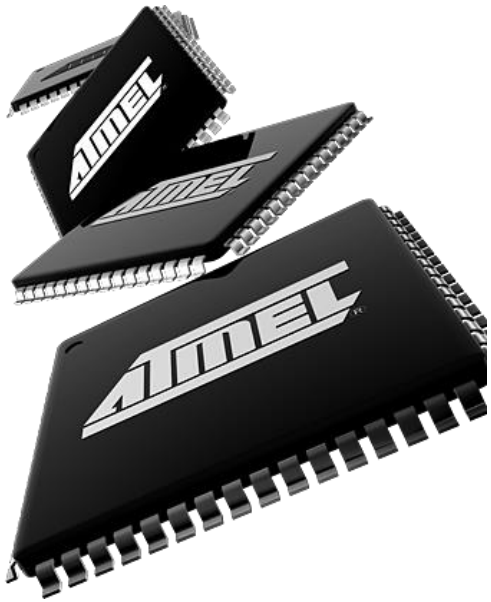
Principles and Applications of Microcontrollers

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

Today:

- AVR clock system
- Timer and counter



Review – Bit-wise Operation

- Bit-wise operators: <<, ~, &, |, ^

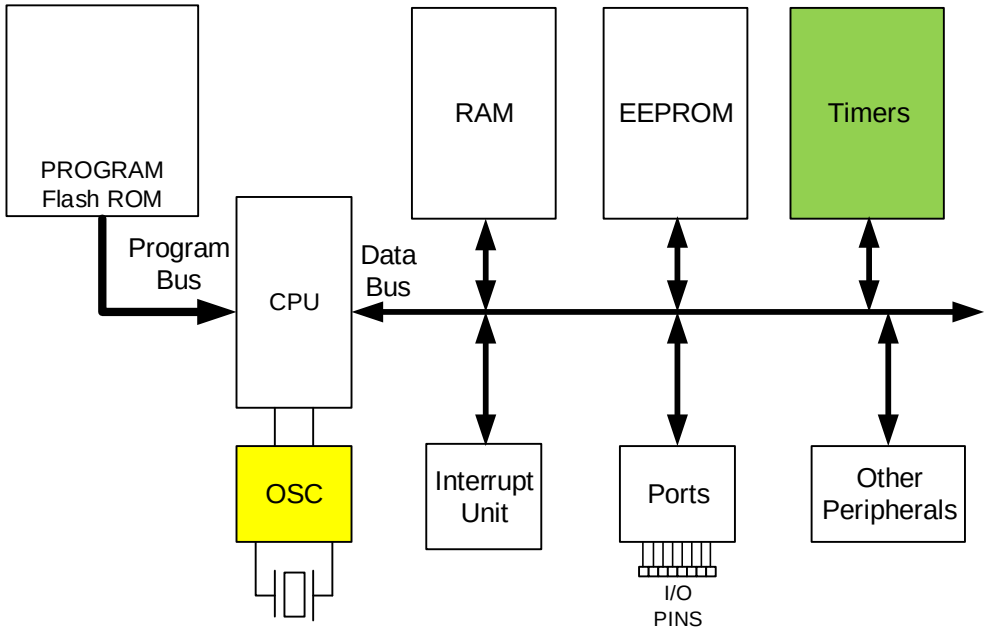
Bits:	7	6	5	4	3	2	1	0

	7	6	5	4	3	2	1	0
CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0

- Pseudo code:
  - if(CLKPR[CLKPS2]==1)
  - if(CLKPR[CLKPS2]==0)
  - CLKPR[CLKPS2]=1;
  - CLKPR[CLKPS2]=0;

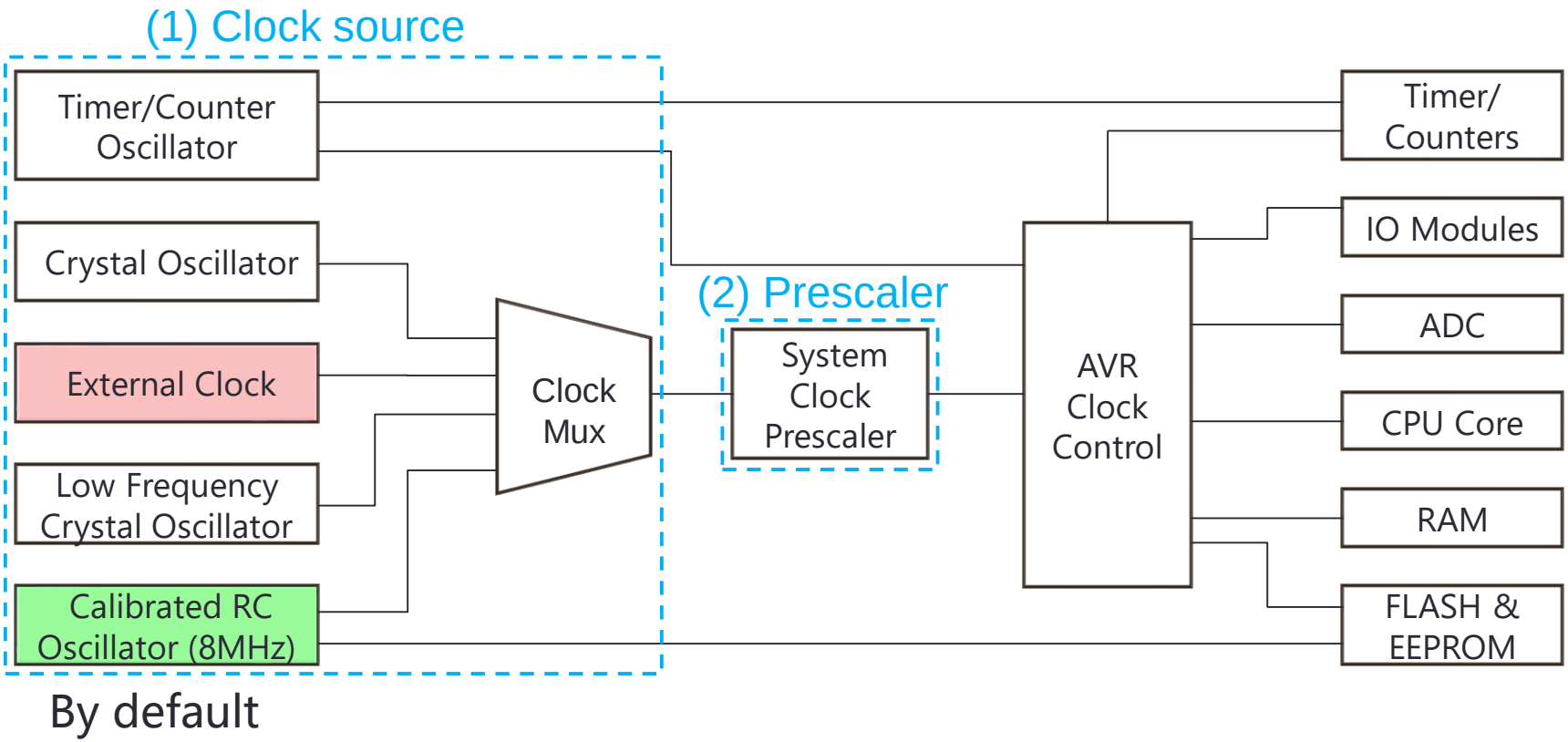
Outline

- Clock system
  - Source option
  - Prescaler option
- Timer/counter
  - Timer0 registers
  - Timer0 normal mode
  - Timer0 CTC mode
  - Timer2 and Timer1
- Getting started



AVR ATmega328P Clock System

- Determined by (1) clock source and (2) prescaler
- Default clock is internal 8MHz oscillator and ÷ 1 prescale, yielding 8MHz CPU clock (up to 5-10% error)

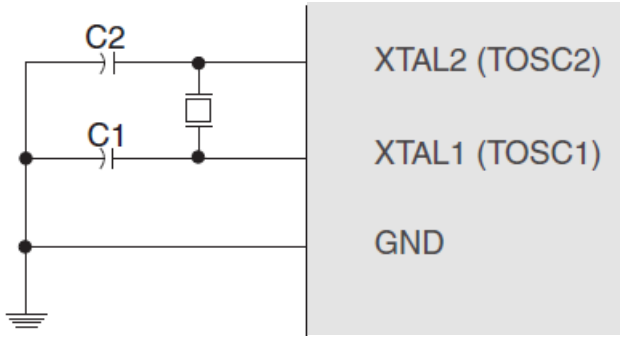


(1) Clock Source Options

- Clock source selected by fuse bits CKSEL3:CKSEL0

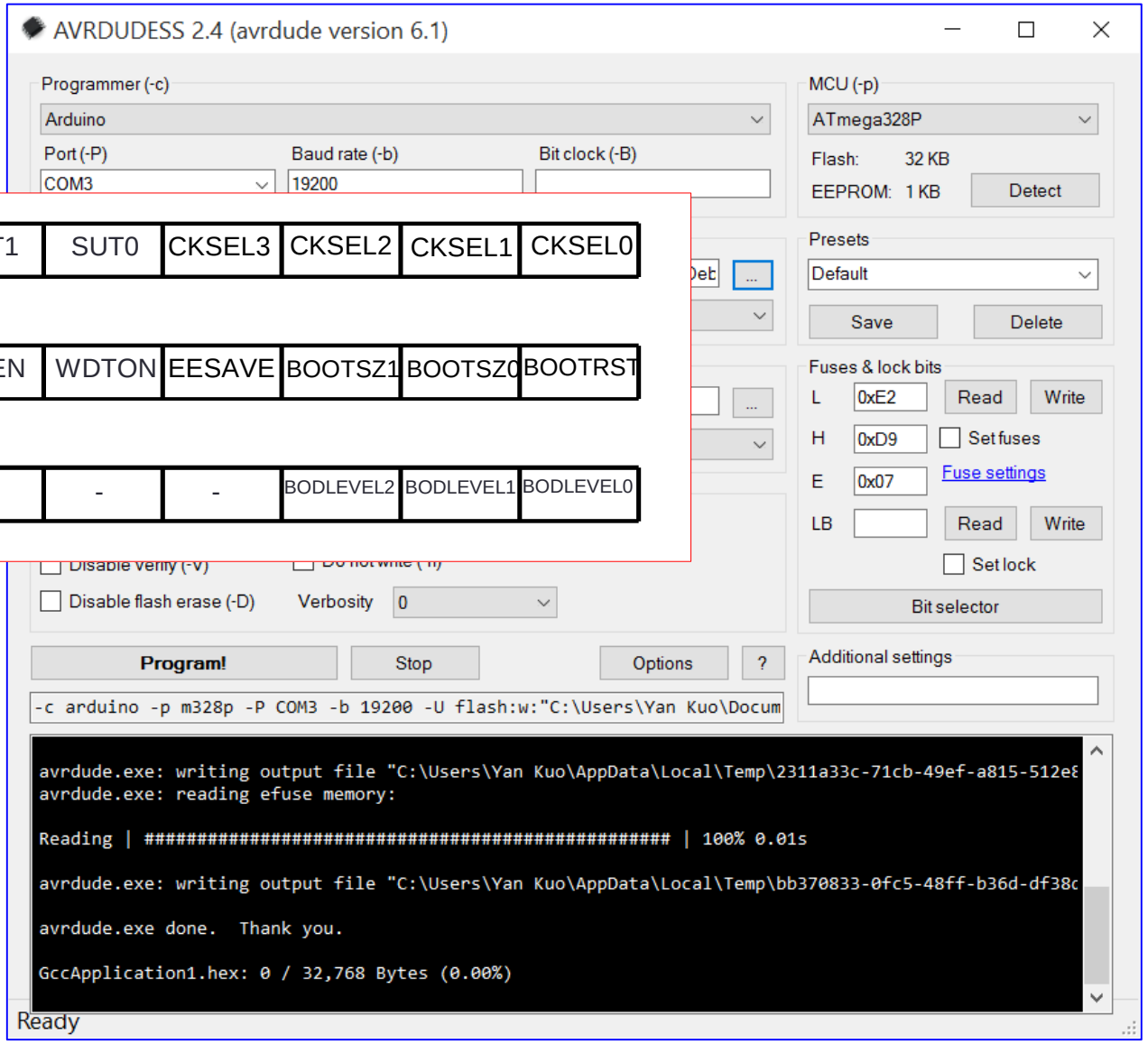
Device Clocking Option	CKSEL3...0
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

- Crystal oscillator connections:



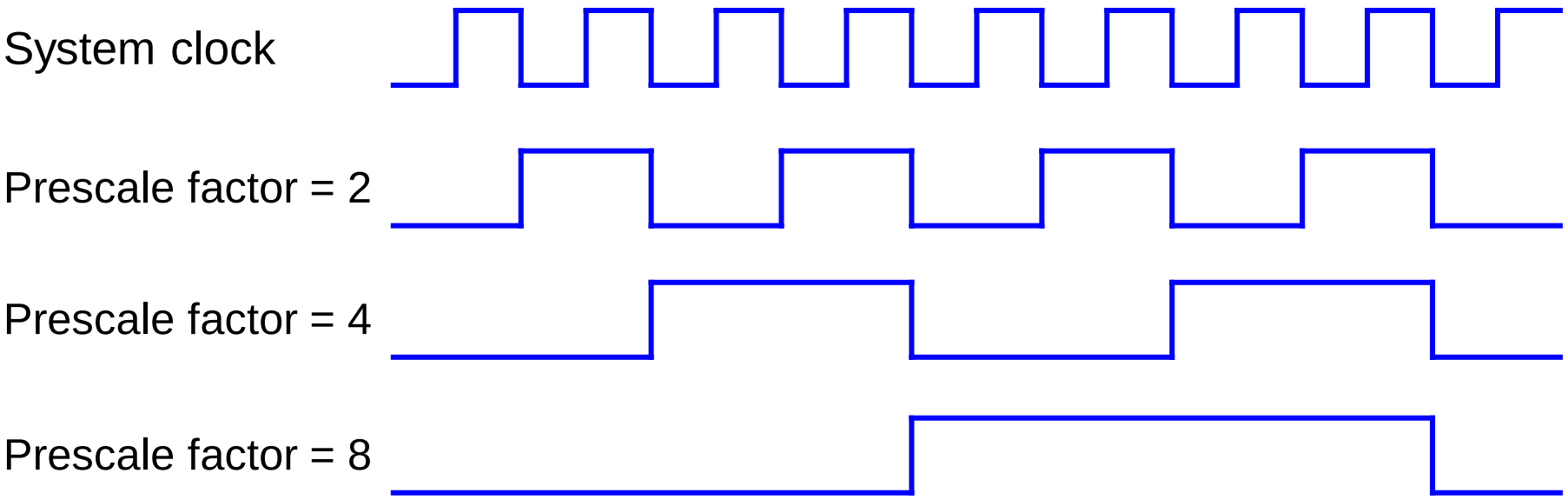
Setting Fuse in AVRDUDESS

Fuse Low	CKDIV8	CKOUT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
Fuse High	RSTDISBL	DWEN	SPIEN	WDTON	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST
Fuse Ext.	-	-	-	-	-	BODLEVEL2	BODLEVEL1	BODLEVEL0



(2) Prescalar

- Used to reduce a high frequency digital pulse signal (system clock) to a lower frequency digial pulse signal

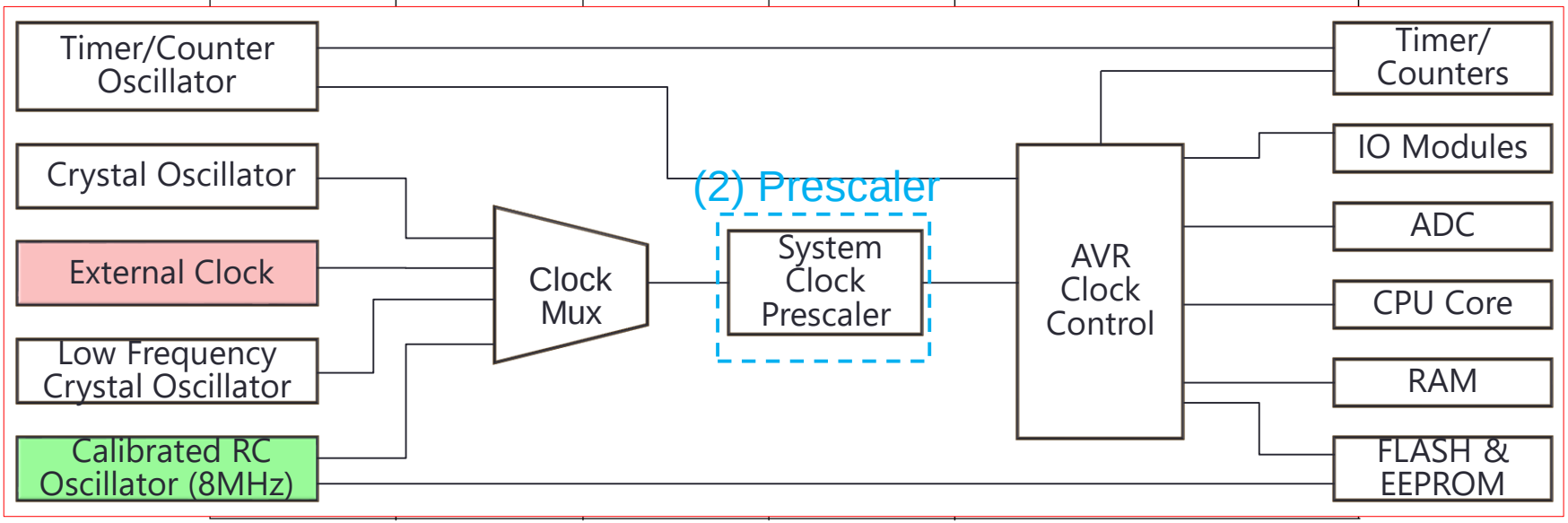


Prescaler Options – Register CLKPR

CLKPR

7	6	5	4	3	2	1	0
CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16

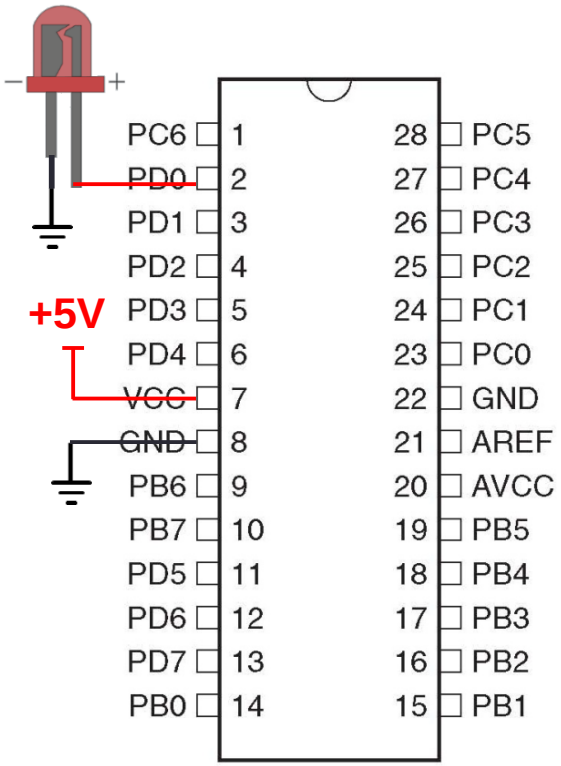


Example: Change Clock Prescaler

- Adjust the clock prescaler to make the MCU run at 1MHz
- Connect an LED to PD0
- LED flashes at 1Hz

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;
    DDRD=0b11111111;
    while (1){
       PORTD=0b00000001;
       _delay_ms(500);
       PORTD=0b00000000;
       _delay_ms(500);
    }
}
```

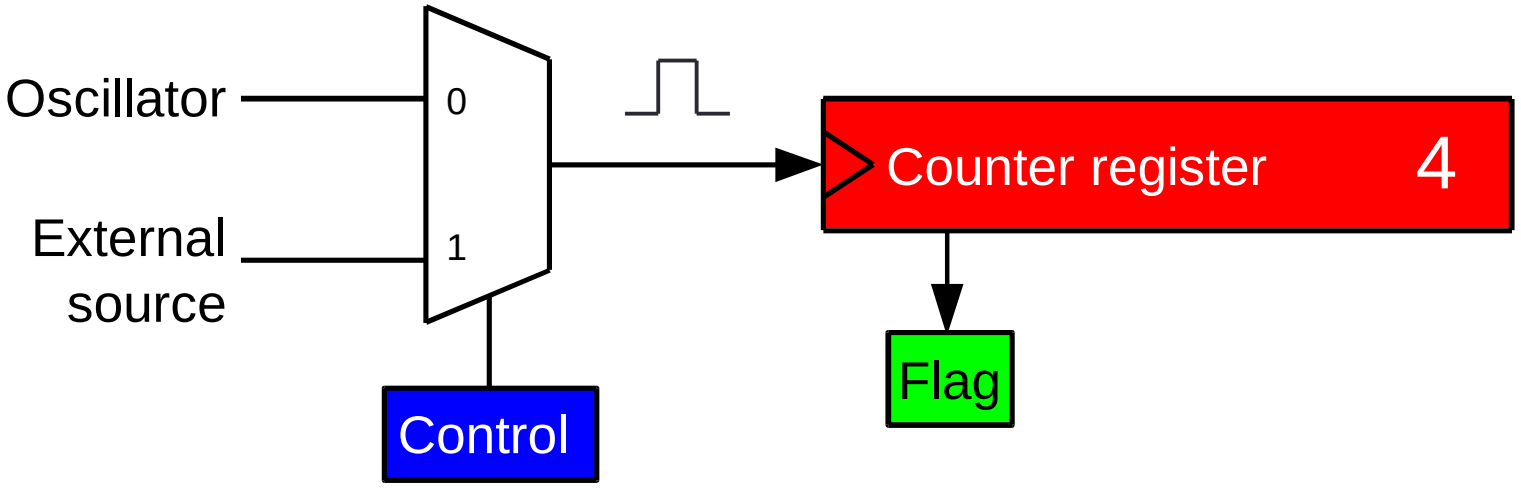


Outline (Cont'd)

- Clock system
  - Source option
  - Prescaler option
- Timer/counter
  - Timer0 registers
  - Timer0 normal mode
  - Timer0 CTC mode
  - Timer2 and Timer1
- Getting started

The Concept of A Timer/Counter

- Counts the number of pulses it received
- Used as a timer when the period of pulses are known
- A generic timer is composed of at least three registers
  - One for control purposes – control register
  - One for counting purposes – data register
  - One for flag purposes – status register



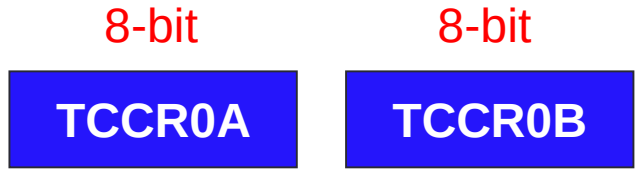
Timers in AVR

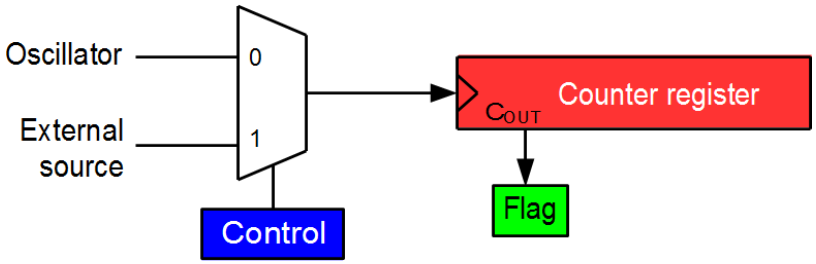
- Three timers in ATmega328P
  - Timer0: 8-bit
  - Timer1: 16-bit
  - Timer2: 8-bit
- Features
  - Single compare unit counter
  - Clear timer on compare match (auto reload)
  - External event counter
  - 10-bit clock prescaler


LOWER CARGO FIRE PROTECTION
MAIN DECK CARGO SMOKE
DETECTOR TEST
AFT CARGO

Timer0

- TCNT0 (Timer/counter register)
- TCCR0A/B (Timer/counter control register)
- TOV0 (Timer overflow flag)
  (as a bit in register TIFR0)



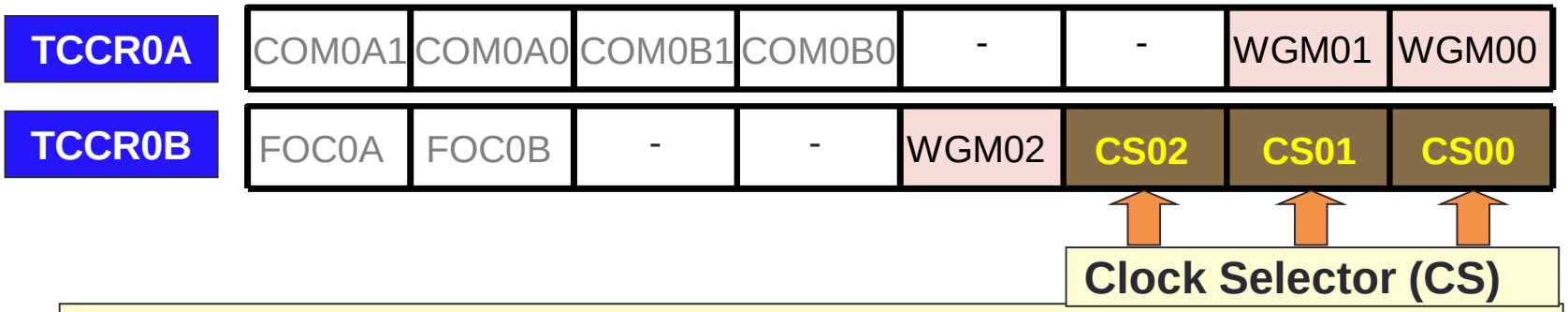


Control Registers – TCCR0A/B

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**

- Determine (1) clock source and (2) operation mode
- CS0n WGM0n
- COM0n – compare output mode
- FOC0n – force compare match

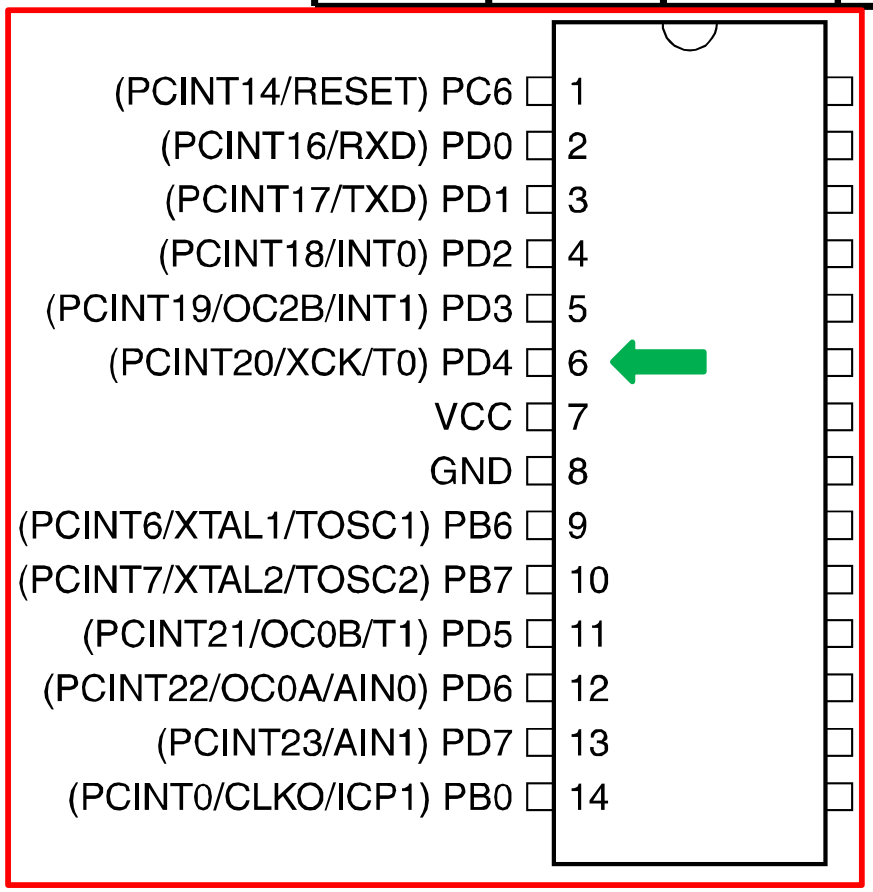
Clock Selector

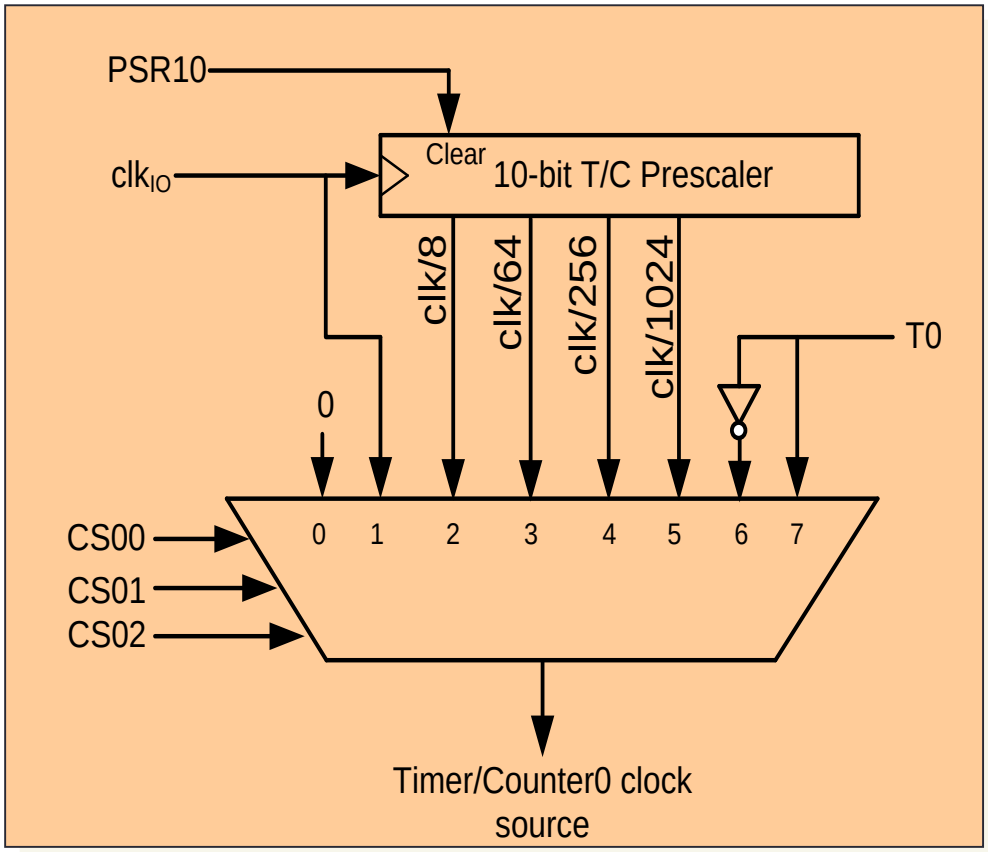


CS02	CS01	CS00	Comment
0	0	0	No clock source (Timer/counter stopped)
0	0	1	clk (No prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin; clock on falling edge
1	1	1	External clock source on T0 pin; clock on rising edge

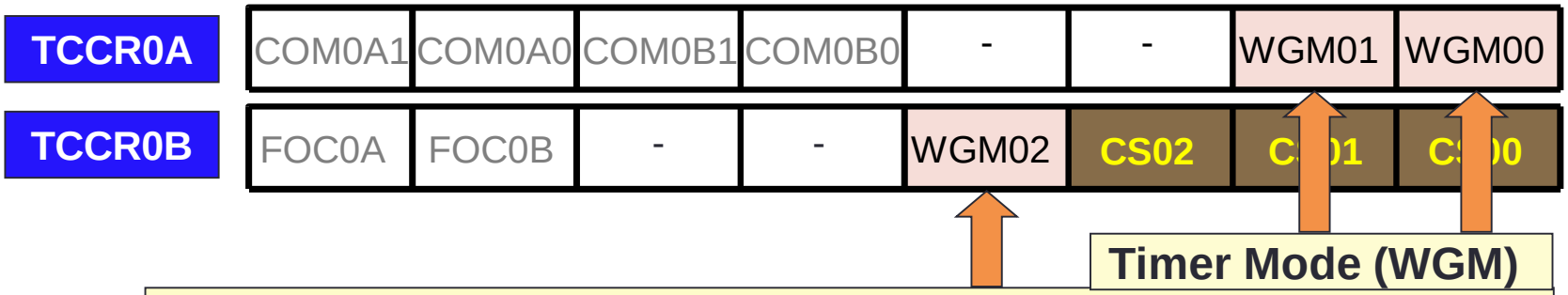
Control Registers – TCCR0A/B

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**





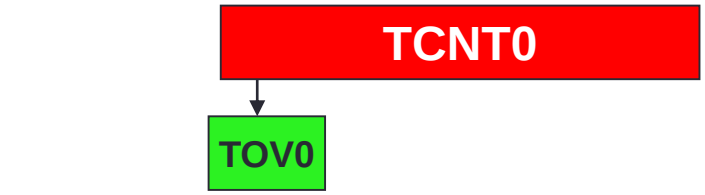
Timer Mode

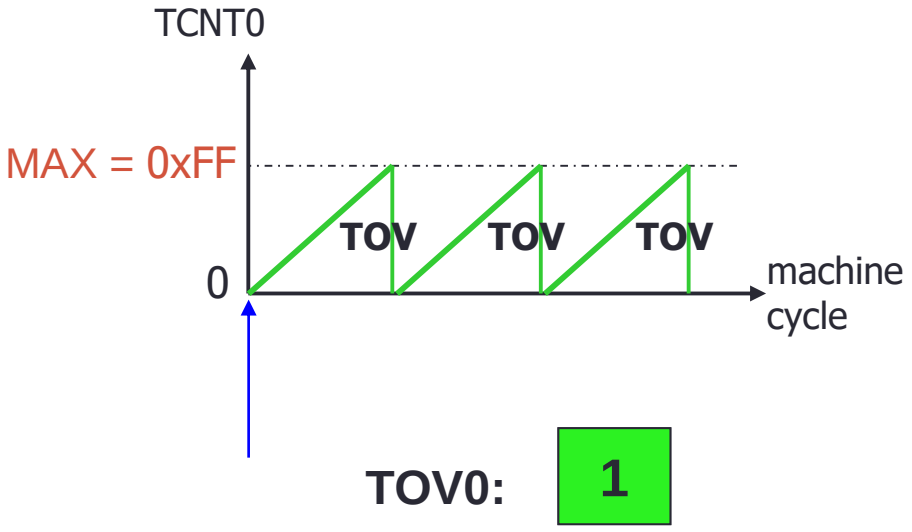


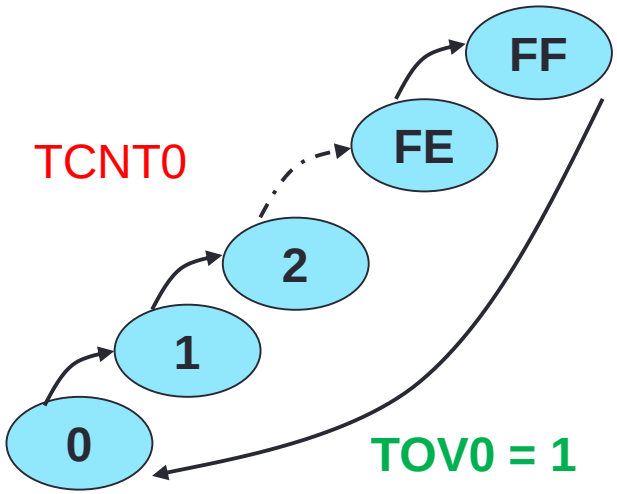
WGM02	WGM01	WGM00	Comment
0	0	0	Normal
0	0	1	PWM, phase correct
0	1	0	CTC (Clear timer on compare match)
0	1	1	Fast PWM
1	0	0	Reserved
1	0	1	PWM, phase correct
1	1	0	Reserved
1	1	1	Fast PWM

Normal Mode

- The content of the timer counts up until it reaches its max of 0xFF (MAX)
- The flag TOV0 = 1 when TCNT0 == 0





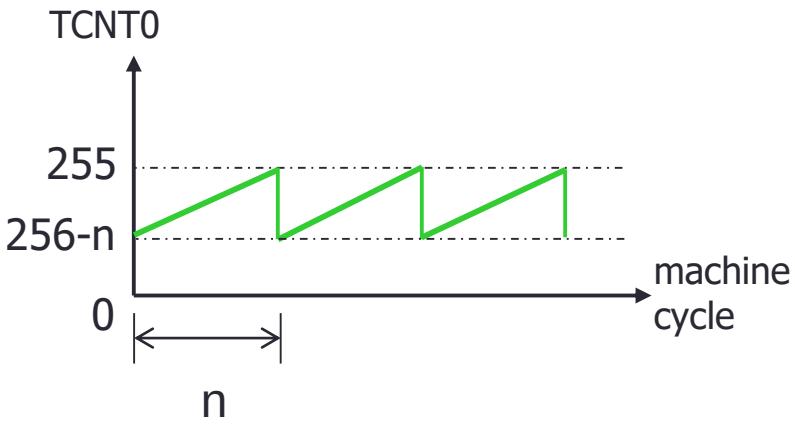


Watch how TOV0 change!!

Example: Counting 14 Machine Cycles (Normal)

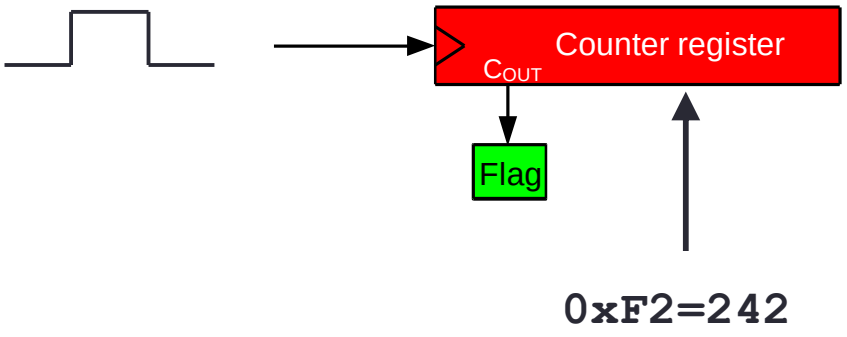
- Use Timer0 to count 14 machine cycles indefinitely
- Strategy:

  Set TCNT0 = 256 – n, where n is the machine cycles to count




```
  0x100 = 256
-0x  0E =  14
-----------
  0x  F2 = 242
```

TCNT0



Flowchart (Normal)

- What value do we set the controller registers?

TCCR0A

						0	0
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

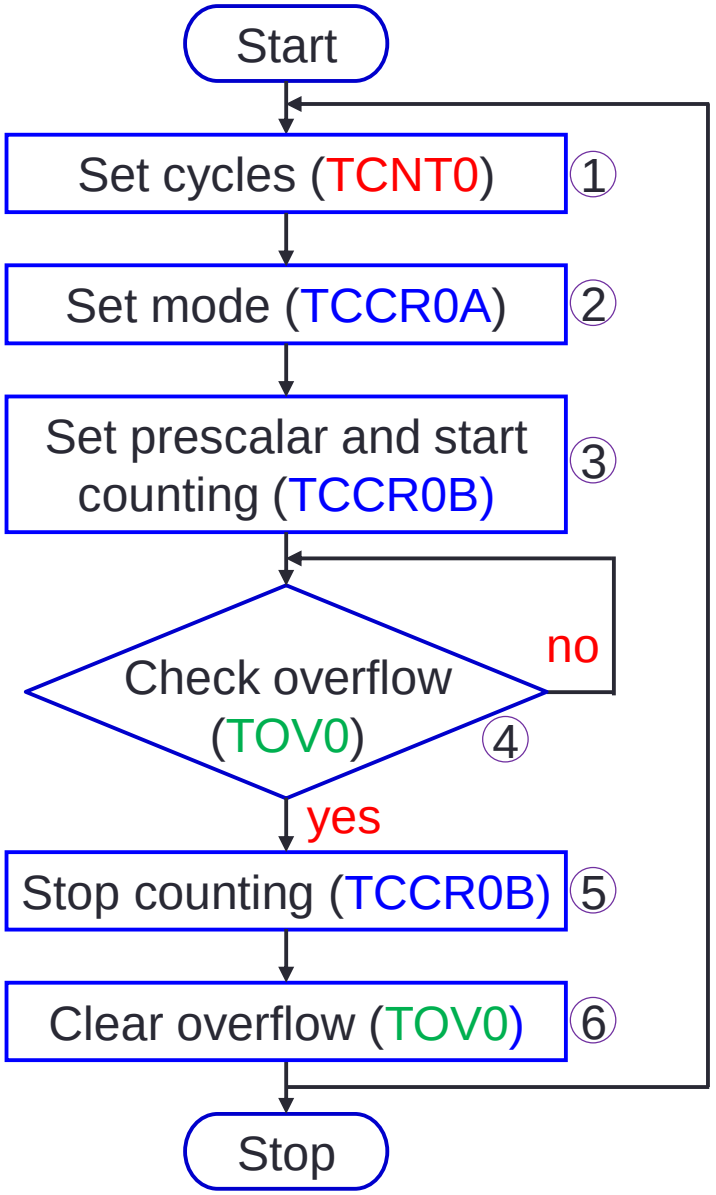
TCCR0B 0 0 0 1

Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	0	0	Normal

Clock Selector (CS)

CS02	CS01	CS00	Comment
0	0	1	clk (No Prescaling)



Counting14 Machine Cycles (Normal)

TCNT0 = 0xF2

TCCR0A = 0x00

TCCR0B = 0x01

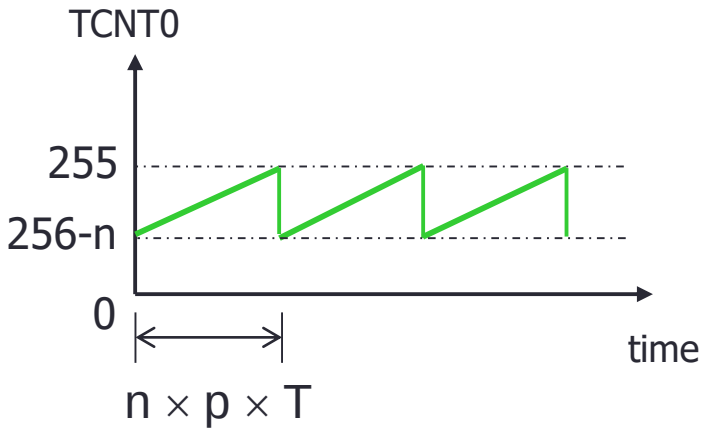
Note: TOV0 is cleared by written 1 into the bit

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

int main(void)
{
    while (1){
     ① TCNT0=0xF2;                          // n=14
     ② TCCR0A=0x00;                         // normal mode, int clk
     ③ TCCR0B=0x01;                         // start Timer0
     ④ while ((TIFR0&(0b00000001))==0);     // wait for flag TOV0=1
     ⑤ TCCR0B=0x00;                         // stop Timer0
     ⑥ TIFR0=TIFR0|(1<<TOV0);               // clear TOV0
    }
}
```

Time Delay Produced by Timer

- Procedure to calculate the delay generated by the timer:
  1. Machine cycles: n
  2. Prescalar factor: p
  3. Machine cycle period: T = 1/f
- Time delay: $t = n \times p \times T$
- Suppose the clock of a MCU runs at 1 MHz
- For the previous example:



1. n = 14
2. p = 1
3. T = 1/f = 1/1MHz = 1μs

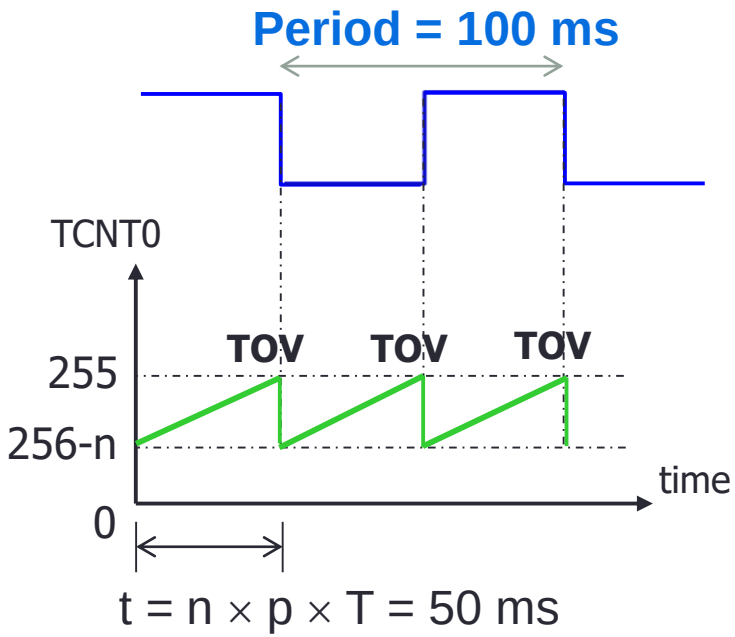
$t = 14 \times 1 \times 1\mu s = 14\mu s$

Example: Square Wave at 10Hz

- Suppose a MCU runs at 1MHz (i.e., the clock period is 1μs)
- Write a program to generate a square wave at a frequency of 10Hz on pin 0 of Port D (PD0)
- The period is 0.1s (100ms)
- Need a time delay of 50ms

- TCNT0 setup:
  1. Prescaling factor p = 1024
  2. Period of clock T = 1μs
  3. Number of machine cycles
     n = 50ms / 1μs / 1024 ≈ 50

TCNT0 = 256 - 50 = 206



Square Wave at 10Hz

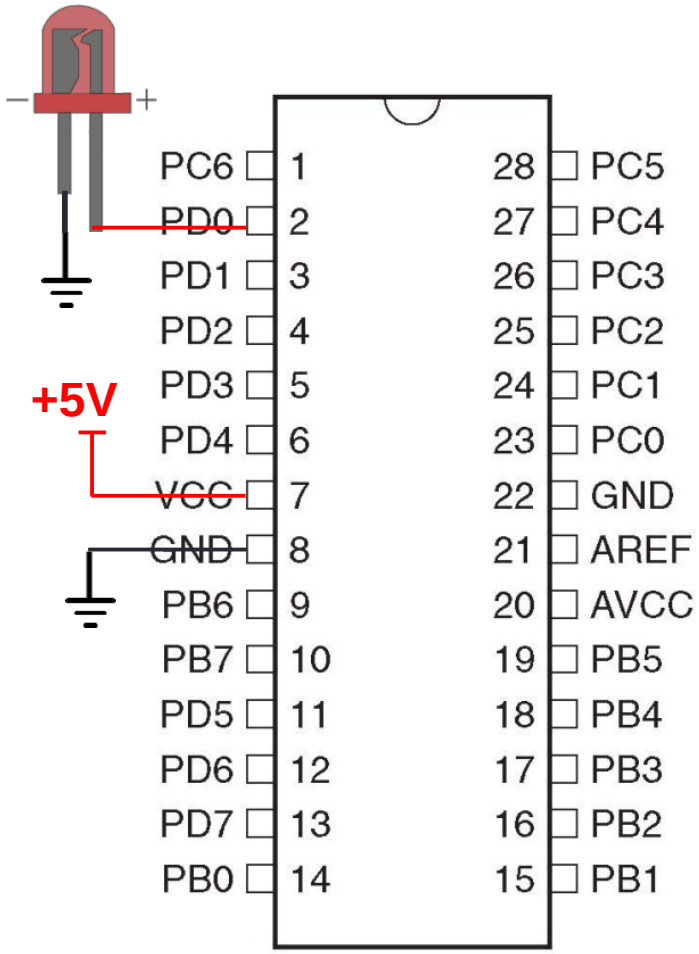
PD0	1	PD0^1
0	1	1
1	1	0

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

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                    // set clk to 1Mhz
    DDRD=0b00000001;                     // PD0 as output
    PORTD=0;                             // initial output 0
    while (1){
       TCNT0=206;
       TCCR0A=0;                         // normal mode, int clk
       TCCR0B=0b00000101;                // p=1024, start Timer
       while ((TIFR0&(1<<TOV0))==0);     // wait for flag TOV0=1
       TCCR0B=0;                         // stop Timer
       TIFR0=TIFR0|(1<<TOV0);            // clear TOV0
       PORTD=PORTD^0b00000001;
    }
}
```

Practice: Square Wave at 10Hz

- Connect an LED to PD0 and flash it at a frequency of 10Hz



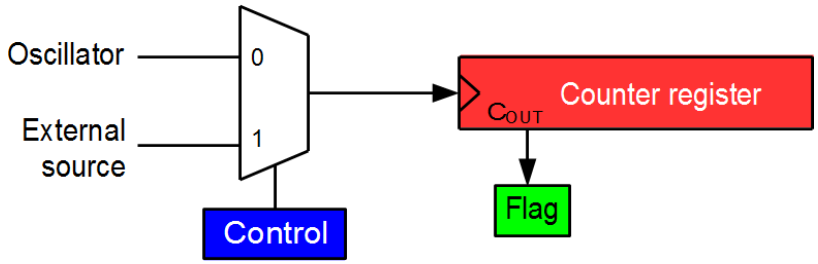
Outline (Cont'd)

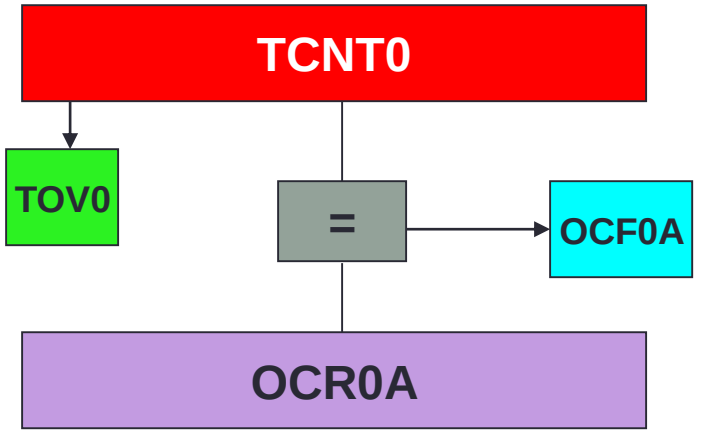
- Clock system
  - Source option
  - Prescaler option
- Timer/counter
  - Timer0 registers
  - Timer0 normal mode
  - Timer0 CTC mode
  - Timer2 and Timer1
- Getting started

2014 Porsche 911: Sport & Sport +

Timer0

- TCNT0 (Timer/counter register)
- TCCR0A/B (Timer/counter control register)
- TOV0 (Timer overflow flag)
- OCR0A (Output compare register)
- OCF0A (Output compare match flag)



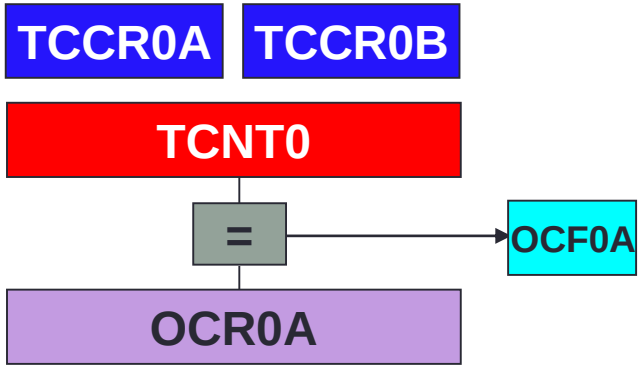


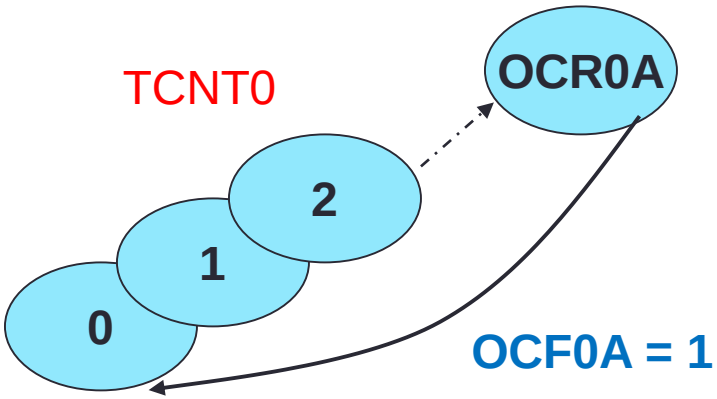
TCCR0A TCCR0B

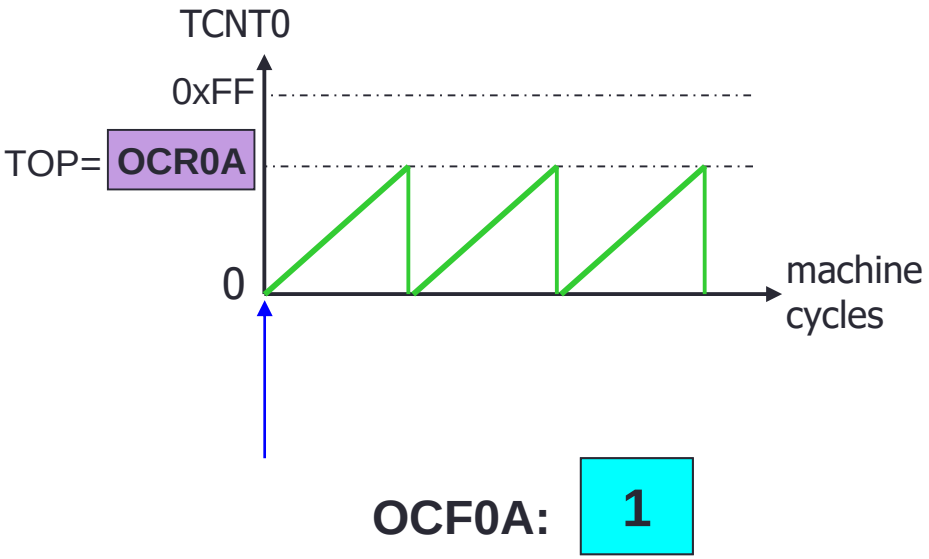
TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0

Clear Timer on CTC Mode

- The timer counts up until
  TCNT0 == OCR0A (**NOT** OCR0B)
- The flag OCF0A = 1 when
  TCNT0 == 0





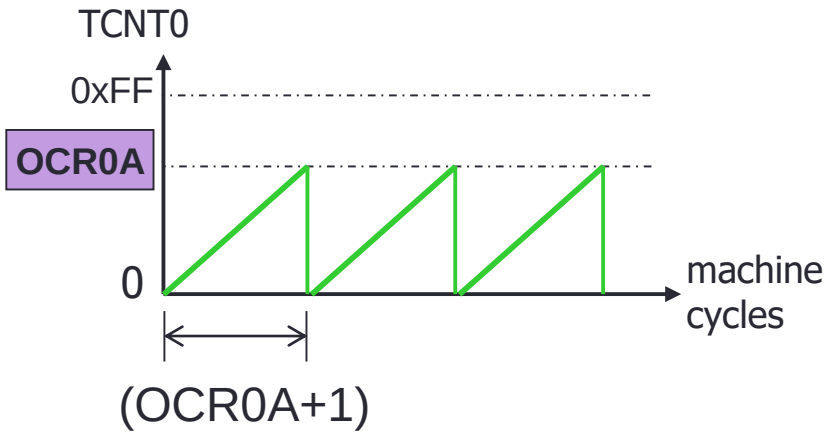


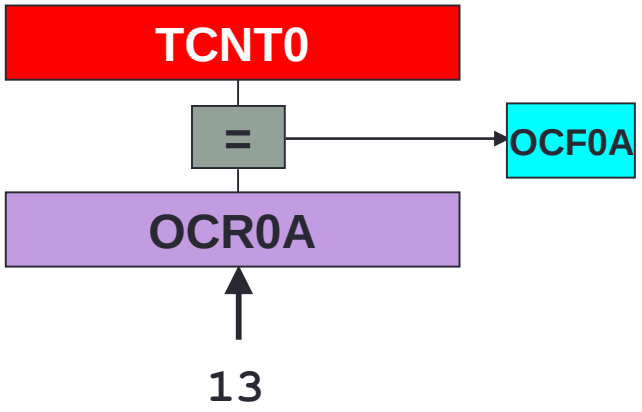
Watch how TOV0 change!!

Example: Counting14 Machine Cycles (CTC)

- Use Timer0 to count 14 machine cycles indefinitely
- CTC mode
- Strategy:

  Set OCR0A = n -1, where n is the machine cycles to count





Flowchart (CTC)

- What value do we set the controller registers?

TCCR0A

COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
						1	0
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

TCCR0B

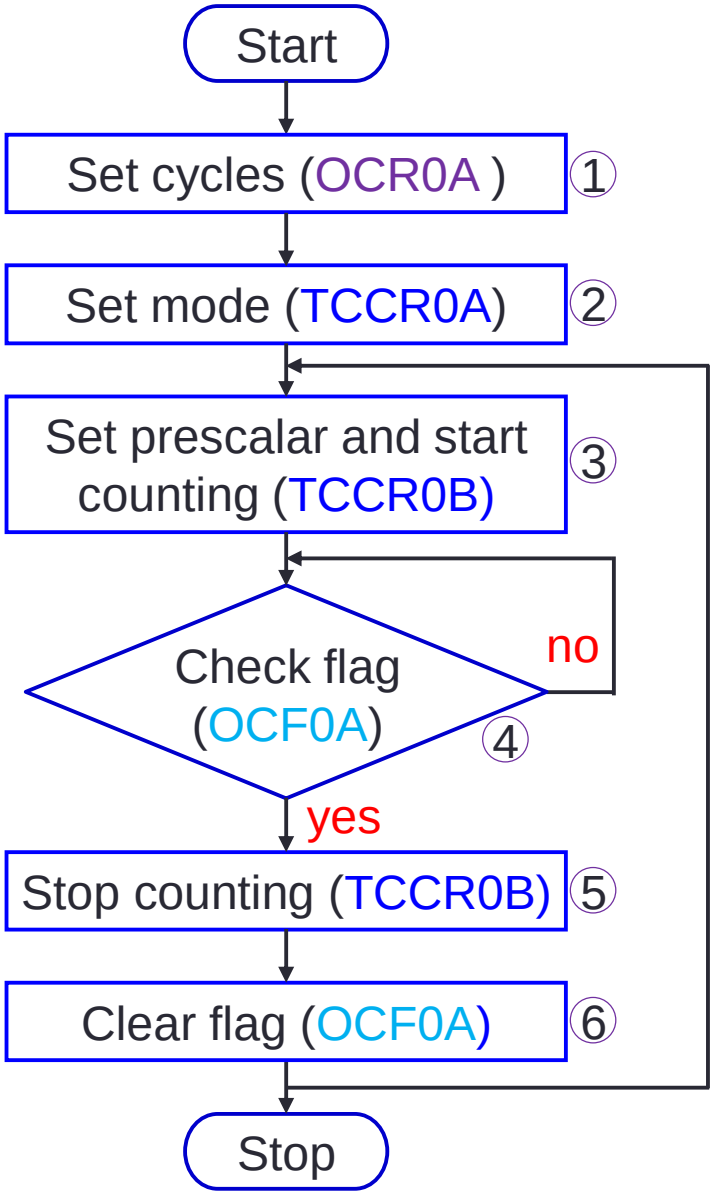
0	0	0	1

Timer Mode (WGM)

WGM02	WGM01	WGM00	Comment
0	1	0	CTC

Clock Selector (CS)

CS02	CS01	CS00	Comment
0	0	1	clk (No Prescaling)



Counting14 Machine Cycles (CTC)

OCR0A = 13

TCCR0A = 0x02

TCCR0B = 0x01

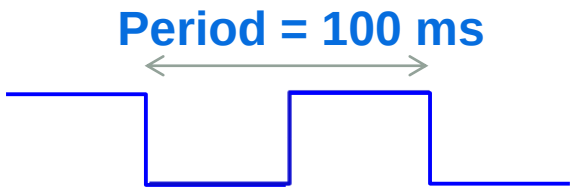
Note: OCF0A is cleared by written 1 into the bit

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

int main(void)
{
  ① OCR0A=13;                                  // n=14
  ② TCCR0A=0x02;                               // CTC mode, int clk
    while (1){
  ③    TCCR0B=0x01;                            // start Timer0
  ④    while ((TIFR0&(1<<OCF0A))==0);          // wait for flag TOV0=1
  ⑤    TCCR0B=0x00;                            // stop Timer0
  ⑥    TIFR0=TIFR0|(1<<OCF0A);                 // clear OCR0A
    }
}
```

Practice: Square Wave at 10Hz (CTC)

- Suppose a MCU runs at 1MHz
- Generate a square wave at a frequency of 10Hz on PD0
- Use time in the CTC mode



Outline (Cont'd)

- Clock system
  - Source option
  - Prescaler option
- Timer/counter
  - Timer0 registers
  - Timer0 normal mode
  - Timer0 CTC mode
  - Timer2 and Timer1
- Getting started

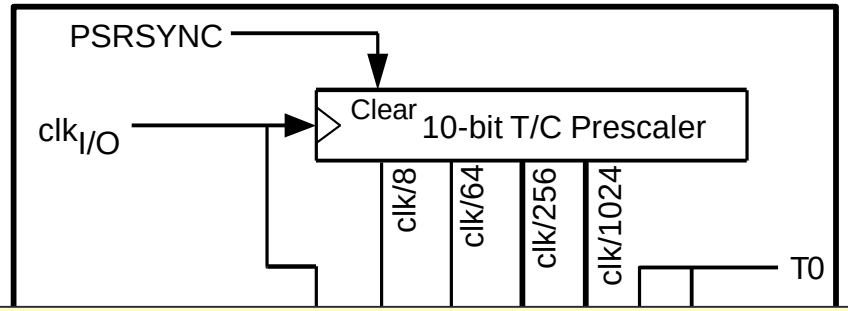
Register: Timer0 vs. Timer2

- Timer0
- Timer2

TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
TCCR0B	FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**
TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0
TCCR2A	COM2A1	COM2A0	COM2B1	COM0B0	-	-	WGM21	WGM20
TCCR2B	FOC2A	FOC2B	-	-	WGM22	**CS22**	**CS21**	**CS20**
TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2

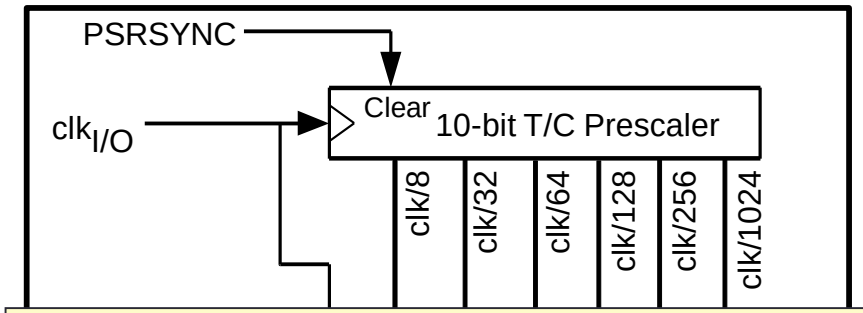
Prescaler: Timer0 vs. Timer2

- Timer0



CS02	CS01	CS00	Comment
0	0	0	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 (falling edge)
1	1	1	External clock (rising edge)

- Timer2



CS22	CS21	CS20	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 32
1	0	0	clk / 64
1	0	1	clk / 128
1	1	0	clk / 256
1	1	1	clk / 1024

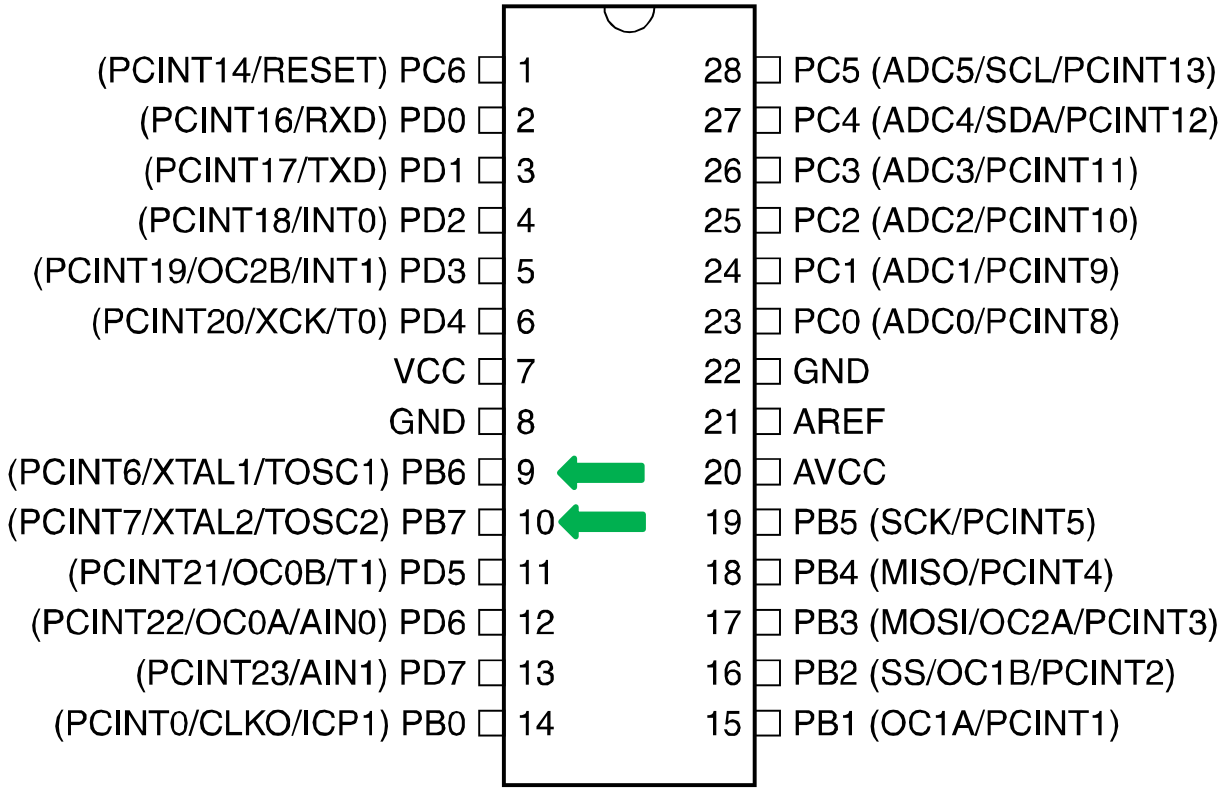
Example: Square Wave at 10Hz (CTC, Time2)

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

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=0b00000011;                    // set clk to 1Mhz
    DDRD=0b00000001;                     // PD0 as output
    PORTD=0;                             // initial output 0
    OCR2A=49;                            // n=14
    TCCR2A=(1<<WGM21);                   // normal mode, int clk
    while (1){
       TCCR2B=(1<<CS22)|(1<<CS21)|(1<<CS20);// p=1024, start Timer
       while ((TIFR2&(1<<OCF2A))==0);    // wait for flag TOV0=1
       TCCR2B=0;                         // stop Timer
       TIFR2|=(1<<OCF2A);                // clear TOV0
       PORTD^=0b00000001;                // toggle
    }
}
```

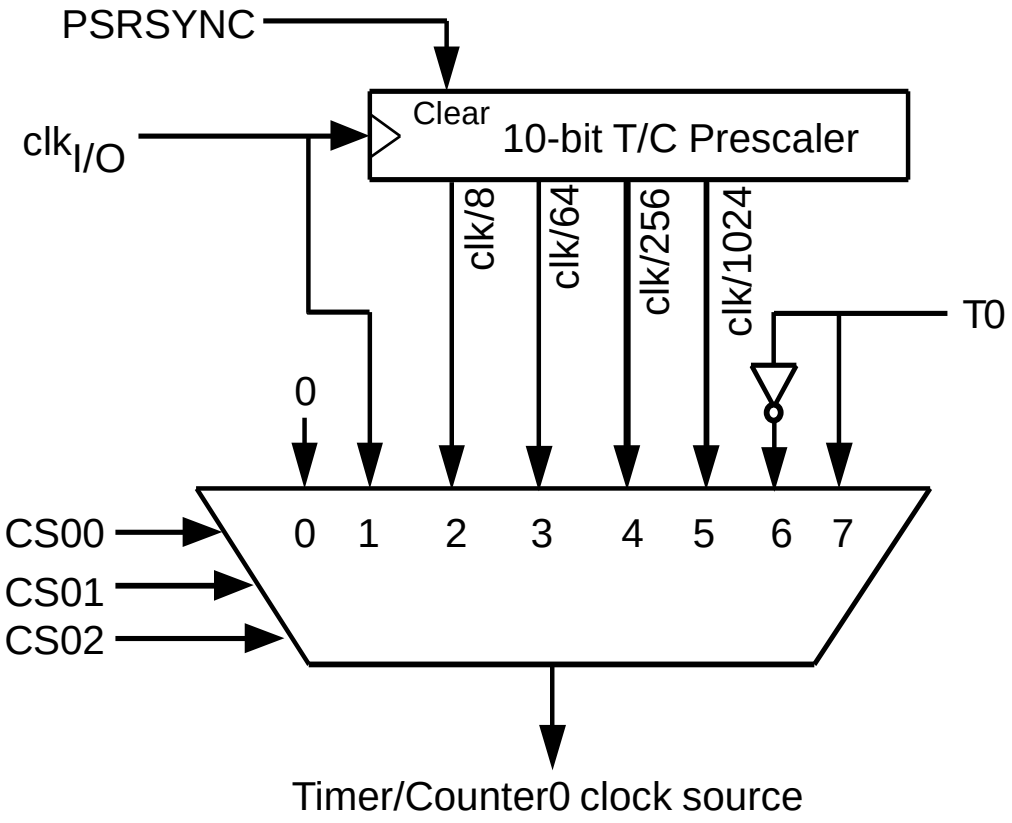
Timer2 as A Real Time Counter

- Enable external oscillator by setting AS2 bit in ASSR
- Connect TOSC1 and TOSC2 to a crystal of 32.768KHz too be a real time counter



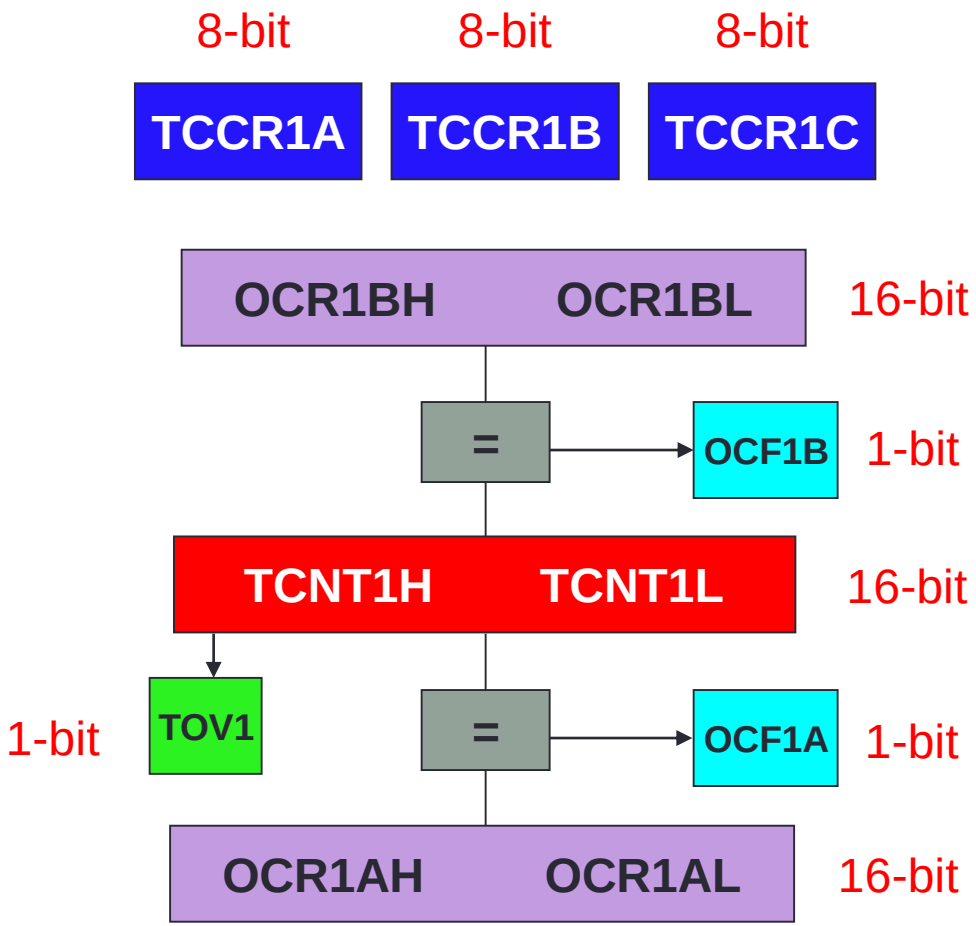
Generating Large Time Delays

- Using loop
- Prescaler
- Bigger counters



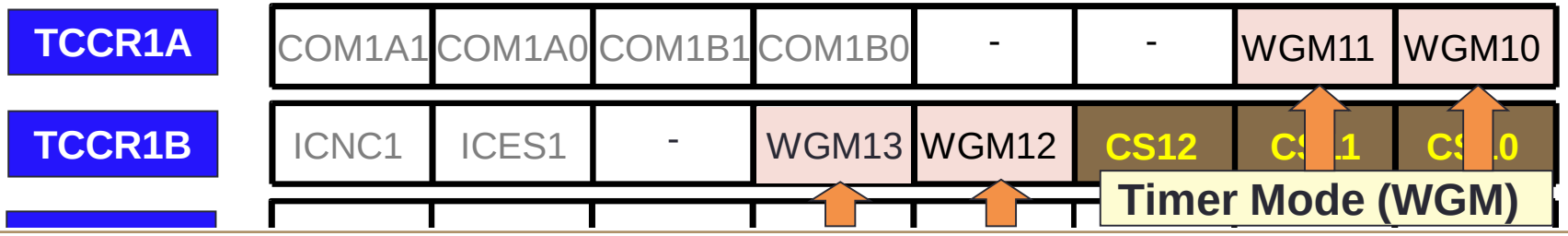
Timer1

- 16-bit register
  - TCNT1H + TCNT1L
  - OCR1AH + OCR1BH
  - OCR1BH + OCR1BL
- 8-bit register
  - TCCR1A
  - TCCR1B
  - TCCR1C
- Flag in register TIFR1
  - TOV1 in bit 0
  - OCF1A in bit 1
  - OCF1B in bit 2
  - ICF1 in bit 5



TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

Timer1 Control Register



Mode	WGM 13	WGM 12	WGM 11	WGM 10	Timer/counter mode	TOP	Update of OCR1x at	TOV1 flag set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Timer1 Control Register

TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
TCCR1C	FOC1A	FOC1B	-	-	-			

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 T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

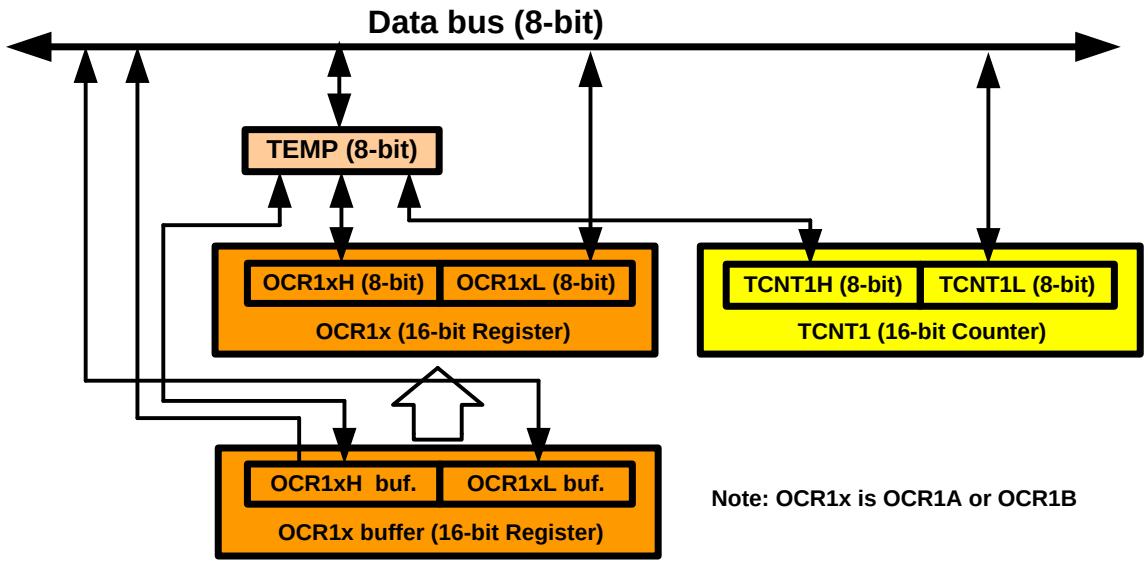
Read and Write 16-bit Registers

- To write a 16-bit register, the high byte must be written first
- To read a 16-bit register, the low byte must be read first

TCNT1H	TCNT1L

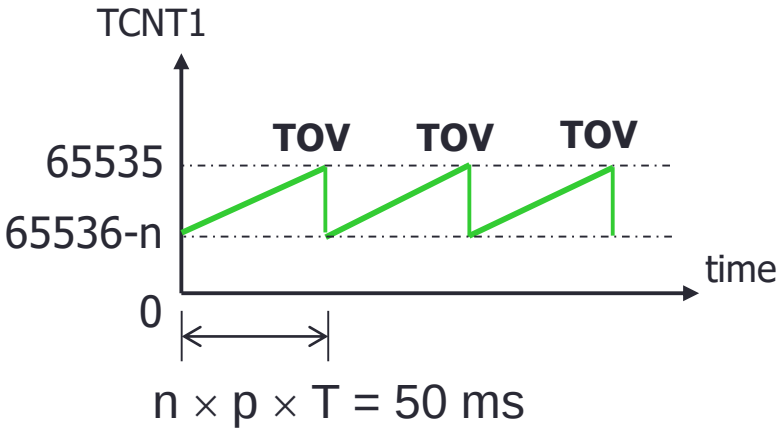
OCR1AH	OCR1AL

OCR1BH	OCR1BL



Example: Square Wave at 10Hz (Time1)

- A MCU runs at 1MHz
- Generate a square wave at 10Hz on PD0
- Use Timer1



- Timer setup:
  1. No prescaling, p = 1
  2. Period of clock T = 1/1MHz = 1µs
  3. Number of machine cycles n = 50ms / 1 µs / 1 = 50,000

TCNT1 = 65,536 – 50,000 = 15,536 = 0x3CB0

TCNT1H TCNT1L

Square Wave at 10Hz (Time1, Normal)

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

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=(1<<CLKPS1)|(1<<CLKPS0);        // set clk to 1Mhz
    DDRD=(1<<PORTD0);                     // PD0 as output
    PORTD=0;                              // initial output 0
    TCCR1A=0;                             // normal mode, int clk
    TCCR1C=0;
    while (1){
       TCNT1H=0x3C;
       TCNT1L=0xB0;
       TCCR1B=(1<<CS10);                  // p=1, start Timer
       while ((TIFR1&(1<<TOV1))==0);      // wait for flag TOV1=1
       TCCR1B=0;                          // stop Timer
       TIFR1=(1<<TOV1);                   // clear TOV1
       PORTD^=0b00000001;
    }
}
```

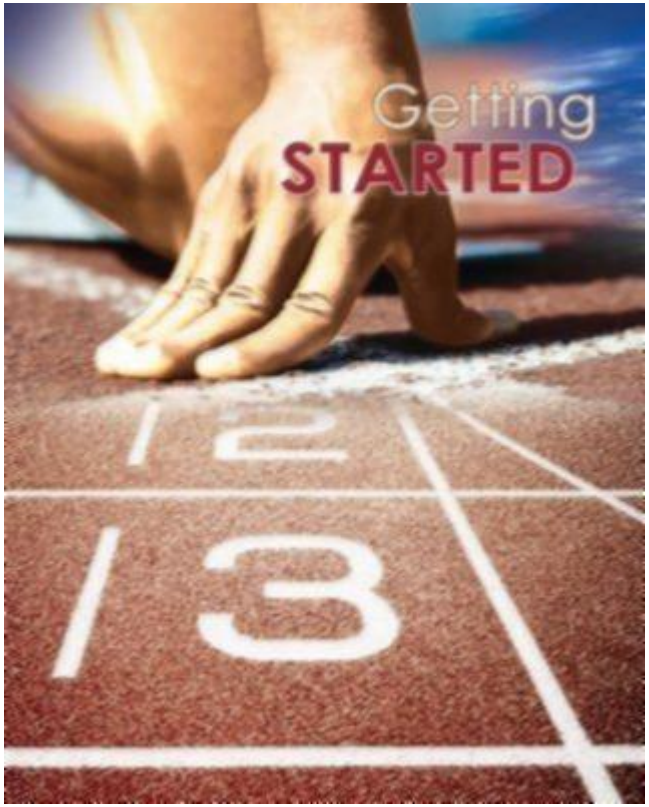
Square Wave at 10Hz (Time1, CTC)

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

int main(void)
{
    CLKPR=(1<<CLKPCE);
    CLKPR=(1<<CLKPS1)|(1<<CLKPS0);        // set clk to 1Mhz
    DDRD=(1<<PORTD0);                     // PD0 as output
    PORTD=0;                              // initial output 0
    TCCR1A=0;                             // normal mode, int clk
    TCCR1C=0;
    OCR1AH=0xC3;                          // OCR1A=49,999
    OCR1AL=0x4F;
    while (1){
       TCCR1B=(1<<WGM12)|(1<<CS10);       // p=1, start Timer
       while ((TIFR1&(1<<OCF1A))==0);     // wait for flag TOV1=1
       TCCR1B=0;                          // stop Timer
       TIFR1=(1<<OCF1A);                  // clear TOV1
       PORTD^=0b00000001;
    }
}
```

Outline (Cont'd)

- Clock system
  - Source option
  - Prescaler option
- Timer/counter
  - Timer0 registers
  - Timer0 normal mode
  - Timer0 CTC mode
  - Timer2 and Timer1
- Getting started



Reference

- ATmega328P data sheet
- AVR 8-bit instruction set
- AVR072: Accessing 16-bit I/O Registers
- AVR130: Setup and Use the AVR Timers
- M. A. Mazidi, S. Naimi, and S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*, Prentice Hall, 2010
- AVR GCC library help http://nongnu.org/avr-libc/user-manual/modules.html