Microprocessors & Interfacing

Interrupts (II)

Lecturer : Dr. Annie Guo

Lecture Overview

- Interrupts in AVR
  - External interrupts
  - Internal interrupts
    - Timers/Counters

External Interrupts

- The external interrupts are triggered by the INT7:0 pins.
  - If enabled, the interrupts will trigger even if the INT7:0 are configured as outputs
    - This feature provides a way of generating a software interrupt.
  - Can be triggered by a falling or rising edge or a logic level
    - Specified in External Interrupt Control Register
      - EICRA (for INT3:0)
      - EICRB (for INT7:4)

External Interrupts (cont.)

- To enable an interrupt, two bits must be set
  - I bit in SREG
  - INTx bit in EIMSK
- To activate an interrupt, the following must be met:
  - The interrupt must be enabled
  - The associated external pin must have a designed signal asserted.

EIMSK

- External Interrupt Mask Register
  - A bit is set to enable the related interrupt

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

EICRA

- External Interrupt Control Register A
  - For INT0-3
  - Defines the type of signals that activates the external Interrupt

7	6	5	4	3	2	1	0
ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

EICRB

- External Interrupt Control Register B
  - For INT4-7
  - Defines the type of signals that activates the External Interrupt

7	6	5	4	3	2	1	0
ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request
1	0	The falling edge between two samples of INTn generates an interrupt request.
1	1	The rising edge between two samples of INTn generates an interrupt request.

EIFR

- Interrupt flag register
  - A bit is set when an event-triggered interrupt is enabled and the related event on the related INT pin happens.
    - Event-triggered interrupt: signal edge activated.

7	6	5	4	3	2	1	0
INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0

Example 1

- Design a system, where the state of LEDs toggles under the control of the user.

Example 1 (solution)

- Use an external interrupt
  - Connect the external interrupt pin to a push button
  - When the button pressed, the interrupt is generated
- In the assembly code
  - Set up the interrupt
    - Set up the interrupt vector
    - Enable the interrupt
  - Write a service routine for this interrupt
    - Change the display pattern
    - Write the pattern to the port connected to the LEDs

Code for Example 1

```
.include "m64def.inc“

.def    temp =r16
.def    output = r17
.def    count = r18
.equ    PATTERN = 0b01010101

                                ; set up interrupt vectors
        jmp RESET
.org    INT0addr
        jmp EXT_INT0

RESET:
        ldi temp, low(RAMEND)   ; initialize stack
        out SPL, temp
        ldi temp, high(RAMEND)
        out SPH, temp

        ser temp                ; set Port C as output
        out DDRC, temp
        out PORTC, temp
        ldi output, PATTERN
                                ; continued
```


Code for Example 1

```
; continued
        ldi temp, (2 << ISC00)      ; set INT0 as falling edge triggered interrupt
        sts EICRA, temp

        in temp, EIMSK              ; enable INT0
        ori temp, (1<<INT0)
        out EIMSK, temp

        sei                         ; enable Global Interrupt
        jmp main

EXT_INT0:
        push temp                   ; save register
        in temp, SREG               ; save SREG
        push temp

        com output                  ; flip the pattern
        out PORTC, output
        inc count

        pop temp                    ; restore SREG
        out SREG, temp
        pop temp                    ; restore register
        reti
```

Code for Example 1

```
; continued

                    ; main - does nothing but increment a counter
main:
        clr count
        clr temp
loop:
        inc temp            ; a dummy task in main
        rjmp loop
```

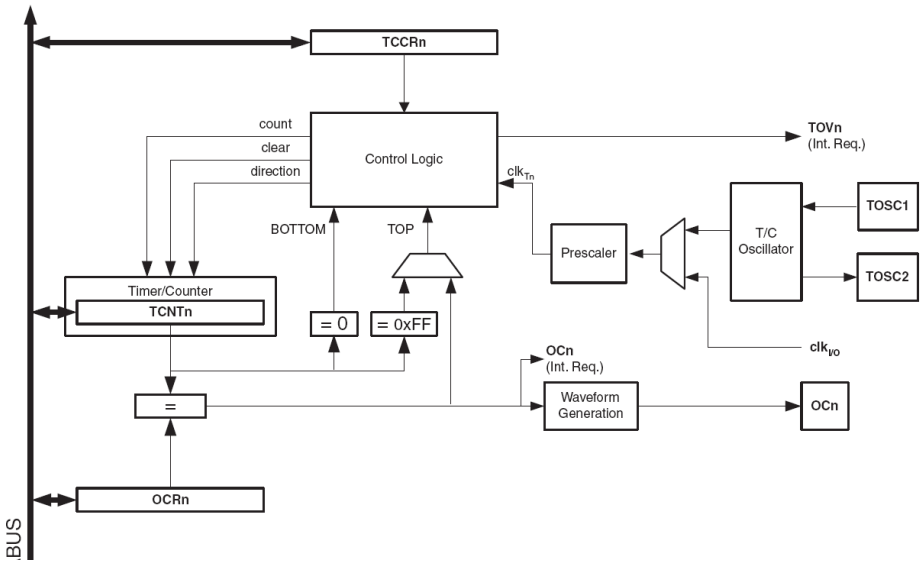

Timer/Counters

- Simply binary counters
- Used in two different modes:
  - Timer
    - Counting time periods
  - Counter
    - Counting the events or pulse or something of this nature
- Can be used to
  - Measure time periods, speed, frequency
  - Generate PWM signals
  - Schedule real-time tasks
  - etc.

Timer/Counters in AVR

- In AVR, there are 8-bit and 16-bit timer/counters.
  - Timer 0 and Timer 2: 8-bit
  - Timer 1 16-bit

8-bit Timer/Counter Block Diagram

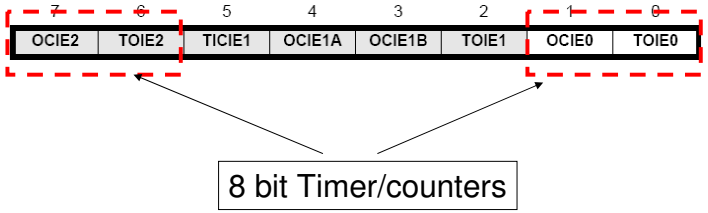


8-bit Timer/Counter

- The counter can be initialized with
  - 0 (controlled by *reset*)
  - a number (controlled by *count signal*)
- Can count up or down
  - controlled by *direction signal*
- Those controlled signals are generated by hardware control logic
  - The control logic is further controlled by programmer by
    - Writing control bits into TCCRn
- Output
  - Overflow interrupt request bit
  - Output Compare interrupt request bit
  - OCn bit: Output Compare bit for waveform generation

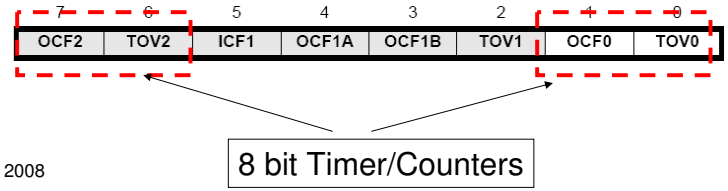
TIMSK

- Timer/Counter Interrupt Mask Register
  - Set TOIEx (and I-bit in SREG) to enable the Overflow Interrupt
  - Set OCIEx (and I bit in SREG) to enable Compare Match Interrupt



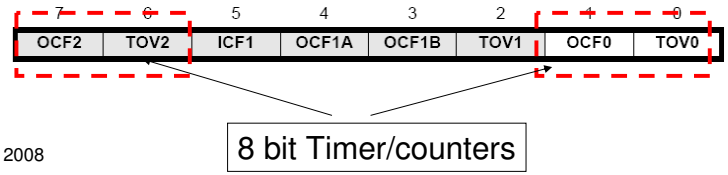
TIFR

- Timer/Counter Interrupt Flag Register
  - OCFx bit is set when a Compare Match between the counter and the data in OCRx (Output Compare Register).
    - When (I=1)&&(OCIEx=1)&&(OCFx=1), the related Timer/Counter Compare Match Interrupt is executed.
  - OCFx bit is cleared by hardware when the related interrupt is handled or can be cleared by writing a logic 0 to the flag



TIFR (cont.)

- Timer/Counter Interrupt Flag Register
  - TOVx bit is set when an overflow occurs in the counter.
    - When (I=1)&&(TOIEx=1)&&(TOVx=1), the related Timer/Counter Overflow Interrupt is executed.
    - In PWM mode, this bit is set when the counter changes counting direction at 0x00
  - OCFx bit is cleared by hardware when the related interrupt is handled or can be cleared by writing a logic 0 to the flag



TCCRx

- Timer Counter Control Register
  - E.g. TCCR0
    - For Timer/Counter0

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

TCCR0 Bit Description

- Bit 7:3:
  - control the mode of operation
    - the behavior of the Timer/Counter and the output, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits.
    - The simplest mode of operation is the Normal Mode (WGM01:00 =00). In this mode the counting direction is always up. The counter rolls over when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00).
- Refer to Mega64 Data Sheet (pages 96~100) for details.

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

TCCR0 Bit Description (cont.)

- Bit 2:0
  - Control the clock selection

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

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

Example 2

- Implement a scheduler that can execute a task every one second.

Example 2 (solution)

- Use Timer0 to count the time
  - Let's set Timer0 prescaler to 8
    - The time-out for the setting should be
      - 256*(clock period) = 256*8/(7.3728 Mhz) = 278 us
        - » Namely, we can set the Timer0 overflow interrupt that is to occur every 278 us.
        - » Note, Clk_{tos} = 1/7.3728 Mhz (obtained from the data sheet)
    - For one second, there are
      - 1000000/278 = 3597 interrupts
- In code,
  - Set Timer0 interrupt to occur every 278 microseconds
  - Use a counter to count to 3597 interrupts for counting 1 second
  - To observe the 1 second time period, use LEDs that toggles every one second.

Example 2

```
; This program implements a timer that counts one second using
; Timer0 interrupt

.include "m64def.inc"

.equ PATTERN=0b11110000
.def temp=r16
.def leds = r17


; The macro clears a word (2 bytes) in a memory
; the parameter @0 is the memory address for that word
.MACRO Clear
        ldi r28, low(@0)        ; load the memory address to Y
        ldi r29, high(@0)
        clr temp
        st y+, temp             ; clear the two bytes at @0 in SRAM
        st y, temp
.ENDMACRO
                                ; contined
```


Example 2

```
; continued
.dseg
 SecondCounter:
    .byte 2                 ; Two-byte counter for counting seconds.
TempCounter:
    .byte 2                 ; Temporary counter. Used to determine
                            ; if one second has passed
.cseg
.org 0x0000
    jmp RESET
    jmp DEFAULT             ; No handling for IRQ0.
    jmp DEFAULT             ; No handling for IRQ1.
    …
    jmp Timer0OVF           ; Jump to the interrupt handler for Timer0 overflow.
    …
    jmp DEFAULT             ; default service for all other interrupts.
DEFAULT:  reti              ; no service
                                    ; continued
```


Example 2

```
; continued

RESET:  ldi temp, high(RAMEND)      ; Initialize stack pointer
        out SPH, temp
        ldi temp, low(RAMEND)
        out SPL, temp

        ser temp                    ; set Port C as output
        out DDRC, temp

        rjmp main
                                    ; continued
```

Example 2

```
; continued

Timer0OVF:                  ; interrupt subroutine to Timer0
        in temp, SREG
        push temp           ; Prologue starts.
        push r29            ; Save all conflict registers in the prologue.
        push r28
        push r25
        push r24            ; Prologue ends.
        ldi r28, low(TempCounter)    ; Load the address of  the temporary
        ldi r29, high(TempCounter)   ; counter.
        ld r24, y+          ; Load the value of the temporary counter.
        ld r25, y
        adiw r25:r24, 1     ; Increase the temporary counter by one.
                                                    ; continued
```


Example 2

```
; continued
        cpi r24, low(3597)          ; Check if (r25:r24)=3597
        ldi  temp, high(3597)       ; 3597= 10^6/278
        cpc r25, temp
        brne  NotSecond
        com leds
        out PORTC, leds
        Clear TempCounter           ; Reset the temporary counter.
        ldi r30, low(SecondCounter)    ; Load the address of  the second
        ldi r31, high(SecondCounter)   ; counter.
        ld r24, z+          ; Load the value of the second counter.
        ld r25, z
        adiw r25:r24, 1     ; Increase the second counter by one.
                                                    ; continued
```


Example 2

```
; continued
        st z, r25           ; Store the value of the second counter.
        st -z, r24
        rjmp EndIF
NotSecond:
        st y, r25           ; Store the value of the temporary counter.
        st -y, r24
EndIF:
        pop r24             ; Epilogue starts;
        pop r25             ; Restore all conflict registers from the stack.
        pop r28
        pop r29
        pop temp
        out SREG, temp
        reti                ; Return from the interrupt.
                                                    ; continued
```


Example 2

```
; continued

main:
        ldi leds, 0xff
        out PORTC, leds
        ldi leds, PATTERN
        Clear TempCounter           ; Initialize the temporary counter to 0
        Clear SecondCounter         ; Initialize the second counter to 0
        ldi temp, 0b00000010
        out TCCR0, temp             ; Prescaling value=8
        ldi temp,  1<<TOIE0         ; =278 microseconds
        out TIMSK, temp             ; T/C0 interrupt enable
        sei                         ; Enable global interrupt
loop:   rjmp loop                   ; loop forever
```

Reading Material

- Chapter 8: Interrupts and Real-Time Events. Microcontrollers and Microcomputers by Fredrick M. Cady.
- Mega64 Data Sheet.
  - External Interrupts.
  - Timer0

Homework

1. What do you need to do to set up an Timer0 Output Compare Match Interrupt?

Homework

2. Based on the Example 1 in this week lecture slides, implement a software interrupt such that when there is an overflow in the counter that counts the number of LED toggles, all LEDs are turned on.