# CSE 315
# Microprocessors & Microcontrollers

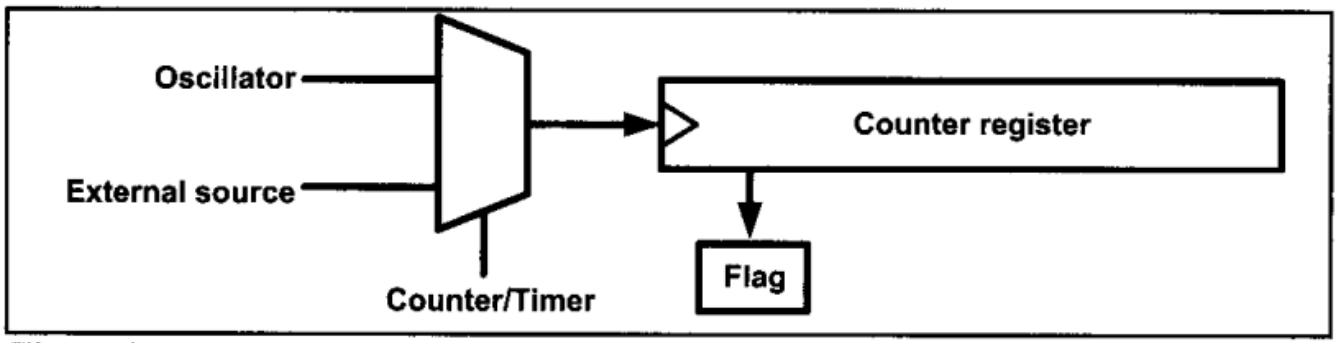
Tanvir Ahmed Khan

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology.

September 16, 2014

# *Recap*

# Timer/Counter Basics



# Timer Basic

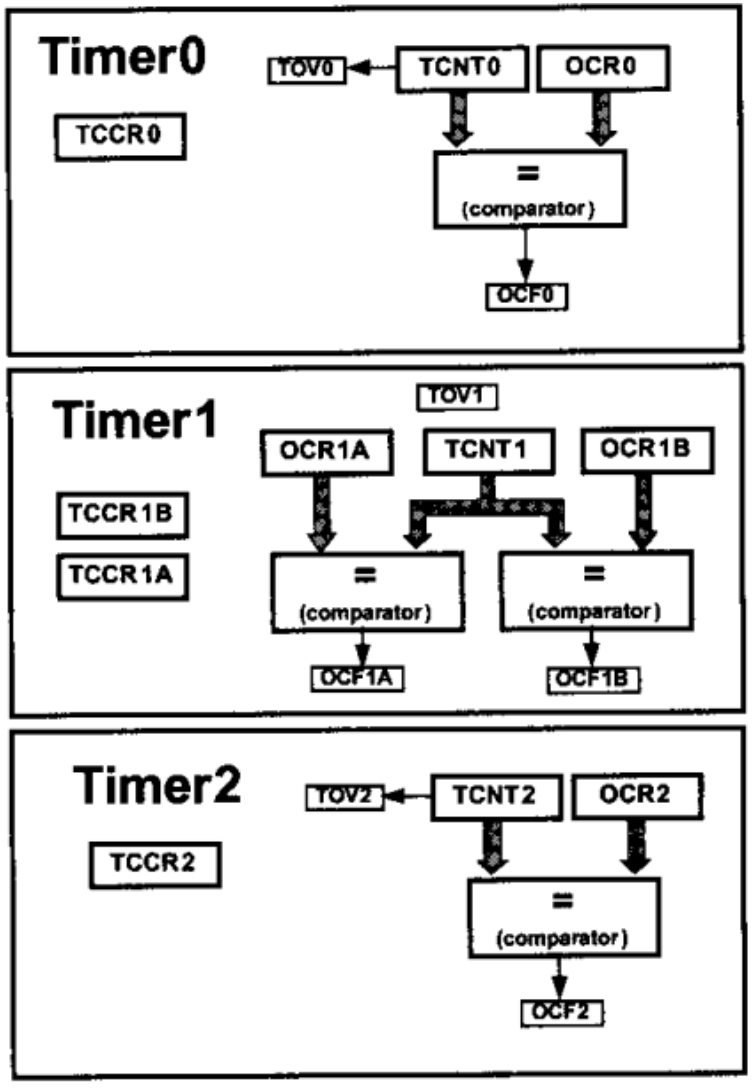
Two ways to generate a time delay,

- clear the counter & wait until the counter reaches a certain number
- **load the counter & wait until the counter overflows**

# ATmega16 Timers

- Timer0, 8-bit
- Timer1, 16-bit
- Timer2, 8-bit

# Timer Basic Registers




Timer/Counter Control Register (TCCR0)

| FOC0 | WGM00 | COM01 | COM00 | WGM01 | CS02 | CS01 | CS00 |
|---|---|---|---|---|---|---|---|
| 7 | | | | | | | 0 |

Timer/Counter Register (TCNT0)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | | | | | | | 0 |

Output Compare Register (OCR0)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | | | | | | | 0 |

Timer/Counter Interrupt Mask Register (TIMSK)

| OCIE2 | TOIE2 | TICIE1 | OCIE1A | OCIE1B | TOIE1 | OCIE0 | TOIE0 |
|---|---|---|---|---|---|---|---|
| 7 | | | | | | | 0 |

Timer/Counter Interrupt Flag REgister (TIFR)

| OCF2 | TOV2 | ICF1 | OCF1A | OCF1B | TOV1 | OCF0 | TOV0 |
|---|---|---|---|---|---|---|---|
| 7 | | | | | | | 0 |

| **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. |

# Programming Timer0

- Configure TCCR0,
  - `TCCR0 = 0b00000001;`
- Load TCNT0 with initial value,
  - `TCNT0 = 0b11110010;`
- Check continuously TOV0,
  - `while((TIFR & 0b00000001) == 0);`
- Stop the timer,
  - `TCCR0 = 0;`
- Clear TOV0,
  - `TIFR = TIFR | 0b00000001;`

# Simple Blink Example

```
 4
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     int i,j;
 9     for(i=0;i<100;i++)
10     {
11         for(j=0;j<100;j++)
12         {
13             asm volatile("nop");
14         }
15     }
16 }
17
18 int main(void)
19 {
20
21     DDRB  = DDRB  | 0b00000001;
22     PORTB = PORTB & 0b11111110;
23
24     while(1)
25     {
26         delay();
27         PORTB = PORTB ^ 0b00000001;
28     }
29     return 0;
30 }
31
```

```
 4
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     TCCR0 = 0b00000111;
 9     TCNT0 = 0b00001111;
10     while((TIFR & 0b00000001) == 0);
11     TCCR0 = 0;
12     TIFR = TIFR | 0b00000001;
13 }
14
15 int main(void)
16 {
17
18     DDRB  = DDRB  | 0b00000001;
19     PORTB = PORTB & 0b11111110;
20
21     while(1)
22     {
23         delay();
24         PORTB = PORTB ^ 0b00000001;
25     }
26     return 0;
27 }
28
```

# Practice Problems

- Finding the delay for a specific TCNT0 value
  - Clock = 8 MHz, TCNT0 = 0x3E, TCCR0 = 1
  - Clock = 8 MHz, TCNT0 = 0x00, TCCR0 = 5
- Finding the value of TCNT0 for a specific delay
  - Clock = 8 MHz, output signal frequency = 16 KHz
  - Clock = 8 MHz, output signal frequency = 125 Hz, with pre-scaler = 256

# Today's Topic

ATmega16 Interrupt

# Interrupts vs. Polling

# Interrupts vs. Polling

- Efficiency

# Interrupts vs. Polling

- Efficiency
- Monitoring several devices

# Interrupts vs. Polling

- Efficiency
- Monitoring several devices
- Priority

# Interrupts vs. Polling

- Efficiency
- Monitoring several devices
- Priority
- Ignoring a device request

# ATmega16 Interrupts

| Vector No. | Program Address[2] | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $000[1] | RESET | External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset |
| 2 | $002 | INT0 | External Interrupt Request 0 |
| 3 | $004 | INT1 | External Interrupt Request 1 |
| 4 | $006 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 5 | $008 | TIMER2 OVF | Timer/Counter2 Overflow |
| 6 | $00A | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 7 | $00C | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 8 | $00E | TIMER1 COMPB | Timer/Counter1 Compare Match B |
| 9 | $010 | TIMER1 OVF | Timer/Counter1 Overflow |
| 10 | $012 | TIMER0 OVF | Timer/Counter0 Overflow |
| 11 | $014 | SPI, STC | Serial Transfer Complete |
| 12 | $016 | USART, RXC | USART, Rx Complete |
| 13 | $018 | USART, UDRE | USART Data Register Empty |
| 14 | $01A | USART, TXC | USART, Tx Complete |
| 15 | $01C | ADC | ADC Conversion Complete |
| 16 | $01E | EE_RDY | EEPROM Ready |
| 17 | $020 | ANA_COMP | Analog Comparator |
| 18 | $022 | TWI | Two-wire Serial Interface |
| 19 | $024 | INT2 | External Interrupt Request 2 |
| 20 | $026 | TIMER0 COMP | Timer/Counter0 Compare Match |
| 21 | $028 | SPM_RDY | Store Program Memory Ready |

# Interrupts Service Routine

- program associated with the interrupt

# Interrupt Vector Table

- group of memory locations holding the addresses of ISR

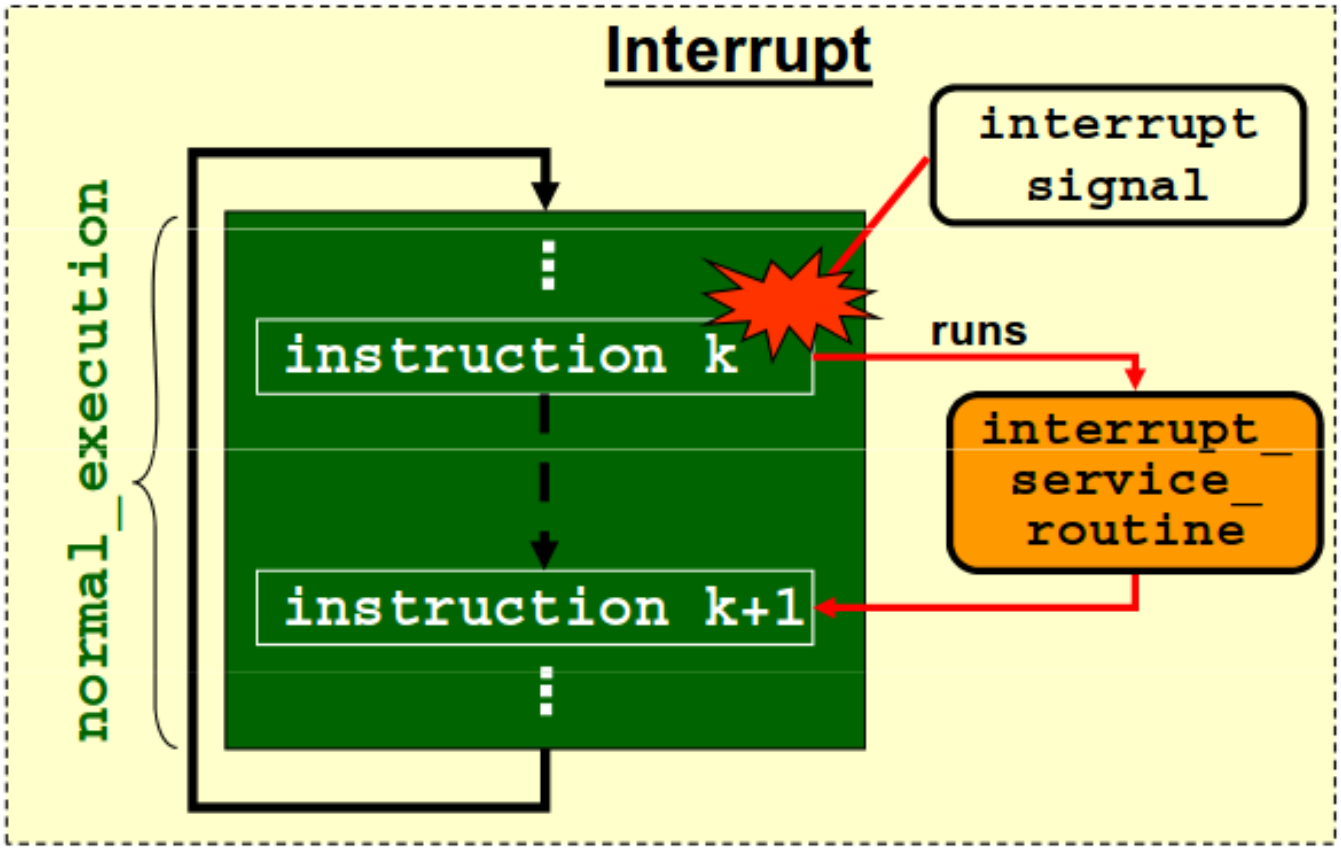
# Interrupt Vector Table

- group of memory locations holding the addresses of ISR

**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 |

# Interrupt Execution



# Steps in Interrupt Execution

# Steps in Interrupt Execution

- finishes the instruction currently executing

# Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt

# Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt
- saves Program Counter (also current context) onto stack

# Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt
- saves Program Counter (also current context) onto stack
- jumps to *interrupt vector table* which redirects to the address of the **interrupt service routine**

# Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt
- saves Program Counter (also current context) onto stack
- jumps to *interrupt vector table* which redirects to the address of the **interrupt service routine**
- executes ISR upto *RETI* statement

# Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt
- saves Program Counter (also current context) onto stack
- jumps to *interrupt vector table* which redirects to the address of the **interrupt service routine**
- executes ISR upto *RETI* statement
- retrieves the original context and PC by popping the first byte of stack

# Reference

- The avr microcontroller & embedded system, *Chapter 10*
  - Muhammad Ali Mazidi
  - Sarmad Naimi
  - Sepehr Naimi