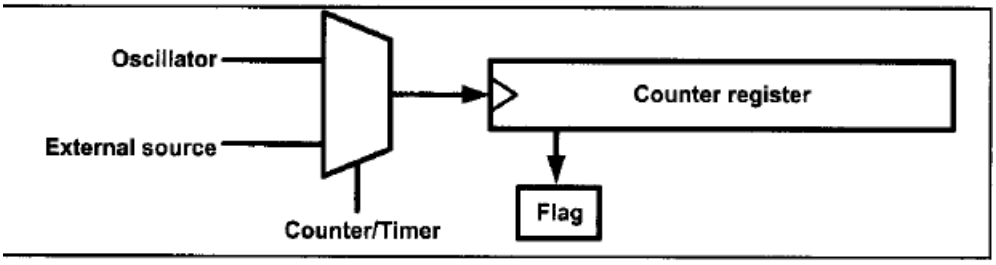
Timer

January 14, 2019

1 AVR Timer

1.1 Tujuan:

```
* Bisa menyebutkan timer pada AVRMega32 dan register terkait
* Mendeskripsikan mode Normal dan CTC dari AVR
* Memprogram timer dan membuat delay
* Memprogram AVR counter sebagai event counter
```



Counter/Timer pada AVR

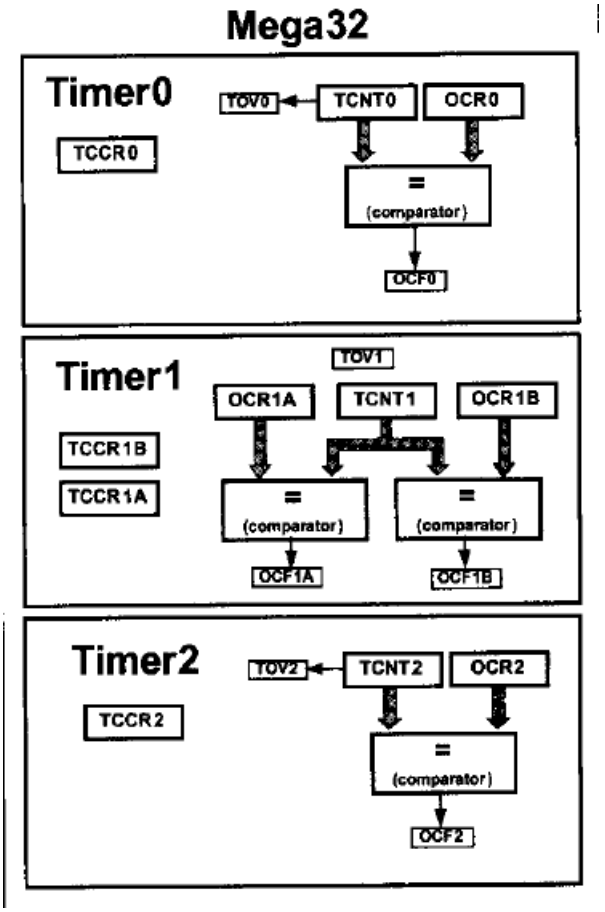
- Timer bisa berfungsi sebagai event counter dan pembuat time delay
  - event counter: koneksi ke external source, jika event terjadi, counter menghitung naik
  - time delay: koneksi ke oscillator, untuk menaikkan nilai counter jika oscillator berdetak

1.2 Timer/Counter pada AVRMega32

- Timer0 dan Timer2 : 8 bit
  - Timer2 bisa digunakan untuk realtime counter
  - ada perbedaan arti pada nilai Clock selector untuk timer0 dan timer2
- Timer1 : 16bit

1.3 Register untuk Timer

- Timer0 dan Timer2 pada AVRMega32 adalah 8 bit timer
- Timer1 merupakan 16 bit timer
- Register untuk masing-masing counter/timer:
  - TCNTn (Timer Counter) : counter
  - TIFR (Timer/counter Interrupt Flag Register) berisi:
    - TOVn (Timer Overflow): flag jika timer melampaui batas atas
    - OCRn (Output Compare Register): register untuk nilai yang dibandingkan dengan nilai TCNTn
    - OCFn (Output Compare Flag)
  - TCCRn (Timer/Counter Control register): menentukan mode



Counter/Timer pada AVR

1.4 Pemrograman Timer0

Timer/Counter Register 0: * 8 bit register * Berapa maksimal nilainya?

TCNT0	D7	D6	D5	D4	D3	D2	D1	D0

Timer/Counter 0 pada AVRMega32

1.5 Pengaturan Mode Timer0

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

FOC0 D7 Force compare match: This is a write-only bit, which can be used while generating a wave. Writing 1 to it causes the wave generator to act as if a compare match had occurred.

WGM00, WGM01

D6	D3	Timer0 mode selector bits
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correct
1	1	Fast PWM

COM01:00 D5 D4 Compare Output Mode:
These bits control the waveform generator (see Chapter 15).

CS02:00

D2	D1	D0	Timer0 clock selector
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
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.

TCCR0

1.5.1 Contoh 1

- Berapa nilai TCCR0 jika kita ingin memprogram Timer0 dalam mode normal, tanpa prescaler, dengan sumber clock adalah oscillator AVR
- Bagaimana perintah untuk menentukan nilai TCCR0?

TCCR0 =	0	0	0	0	0	0	0	1
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

- Perintah Assembler:

```
LDI R20,0x01
OUT TCCR0,R20
```

- Perintah C:

```
TCCR0 = 0x01;
```

1.5.2 Contoh 2:

Berapa masing-masing frekuensi dari timer dan periodenya , dengan frekuensi kristal oscilator pada AVR sebagai berikut? a. 10 MHz b. 8 MHz c. 1 MHz

a. F = 10 MHz, T = 1/10 MHz = 0,1μs
b. F = 8 MHz, T = 1/8 MHz = 0,125μs
c. F = 1 MHz, T = 1/1 MHz = 1μs

1.6 Mode Timer

- Ada 4 mode timer:
  - normal
  - PWM dengan fase yang benar
  - CTC (Clear Timer on Compare Match)
  - Fast PWM

1.7 Register Flag untuk Timer

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

TOV0 D0 Timer0 overflow flag bit
0 = Timer0 did not overflow.
1 = Timer0 has overflowed (going from $FF to $00).

OCF0 D1 Timer0 output compare flag bit
0 = compare match did not occur.
1 = compare match occurred.

TOV1 D2 Timer1 overflow flag bit

OCF1B D3 Timer1 output compare B match flag

OCF1A D4 Timer1 output compare A match flag

ICF1 D5 Input Capture flag

TOV2 D6 Timer2 overflow flag

OCF2 D7 Timer2 output compare match flag

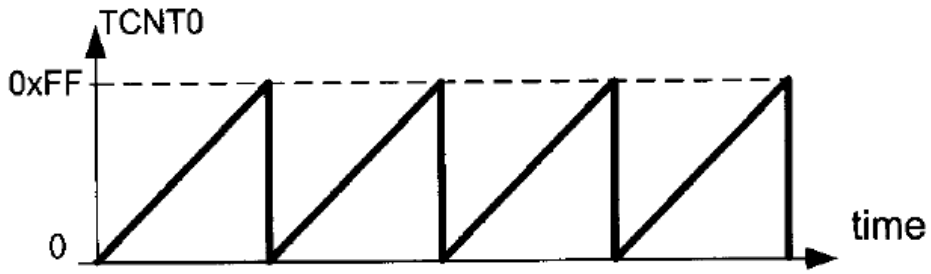
1.7.1 Clear Flag

- Jika terjadi event tertentu, flah ter set menjadi 1
- Perntah untuk clear:

```
LDI R20,0x01
OUT TIFR,R20
```

1.7.2 Normal Mode

- Pada mode normal, isi counter naik sampai mencapai maksimal 0xFF
- Jika nilai mengulang dari 0xFF ke 0x00,flag TOV0 menjadi 1



1.7.3 Membuat timer delay pada mode normal

1. Load nilai awal pada TCNT0
2. Set nilai TCCR0 untuk menentukan mode dan prescaler
3. monitor TOV0, keluar dari loop jika TOV0 high
4. Stop timer dengan memutus sumber clock `LDI R20, 0x00  OUT TCCR0, R20`
5. Clear TOV0 flag untuk delay berikutnya
6. Mulai lagi dari langkah 1

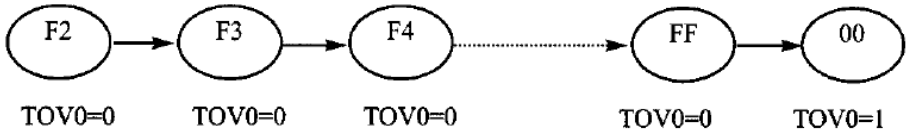
In the following program, we are creating a square wave of 50% duty cycle (with equal portions high and low) on the PORTB.5 bit. Timer0 is used to generate the time delay. Analyze the program.

```
.INCLUDE "M32DEF.INC"
.MACRO      INITSTACK          ;set up stack
      LDI   R20,HIGH(RAMEND)
      OUT   SPH,R20
      LDI   R20,LOW(RAMEND)
      OUT   SPL,R20
.ENDMACRO
      INITSTACK
      LDI   R16,1<<5     ;R16 = 0x20 (0010 0000 for PB5)
      SBI   DDRB,5       ;PB5 as an output
      LDI   R17,0
      OUT   PORTB,R17    ;clear PORTB
BEGIN:RCALL DELAY        ;call timer delay
      EOR   R17,R16      ;toggle D5 of R17 by Ex-Oring with 1
      OUT   PORTB,R17    ;toggle PB5
      RJMP BEGIN
;--------------------Time0 delay
DELAY:LDI   R20,0xF2     ;R20 = 0xF2
      OUT   TCNT0,R20    ;load timer0
      LDI   R20,0x01
      OUT   TCCR0,R20    ;Timer0, Normal mode, int clk, no prescaler
AGAIN:IN    R20,TIFR     ;read TIFR
      SBRS  R20,TOV0     ;if TOV0 is set skip next instruction
      RJMP  AGAIN
      LDI   R20,0x0
      OUT   TCCR0,R20    ;stop Timer0
      LDI   R20,(1<<TOV0)
      OUT   TIFR,R20     ;clear TOV0 flag by writing a 1 to TIFR
      RET
```

Solution:

In the above program notice the following steps:

1. 0xF2 is loaded into TCNT0.
2. TCCR0 is loaded and Timer0 is started.
3. Timer0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of F3, F4, F5, F6, F7, F8, F9, FA, FB, and so on until it reaches 0xFF. One more clock rolls it to 0, raising the Timer0 flag (TOV0 = 1). At that point, the "`SBRS R20,TOV0`" instruction bypasses the "`RJMP AGAIN`" instruction.
4. Timer0 is stopped.
5. The TOV0 flag is cleared.



Example 9-4

In Example 9-3, calculate the amount of time delay generated by the timer. Assume that XTAL = 8 MHz.

Solution:

We have 8 MHz as the timer frequency. As a result, each clock has a period of T = 1 /8 MHz = 0.125 μs. In other words, Timer0 counts up each 0.125 μs resulting in delay = number of counts × 0.125 μs.
The number of counts for the rollover is 0xFF – 0xF2 = 0x0D (13 decimal). However, we add one to 13 because of the extra clock needed when it rolls over from FF to 0 and raises the TOV0 flag. This gives 14 × 0.125 μs = 1.75 μs for half the pulse.

In Example 9-3, calculate the frequency of the square wave generated on pin PORTB.5. Assume that XTAL = 8 MHz.

Solution:

To get a more accurate timing, we need to add clock cycles due to the instructions.

```
                                    Cycles
      LDI   R16,0x20
      SBI   DDRB,5
      LDI   R17,0
      OUT   PORTB,R17
BEGIN:RCALL DELAY                     3
      EOR   R17,R16                   1
      OUT   PORTB,R17                 1
      RJMP  BEGIN                     2
DELAY:LDI   R20,0xF2                  1
      OUT   TCNT0,R20                 1
      LDI   R20,0x01                  1
      OUT   TCCR0,R20                 1
AGAIN:IN    R20,TIFR                  1
      SBRS  R20,0                 1 / 2
      RJMP  AGAIN                     2
      LDI   R20,0x0                   1
      OUT   TCCR0,R20                 1
      LDI   R20,0x01                  1
      OUT   TIFR,R20                  1
      RET                             4
                                     24
```

T = 2 × (14 + 24) × 0.125 μs = 9.5 μs and F = 1 / T = 105.263 kHz.

Find the delay generated by Timer0 in the following code, using both of the methods of Figure 9-8. Do not include the overhead due to instructions. (XTAL = 8 MHz)

```
.INCLUDE "M32DEF.INC"
      INITSTACK          ;add its definition from Example 9-3
      LDI   R16,0x20
      SBI   DDRB,5       ;PB5 as an output
      LDI   R17,0
      OUT   PORTB,R17
BEGIN:RCALL DELAY
      EOR   R17,R16      ;toggle D5 of R17
      OUT   PORTB,R17    ;toggle PB5
      RJMP  BEGIN
DELAY:LDI   R20,0x3E
      OUT   TCNT0,R20    ;load timer0
      LDI   R20,0x01
      OUT   TCCR0,R20    ;Timer0, Normal mode, int clk, no prescaler
AGAIN:IN    R20,TIFR     ;read TIFR
      SBRS  R20,TOV0     ;if TOV0 is set skip next instruction
      RJMP  AGAIN
      LDI   R20,0x00
      OUT   TCCR0,R20          ;stop Timer0
      LDI   R20,(1<<TOV0)      ;R20 = 0x01
      OUT   TIFR,R20           ;clear TOV0 flag
      RET
```

(a) (FF - 3E + 1) = 0xC2 = 194 in decimal and 194 × 0.125 μs = 24.25 μs.

(b) Because TCNT0 = 0x3E = 62 (in decimal) we have 256 - 62 = 194. This means that the timer counts from 0x3E to 0xFF. This plus rolling over to 0 goes through a total of 194 clock cycles, where each clock is 0.125 μs in duration. Therefore, we have 194 × 0.125 μs = 24.25 μs as the width of the pulse.