register (**TCNT0**).
 - We can write to and read from the register.
 - It can be “clocked” by:
 - Internal clock (that can be prescaled, i.e., slowed by some multiple).
 - External clock on **T0** pin (mapped to **PORTB0**).
- Definitions:
 - The counter reaches *BOTTOM* when value = 0.
 - The counter reaches *MAX* when value = 255.
 - The counter reaches *TOP* when it reaches its highest value:
 - *MAX* or
 - The value stored in the Output Compare Register (**OCR0**).
 - Which value is considered the *TOP* depends on the mode of operation.
- General operation:
 - Each time the clock cycles, **TCNT0** is incremented.
 - When **TCNT0** overflows, the Timer/Counter0 Overflow Flag (**TOV0** in **TIFR** register) is set.
 - If configured appropriately, this can cause an interrupt.
 - When **TCNT0** = **OCR0** the Timer/Counter0 Output Compare Flag (**OCF0** in **TIFR** register) is set.
 - If configured appropriately, this can cause either or both of the following:
 - An interrupt.
 - Pin action on **OC0** pin (connected to **PORTB3**).

Relevant I/O Registers

- **TCNT0** – Timer/Counter0 (8-bits).

Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							

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

- Read and write accessible.
- A clock is associated with the register.
- Each time the clock ticks, **TCNT0** is incremented.
- When **TCNT0** goes from 255 to 0, an overflow flag (**TOV0** flag in **TIFR**) gets set.

- **OCR0** – Output Compare Register (8-bits).

Bit	7	6	5	4	3	2	1	0
	OCR0[7:0]							

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

- Read and write accessible.
- The value in **OCR0** is compared with the value in **TCNT0**.
- When **OCR0** and **TCNT0** are equal, we can trigger other actions, e.g.,
 - **OC0** (connected to Pin 3 on **PORTB**) can be toggled (depends on **COM01:0** settings in **TCCR0**).
 - Output Compare Flag 0 (**OCF0** from **TFIR**) is set.
 - If Timer/Counter0 Overflow Interrupt Enable (**TOIE0** from **TIMSK**) is set, an interrupt occurs when **OCF0** goes high.

- **TIMSK** – Timer mask Register.

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

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

- **OCIE0** – Timer/Counter0 Output Compare Match Interrupt Enable (active high).
- **TOIE0** – Timer/Counter0 Overflow Interrupt Enable (active high).

- **TIFR** – Timer Flag Register.

Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

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

- **OCF0** – Output Compare Flag 0.
 - Set when a compare match between **TCNT0** and **OCR0** occurs.

- If **OCIE0** is set, then an interrupt occurs when **OCF0** is set.
- Can be cleared two ways:
 - Manually by **SBI TIFR, OCF0** (yes *SBI*)
 - Automatically cleared by hardware when executing the corresponding interrupt handling vector.
- **TOV0** – Timer/Counter0 Overflow Flag.
 - Set when **TCNT0** overflows (goes from 0xff to 0x00).
 - If **TOIE0** is set, then an interrupt occurs when **TOV0** is set.
 - Can be cleared two ways:
 - Manually by **SBI TIFR, TOV0** (yes *SBI*)
 - Automatically cleared by hardware when executing the corresponding interrupt handling vector.
- **TCCR0** – Timer/Counter Control Register. (see pp. 25-26 of Atmega32 reference guide)

Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	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

- **FOC0** – Force Output Compare, we'll just leave this cleared.
- **WGM01:0** – Waveform Generation Mode.
 - Lookup table to select the counting sequence of the counter.
 - We will leave these two bits cleared.
- **COM01:0** – Compare Match Output Mode.
 - Lookup table to determine pin action.
 - In order for pin action to occur, **PORTB3** must be configured for output.
 - With **WGMM01:0** cleared, the options are:

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected
0	1	Toggle OC0 on compare match (TCNT0 = OCR0)
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

-
- **CS02:0** – Clock Select.
 - Lookup table to select the clock source and speed.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk
0	1	0	clk/8

0	1	1	clk/64
1	0	0	clk/256
1	0	1	clk/1024
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.

Sample Code

Using Timer/Counter0 to Create a Delay with Polling

- Prescale by 1024, with no pin action: **TCCR0** = 0b00000101
- Clear **TCNT0**
- Clear **TOV0**

```
.def temp = r16
    ldi    temp, 1<<CS02 | 1<<CS00
    out   TCCR0, temp
    ldi    temp, 0x00
    out   TCNT0, temp
    ldi    temp, 1<<TOV0
    out   TIFR, temp
```

- Notes:
 - The first *LDI* instruction could have been written as *LDI temp, 0x05*.
 - The m32def.inc file has definitions for all of the pin labels.

```
; ...
.equ CS00 = 0
.equ CS01 = 1
.equ CS02 = 2
; ...
```

- - - *LDI temp, 1<<CS02 | 1<<CS00* is equivalent to *LDI temp, 1<<2|1<<0*
 - $1 \ll 2 = 0b00000100$ and $1 \ll 0 = 0b00000001$ which, when bit-wise OR'd together, gives us $0b00000101 = 0x05$.
 - Writing using the bit labels makes our code more expressive (a good thing).
 - You may wonder why we didn't use *SBI TIFR, TOV0* instead of the last two instructions above.
 - It turns out that the bit related instructions (like *SBI*) will not work on **TIFR**.
 - **TIFR** is mapped to a memory address that is above the range of addresses for which the bit instructions work.
- The increment frequency for **TCNT0** is: $8 \times 10^6 \text{ cycles/sec} / 1024 = 7812\text{Hz}$

- The increment interval is $1/7812 = 128 \times 10^{-6}$ sec
- The overflow interval is: 128×10^{-6} sec/clock \times 256 clock = 32.768 ms.
- If we wanted a 10ms delay, we would need $10\text{ms}/128 \times 10^{-6} = 78$ clocks. Instead of writing 0x00 into **TCNT0**, we would give it 177 (=255-78).

```

; Function that provides a 10ms delay routine
delay10ms:
.def temp = r16
    push    temp
    ldi     temp, 0xff-78
    out     TCNT0, temp
    ldi     temp, 1<<CS02|1<<CS00
    out     TCCR0, temp
    in      temp, TIFR          ; See alternate approach above
    ori     temp, 1<<TOV0
    out     TIFR, temp

d10Poll:
    in      temp, TIFR          ; Cannot do sbrs on TIFR
    sbrs   temp, TOV0
    rjmp   d10Poll

    pop    temp
    ret

```

- Note: we need to read **TIFR** into a register since *SBS* doesn't work on **TIFR**.

Using Timer/Counter0 to Create a Delay with Interrupts

- Steps:
 - Set interrupt vector.
 - Initialize stack.
 - Configure timer/counter0.
 - Enable local Int mask.
 - Set **I** bit in **SREG**.

```

.include "m32def.inc"

.def temp = r16

.org 0x00
    rjmp   initStack
.org OVf0addr
    rjmp   ovr0ISR
.org 0x2a
initStack:

start:
    rcall  initTimer0
    sei                       ; Enable global interrupts

```

```

; Initialize timer/counter0 to interrupt every 10ms.
initTimer0:
    push    temp

    ldi    temp, 0xff-78      ; make overflow occur in 78 clk ticks.
    out    TCNT0, temp
    ldi    temp, 1<<CS02|1<<CS00
    out    TCCR0, temp
    in     temp, TIFR
    sbr    temp, 1<<TOV0      ; Could have use ori
    out    TIFR, temp
    in     temp, TIMSK
    ori    temp, 1<<TOIE0    ; Enables timer/counter0 interrupt
    out    TIMSK, temp

    pop    temp
    ret

; ISR for timer/counter0 interrupt
; Schedules next interrupt to happen 10ms from now.
ovr0ISR:
    push    temp
    in     temp, SREG
    push    temp

    ldi    temp, 0xff-78      ; make overflow occur in 78 clk ticks.
    out    TCNT0, temp

    pop    temp
    out    SREG, temp
    pop    temp
    reti

```