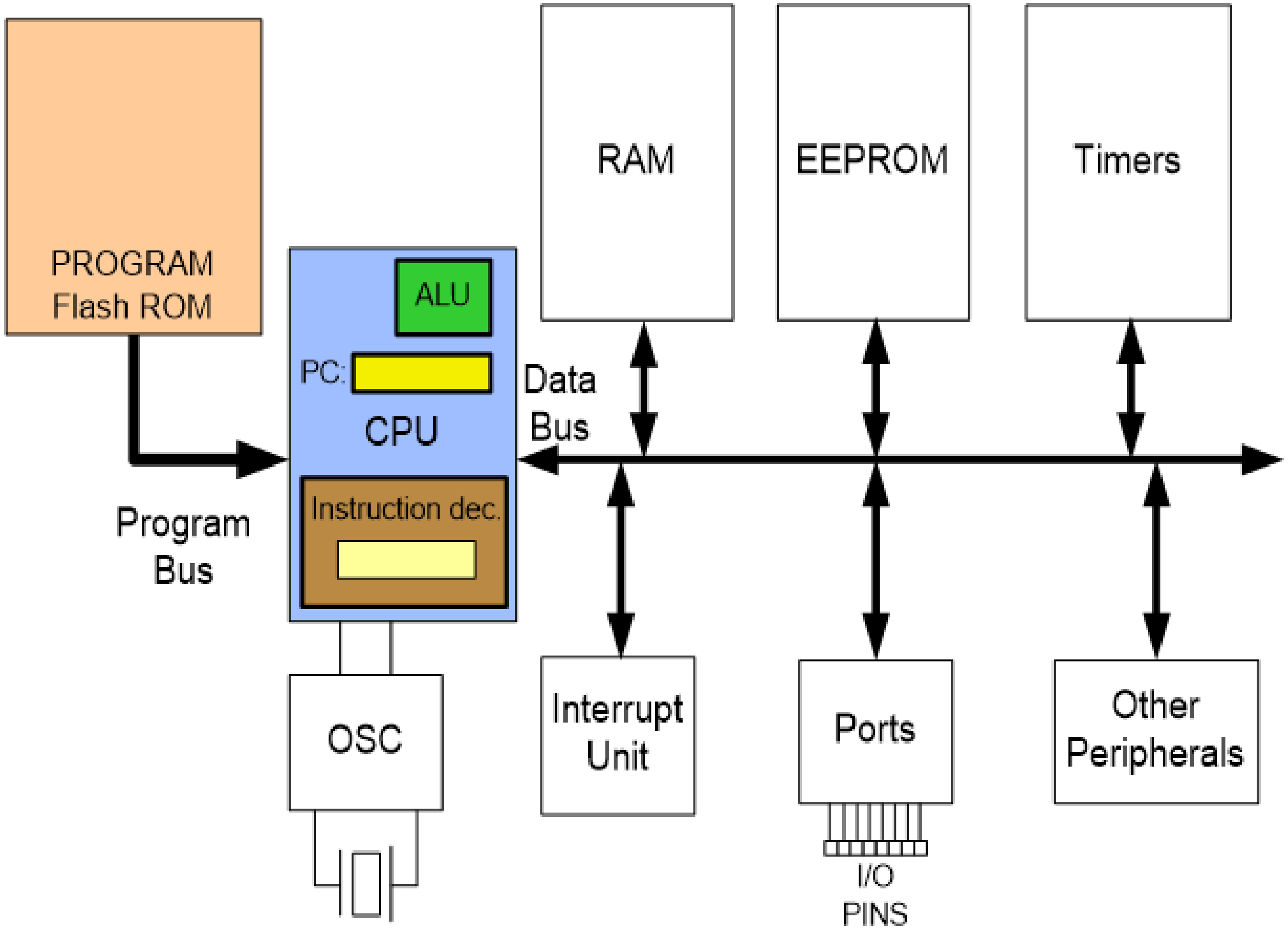
EE-222: Microprocessor Systems

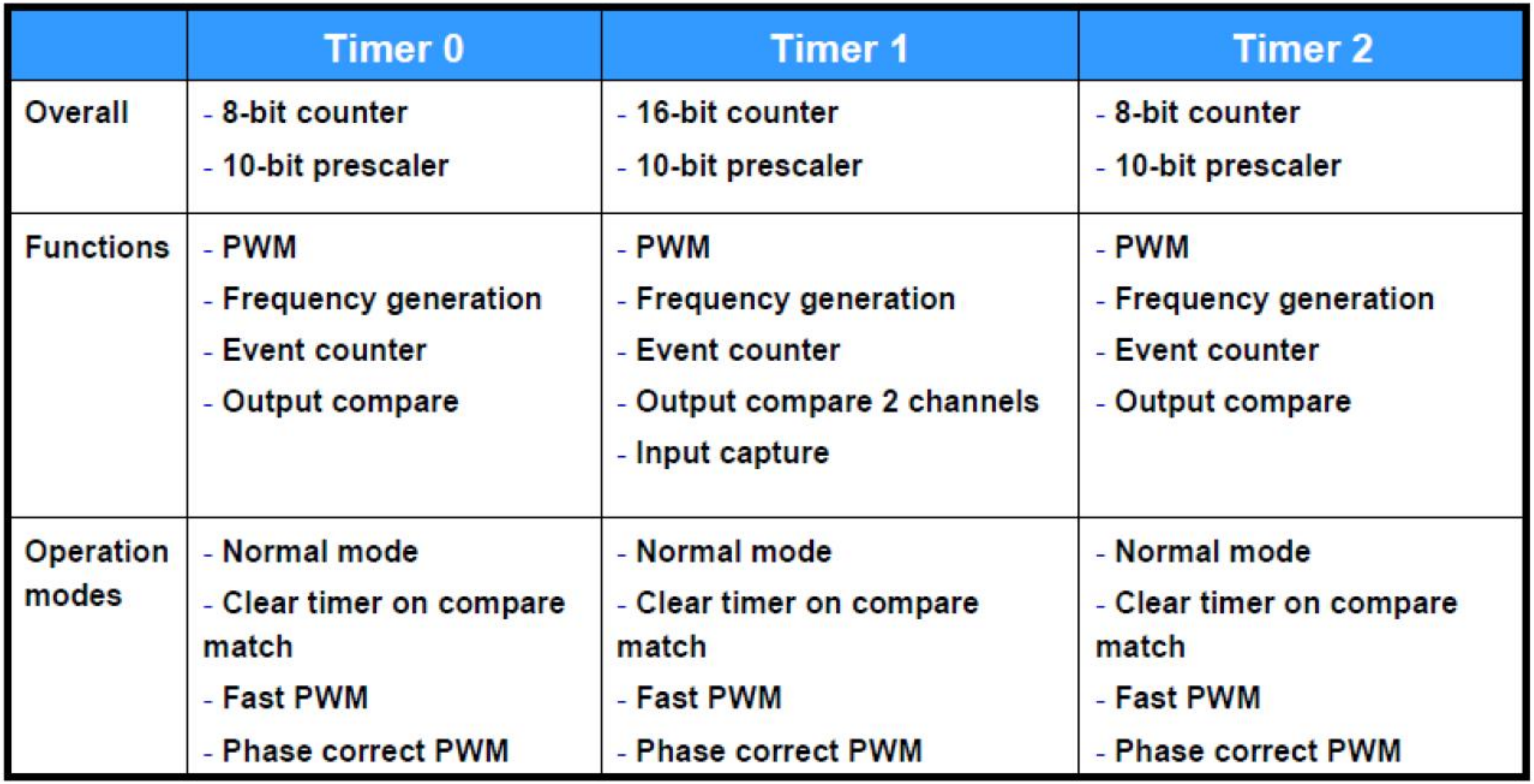
AVR Timers

Instructor: Dr. Arbab Latif

Timers


PROGRAM
Flash ROM
Program
Bus
ALU
PC:
CPU
Instruction dec.
OSC
Data
Bus
RAM
EEPROM
Timers
Interrupt
Unit
Ports
I/O
PINS
Other
Peripherals

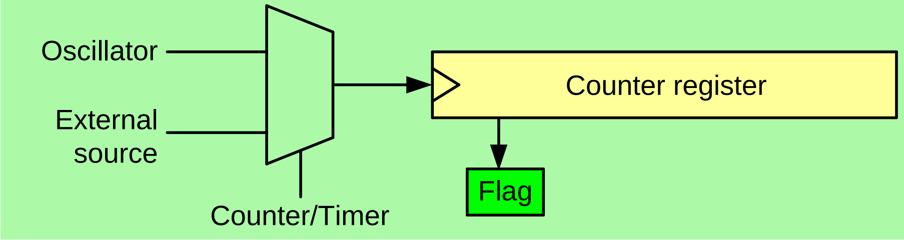
Overview of Atmega16 Timers

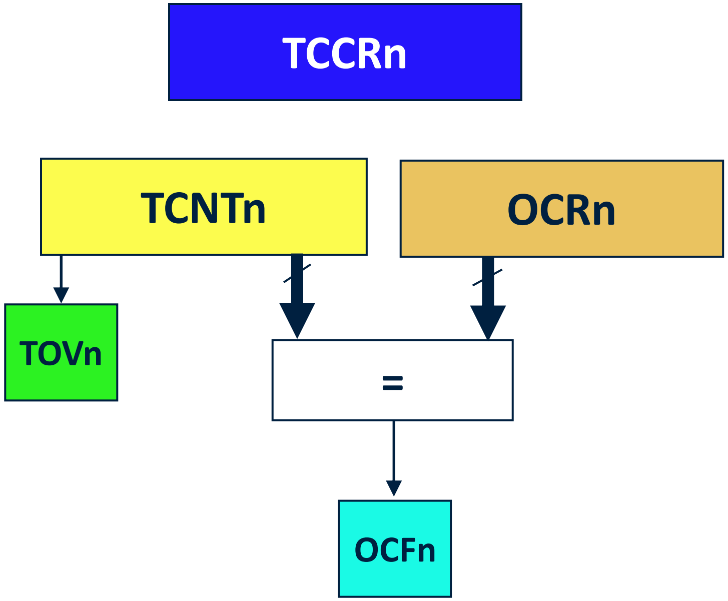


	Timer 0	Timer 1	Timer 2
Overall	- 8-bit counter - 10-bit prescaler	- 16-bit counter - 10-bit prescaler	- 8-bit counter - 10-bit prescaler
Functions	- PWM - Frequency generation - Event counter - Output compare	- PWM - Frequency generation - Event counter - Output compare 2 channels - Input capture	- PWM - Frequency generation - Event counter - Output compare
Operation modes	- Normal mode - Clear timer on compare match - Fast PWM - Phase correct PWM	- Normal mode - Clear timer on compare match - Fast PWM - Phase correct PWM	- Normal mode - Clear timer on compare match - Fast PWM - Phase correct PWM

Timer Registers

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)

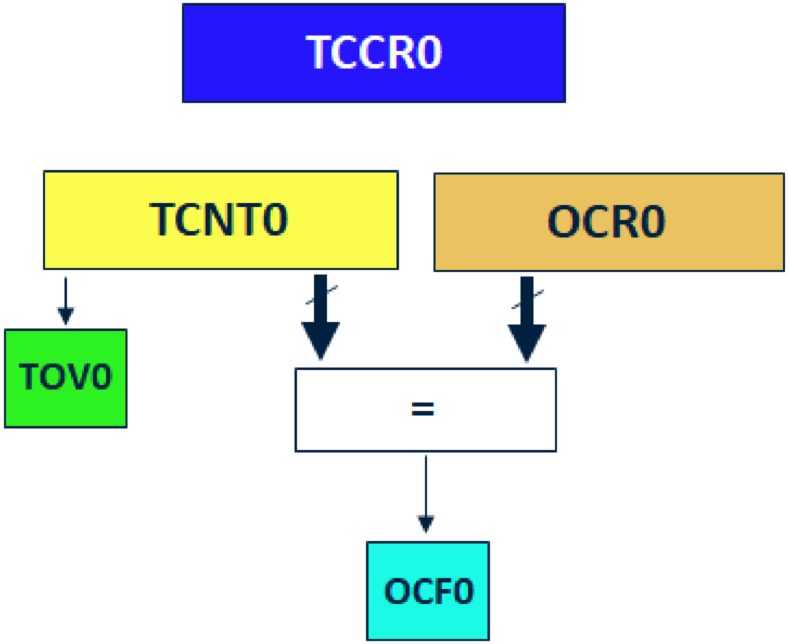




Programming Timer 0

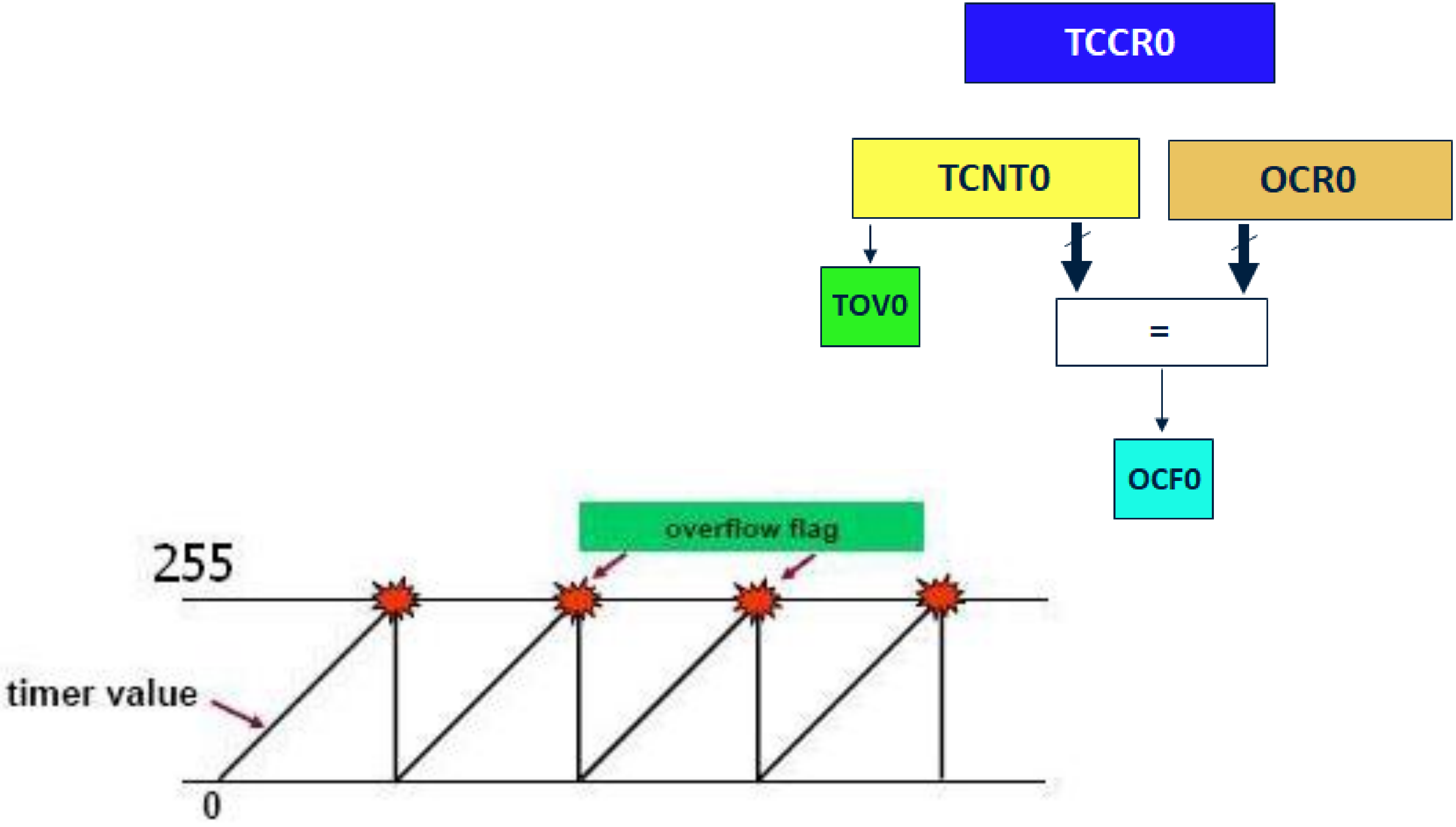
TCNT0 Register

- TCNT0 [Timer/Counter] Register:
  - R/W
  - ZERO upon RESET
  - Contents of timer/counter can be accessed through this register.



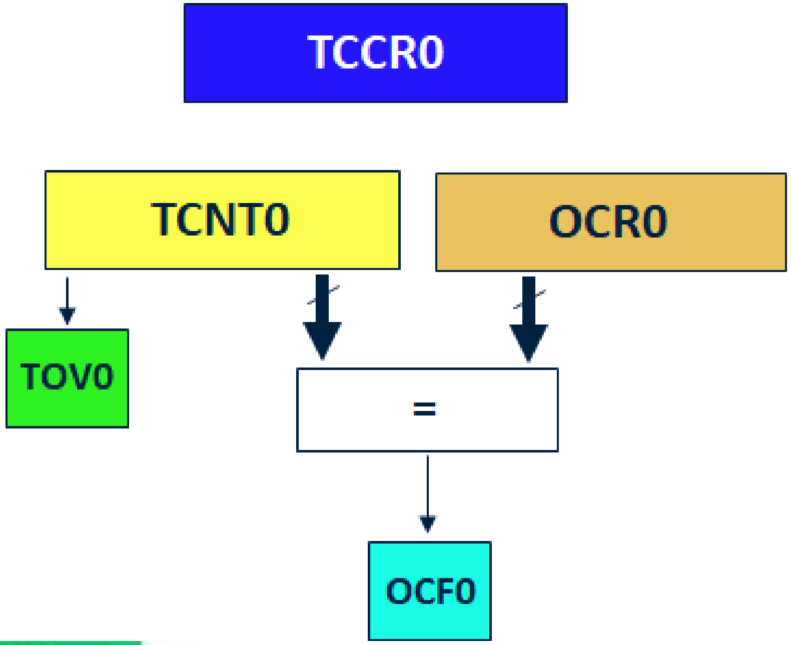
TOV0 Flag

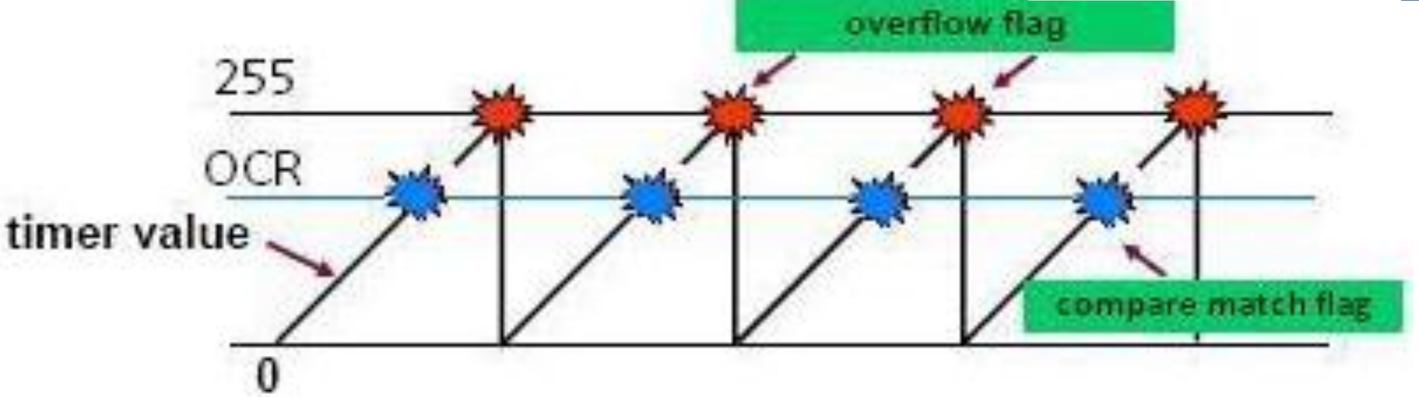
- TOV0 [Timer Overflow] Flag Register:
  - TOV0 sets, when a timer overflows



OCR0 Flag

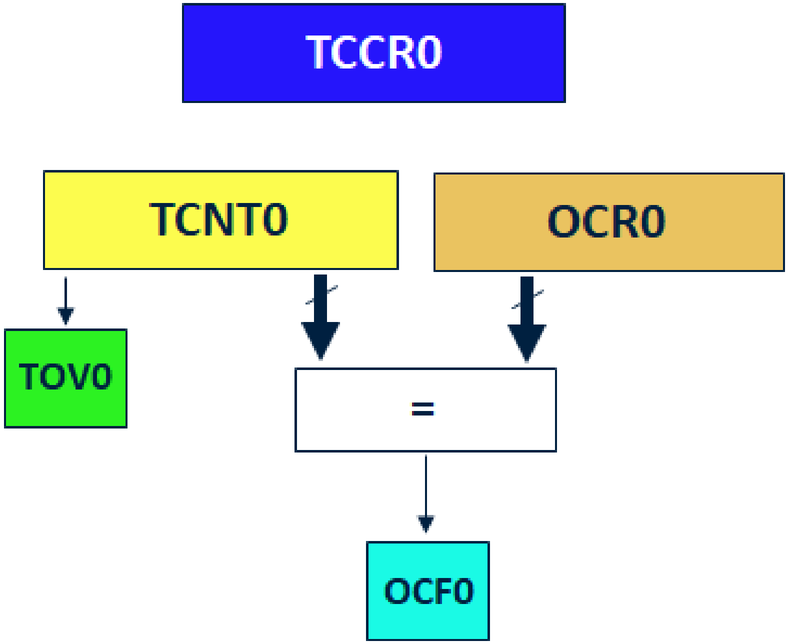
- OCR0 [Output Compare] Flag:
  - The content of the OCR0 is compared with the contents of the TCNT0
    - OCR0 flag is set when both are equal.





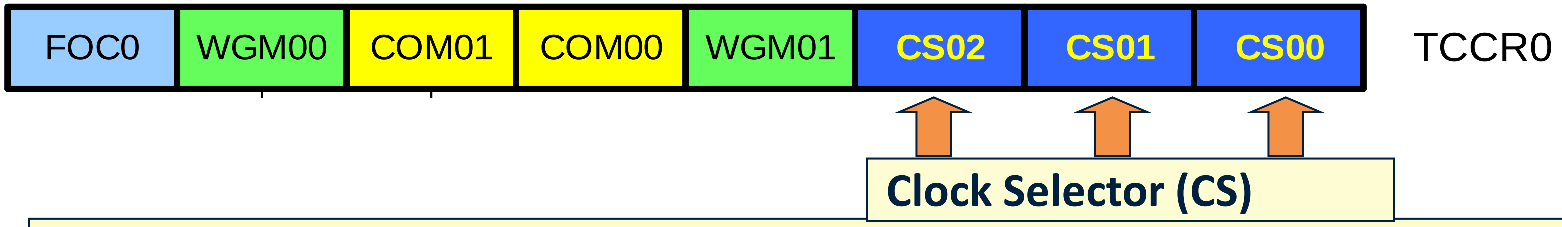
TCCR0 Register

- TCCR0 [Timer/Counter Control] Register:
  - Used for various settings (see next)



FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**	TCCR0

TCCR0: Clock Selector



FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0

Clock Selector (CS)

CS02	CS01	CS00	Comment
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

TCCR0: Mode Selector

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

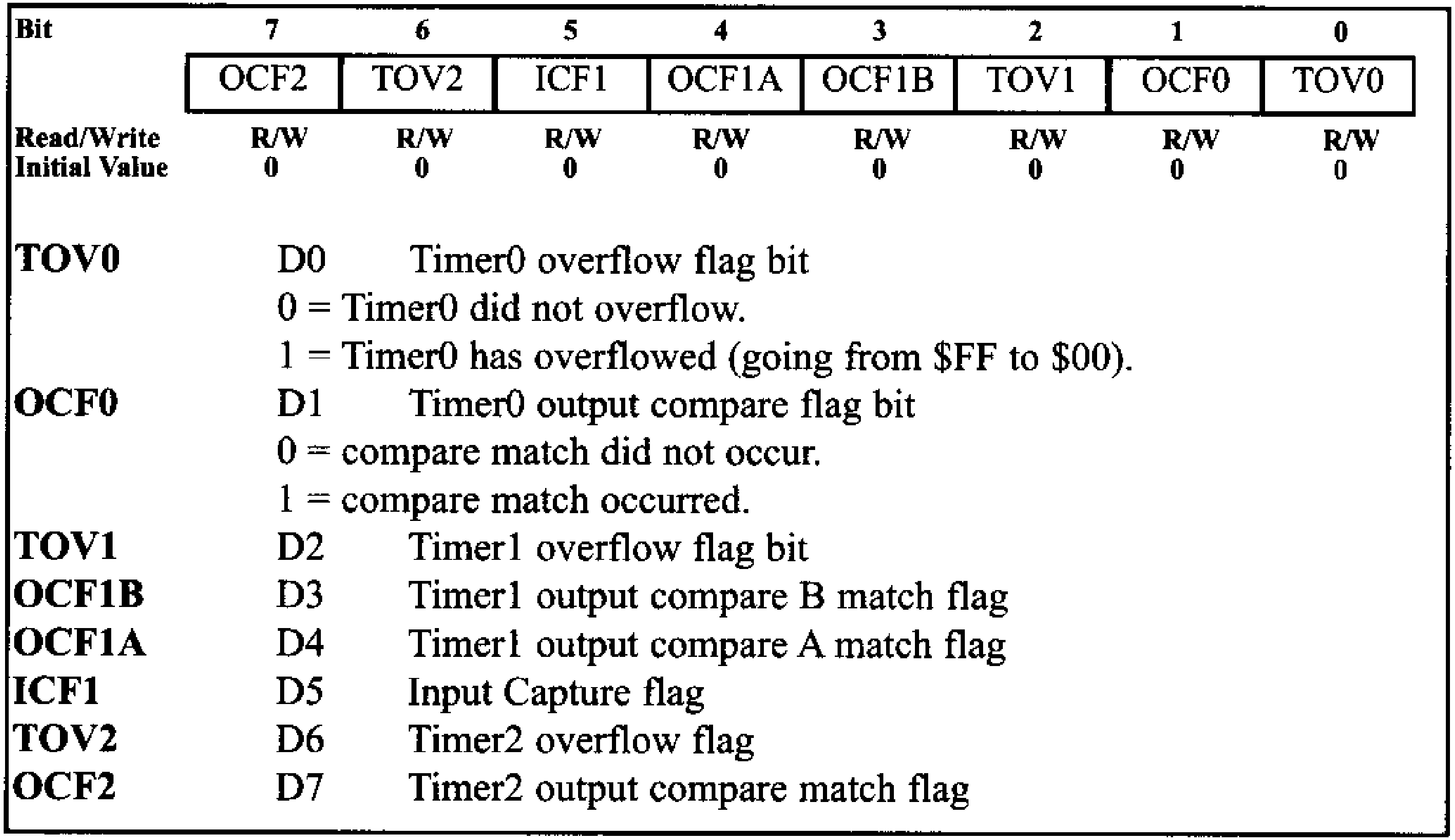
TCCR0

Timer Mode (WGM)

WGM00	**WGM01**	**Comment**
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	Phase correct PWM
1	1	Fast PWM

TIFR (Timer/Counter Interrupt Flag Register)

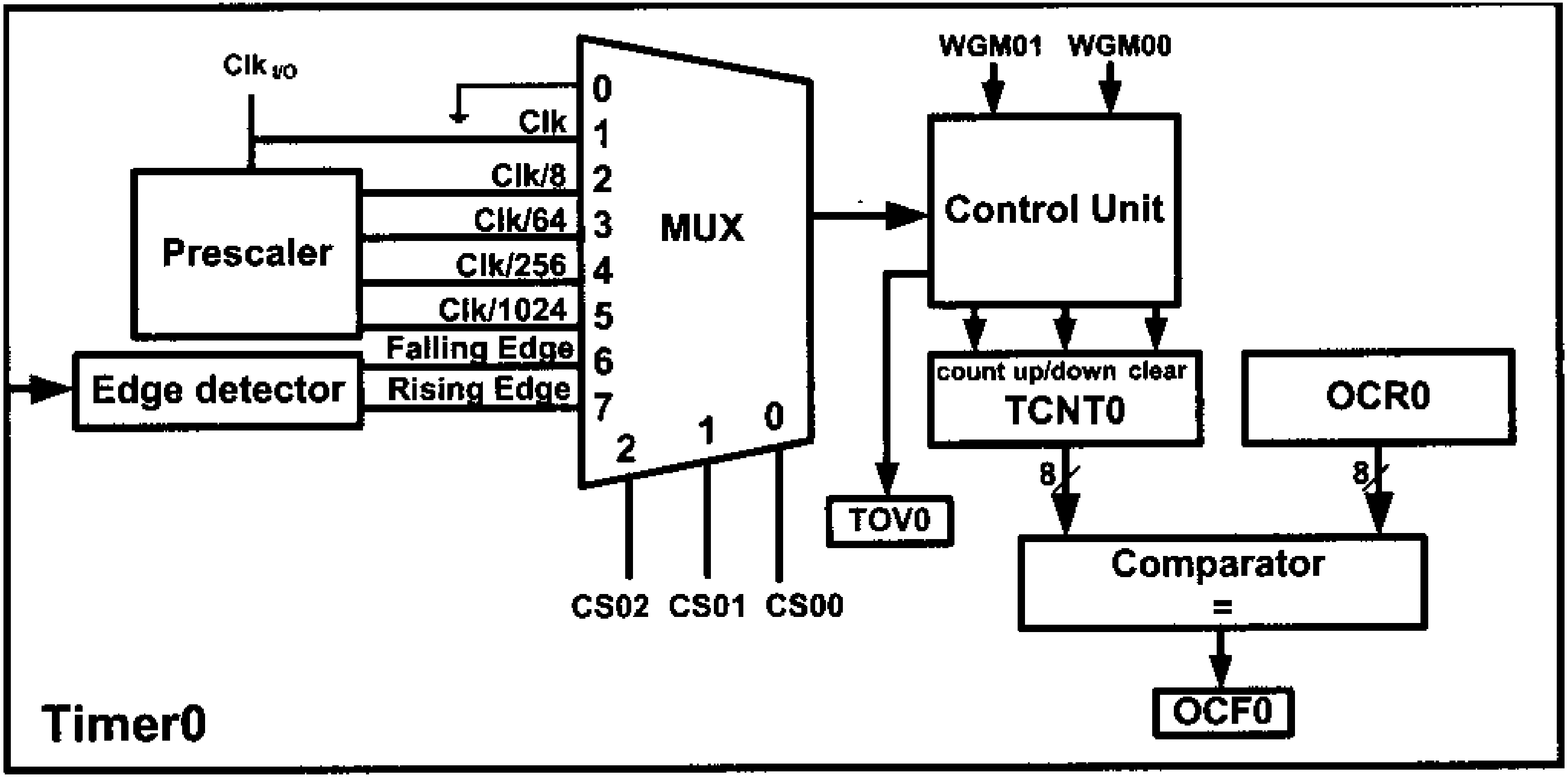
- TOV0 and OCF0 are part of TIFR register



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

Overall: Timer 0 Hardware Organization



Steps to Program Timer 0 in Normal Mode

Steps to Program Timer0 in Normal Mode

1. Load the TCNT0 with the initial count value.

2. Configure timer/counter mode through TCCR0 register.

3. Keep monitoring the timer overflow flag (TOV0):
   - Get out of the loop when TOV0 becomes high

4. Stop the timer by disconnecting the clock source:
   - LDI R20, 0x00
   - TCCR0,R20

5. Clear the TOV0 flag for the next round.

6. Go back to Step 1 to load TCNT0 again.

Timer 0 Demo

1. Load the TCNT0
2. Configure TCCR0 register
3. Monitor TOV0
4. Stop the timer
5. Clear the TOV0

```c
#include <avr/io.h>

int main()

{

 TCNT0 = 0xF2;

 TCCR0 = 0x01;     //WGM=0000 (Normal)

 while ((TIFR & (1 << TOV0)) ==0)
//wait for TOV0 to roll over

 TCNT0 = 0;

 TIFR = 0x01;

}
```

```
LDI R20, 0xF2
OUT TCNT0, R20

LDI R20, 0x01
OUT TCCR0, R20

AGAIN: IN R20,TIFR
       SBRS R20,TOV0
       RJMP AGAIN


LDI R20,0x0
OUT TCCR0,R20

LDI R20,0x01
OUT TIFR, R20
```

In example 1 calculate the delay. XTAL = 10 MHz.

Solution 1 (inaccurate)**:**

1) **Calculating T:**

T = 1/f = 1/10M = 0.1µs

2) **Calculating num of machine cycles:**

```
 $100
-$F2
 ----
  $0E = 14
```

3) **Calculating delay**

14 * 0.1µs = 1.4 0µs

```
         LDI      R16,0x20
         SBI      DDRB,5   ;PB5 as an output
         LDI      R17,0
         OUT      PORTB,R17
BEGIN:   LDI      R20,0xF2
         OUT      TCNT0,R20         ;load timer0
         LDI      R20,0x0
         OUT      TCCR0A,R20
         LDI      R20,0x01
         OUT      TCCR0B,R20 ;Normal mode, inter. clk
AGAIN:   SBIS     TIFR0,TOV0 ;if TOV0 is set skip next
         RJMP     AGAIN
         LDI      R20,0x0
         OUT      TCCR0B,R20        ;stop Timer0
         LDI      R20,(1<<TOV0)     ;R20 = 0x01
         OUT      TIFR0,R20         ;clear TOV0 flag

         EOR      R17,R16           ;toggle D5 of R17
         OUT      PORTB,R17         ;toggle PB5
         RJMP     BEGIN
```

Accurate calculating

Other than timer, executing the instructions consumes time; so if we want to calculate the accurate delay a program causes we should add the delay caused by instructions to the delay caused by the timer

```
          LDI       R16,0x20
          SBI       DDRB,5
          LDI       R17,0
          OUT       PORTB,R17
BEGIN:    LDI       R20,0xF2                          1
          OUT       TCNT0,R20                         1
          LDI       R20,0x00                          1
          OUT       TCCR0A,R20                1
          LDI       R20,0x01                          1
          OUT       TCCR0B,R20                1
AGAIN:    SBIS      TIFR0,TOV0                        1 / 2
          RJMP      AGAIN                             2
          LDI       R20,0x0                           1
          OUT       TCCR0B,R20                1
          LDI       R20,0x01                          1
          OUT       TIFR0,R20                         1
          EOR       R17,R16                           1
          OUT       PORTB,R17                         1
          RJMP      BEGIN                             2
                                                    18
```

Delay caused by timer = 14 * 0.1µs = 1.4 µs **Delay caused by instructions = 18 * 0.1µs = 1.8**

Total delay = 3.2 µs ➔ wave period = 2*3.2 µs = 6.4 µs ➔ wave frequency = 156.25 KHz

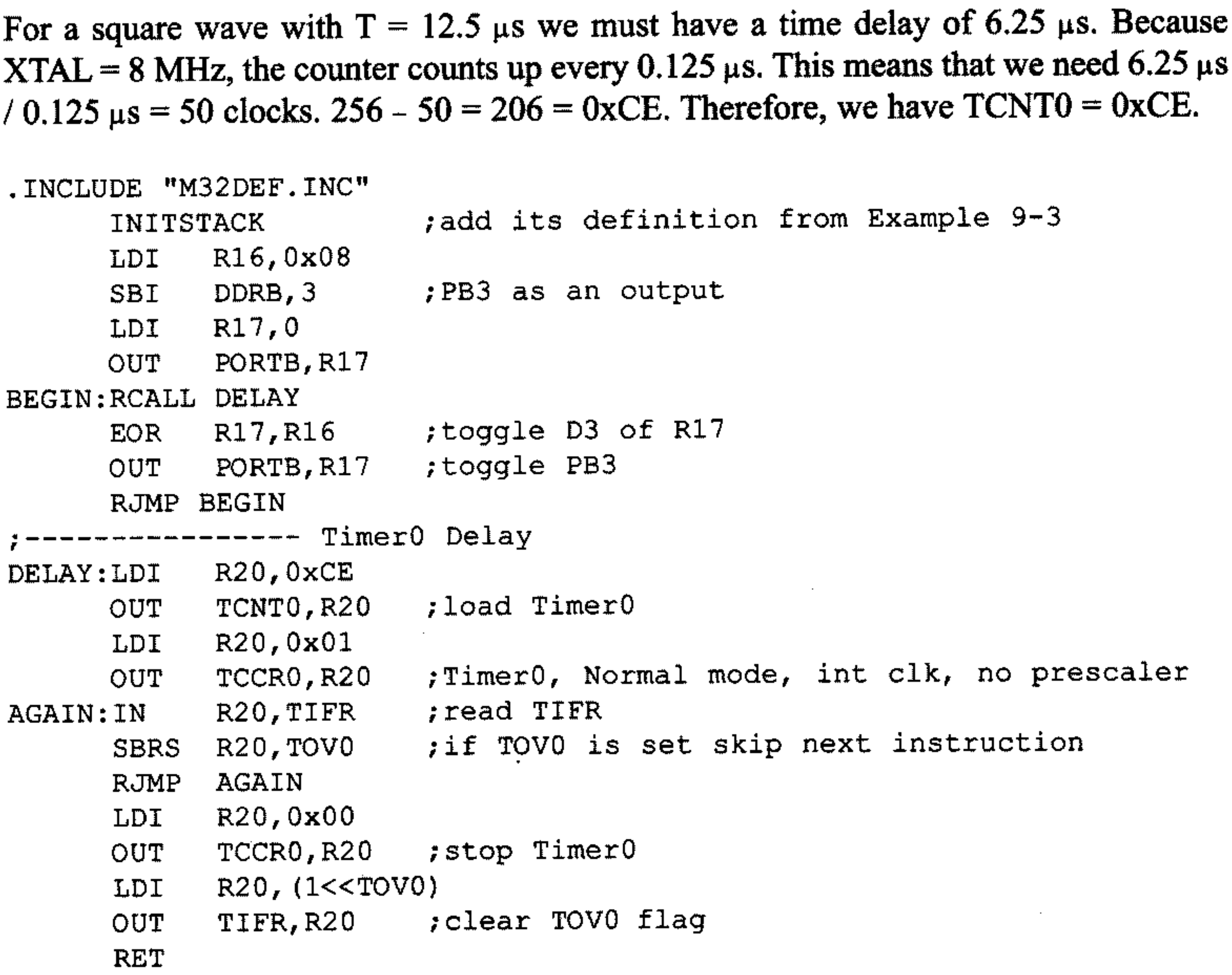
Finding values to be loaded into the Timer

1. Calculate the period of clock source.
   - Period = 1 / Frequency
     - E.g. For XTAL = 16 MHz ➔ T = 1/16MHz
2. Divide the desired time delay by period of clock.
3. Perform 256 - n, where n is the decimal value we got in Step 2.
4. Set TCNT0 = 256 - n

Example

- Assuming XTAL = 8 Mhz, write a program to generate a square wave with a period of 12.5 us on PIN PORTB.3.

Solution



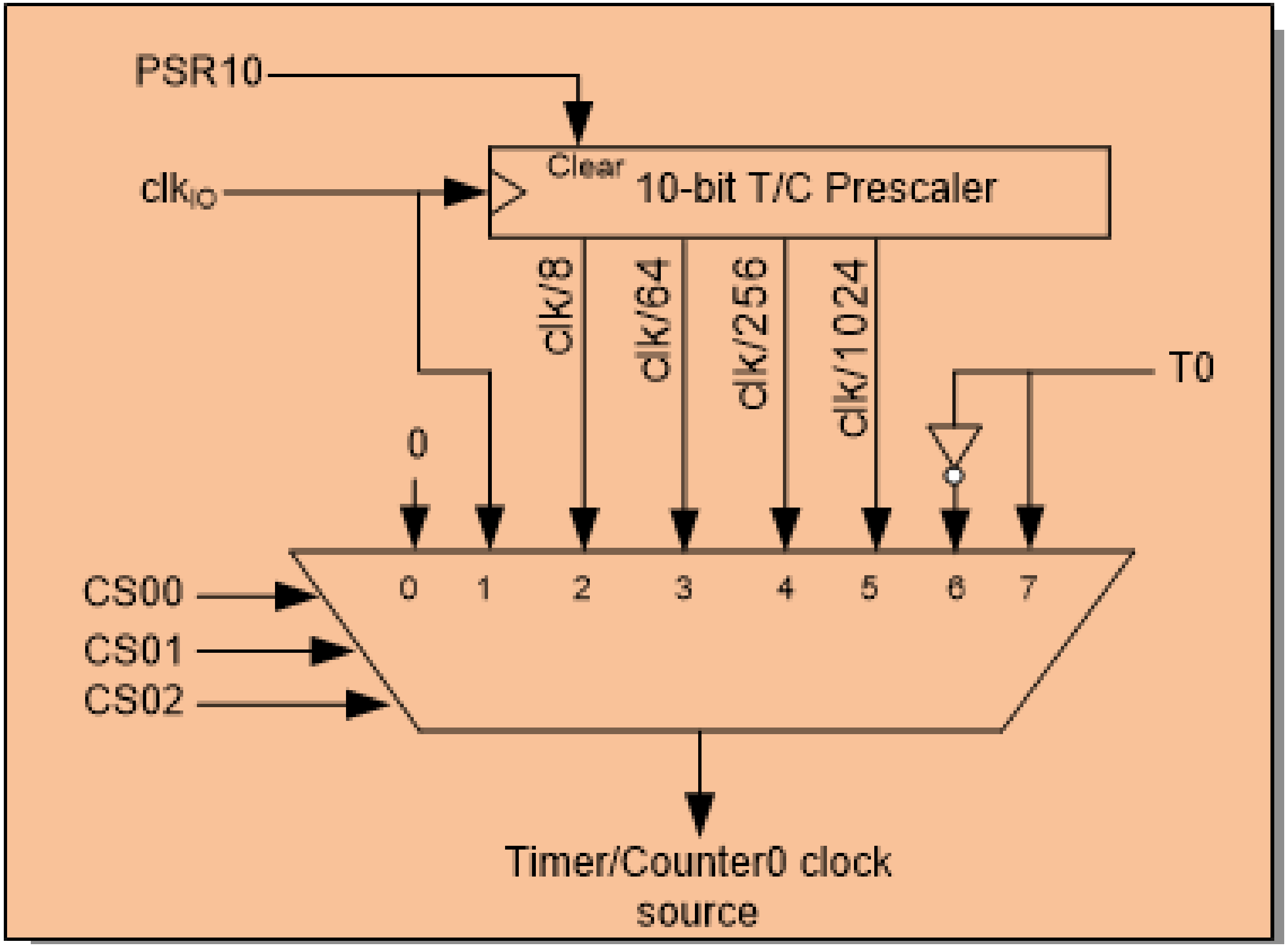
For a square wave with T = 12.5 µs we must have a time delay of 6.25 µs. Because XTAL = 8 MHz, the counter counts up every 0.125 µs. This means that we need 6.25 µs / 0.125 µs = 50 clocks. 256 – 50 = 206 = 0xCE. Therefore, we have TCNT0 = 0xCE.

```
.INCLUDE "M32DEF.INC"
      INITSTACK         ;add its definition from Example 9-3
      LDI   R16,0x08
      SBI   DDRB,3      ;PB3 as an output
      LDI   R17,0
      OUT   PORTB,R17
BEGIN:RCALL DELAY
      EOR   R17,R16     ;toggle D3 of R17
      OUT   PORTB,R17   ;toggle PB3
      RJMP BEGIN
;---------------- Timer0 Delay
DELAY:LDI   R20,0xCE
      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   R20,0x00
      OUT   TCCR0,R20   ;stop Timer0
      LDI   R20,(1<<TOV0)
      OUT   TIFR,R20    ;clear TOV0 flag
      RET
```

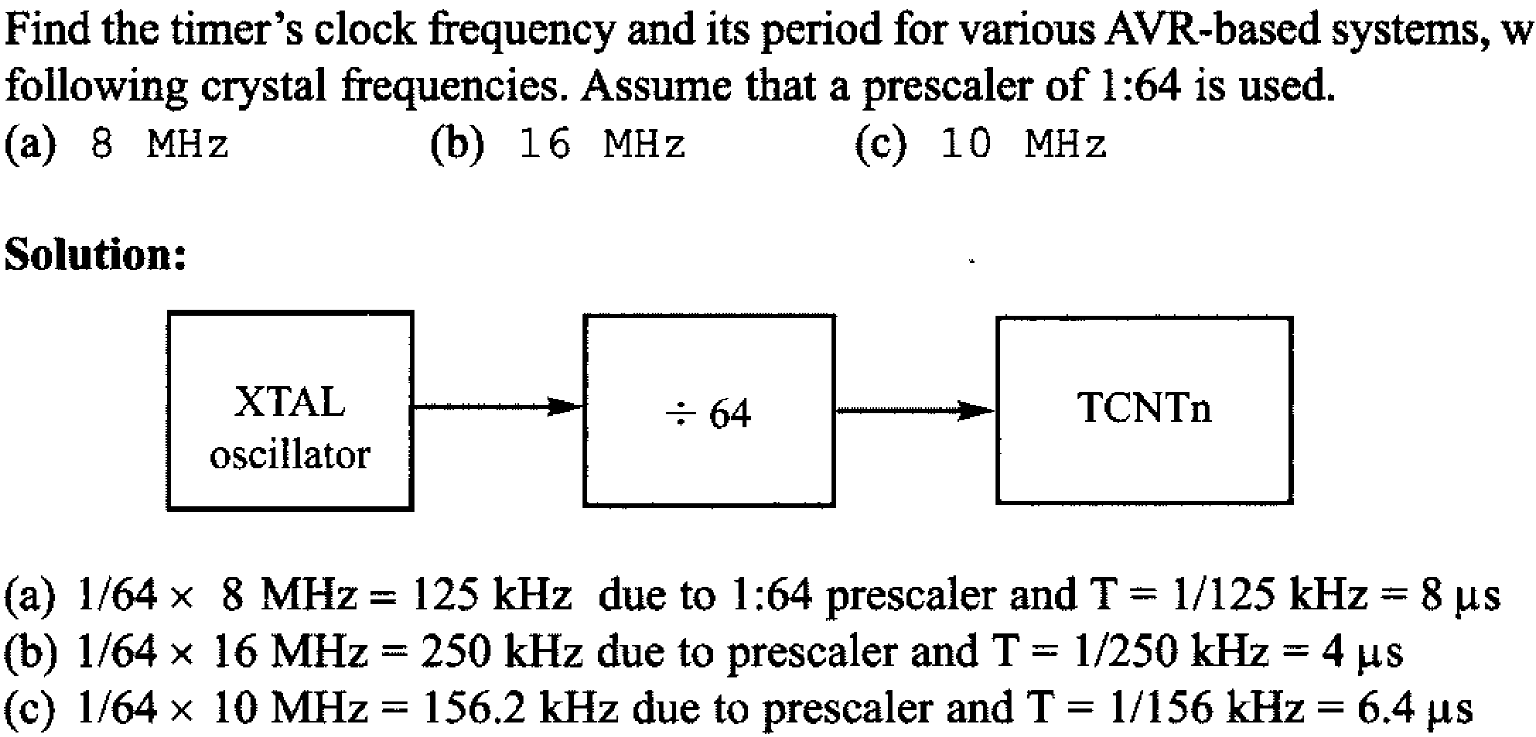
Prescalar and Generating a Large Time Delay

- Time delay depends on:
  - Crystal Frequency
  - Timer's 8-bit register
- Both are fixed
- How to generate large time delay?
  - Use prescalar to increase the delay by reducing the clock time period

Prescalar and Generating a Large Time Delay



Example



Find the timer's clock frequency and its period for various AVR-based systems, with the following crystal frequencies. Assume that a prescaler of 1:64 is used.

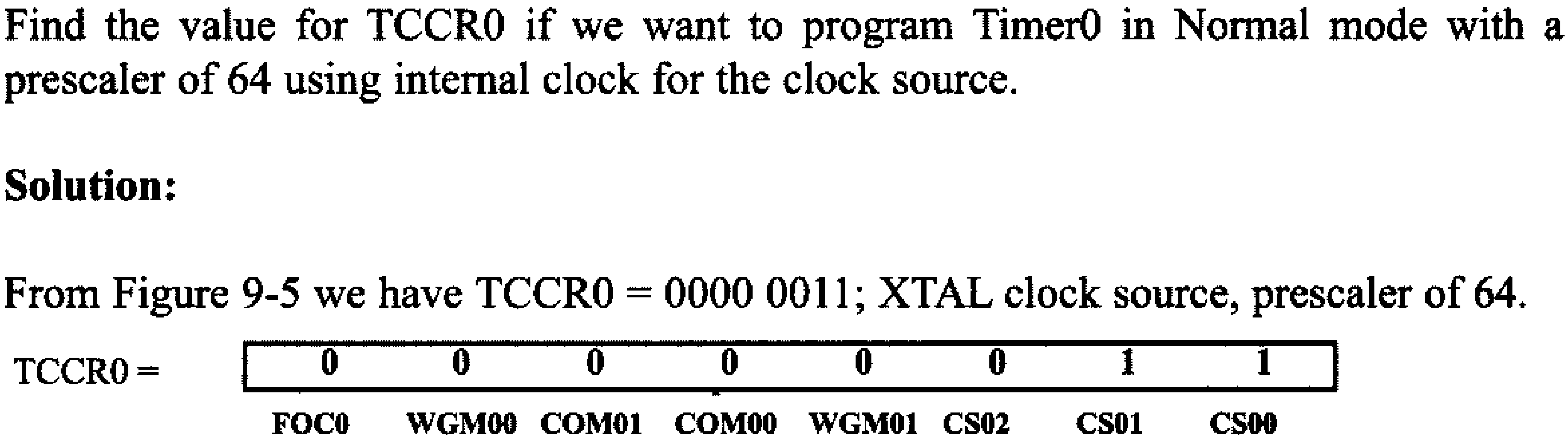
(a) `8 MHz` (b) `16 MHz` (c) `10 MHz`

Solution:

XTAL oscillator
÷ 64
TCNTn

(a) 1/64 × 8 MHz = 125 kHz due to 1:64 prescaler and T = 1/125 kHz = 8 µs
(b) 1/64 × 16 MHz = 250 kHz due to prescaler and T = 1/250 kHz = 4 µs
(c) 1/64 × 10 MHz = 156.2 kHz due to prescaler and T = 1/156 kHz = 6.4 µs

Example



Find the value for TCCR0 if we want to program Timer0 in Normal mode with a prescaler of 64 using internal clock for the clock source.

Solution:

From Figure 9-5 we have TCCR0 = 0000 0011; XTAL clock source, prescaler of 64.

TCCR0 =	0	0	0	0	0	0	1	1
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Timer as Counter

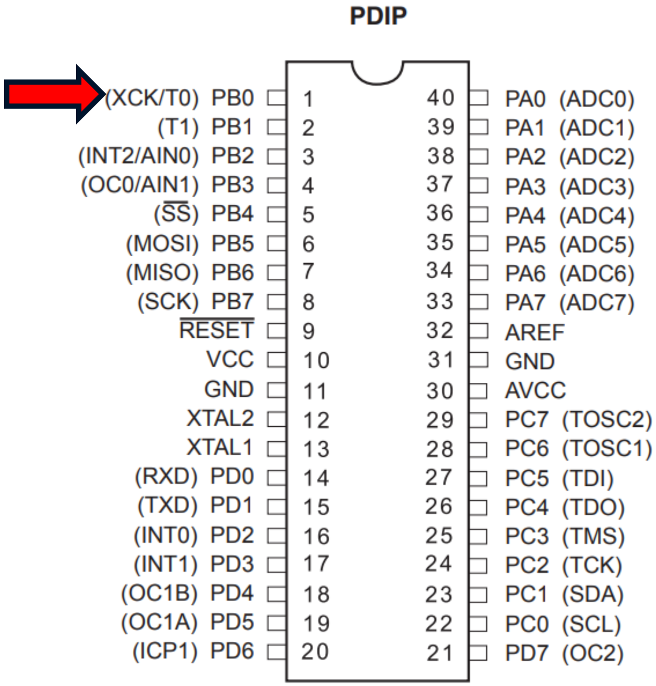
Timer as Counter

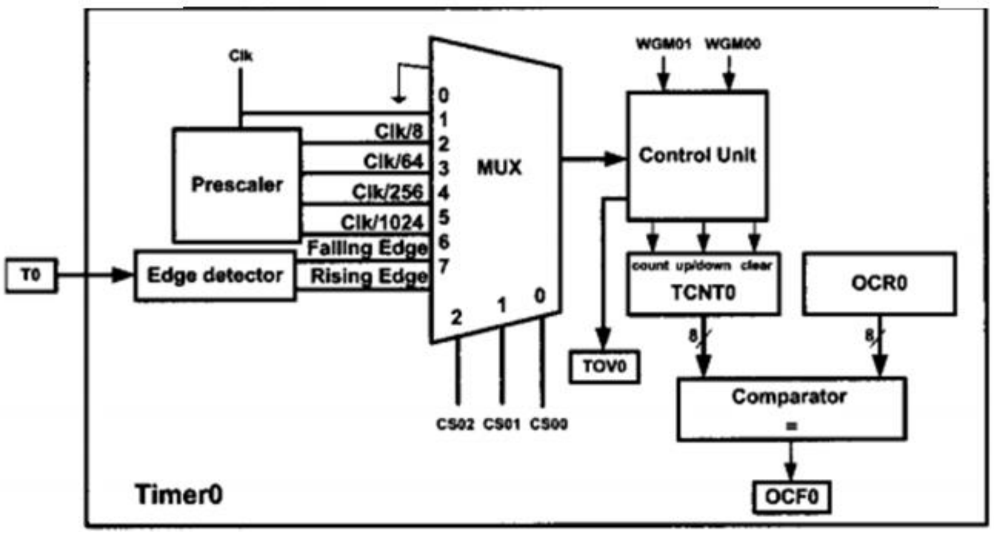
- So far, we have used timers to generate time delays

- The AVR timer can also be used to count, detect and measure time of events happening outside of AVR.

Timer as Counter

- When the timer is used as a timer, the source of the frequency is the AVR’s internal crystal.

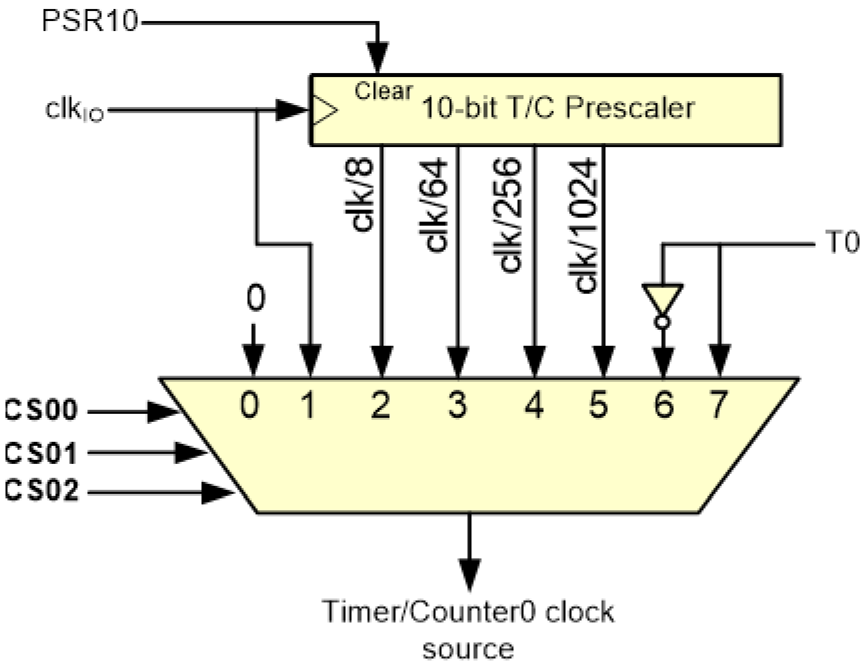

PDIP
(XCK/T0) PB0 1 40 PA0 (ADC0)
(T1) PB1 2 39 PA1 (ADC1)
(INT2/AIN0) PB2 3 38 PA2 (ADC2)
(OC0/AIN1) PB3 4 37 PA3 (ADC3)
(SS) PB4 5 36 PA4 (ADC4)
(MOSI) PB5 6 35 PA5 (ADC5)
(MISO) PB6 7 34 PA6 (ADC6)
(SCK) PB7 8 33 PA7 (ADC7)
RESET 9 32 AREF
VCC 10 31 GND
GND 11 30 AVCC
XTAL2 12 29 PC7 (TOSC2)
XTAL1 13 28 PC6 (TOSC1)
(RXD) PD0 14 27 PC5 (TDI)
(TXD) PD1 15 26 PC4 (TDO)
(INT0) PD2 16 25 PC3 (TMS)
(INT1) PD3 17 24 PC2 (TCK)
(OC1B) PD4 18 23 PC1 (SDA)
(OC1A) PD5 19 22 PC0 (SCL)
(ICP1) PD6 20 21 PD7 (OC2)

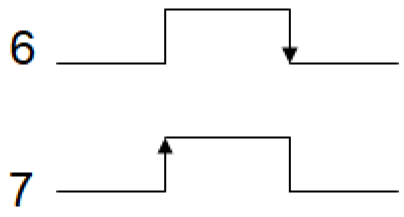


- When the timer is used as a counter, the pulse outside the AVR increments the TCNT register.

Timer as Counter

- When the timer is used as a timer, the source of the frequency is the AVR’s internal crystal.

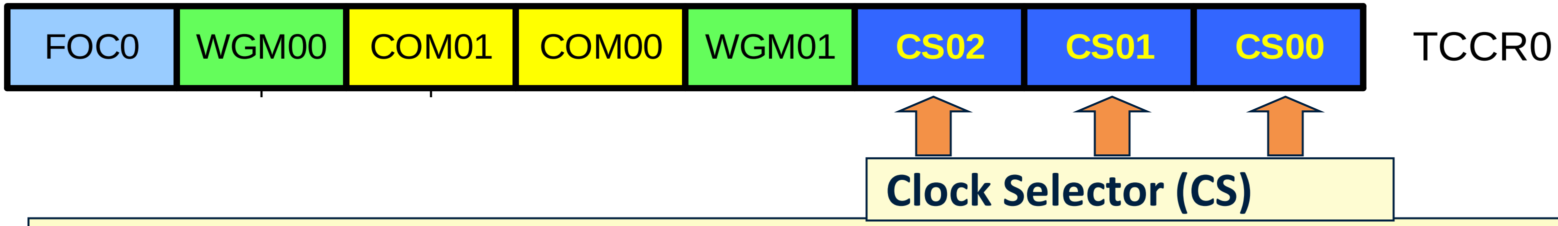




- When the timer is used as a counter, the pulse outside the AVR increments the TCNT register.

Timer as Counter: Example

- Find the value for TCCR0 if we want to program Timer0 as a Normal mode counter. Use an external clock for the clock source and increment on the positive edge.



FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0

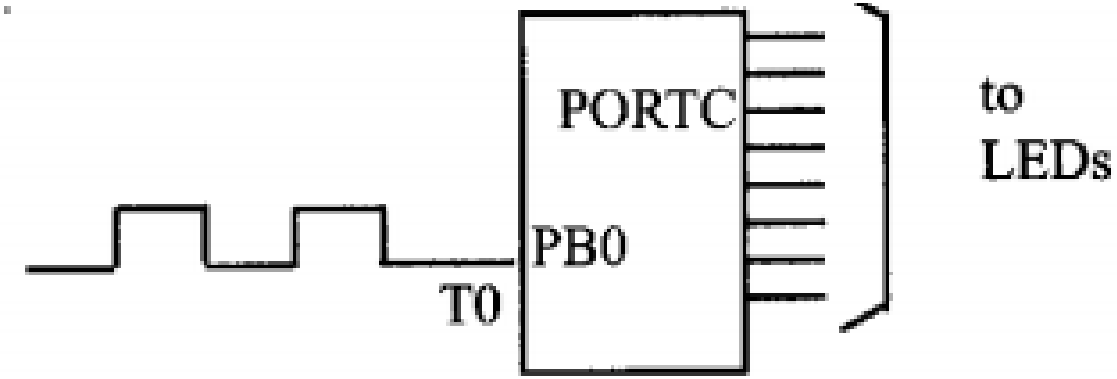
Clock Selector (CS)

CS02	CS01	CS00	Comment
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

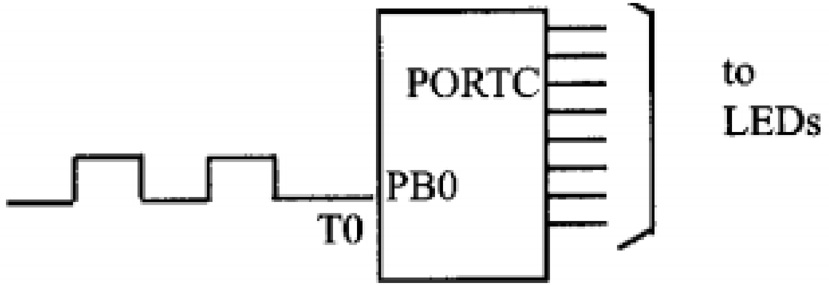
Sol: TCCR0 = 0000 0111 Normal, external clock source, no prescalar

Timer as Counter: Example

- Assuming that a 1 Hz clock pulse is fed into pin T0 (PB0),
  - Write a program for Counter0 in normal mode to count the pulses on falling edge and display the state of the TCNT0 count on PORTC.



Timer as Counter: Example

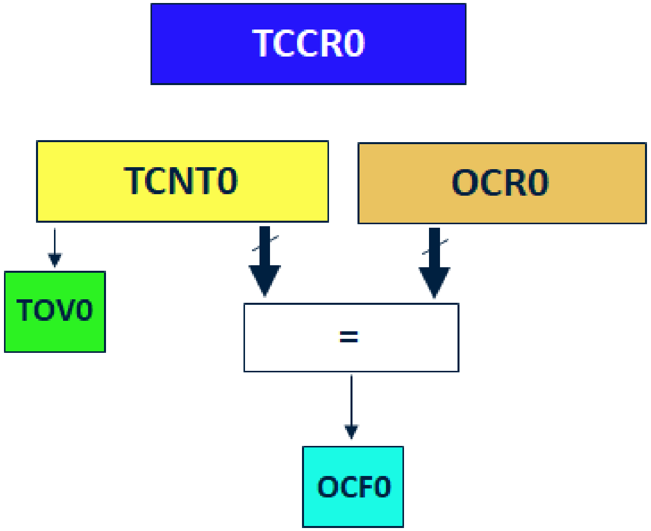


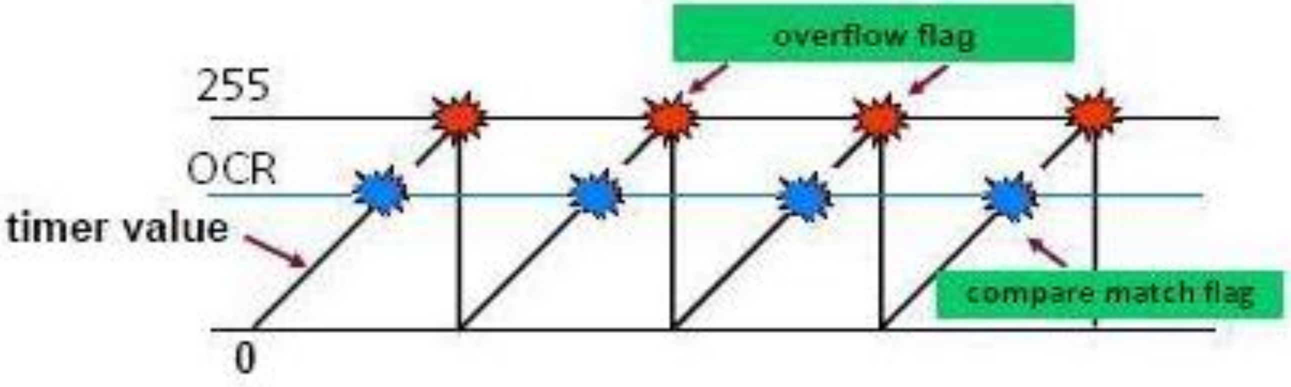
```
        CBI   DDRB,0             ;make T0 (PB0) input
        LDI   R20,0xFF
        OUT   DDRC,R20           ;make PORTC output
        LDI   R20,0x06
        OUT   TCCR0,R20          ;counter, falling edge
AGAIN:
        IN    R20,TCNT0
        OUT   PORTC,R20          ;PORTC = TCNT0
        IN    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
```

Clear Timer on Compare Match (CTC) Mode Programming

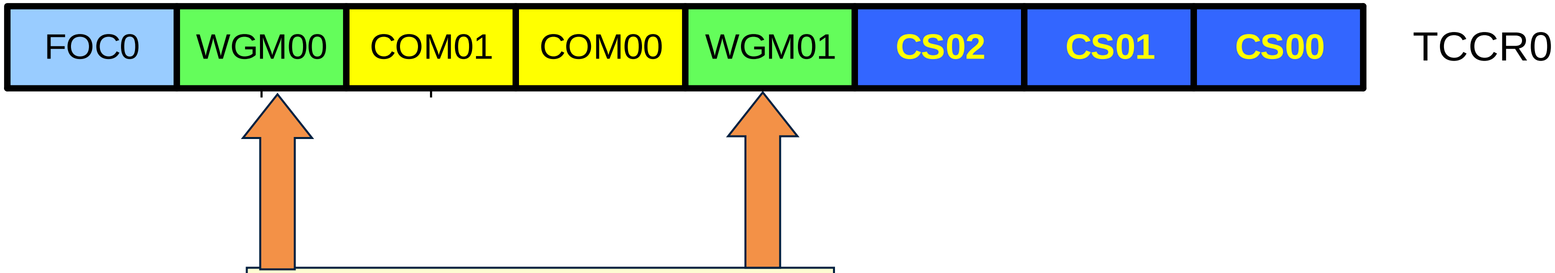
Recall: OCR0 Flag

- OCR0 [Output Compare] Flag: [in CTC Mode]
  - The content of the OCR0 is compared with the contents of the TCNT0, when both are equal:
    - OCR0 flag is set AND
    - Timer is cleared





TCCR0: Mode Selector



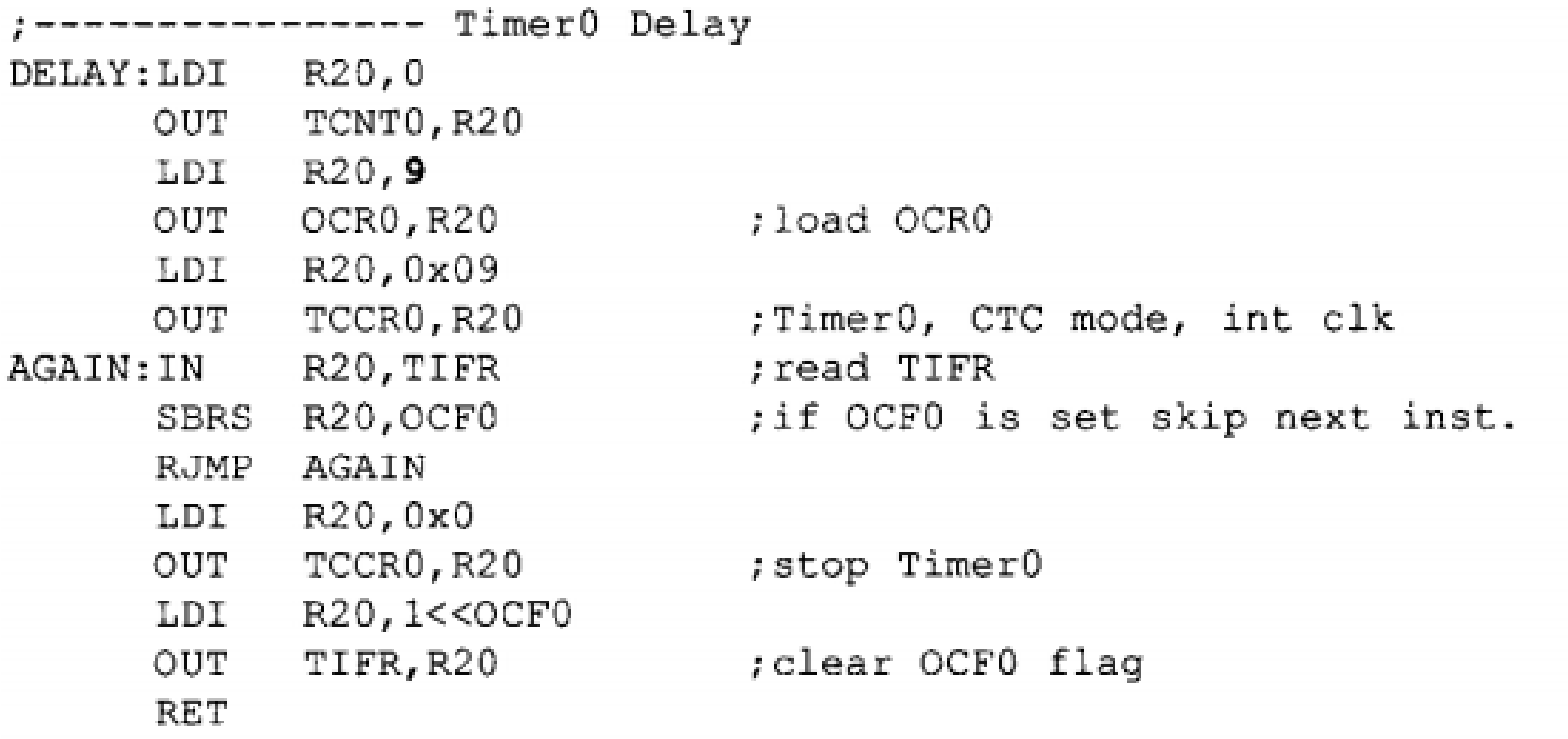
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

TCCR0

Timer Mode (WGM)

WGM00	WGM01	Comment
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	Phase correct PWM
1	1	Fast PWM

OCR0 Example



```
;----------------- Timer0 Delay
DELAY:LDI    R20,0
      OUT    TCNT0,R20
      LDI    R20,9
      OUT    OCR0,R20             ;load OCR0
      LDI    R20,0x09
      OUT    TCCR0,R20            ;Timer0, CTC mode, int clk
AGAIN:IN     R20,TIFR             ;read TIFR
      SBRS   R20,OCF0             ;if OCF0 is set skip next inst.
      RJMP   AGAIN
      LDI    R20,0x0
      OUT    TCCR0,R20            ;stop Timer0
      LDI    R20,1<<OCF0
      OUT    TIFR,R20             ;clear OCF0 flag
      RET
```

Recommended Reading

- The AVR Microcontroller and Embedded Systems: Using Assembly and C by Mazidi et al., Prentice Hall
  - Chapter-9

THANK YOU