Microprocessors & Interfacing

Interrupts (II)

Lecturer : Annie Guo

Lecture Overview

- Interrupts in AVR
  - Internal interrupt
    - Timer

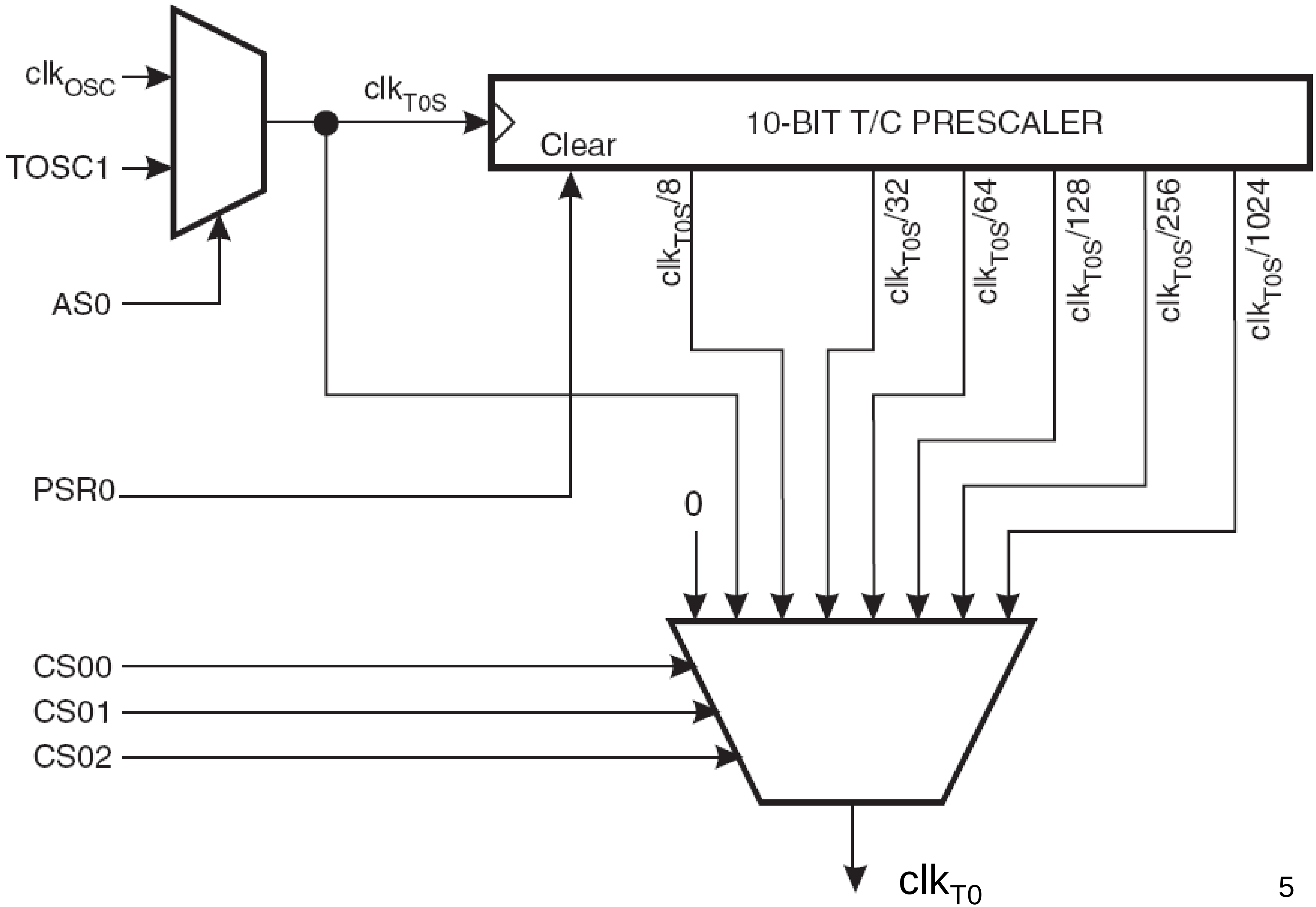
Timer

- A timer is simply a binary counter
- Can be used to
  - Measure time duration
  - Generate PWM signals
  - Schedule real-time tasks
  - etc.

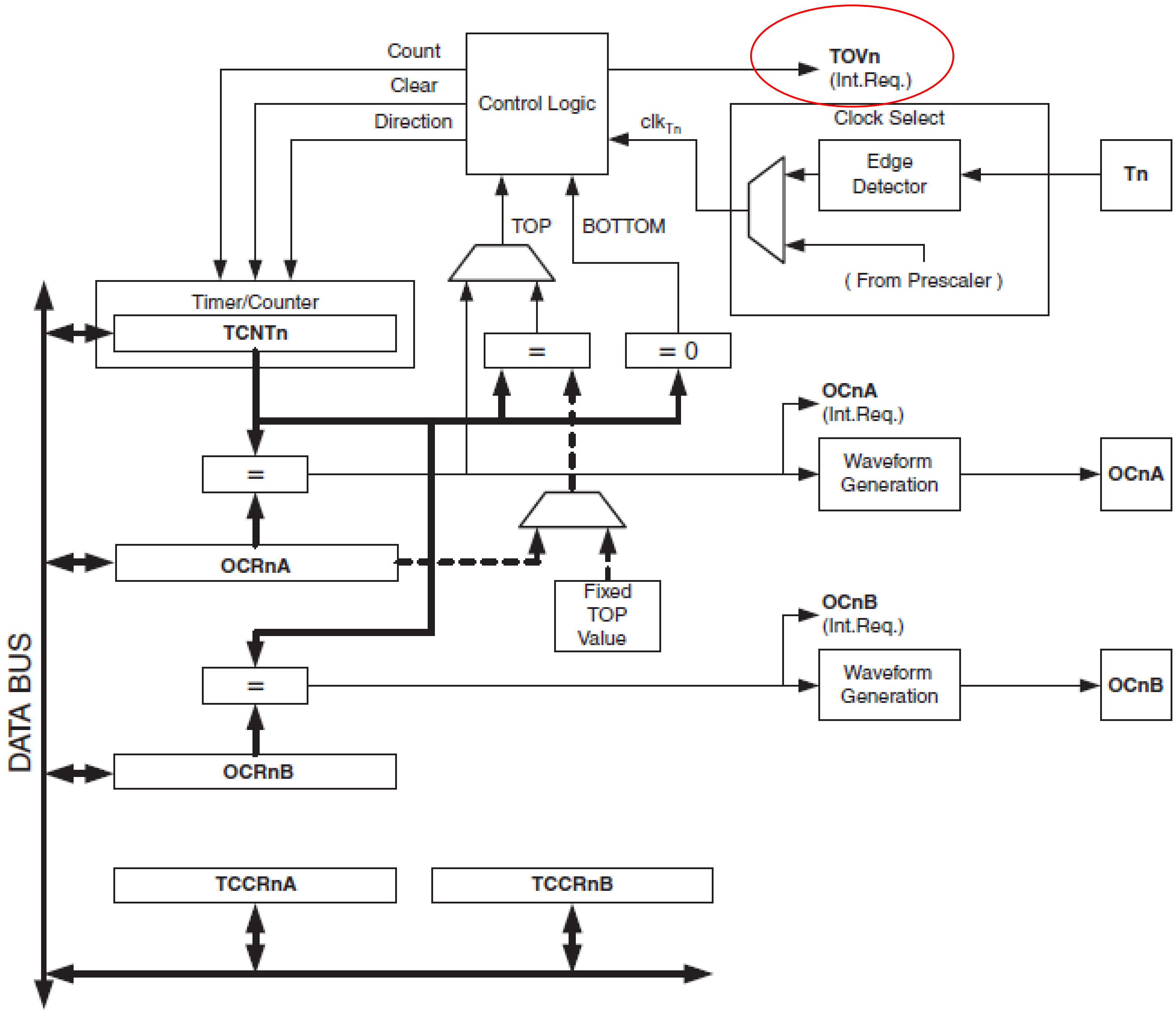
Timers in AVR

- In AVR, there are 8-bit and 16-bit timers.
  - Timer 0 and Timer 2
    - 8-bit counters
  - Timer 1, 3-5
    - 16-bit counters
- Timer0 is covered in the next slides
  - Similar designs can be found for other timers
    - See the Atmega2560 data sheet

Timer0 Clock Source*



8-bit Timer Block Diagram*



8-bit Timer

- 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 TCCRnA/TCCRnB
- Output
  - Overflow interrupt request bit
  - Output Compare interrupt request bit
  - OCn bit: Output Compare bit for waveform generation

TIMSK0

- Timer/Counter Interrupt Mask Register
  - Set TOIE0 (and I-bit in SREG) to enable the Overflow Interrupt
  - Set OCIE0 (and I bit in SREG) to enable Compare Match Interrupt

Bit	7	6	5	4	3	2	1	0
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Control bits for Timer/Counter0

TCCR0A/B

- Timer Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0 Bit Description

- COM0xn/WGM0n/FOC0:
  - control the mode of operation
    - the behavior of the Timer/Counter and the output, is defined by the combination of the Waveform Generation mode (WGM02 WGM00) and Compare Output mode (COM0x1:0) bits.
    - The simplest mode of operation is the Normal Mode (WGM02:00 =000). In this mode the counting direction is up. The counter rolls over when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00).
- Refer to Mega2560 Data Sheet (pages 118~194) for details.

TCCR0 Bit Description (cont.)

- Bit 2:0 in TCCR0B
  - Control the clock selection

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
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

T_{clk}

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Example 1

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

Example 1 (solution)

- Use 8-bit Timer0 to count the time
  - Let's set Timer0 prescaler to /64 (i.e. the system frequency divided by 64)
    - The time-out for the setting should be
      - 256*(clock period) = 256*64/(16 MHz)

        = 1024 us
        - » Namely, we can set the Timer0 overflow interrupt that is to occur every 1024 us.
        - » Note, Clk_{TOS} = 1/16 MHz (obtained from the data sheet)
    - For one second, there are
      - 1000000/1024 =~ 1000 interrupts
- In code,
  - Set Timer0 interrupt to occur every 1024 microseconds
  - Use a counter to count to 1000 interrupts for counting 1 second
  - To observe the 1 second time period, use LEDs that toggles every 1000 interrupts (i.e. one second).

Example 1

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

.include "m2560def.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 YL, low(@0)            ; load the memory address to Y
        ldi YH, high(@0)
        clr temp
        st Y+, temp                ; clear the two bytes at @0 in SRAM
        st Y, temp
 .endmacro
                                                    ; continued
```

Example 1

```
; continued
.dseg
 SecondCounter:
       .byte 2                          ; Two-byte counter for counting seconds.
TempCounter:
       .byte 2                          ; Temporary counter. Used to determine
                                        ; if one second has passed (when TempCounter=1000)
 .cseg
 .org 0x0000
       jmp RESET
       jmp DEFAULT                      ; No handling for IRQ0.
       jmp DEFAULT                      ; No handling for IRQ1.
       …
.org OVF0addr
       jmp Timer0OVF                    ; Jump to the interrupt handler for Timer0 overflow.
       …
       jmp DEFAULT                      ; default service for all other interrupts.
DEFAULT:  reti                          ; no service
                                                                ; continued
```

Example 1

```
; continued

RESET:

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

        rjmp main
                                        ; continued
```

Example 1

```
; continued

Timer0OVF:                    ; interrupt subroutine for Timer0
        ; in temp, SREG
        push temp             ; Prologue starts.
         push Yh              ; Save all conflict registers in the prologue.
         push YL
         push r25
         push r24             ; Prologue ends.
         ldi YL, low(TempCounter)    ; Load the address of  the temporary
         ldi YH, 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 1

```
; continued
        cpi r24, low(1000)              ; Check if (r25:r24)=1000
        brne NotSecond
        cpi  r25, high(1000)            ; 1000 = 10^6/1024
        brne  NotSecond
        com leds
        out PORTC, leds
        Clear TempCounter               ; Reset the temporary counter.

        ldi ZL, low(SecondCounter)      ; Load the address of  the second
        ldi ZH, 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 1

```
; 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 YL
        pop YH
        pop temp
        ; out SREG, temp
        reti                     ; Return from the interrupt.

                                                  ; continued
```

Example 1

```
; continued

main:
        ldi leds, 0xff                  ; Init pattern displayed
        out PORTC, leds
        ldi leds, PATTERN
        Clear TempCounter               ; Initialize the temporary counter to 0
        Clear SecondCounter             ; Initialize the second counter to 0
        ldi temp, 0b00000000
        out TCCR0A, temp
        ldi temp, 0b00000011
        out TCCR0B, temp                ; Prescaling value=64, counting 1024 us
        ldi temp,  1<<TOIE0
        sts TIMSK0, temp                ; T/C0 interrupt enable
        sei                             ; Enable global interrupt
  loop: rjmp loop                       ; loop forever
```

Reading Material

- Chapter 10: Interrupts and Real-Time Events. Microcontrollers and Microcomputers by Fredrick M. Cady.
- Mega2560 Data Sheet.
  - External Interrupts.
  - Timer0

Homework

1. An underground oil tank monitor system has the following functions:
    1. read(): to read the tank oil level
    2. display(): to display the oil level
    3. main(): process a few of basic tasks: if the oil level is below the low limit, do something; if oil level is over the high limit, do something else; and other routine work.

    It is required that the display should be updated every 1 minute, reading should be done at least every 10 seconds. Assume read() and display() take 1 ms and 5 ms, respectively. Design a scheduling controller for those functions so that the above requirements can be met and the design leads to an easy assembly code implementation.