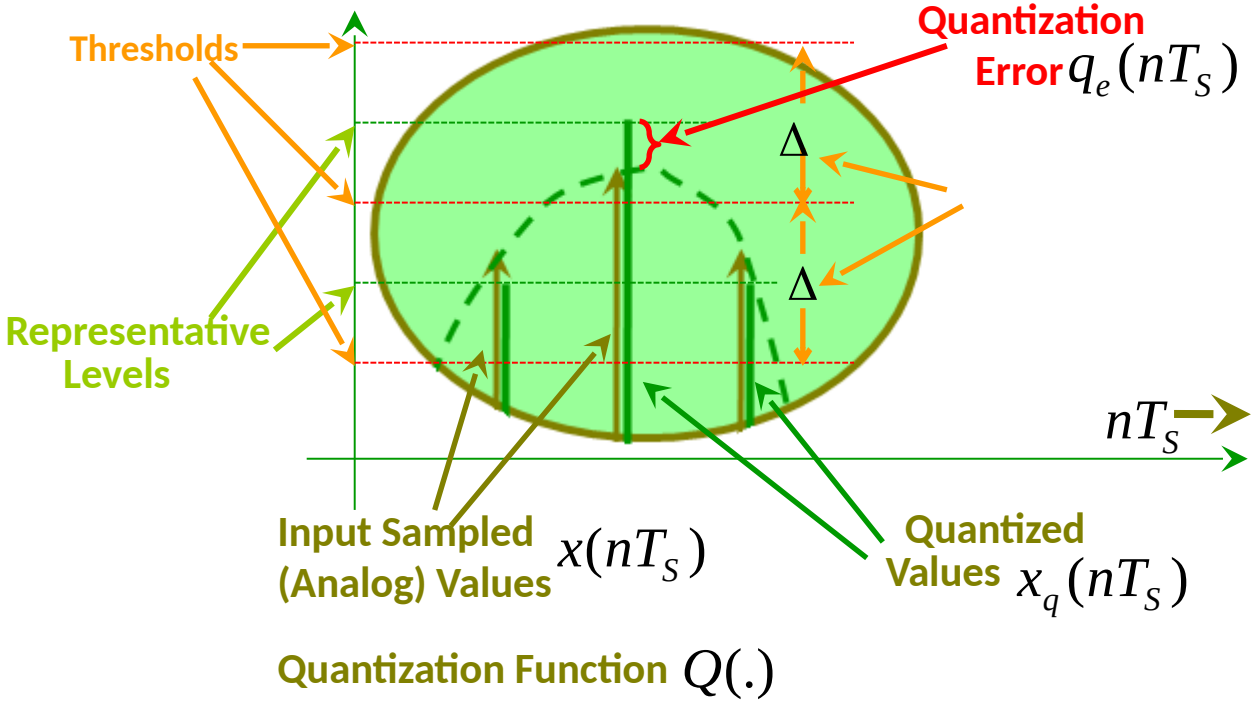
EE2016 Microprocessor Theory & Lab, Fall, 2024

Week 10: Timers and Counters
(Timers, Timers using Polling, Timers using Interrupt)

Dr. R. Manivasakan, OSWM Lab, EE Dept, IIT, Madras

ADC Basics



$$x_q(nT_S) = Q(x(nT_S))$$

$$x_q(nT_S) = x(nT_S) + q_e(nT_S)$$

ADC C- Code

- Definitions
  - LEDs on PORT0, ADC_DONE, OVERRUN etc
- main
  - Infinite loop
  - ADC output with input ch2, throw away the 2 LSB bits
  - To get 8 bits displayed on 8 LEDs
- Init_ADC0()
  - PINSEL
  - AD0CR

LPC2148 UM10139, page nos: 286 - 289

- Read_ADC0()
  - AD0CR
  - Infinite loop { check whether conversion over? If overrun, take right steps.

    Extract the digital output only and

    ignore the 2 LSBs

    return }

_

DAC C- Code

- Definitions
  - LEDs on PORT0, ADC_DONE, OVERRUN etc
- main
  - Infinite loop
  - ADC output with input ch2, throw away the 2 LSB bits
  - To get 8 bits displayed on 8 LEDs
- Init_ADC0()
  - PINSEL
  - AD0CR

LPC2148 UM10139, page nos: 286 - 289

- Read_ADC0()
  - AD0CR
  - Infinite loop { check whether conversion over? If overrun, take right steps.

    Extract the digital output only and

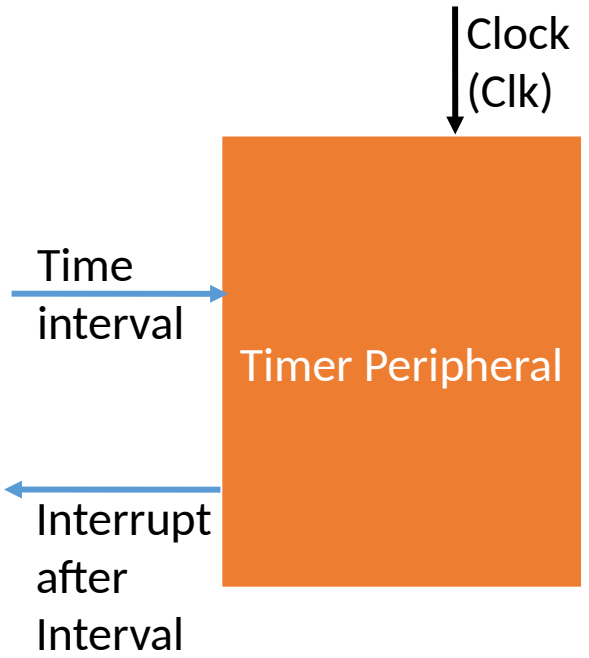
    ignore the 2 LSBs

    return }

_

Timer Peripheral

- One of the most common peripherals
  - Available on every microcontroller
- Used to execute something after a certain period of time
  - Takes in the time period, gives an interrupt when after given time interval
  - CLK could be processor CLK or its sub-multiple
- Use cases
  - Periodic tasks like polling
  - Scheduling a task
  - Controlling time intervals while scheduling positions in a stepper motor



Ref: Mazidi et al, page nos: 311 - 361

Timer versus Counter

- Timer / Counter needs clock to tick
- Clock sources can be internal or external
  - Internal clock source is usually crystal oscillator
    - Time delay generation
    - Stop watch (stop by external interrupt)
      - Time measurement between two random events in basic sciences experiments
  - By choosing external clock, we send (at random time) pulses through AVR pins --> counter
- S

Timer

- Methods to generate the time interval or measure time interval between two random events (stop watch)
  - 1st method: Time interval (or time delay) generation is by clearing the counter at the start and allow the counter to tick till the counter value reaches a value.
    - Comparator is required, with one input the threshold
  - 2nd method: muC's have flag register to flag the overflow of the counter. Set, a required value in the counter, allow it to overflow. 0xFD till 0xFF
- Usually the clock source is internal (crystal oscillator)

Timer Controller

- Does the job of setting the time interval in the timer by processor
- Can control multiple timers
- Commands indicated through Control Registers
  - Select Timer
  - Set timer interval
  - Set clock for timer
  - Reset Timer
- Data Register: Value to be set in timer, time to be set on clock
- Status
  - Which timer generated interrupt

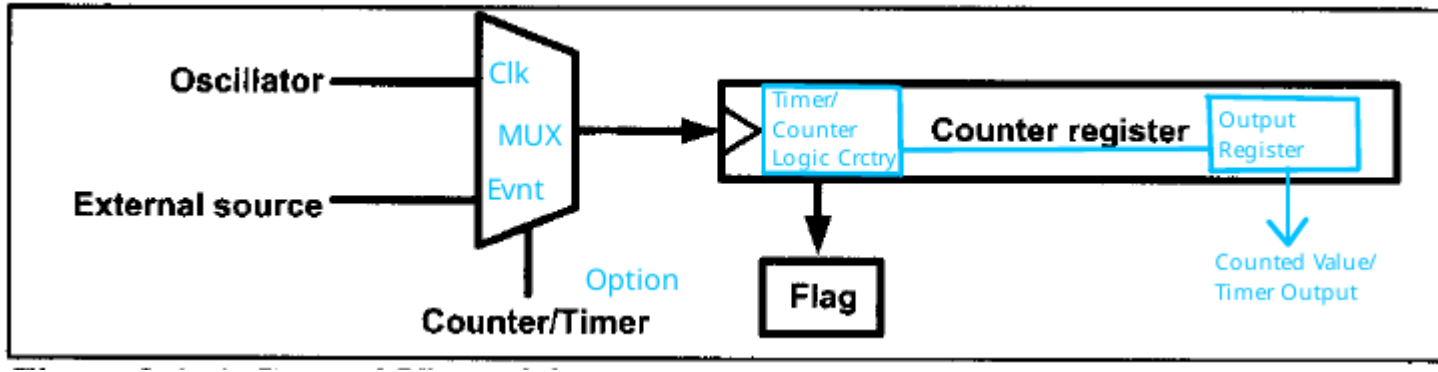
Application of Timers

- Watchdog timers
  - to detect and recover from computer malfunctions.
  - Automatic correction temporary hardware faults or to prevent errant or malevolent software from disrupting system operation
- Timer in assembly line in industries
  - Some mechanical processes are executed for a specified time interval → painting --> need timers
- Timer in automobile embedded systems (eg. ECU)
  - Used to read out a physical quantity say temperature of engine . To maintain consistency sensed for a prefixed interval (sample interval)
- Timers in Real-time applications and mission critical applications
  - eg. safety air bags in cars – after opening, after certain time, gas is released
  - (Difference between real-time & mission critical)

Application of Timers

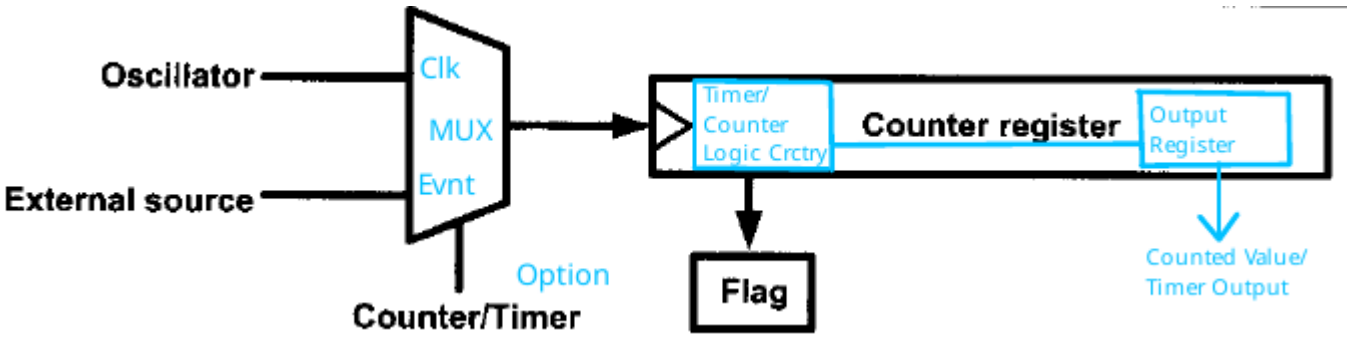
- Timers in Basic Sciences
  - to measure the time interval between two consecutive random events in many disciplines in Physics, Chemistry & Biology
  - To schedule data acquisition system initiation
- Timers in engineering
  -
-

Timers in AVR



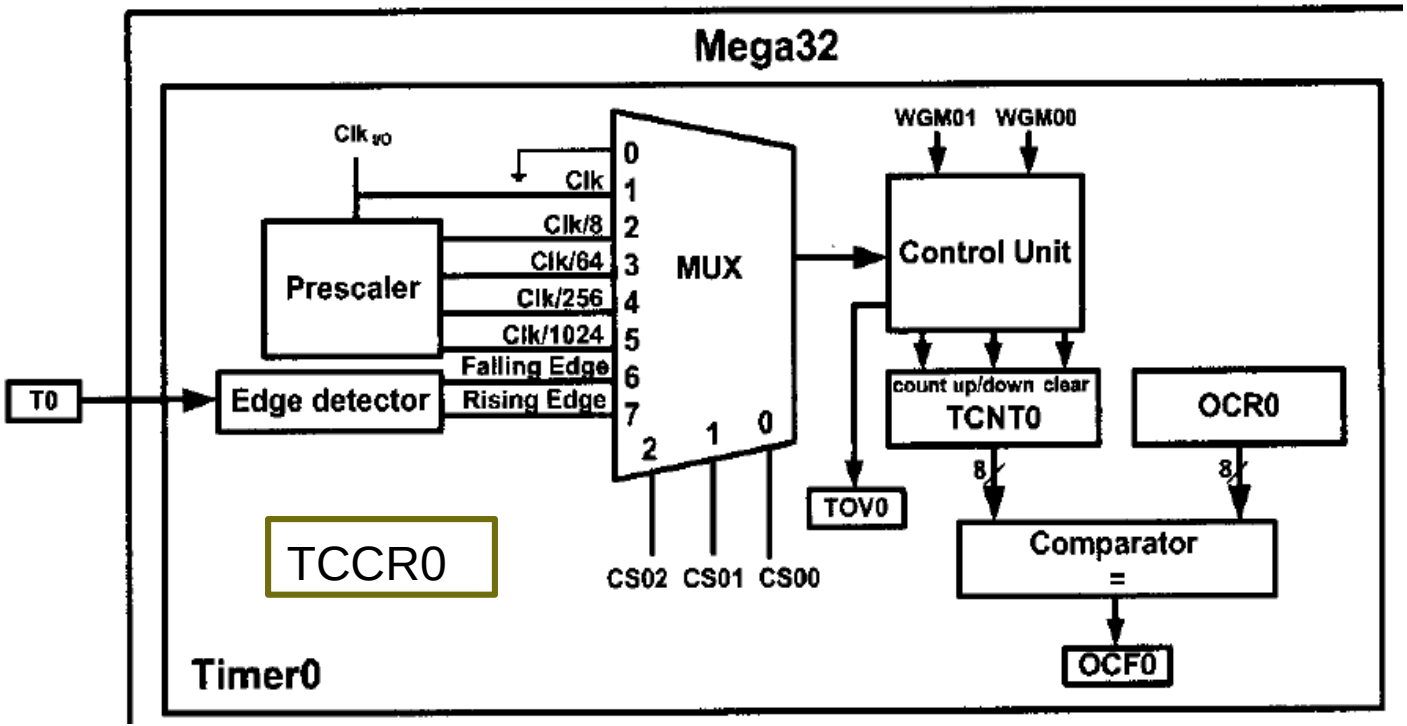
- 1 to 6 timers in AVR
- ATmega32 has 3 timers
  - Timer0 & timer2 has 8 bits
  - Timer1 is a16 bit timer
- Timers – Anatomy
  - MUX, Counter Registers, Flag
- Timers – Applications
  - Counters
    - (Often random) Events Source connected to clock input of Counter Register through MUX
  - Timers
    - Oscillator connected to clock input of Counter Register through MUX, generates a fixed, predetermined
-

Block diagram of Timer / Counter



Timers & Basic Registers of Timers in AVR

- Timer / Counter Register – TCNTn, n=0, 1, 2
  - Contents of Timer/ Counter can be accessed through TCNTn
  - Upon reset, 0x00
- Timer Overflow flag register - TOVn flag will be set when timer overflows
- TCCRn timer / counter control register
  - Sets modes of operation, timer / counter



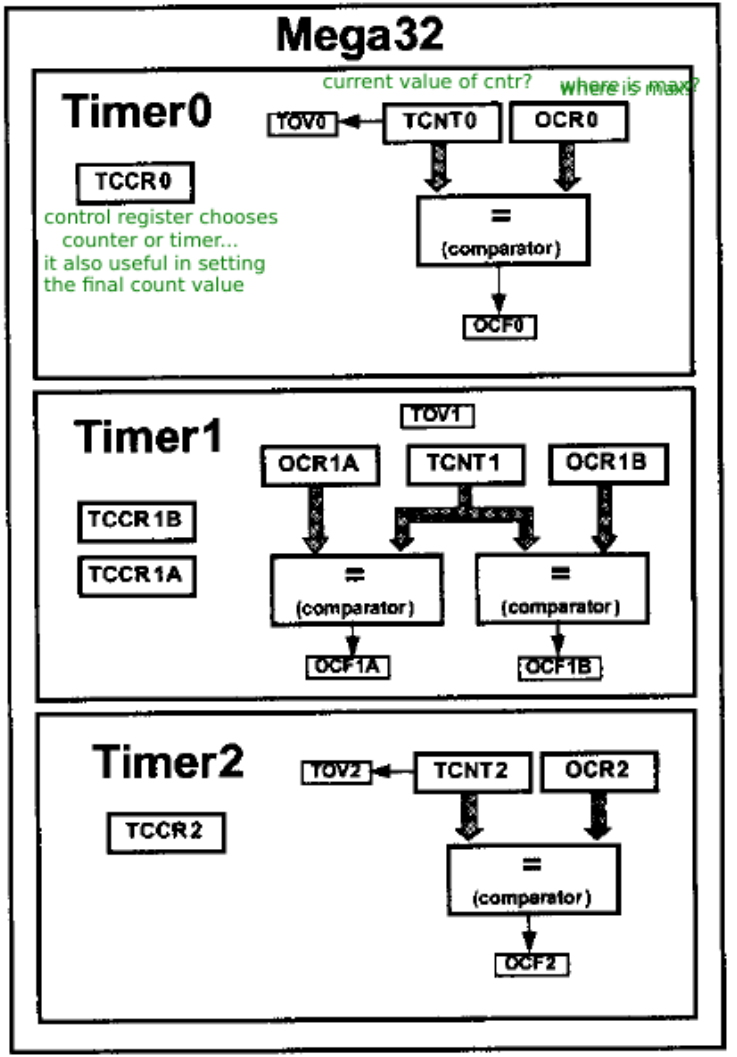
- Timer / Counter has – Output Compare Register, OCRn
  - Contents of OCRn are compared with TCNTn
- When above is equal, Output Compare Flag (OCFn)
- Timer registers are located in the I/O register memory.

Timers in AVR

- Three timer / counters in AVR TCNTn, n =0, 1, 2

TCNT0	D7	D6	D5	D4	D3	D2	D1	D0

- Timer / counter registers are located in the I/O register memory.
  - Read / Write using IN and OUT instructions
  - LDI R20, 25 ;
  - OUT TCNT0, R20 ;
- TCCRn Timer/ Counter Control Register n = 0,1,2



TCCRn Timer / Counter Control Register

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

- TCCRn Timer/ Counter Control Register n = 0,1,2
- D7 Force compare
- D6 D3 timer selector bits
- D5, D4
- D3

TCCRn Timer / Counter Control Register

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

CS02:00 — Timer0 clock selector

D2	D1	D0	
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	Internal clk (No Prescaling) internal cloack, normal mode, no prescaling
0	1	0	Internal clk / 8 till 101 normal mode
0	1	1	Internal clk / 64 precaling is same as frequency division?
1	0	0	Internal clk / 256
1	0	1	Internal clk / 1024 this means that the number
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.

TIFRn Timer / Counter Interrupt Flag Register

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

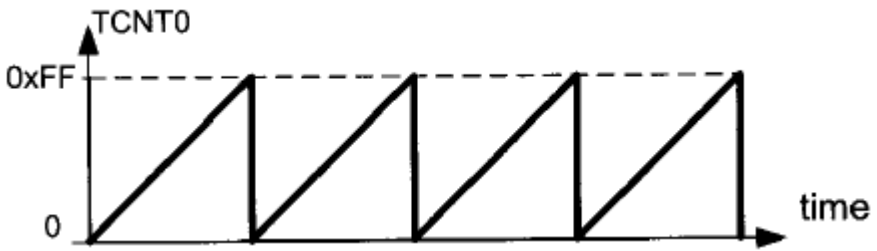
ICF1 D5 Input Capture flag

TOV2 D6 Timer2 overflow flag

OCF2 D7 Timer2 output compare match flag

TOV0 (Timer0 Overflow)

- Flag is set when Timer / counter overflows
  - Going from 0xFF to 0x00
    - TOV0 flag is set to 1 till it is cleared by software
    - To clear we need to write 1 to it
      - (applies to all flags of AVR?)
- To clear a given flag, write 1 & 0 to other bits
- LDI R20, 0x01
- OUT TIFR, R20
- In normal mode counts upto 0xFF, rolls over to 0x00 and starts all over again from 0x00 counts upto 0xFF
- Timer flag has to be monitored and the TCNTn follows the following graph



Square wave generation

```
.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
```

Square wave generation

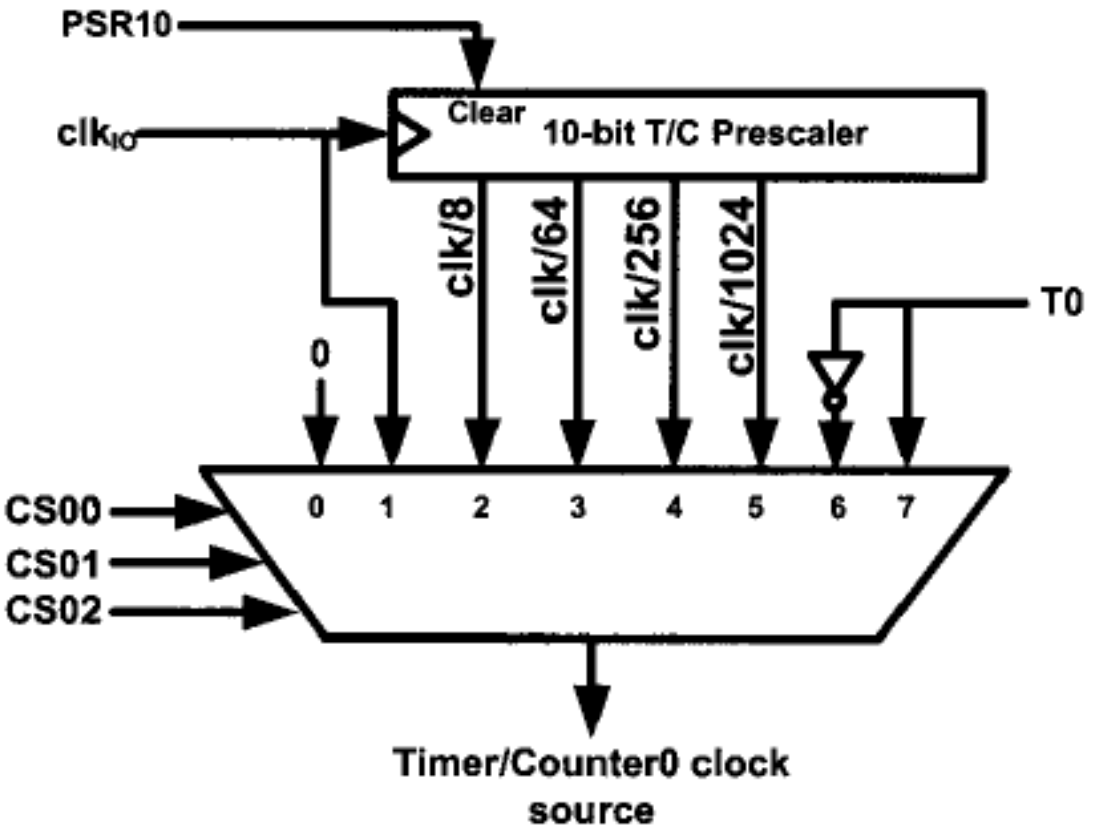
```
;--------------------Time0 delay
DELAY:LDI    R20,0xF2     ;R20 = 0xF2
      OUT    TCNT0,R20    ;load timer0
      LDI    R20,0x01
      OUT    TCCR0,R20    ;Timer0, Normal mode, int clk, no prescaler
AGAIN:IN     R20,TIFR     ;read TIFR
      SBRS   R20,TOV0     ;if TOV0 is set skip next instruction
      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
```

- Calculate the frequency of the square wave in example above, generated on pin PORTB.5?
  - Xtal frequency = 8 MHz
    - Given the clock cycles of each instruction as given under
- S
- S

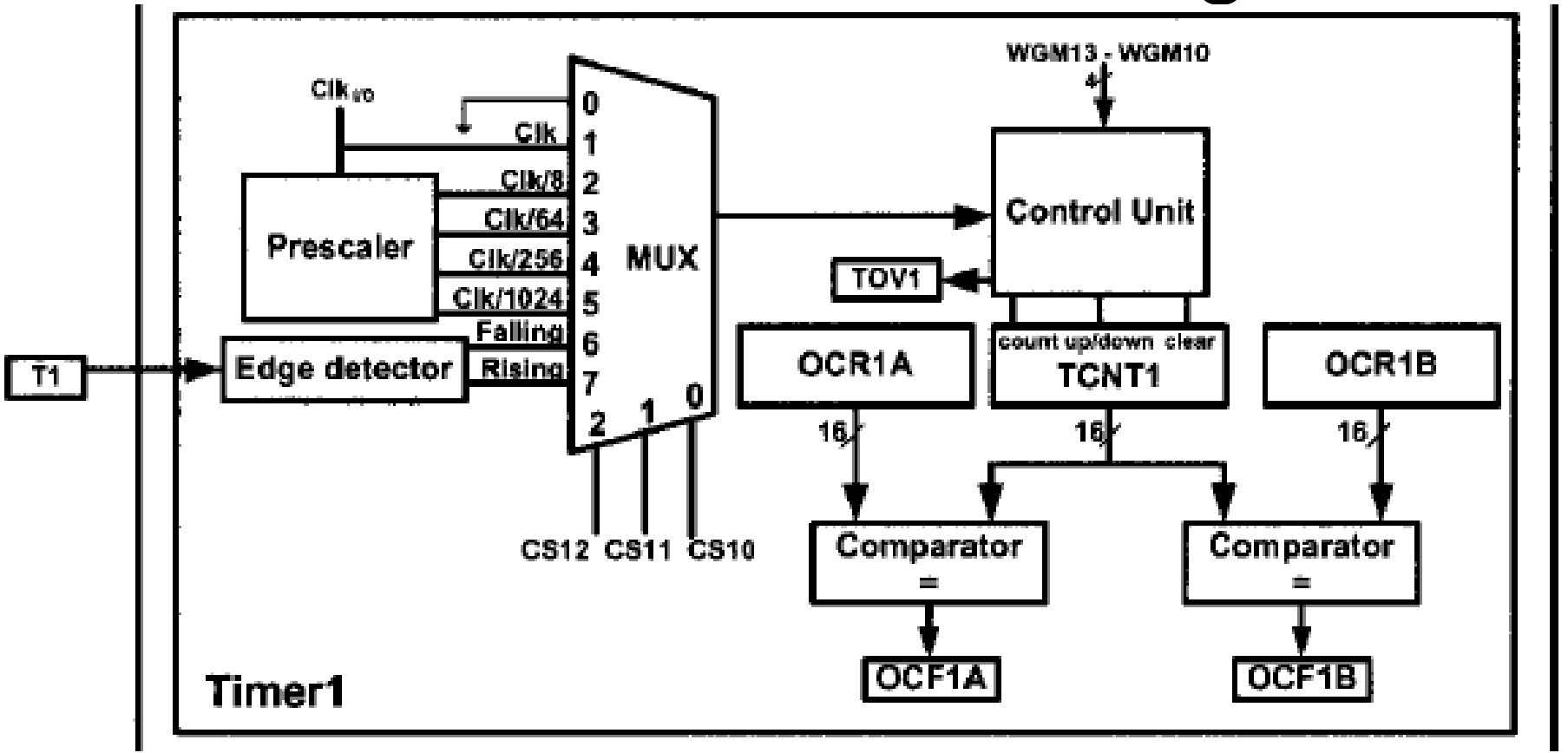
```
                                          Cycles
        LDI   R16,0x20
        SBI   DDRB,5
        LDI   R17,0
        OUT   PORTB,R17
BEGIN:  RCALL DELAY                         3
        EOR   R17,R16                       1
        OUT   PORTB,R17                     1
        RJMP  BEGIN                         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
```

$T = 2 \times (14 + 24) \times 0.125\ \mu s = 9.5\ \mu s$ and $F = 1 / T = 105.263$ kHz.

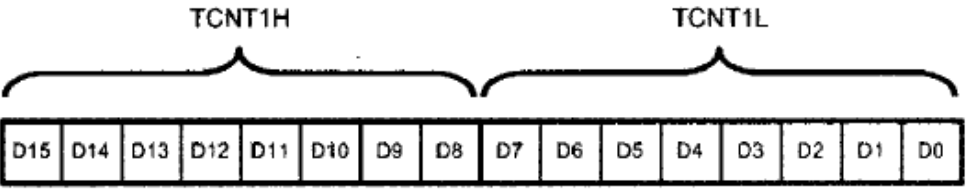
Prescaler

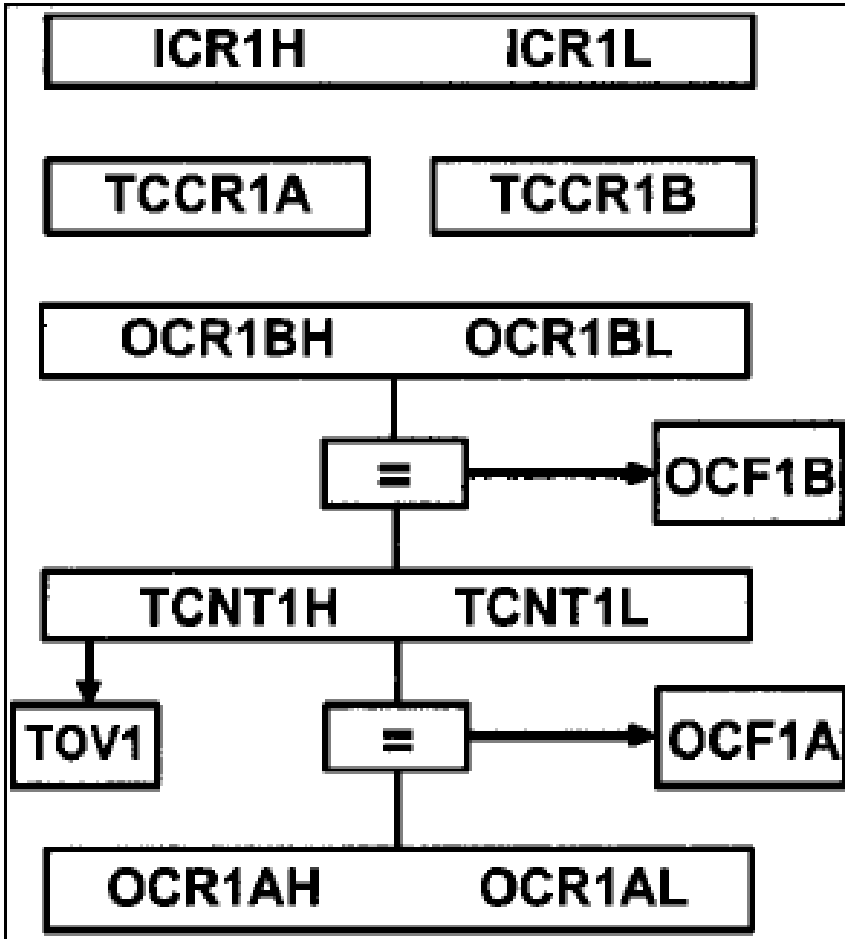

PSR10
clk$_{IO}$
Clear
10-bit T/C Prescaler
clk/8
clk/64
clk/256
clk/1024
T0
0
CS00
CS01
CS02
0 1 2 3 4 5 6 7
Timer/Counter0 clock source

16-bit Timer1 in ATmega32



16-bit timer


TCNT1H
TCNT1L
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0


ICR1H ICR1L
TCCR1A
TCCR1B
OCR1BH OCR1BL
=
OCF1B
TCNT1H TCNT1L
TOV1
=
OCF1A
OCR1AH OCR1AL

Counter Programming

- AVR timer/counter can be configured to count (the random) events (number of such events) happening in the real-world (natural sciences, engineering etc)
  - When AVR timer/counter is used as counter, external pulse is used to drive the unit (incrementing TCNTx register)
    - TCCR, OCR and TCNT are same as in timer
    - Configuration input to TCCR is different
- Connection to local clock (xtal) is removed, while external pulse generator (in response to a sensor, sensing the engineering quantity)
  - The above is done, by choosing CS02 – CS00 is 6 or 7.
- S

Counter Programming: 9:35

```
.INCLUDE "M32DEF.INC"
      CBI   DDRB,0            ;make T0 (PB0) input
      LDI   R20,0xFF
      OUT   DDRC,R20          ;make PORTC output
      LDI   R20,0x06
      OUT   TCCR0,R20         ;counter, falling edge
AGAIN:
      IN    R20,TCNT0
      OUT   PORTC,R20         ;PORTC = TCNT0
      IN    R16,TIFR
      SBRS  R16,TOV0          ;monitor TOV0 flag
      RJMP  AGAIN             ;keep doing if Timer0 flag is low
      LDI   R16,1<<TOV0
      OUT   TIFR, R16         ;clear TOV0 flag
      RJMP  AGAIN             ;keep doing it
```

Programming Timer Interrupts

- Chap 9 Timers 0, 1, 2 with the polling method
  - SBRS R20, TOV0
  - MuP is tied up to polling, cant do anything else including mother programming
- Chap 10 interrupts to program the AVR timers
  - TOV0 is raised whenever timer rolls over and muC jumps to IVT to serve the ISR.
  - MuP can do mother program untill it is notified of the roll over
- Procedure for programming Timer interrupts
  - Enable timer interrupts
  - TOIEx bit enables the interrupt for a given timer
- Examples 9:36, 9:37, 9:38 in Mazidi

Timer Interrupt in Mazidi

```
                .INCLUDE "M32DEF.INC"
0               .ORG 0x0000                     ; reset location
1               JMP     MAIN
2               .ORG 0x16                       ; Timer0 overflow location
3               JMP     T0_OV_ISR
4               ; main program for initialization and keeping CPU busy
5               .ORG 0x100
6 MAIN:         LDI             R20, HIGH(RAMEND); initialise stack
7               OUT             SPH, R20
8               LDI             R20, LOW(RAMEND)
9               OUT             SPL, R20        ; initialization over
10              SBI             DDRB, 5         ; PB5 as an output
11              LDI             R20, (1<<TOIE0) ; Tmr enble bit TIMSK p 368 register
12              OUT             TIMSK, R20      ; enble Timer0 overflow interrupt
13              SEI                             ; set I (enable interrupts globally)
14              LDI             R20, -32        ; timer value for 4 microsec
15              OUT             TCNT0, R20      ; load Timer0 with -32
16              LDI             R20, 0x01       ; TCCR0 p315
```

Counter Examples in Mazidi

```
17              OUT             TCCR0, R20      ; Normal, cnfgrsAsTmr-NtCntr, noPreSclr,
18              LDI             R20, 0x00       ;
19              OUT             DDRC, R20       ; make PORTC input
20              LDI             R20, 0xFF       ;
21              OUT             DDRD, R20       ;makes PORTD output
22              ; ---------- infinite loop --------------
23 HERE         IN              R20, PINC       ;read from PORTC
24              OUT             PORTD, R20      ;give it to PORTD
25              JMP             HERE            ;keeping CPU busy waiting for interrupt
26              ;-----ISR for Timer0 (executed every 4 musec) ---------
27              .ORG 0x200
28 T0_OV_ISR:
29              IN              R16, PORTB      ;read PORTB
30              LDI             R17, 0x20       ;00100000 for toggling PB5
31              EOR             R16, R17        ;
32              OUT             PORTB, R16      ; toggle PB5
33              LDI             R16, -32        ; timer value for 4 muSec
34              OUT             TCNT0, R16      ; load Timer0 with -32 (for next round)
35              RETI                            ; return from interrupt
```

Problem

- Rewrite the AVR program Ex 10.1 in Mazidi
  - In which the interrupt is replaced by polling
  - Use SBRS TOV0 for polling
- Evolve the flow chart first.
- Then code it
- Check the time taken by both programs (which do the same task (a) port C to port D for ever (b) generate square wave in PORTD.5)