Lab 8: Timer (Internal Interrupt)

written by: Achmad Firdaus A - Computer Science 2015

1. Pembukaan

Selamat!

Ini adalah lab terakhir mata kuliah POK!

Dalam Lab kali ini, kita ingin berkenalan dengan Internal Interrupt dengan menggunakan timer sebagai perantaranya.

Note, apabila anda sudah merasa mengerti mengenai materi Timer, anda dapat langsung mengerjakan latihan di **halaman 6**.

2. Apa itu Timer?

Timer adalah sesuatu yang digunakan untuk mengukur interval waktu. Dalam AVR Timer dapat digunakan sebagai penanda untuk melakukan Interrupt dalam jangka waktu tertentu.

Analogi yang dapat digunakan untuk mendeskripsikan guna Timer sebagai Interrupt dalam AVR adalah:

> Dalam sebuah kelas jam yang dimulai pada jam 08.00, mahasiswa yang terlambat hanya boleh masuk ke kelas setiap 15 menit sekali (pada jam 08.15, 08.30, 08.45, dst). Untuk memastikan bahwa mahasiswa masuk tepat pada jam yang disebutkan di atas, dipasang sebuah *stopwatch* di dalam kelas yang bisa menghitung dari 0 hingga 15 menit dan dihubungkan dengan sebuah bel yang akan menyala ketika 15 menit telah berjalan.
>
> Jadi, pada jam 08.00 tepat, stopwatch akan dijalankan dan perkuliahan akan dimulai. Ketika *stopwatch* telah berjalan selama 15 menit, maka bel akan berbunyi. Hal ini menandakan bahwa perkuliahan dihentikan sejenak untuk membiarkan mahasiswa yang terlambat untuk masuk ke dalam kelas.
>
> Setelah mahasiswa yang terlambat telah masuk ke dalam kelas, maka perkuliahan dilanjutkan kembali hingga 15 menit berikutnya ketika bel kembali berbunyi dan perkuliahan kembali dihentikan sejenak untuk membiakan mahasiswa terlambat masuk ke dalam kelas. Hal ini dilakukan hingga perkuliahan pada hari tersebut telah selesai.

Dari cerita diatas, kita dapat menganalogikan bahwa perkuliahan merupakan Instruksi utama yang sedang berjalan, sedangkan aktifitas mahasiswa terlambat yang masuk kedalam kelas adalah instruksi Interrupt yang dijalankan. Sedangkan, *Stopwatch* yang digunakan merupakan Timer yang akan kita coba minggu ini.

3. Jenis-jenis Timer.

Timer dalam AVR dapat dibedakan menjadi beberapa tipe:

Alamat	Sumber	Definisi
$003	Timer1 Capt	Timer/Counter1 Capture Event (Interrupt ketika terdapat sinyal masukan yang menyebabkan nilai timer disimpan dalam Input Capture Register)
$004	Timer1 CompA	Timer/Counter1 Compare Match A (Melakukan Interrupt ketika Timer sudah sama dengan nilai yang telah ditentukan)
$005	Timer1 CompB	Timer/Counter1 Compare Match B (Melakukan Interrupt ketika Timer sudah sama dengan nilai yang telah ditentukan)
$006	Timer1 Ovf	Timer/Counter Overflow (Melakukan Interrupt ketika Timer sudah melebihi kapasitas/*Overflow*)
$007	Timer0 Ovf	Timer/Counter Overflow (Melakukan Interrupt ketika Timer sudah melebihi kapasitas/*Overflow*)
$00E	Timer0 Comp	Timer/Counter0 Compare Match A (Melakukan Interrupt ketika Timer sudah sama dengan nilai yang telah ditentukan)

Perbedaan **Timer0** dengan **Timer1** ?

1. **Timer0** memiliki panjang **8 bit.**
2. **Timer1** memiliki panjang **16 bit.**

4. Register dalam Internal Interrupt?

Minggu lalu kita telah berkenalan dengan **External Interrupt**. Dalam Implementasinya, External Interrupt menggunakan beberapa register:

1. **GIFR (General Interrupt Flag Register)**

Digunakan sebagai indikator bahwa sebuah External Interrupt sedang dijalankan atau tidak.

2. **GICR (General Interrupt Control Register)**

Digunakan untuk memilih jenis eksternal interrupt yang akan diaktifkan.

3. **MCUCR (MCU Control Register)**

Digunakan sebagai menentukan *trigger* yang menyebabkan Interrupt dijalankan.

Minggu ini, kita akan menggunakan kembali beberapa register di atas, dengan berkenalan dengan register-register baru.

Dalam Timer, kita mengenal register baru:

1. **TIFR (Timer Interrupt Flag Register)**

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	–	TOV0	OCF0	TIFR
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	

Hampir mirip seperti GIFR di atas, register ini digunakan sebagai indikator bahwa sebuah Timer Interrupt sedang dijalankan atau tidak.

2. **TIMSK (Timer/Counter Interrupt Mask Register)**

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	TICIE1	–	TOIE0	OCIE0	TIMSK
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	

Digunakan untuk memilih jenis internal interrupt yang akan diaktifkan.

3. **TCCR0 / TCCR1A/B (Timer/Counter Control Register)**

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
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	

Digunakan untuk mengatur kecepatan pada timer. Dapat dilihat dari tabel berikut:

TCCRx			Synchronous Timer0 & Timer1 P_{CK} = CK	Synchronous/Asynchronous Timer2 P_{CK2} = f (AS2)
Bit 2	Bit 1	Bit 0		
CSx2	CSx1	CSx0	$T_{CK0,1}$	T_{CK2}
0	0	0	0 (Timer Stopped)	0 (Timer Stopped)
0	0	1	P_{CK} (System Clock)	P_{CK2} (System Clock/Asynchronous Clock)
0	1	0	P_{CK}/8	P_{CK2}/8
0	1	1	P_{CK}/64	P_{CK2}/32
1	0	0	P_{C}/256	P_{CK2}/64
1	0	1	P_{CK}/1024	P_{CK2}/128
1	1	0	External Pin Tx falling edge	P_{CK2}/256
1	1	1	External Pin Tx rising edge	P_{CK2}/1024

Perhatikan bahwa **Nilai timer didasarkan dari nilai clock.** Dimana satu satuan waktu ditentukan dari nilai kombinasi nilai **CSx2 CSx1** dan **CSx0**. (Nilai x dimaksudkan sebagai jenis timer yang dipilih, e.g. Timer0 berarti CS02, CS01 dan CS00, Timer1 berarti CS12, CS11, CS10)

Semakin besar pembagi akan semakin lama satuan durasi per satuan waktunya. (Misal, durasi per satuan waktu kombinasi 001 (kecepatan timer setara clock biasa) akan lebih pendek dibandingkan dengan 101 (kecepatan timer setara clock/1024)).

Dalam satu detik berapa kali Overflow?

Gunakan rumus berikut:

$$TOV_{CK} = \frac{f_{CK}}{MaxVal}$$

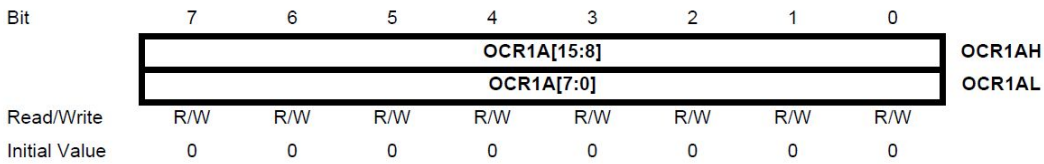
TOV_{ck} = frekuensi timer overflow dalam satu detik

f_{ck} = frekuensi clock (P_{ck}/N)

MaxVal = nilai maksimal TCNT ($2^{\text{jumlah bit pada TCNT}}$).

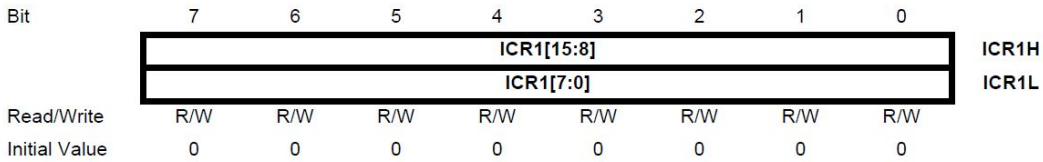
4. **OCR0 (Gambar pertama) dan OCR1A/B (Gambar kedua)**
   (Output Compare Register)

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



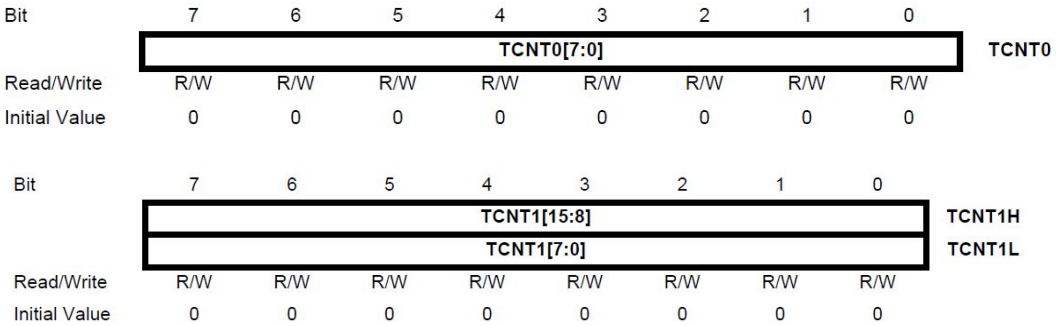
Register yang menyimpan nilai "perbandingan." Nilai timer akan selalu dibandingkan dengan nilai register OCR, apabila nilai timer sudah sama dengan OCR, maka Interrupt akan dijalankan. **(Hanya berlaku apabila kita menggunakan Timer jenis Compare Match).**

5. ICR1 (Input Capture Register)



Register ini berlaku apabila menggunakan **Timer jenis Capture Event**. Isi ICR akan selalu diupdate dengan nilai Timer ketika terjadi perubahan di register ICP1.

6. TCNT0 (Gambar pertama) dan TCNT1 (Gambar kedua)
(Timer/Counter Count Register)



Secara singkat, TCNT berlaku sebagai penunjuk timer, atau dapat juga dibilang **berperan seperti jarum jam pada timer.**

Seperti yang telah dijelaskan pada bagian 3, dapat dilihat bahwa **Timer0** memiliki **kapasitas 8 bit**, dan **Timer1** memiliki kapasitas **16 Bit.**

5. Bagaimana cara menggunakan Timer sebagai Internal Interrupt?

Dalam menggunakan Timer, **perlu dilakukan inisiasi di awal** ketika program pertama kali dijalankan. Berikut adalah langkah singkat melakukan Timer:

1. **Apabila Menggunakan Timer/Counter Overflow Interrupt:**
    1. Tetapkan pengaturan TCCR0/TCCR1A/TCCR1B.
    2. Tetapkan pengaturan TIFR.
    3. Tetapkan pengaturan TIMSK.
    4. Berikan perintah SEI.
    5. Jalankan perintah program seperti biasa.
2. **Apabila Menggunakan Timer/Counter Compare Interrupt:**
    1. Tetapkan pengaturan TCCR0/TCCR1A/TCCR1B.
    2. Tetapkan pengaturan TIFR.
    3. Tetapkan pengaturan TIMSK.
    4. Berikan nilai ke OCR0/OCR1A/OCR1B untuk nantinya dikomparasi dengan TCNT.
    5. Berikan perintah SEI.
    6. Jalankan perintah program seperti biasa.

6. Contoh Program

Berikut adalah contoh program untuk melakukan inversi/komplemen lampu yang menyala (misal dari nyala -> mati dan dari mati -> nyala) :
Dalam contoh program ini, Timer yang digunakan adalah **Timer1 Overflow.** Dengan kecepatan timer **Processor Clock/8.**
Gunakan Konfigurasi Hapsim (L8_example.xml) untuk mencoba program dibawah.

```
.include "m8515def.inc"

; At the start, go to RESET label first for initiating stuff
.org $00
        rjmp RESET

; Yo, when Internal Interrupt from Timer1 Overflow happened, Please run label ISR_TOV1
; Check table on the a few page back for other types of Internal Interrupts and their corresponding address
.org $06
        rjmp ISR_TOV1

RESET:
        ; Set the Stack Pointer pls
        ldi     r16,low(RAMEND)
        out     SPL,r16
        ldi     r16,high(RAMEND)
        out     SPH,r16

        ; Could you adjust the timer speed to (Processor clock / 8) per tick?
        ; (1<<CS10) -> Set 1 to CS01 block on TCCR1B
        ldi r16, (1<<CS01)          ;
        out TCCR1B,r16

        ; Hey, Please do Interrupt when Timer1 Overflow, thx!
        ; (1<<TOV1) -> Set 1 to TOV1 block on TIFR
        ldi r16, (1<<TOV1)
        out TIFR,r16

        ; Sup, I would like to use Timer1 Overflow as a Timer please
```

```
        ; (1<<TOIE1) -> Set 1 to TOIE1 block on TIMSK
        ldi r16, (1<<TOIE1)
        out TIMSK,r16

        ; Yo, I would like ALL port on DDRB as an output
        ser r16
        out DDRB,r16

        ; I'm ready, let's do this!
        sei

; While there are no Interrupt, let's just Infinite loop forever
forever:
        rjmp forever

ISR_TOV1:
        push r16
        in r16,SREG
        push r16
        in r16,PORTB    ; read Port B
        com r16         ; invert bits of r16
        out PORTB,r16   ; write Port B
        pop r16
        out SREG,r16
        pop r16
        reti            ; Interrupt is done, you can do your previous stuff back
```

7. Exercise

Yolo ingin memasak sesuatu menggunakan *microwave*, tetapi ternyata timer *microwave* tersebut tidak dapat berfungsi dengan baik. Untuk itu, Yolo meminta teman-teman mahasiswa POK untuk membuat timer yang menghitung lamanya waktu yang diperlukan agar makanan yang ingin Yolo *microwave* selesai tepat waktu.

Gunakan Konfigurasi Hapsim (l8_exercise.xml) yang telah disediakan.

Sesuaikan pula nilai PORT dari HAPSIM dengan program anda.

Cara mengerjakan:

Terdapat dua buah set LED (1 set = 8 LED):

1. Satu set (ada di **PORTA**) menunjukkan waktu yang tersisa hingga matang.
    a. Ketika program dinyalakan, **semua lampu di set ini akan menyala.**
    b. Seiring berjalannya waktu, **satu demi satu lampu akan mati** dari bawah (LED7) ke atas (LED0). Ketika semua lampu mati, lihat poin 2b.
2. Satu set (ada di **PORTB**) menunjukkan kondisi makanan dalam microwave.
    a. Ketika program pertama dijalankan, **LED0-3 yang berwarna merah akan menyala**, menunjukkan bahwa makanan belum matang.
    b. Ketika **semua LED dalam set nomor (1) telah mati, LED4-7 yang berwarna hijau akan menyala (sedangkan LED0-3 akan mati),**

menunjukkan bahwa makanan sudah matang.

Hint:

Gunakan variabel yang berisi waktu sisa, gunakan Internal Interrupt untuk mengubah nilai waktu sisa tersebut. Kemudian lakukan pengecekan nilai waktu sisa di setiap iterasi Infinite Loop.

Note: Anda dapat melihat video contoh hasil lab ini di **bit.ly/L8res** sebagai referensi.

BONUS ! [10 poin]

Buat timer sehingga dia dapat hampir akurat untuk menunjukkan satu perubahan lampu per detik.

Simpan program dengan nama file **Lab08_[Kode Asdos]_[Nama]_[NPM].asm**

(Tidak perlu di zip)

Reference:

[1] Slide Timer POK

[2] http://www.avr-tutorials.com/interrupts/The-AVR-8-Bits-Microcontrollers-External-Interrupts