Sensors & Microsystem Electronics: microcontrollers

PART 2: MEMORY, TIMERS & INTERRUPTS

Combining timers & interrupts

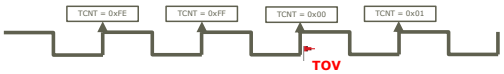
Process flow

- Timers generate an overflow flag when TCNT reaches its max. value
- Setting this flag generates an interrupt request
- If the timer's overflow interrupt is enabled, an interrupt will occur
- Do something in this interrupt

Timers can be used in multiple different ways, we will give the example of the method using the overflow here since this is the simplest to implement. For other uses you can check the datasheet.

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
  - $t = \frac{1}{f_{clk}} * [0xFF - TCNT_{init}]$



See Part 2: Timers

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
- $t_{timer} = \frac{1}{f_{clk}} * [0xFF - TCNT_{init}]$
- $TCNT_{init} = 0xFF - t_{timer} * f_{clk} = 0xFF - \frac{f_{clk}}{f_{timer}}$
- $f_{clk} = prescaled\ clock = \frac{System\ clock_{16Mhz}}{prescaler}$

See Part 2: Timers

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)

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

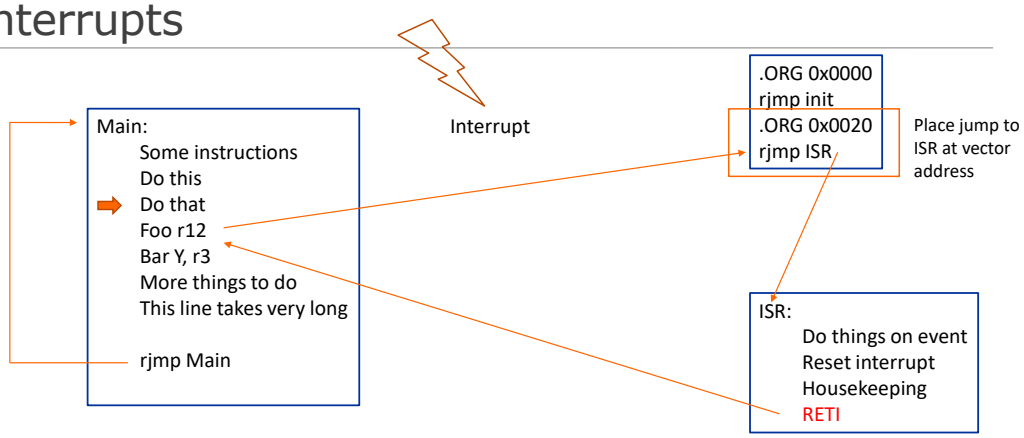
Rounding error -> little different frequency

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.

Interrupts



See Part 2: Interrupts

Interrupts: registers

- Set global interrupt flag → activate interrupt machinery

SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, 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. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

See Part 2: Interrupts

The global interrupt enable flag lives in the SREG (Status register). One can set and clear this flag using normal register operations. But the preferred way of enabling and disabling the Global interrupt flag is using the SEI and CLI instruction respectively.

Interrupts: registers

- Set peripheral interrupt flag → activate specific interrupt

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **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/Counter 0 Interrupt Flag Register – TIFR0.

See Part 2: Interrupts

Each different peripheral can trigger an interrupt for different events, but these also need to be enabled by setting the respective bit in the configuration registers to 1.

Timers & interrupts

- Let timer 0 overflow/interrupt @ 1000 Hz

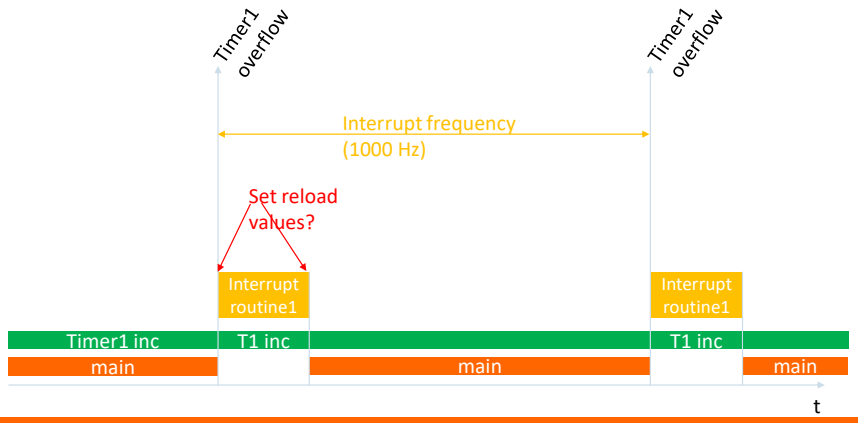
- f_{clk} = 250 kHz (prescaler 64)
- $TCNT_{init} = 256 - \frac{250\,000\ Hz}{1000\ Hz} = 6 = 0x06$

```
.ORG 0x0000
RJMP init
.ORG 0x0020
RJMP Timer0OverflowInterrupt

Init:
   ;set timer0 prescaler to 64
   ;enable global interrupt & timer0 interrupt

Main:
   ;do something
   RJMP main

Timer0OverflowInterrupt:
   ;reset correct 'reload values'
   LDI R16,0x06
   OUT TCNT0,R16
   ;Do what you need to do in the interrupt
   ;
   RETI
```


Each time the timer overflows we need to set the count register back to its initial value. Because when the timer overflows, the timer count register rolls over back to zero. If we do not manually reload this register, the next period will be wrong.

When to reload your TCNT?

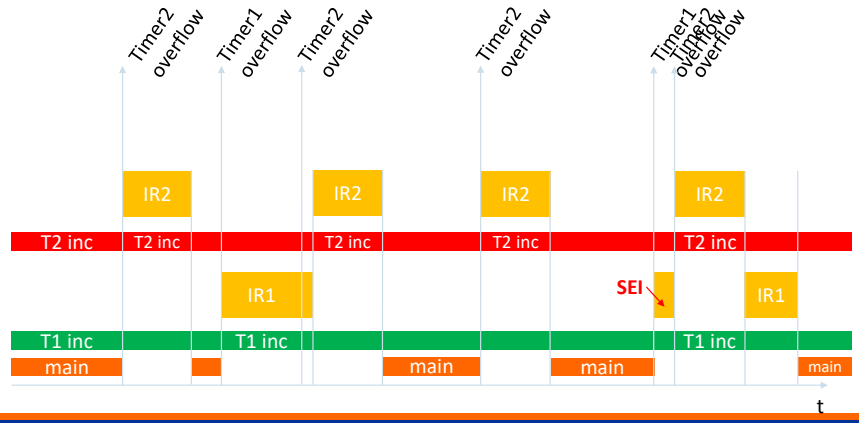
- 1 timer

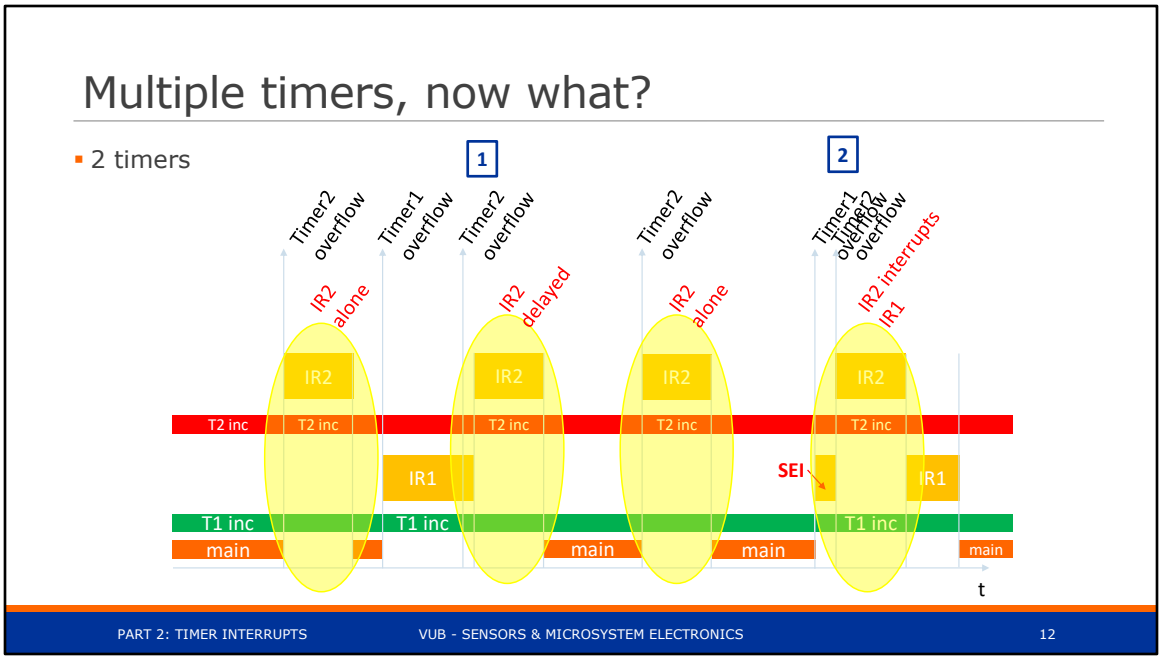


Where does one need to reload the timer count register? If it is done at the end, the period of the timer will have the time from the overflow to the reload added, and thus the frequency will be off. Correct answer: immediately at the beginning of the interrupt routine.

Multiple timers, now what?

- 2 timers





An interrupt call will disable the global interrupt flag. This means no interrupt can be executed while another interrupt is busy. The second interrupt will be postponed till after the first interrupt returns. Yet in some cases this would be wanted behavior for tasks that need precise timing. In this case, you can reenable the interrupt in the interrupt routine to allow it to be interrupted by another.

In this example, at the time indicated with [1] it can be seen that the execution of IR2 is postponed due to IR1 still being executed. Resulting in a time shift

At time [2] we have reenabled the interrupt with SEI inside of IR1. This allows for IR2 to “interrupt the interrupt” and be executed at its intended time.

Summary: Timer + interrupt

- Calculate the TCNT reload value based on your frequency
- Setup the timer using the, TCCR0A, TCCR0B registers
- Setup the interrupts with the TIMSK0 register and SEI

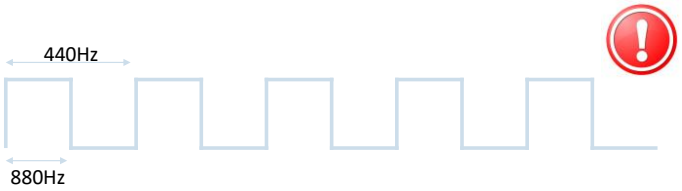
- Write the interrupt handler that reloads the timer to its initial value, and does what the timer needs to do
- Point to your interrupt handler using the .ORG

Task 5 & Task 6

Task 5

Sound the buzzer at a fixed frequency of 440 Hz when the button is pressed

- Set up the timer and interrupt
- Toggle the buzzer in the interrupt
  - There is a hidden feature in the pin register to toggle a pin... Read the datasheet to find out
- A tone of 440 Hz asks for an interrupt @ 880 Hz!



To sound the buzzer the pin of connected to the buzzer (PB1) needs to be toggled. To get a square wave at 440Hz the pin needs to be set and cleared once every period, this comes to a frequency of 880Hz.

HINT: use SBI PINx,1 to toggle a pin (explanation see datasheet or slide about pin config in Part1)

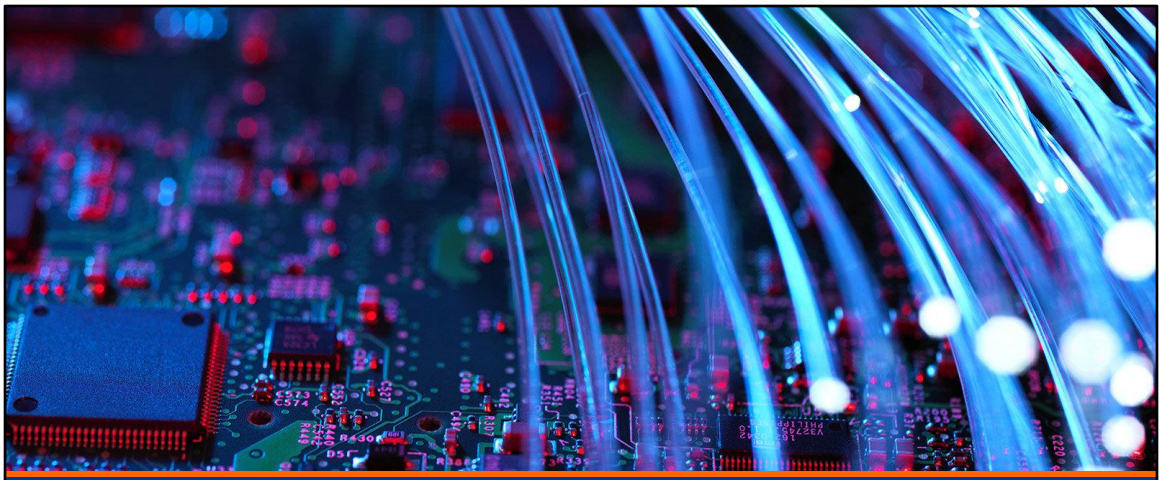
Task 6

Sound the buzzer at a frequency of 440 Hz or 880 Hz depending on the state of the switch

- Hint: Change the reload values depending on the state of the switch either in the main loop or in the interrupt

To sound the buzzer the pin of connected to the buzzer (PB1) needs to be toggled. To get a square wave at 440Hz the pin needs to be set and cleared once every period, this comes to a frequency of 880Hz.

HINT: use SBI PINx,1 to toggle a pin (explanation see datasheet or slide about pin config in Part1)



End of Part 2: Combining timers & interrupts