Interupt

January 14, 2019

1 AVR Interupt

Ada 2 metode layanan AVR terhadap peripheri: * Interupt * Polling

1.1 Interupt

- Jika perangkat membutuhkan service/layanan dari mikrokontroller, perangkat mengirimkan pemberitahuan kepada mikrokontroller berupa sinyal interupt
- Mikrokontroller menghentikan aktivitas saat mendapat interupt, lalu menjalankan program terkait

1.2 Polling

- Mikrokontroller secara kontinyu memonitor perangkat yang ada
- Jika kondisi sesuai, akan menjalankan service
- Setelah selesai lanjut ke perangkat berikutnya

1.3 Keunggulan Interupt dibandingkan polling

- Interrupt lebih efisien
  - mikrokontroller tidak perlu mengecek setiap perangkat
  - Jika flag membutuhkan waktu, mikrokontroller tidak perlu menunggu
- Bisa diterapkan prioritas
- Bisa melakukan mask (mengabaikan perangkat tertentu)

1.4 Interupt Service Rutine

- Untuk setiap setiap interupt harus ada interupt service routine (ISR) atau interupt handler
- ISR adalah program yang menangani interupt terkait

- Grup lokasi memori untuk ISR disebut interrupt vector table

```
In [ ]:
```

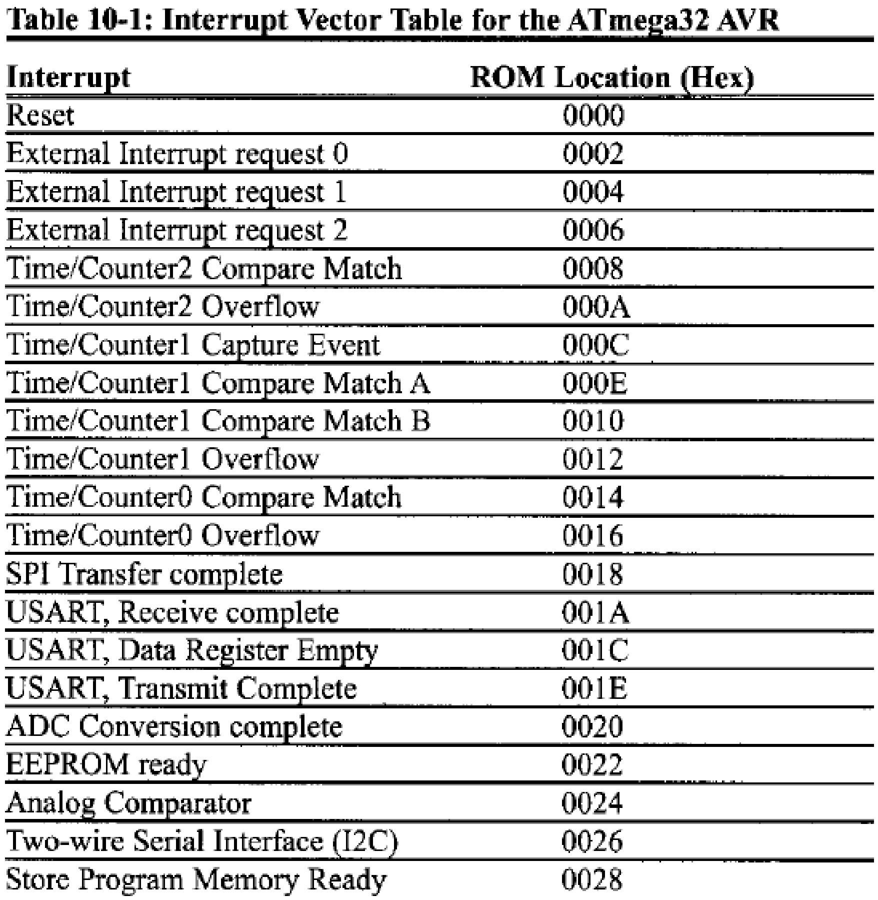


Table 10-1: Interrupt Vector Table for the ATmega32 AVR

Interrupt	ROM Location (Hex)
Reset	0000
External Interrupt request 0	0002
External Interrupt request 1	0004
External Interrupt request 2	0006
Time/Counter2 Compare Match	0008
Time/Counter2 Overflow	000A
Time/Counter1 Capture Event	000C
Time/Counter1 Compare Match A	000E
Time/Counter1 Compare Match B	0010
Time/Counter1 Overflow	0012
Time/Counter0 Compare Match	0014
Time/Counter0 Overflow	0016
SPI Transfer complete	0018
USART, Receive complete	001A
USART, Data Register Empty	001C
USART, Transmit Complete	001E
ADC Conversion complete	0020
EEPROM ready	0022
Analog Comparator	0024
Two-wire Serial Interface (I2C)	0026
Store Program Memory Ready	0028

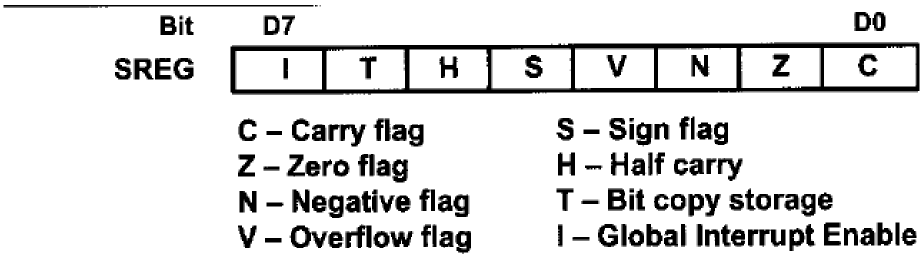
Interupt Vector Table

1.5 Sumber-sumber Interupt

- Timer
  - overflow
  - compare
- external hardware interrupt: INT0 (PD2), INT1 (PD3), INT2 (PB2)
- USART: 1 receive, 2 transmit
- SPI interrupt
- ADC

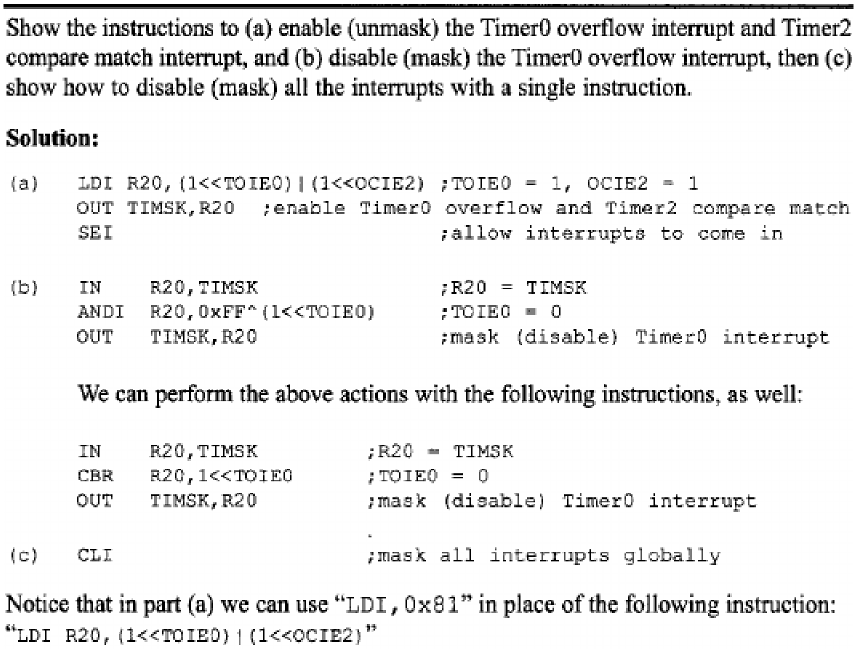
1.6 Mengaktivkan dan Menonaktivkan Interrupt

- Register terkait: SREG, interupt flag (I)
- perintah assembler: CLI (clear interrupt) membuat I = 0 (interrupt non-aktiv secara global)



Status Register

- Enable intrrupt:
  - perintah SEI : set flag I menjadi 1
  - jika I = 1, masing-masing interrupt diaktivkan dengan bit Interrupt Enable (IE)



Show the instructions to (a) enable (unmask) the Timer0 overflow interrupt and Timer2 compare match interrupt, and (b) disable (mask) the Timer0 overflow interrupt, then (c) show how to disable (mask) all the interrupts with a single instruction.

Solution:

```
(a)   LDI R20,(1<<TOIE0)|(1<<OCIE2) ;TOIE0 = 1, OCIE2 = 1
      OUT TIMSK,R20  ;enable Timer0 overflow and Timer2 compare match
      SEI                           ;allow interrupts to come in

(b)   IN    R20,TIMSK               ;R20 = TIMSK
      ANDI  R20,0xFF^(1<<TOIE0)     ;TOIE0 = 0
      OUT   TIMSK,R20               ;mask (disable) Timer0 interrupt
```

We can perform the above actions with the following instructions, as well:

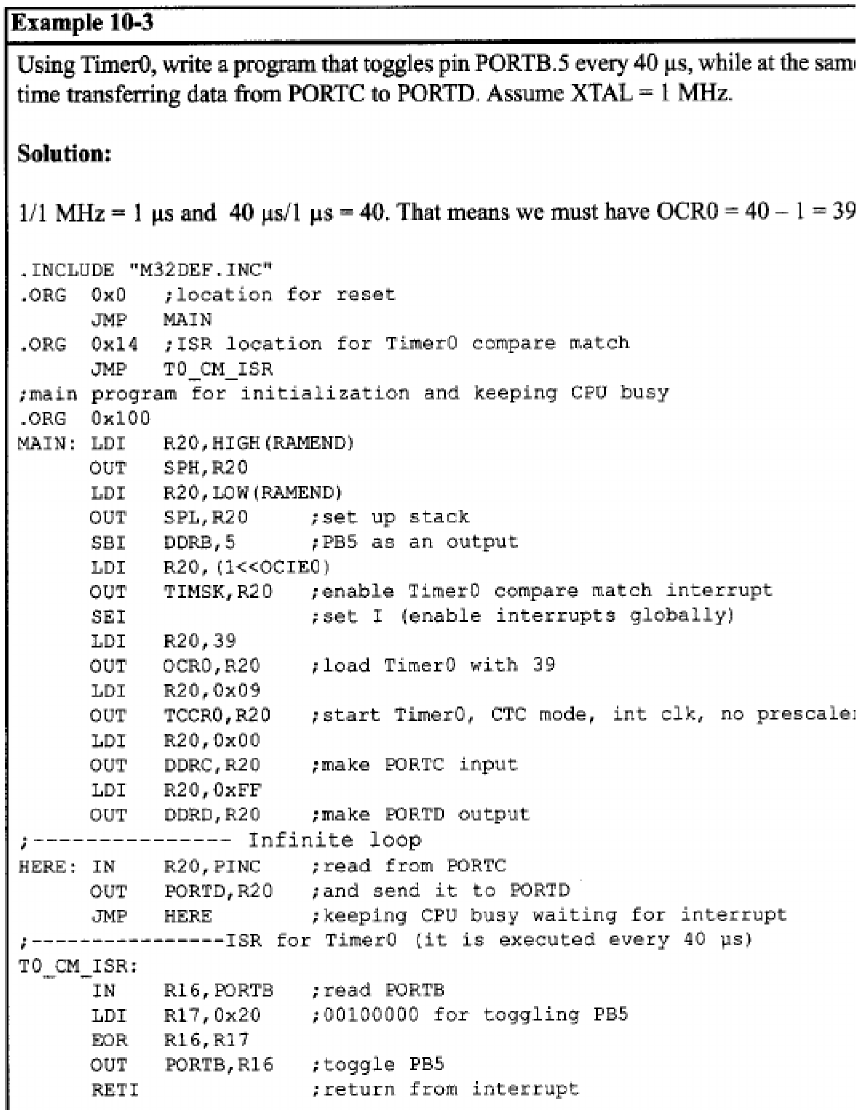
```
      IN    R20,TIMSK          ;R20 = TIMSK
      CBR   R20,1<<TOIE0       ;TOIE0 = 0
      OUT   TIMSK,R20          ;mask (disable) Timer0 interrupt

(c)   CLI                      ;mask all interrupts globally
```

Notice that in part (a) we can use "LDI, 0x81" in place of the following instruction: "LDI R20,(1<<TOIE0)|(1<<OCIE2)"

Enable dan Disable Interrupt

1.7 Tugas

Buat program untuk menyalakan 2 LED secara bergantian masing-masing menyala 10ms dan padam 10s menggunakan interrupt. Asumsi XT = 1 MHz.



Example 10-3

Using Timer0, write a program that toggles pin PORTB.5 every 40 µs, while at the sam
time transferring data from PORTC to PORTD. Assume XTAL = 1 MHz.

Solution:

1/1 MHz = 1 µs and 40 µs/1 µs = 40. That means we must have OCR0 = 40 – 1 = 39

```
.INCLUDE "M32DEF.INC"
.ORG  0x0   ;location for reset
      JMP   MAIN
.ORG  0x14  ;ISR location for Timer0 compare match
      JMP   T0_CM_ISR
;main program for initialization and keeping CPU busy
.ORG  0x100
MAIN: LDI   R20,HIGH(RAMEND)
      OUT   SPH,R20
      LDI   R20,LOW(RAMEND)
      OUT   SPL,R20     ;set up stack
      SBI   DDRB,5      ;PB5 as an output
      LDI   R20,(1<<OCIE0)
      OUT   TIMSK,R20   ;enable Timer0 compare match interrupt
      SEI               ;set I (enable interrupts globally)
      LDI   R20,39
      OUT   OCR0,R20    ;load Timer0 with 39
      LDI   R20,0x09
      OUT   TCCR0,R20   ;start Timer0, CTC mode, int clk, no prescale
      LDI   R20,0x00
      OUT   DDRC,R20    ;make PORTC input
      LDI   R20,0xFF
      OUT   DDRD,R20    ;make PORTD output
;--------------- Infinite loop
HERE: IN    R20,PINC    ;read from PORTC
      OUT   PORTD,R20   ;and send it to PORTD
      JMP   HERE        ;keeping CPU busy waiting for interrupt
;----------------ISR for Timer0 (it is executed every 40 µs)
T0_CM_ISR:
      IN    R16,PORTB   ;read PORTB
      LDI   R17,0x20    ;00100000 for toggling PB5
      EOR   R16,R17
      OUT   PORTB,R16   ;toggle PB5
      RETI              ;return from interrupt
```

Contoh Program dengan Interrupt