# CprE 288 – Introduction to Embedded Systems (Output Compare and PWM)

Instructors:
Dr. Zhao Zhang (Sections A, B, C, D, E)
Dr. Phillip Jones (Sections F, G, J)

# Overview of Today's Lecture

- Announcements
- Output Compare and Pulse Wave Modulation (PWM)

# Announcements

- Homework 7 is due on Thursday

# Output Compare and PWD

**Generate certain digital waveforms for control purposes**

Recall Input Capture: Recognizes digital waveforms

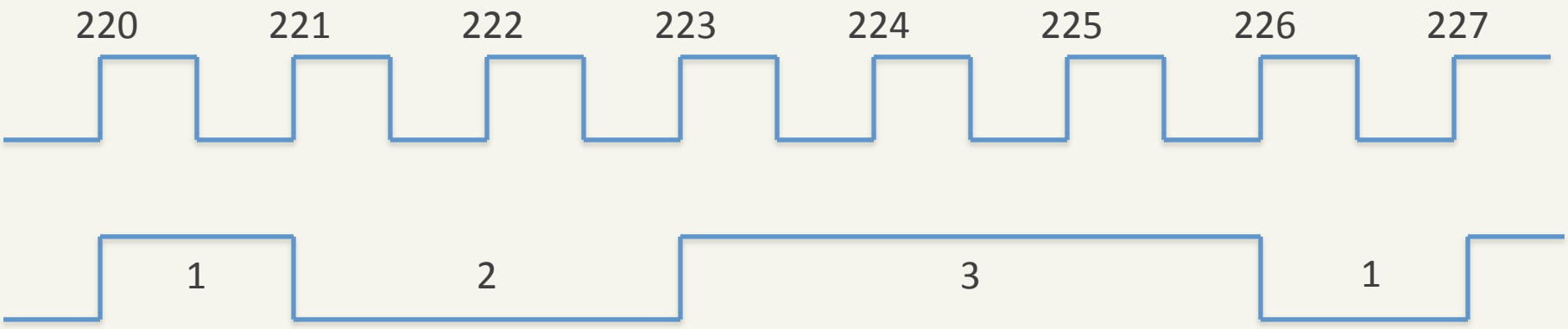
Many applications in microcontroller applications:

- Start analog devices
- Control speed of motors
- Control power output rate
- Communications
- Control servo (lab 8)

# Output Capture

Example: Generate a waveform with output events (transitions) at 220, 221, 223, 226, and 227 with initial state as low

The waveform is 1-cycle high, 2-cycle low, 3-cyle high, and 1-cycle low



# Output Compare: Design Principle

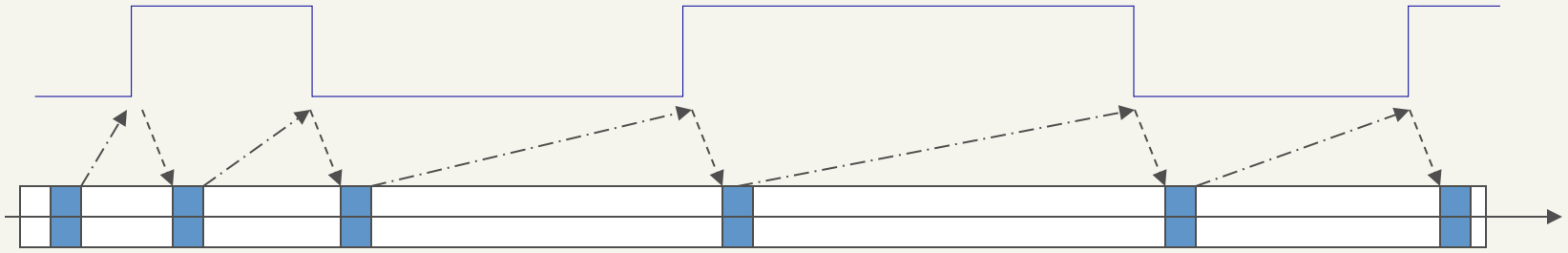
**Time is important**

How could a microcontroller generate events at precise time intervals?

- Use time delay functions?
  - CPU cannot do anything else
  - Not accurate
- Use interrupts?
  - Not necessarily accurate, because of interrupt overhead and possibly delays by other interrupts

# Output Compare: Example

**Solution: Preset the time of each event!**



CPU sets next event time

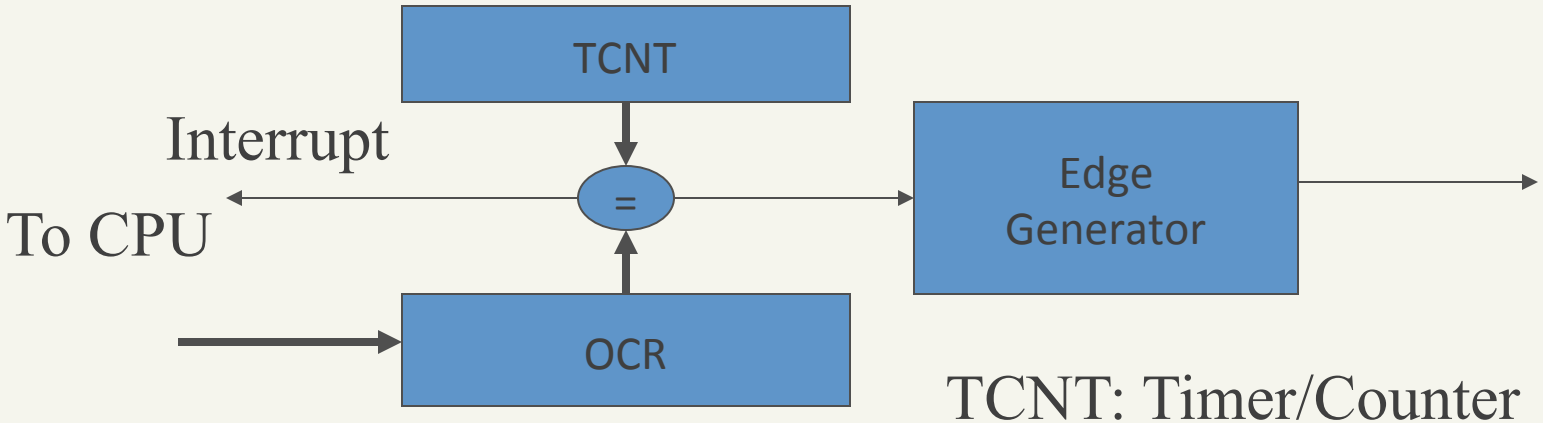
Interrupt to CPU

CPU Interrupt processing

CPU Foreground computation

# Output Compare: Design Principle

Time value (clock count) is set first by the CPU and then used by the output compare unit




TCNT: Timer/Counter
OCR: Output Compare Register

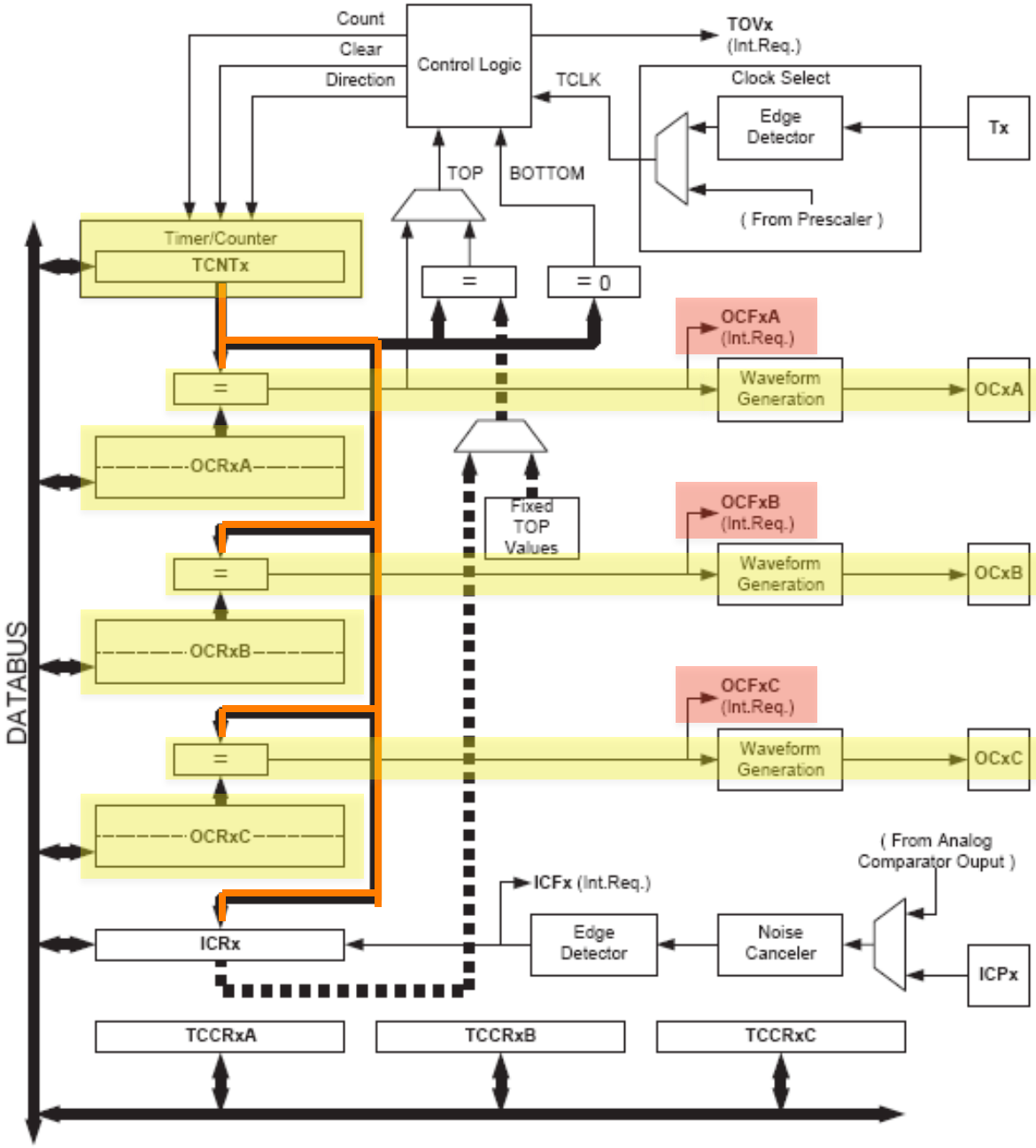
Under what condition does the hardware generate the precise waveform?

A: The CPU is not overloaded by interrupt processing

# ATmega128 16-bit Timer/Counter

ATMega128 has two, multi-purpose 16-bit timer/counter units

- One input capture unit
- **Three independent output compare units**
- Pulse width modulation output
- Frequency generator
- And other features




Output Compare in Timer/Counter

TCNTx: Timer/Counter
OCR*xY*: Output Compare Reg
OC*xY*: Output Compare Pin

# ATmega128 16-bit timer/counter

# ATmega128 16-bit Timer/Counter

Port Connection:

Timer 1, Channels A, B, C

=> Port B pins 5, 6, 7

SERV1A, SERV1B, SERV1C on board

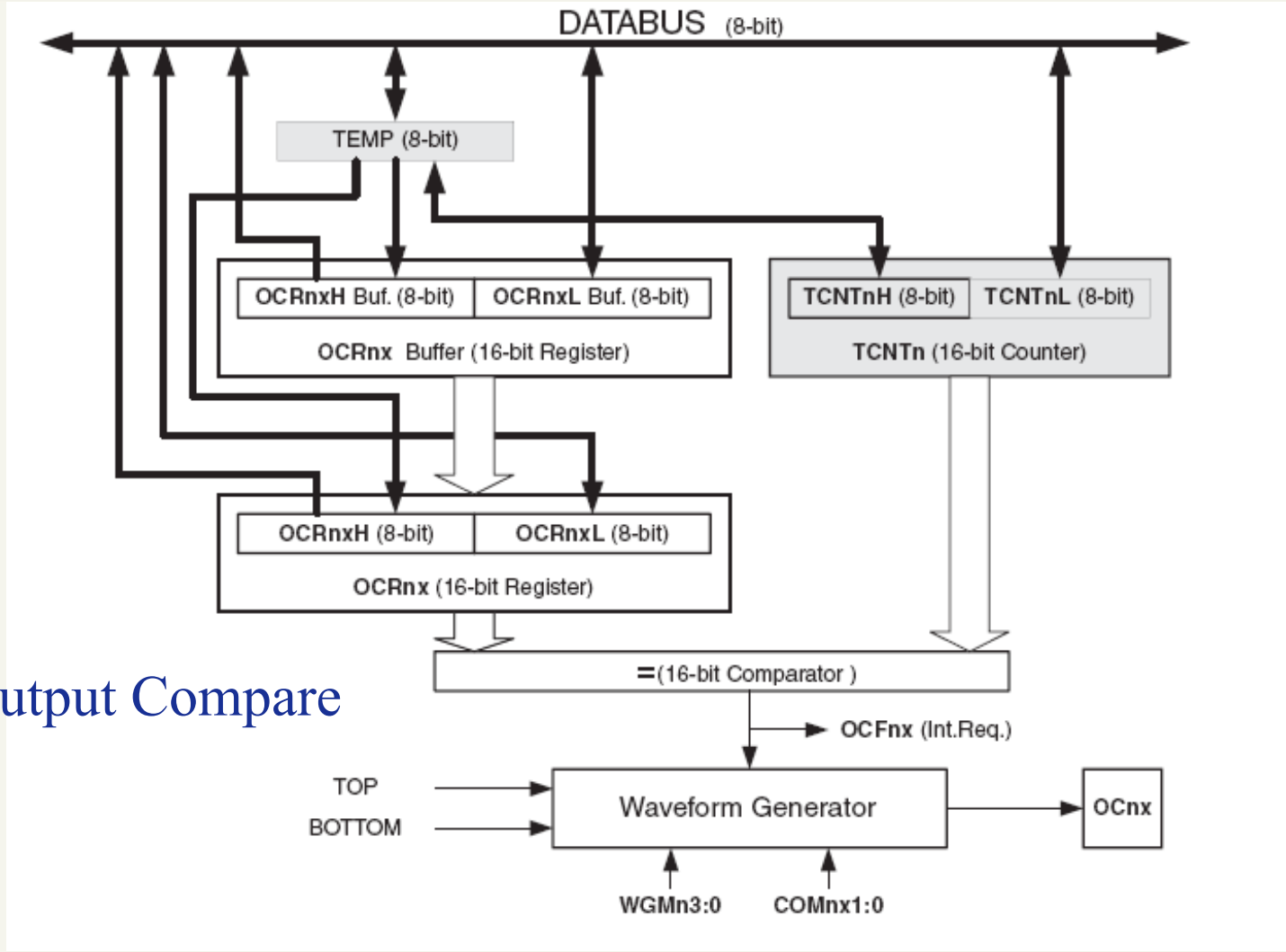
Timer 3, Channels A, B, C

=> Port E pins 3, 4, 5

SERV3A, SERV3B, SERV3C on board

Note: Those pins have to be configured as output pin to be used with OC

# Output Compare and Waveform Generator




OCR: Output Compare Register

# WGM: Waveform Generation Modes

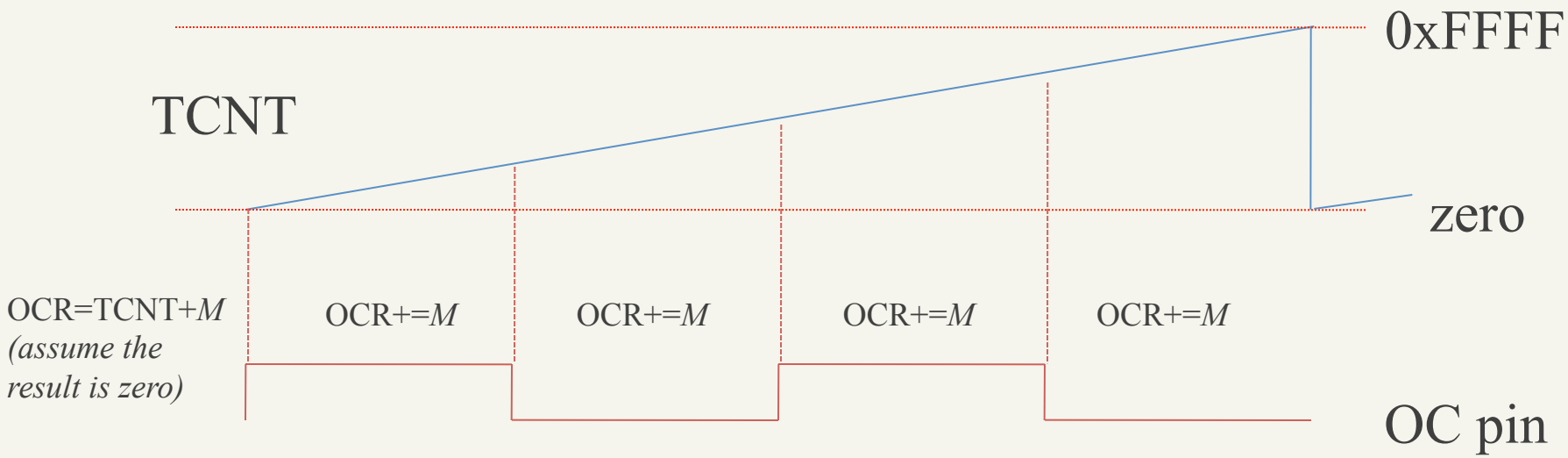
Sixteen WGMs, three categories:

- Normal Mode: ISR sets the new value to OCR(s) after a match event
  - Precisely, output compare match event

- CTC Modes and PWM Modes: Hardware sets the new value in OCR(s) after a match event
  - CTC Modes: To generate square waveforms
  - PWM modes: To generate pulse-width-modulation waveforms

# Normal Mode

Use normal mode to generate square mode




Cycle time: $2 \times M$

# Programming Example

```
// Use Normal Mode to generate a square waveform of
// 2M timer cycles, using Timer/Counter 1

unsigned count = 0;

ISR (TIMER1_COMPA_vect)
{
   count += M;
   OCR1A = count;
}
```

Note:

- May initialize "count" to a different value, e.g. count = TCNT+100;
- For this code, WGM must be 0 (0b0000) (Normal Mode)
- **Output compare mode** must be **toggle on compare match** (see TCCRnA)
- Not really suggested for square waveform because there is a more efficient way (see CTC mode)

# Programming Example: Normal Mode

To generate a **different waveform**, what's the code in the interrupt handler?

- Generate a periodic waveform repeating the following: 100-cycle low, 100 high, 200 low, 200 high, 300 low, 300 high.

# Programming Example: Normal Mode

```
// assume appropriate configuration, and
// the output is initially high

#define SEGS 6
unsigned count[SEGS] = {100, 100, 200, 200,
  300, 300};
int pos = 0;

ISR (TIMER1_COMPA_vect)
{
  OCR1A += count[pos];
  pos = (pos+1) % SEGS;
}
```

# Programming Interface: Output Compare

- **TCCRnA**: Control Register A
- **TCCRnB**: Control Register B
- **TCCRnC**: Control Register C
- **OCRn**: Output Capture Register
- **TIMSK**: Timer/Counter Interrupt Mask
- **ETIMSK**: Extended Timer/Counter Interrupt Mask
- **TCNTn**: Timer/Counter Register

Note

- *Only partial interface related to Output Compare is shown*
- n can be 1 or 3
- Use Timer/Counter 3 in the following discussions

# Programming Interface: Output Compare

Inside TCCRs, regarding Output Compare:

- **COM 1:0** (A): Compare Output Mode
- **WGM 3:0** (A, B): Waveform Generator Mode
- **CS 2:0** (B): Clock Select
- **FOC 2:0**: Force Output Compare

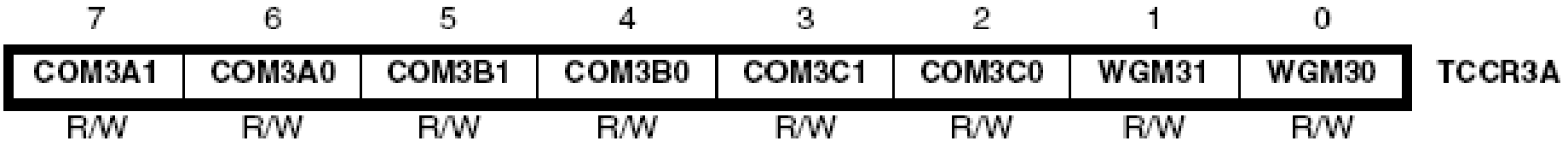
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| COM3A1 | COM3A0 | COM3B1 | COM3B0 | COM3C1 | COM3C0 | WGM31 | WGM30 | TCCR3A |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

**COMn*Xm* bits: Compare Output Mode – what to do when a *match event* happens**

**n – timer index, X – channel index, m – bit index**

COM code for **Normal and CTC Modes**

| COMnA1/COMnB1/ COMnC1 | COMnA0/COMnB0/ COMnC0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OCnA/OCnB/OCnC disconnected. |
| 0 | 1 | Toggle OCnA/OCnB/OCnC on compare match. |
| 1 | 0 | Clear OCnA/OCnB/OCnC on compare match (set output to low level). |
| 1 | 1 | Set OCnA/OCnB/OCnC on compare match (set output to high level). |



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| COM3A1 | COM3A0 | COM3B1 | COM3B0 | COM3C1 | COM3C0 | WGM31 | WGM30 | TCCR3A |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

**WGM31, WGM30**: Waveform Generator Mode to select Timer/Counter function. Four bits in total (**WGM33** and **WGM32** in TCCR3B)

WGM uses 4-bit encoding

1. **Normal mode**
2. **CTC modes**
3. **Fast PWM modes**
4. Phase Correct PWM Mode
5. Phase and Frequency Correct PWM Mode

**Table 61.** Waveform Generation Mode Bit Description

| Mode | WGMn3 | WGMn2 (CTCn) | WGMn1 (PWMn1) | WGMn0 (PWMn0) | Timer/Counter Mode of Operation[1] | TOP | Update of OCRnX at | TOVn 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 | OCRnA | 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 | ICRn | BOTTOM | BOTTOM |
| 9 | 1 | 0 | 0 | 1 | PWM, Phase and Frequency Correct | OCRnA | BOTTOM | BOTTOM |
| 10 | 1 | 0 | 1 | 0 | PWM, Phase Correct | ICRn | TOP | BOTTOM |
| 11 | 1 | 0 | 1 | 1 | PWM, Phase Correct | OCRnA | TOP | BOTTOM |
| 12 | 1 | 1 | 0 | 0 | CTC | ICRn | Immediate | MAX |
| 13 | 1 | 1 | 0 | 1 | (Reserved) | – | – | – |
| 14 | 1 | 1 | 1 | 0 | Fast PWM | ICRn | BOTTOM | TOP |
| 15 | 1 | 1 | 1 | 1 | Fast PWM | OCRnA | BOTTOM | TOP |

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

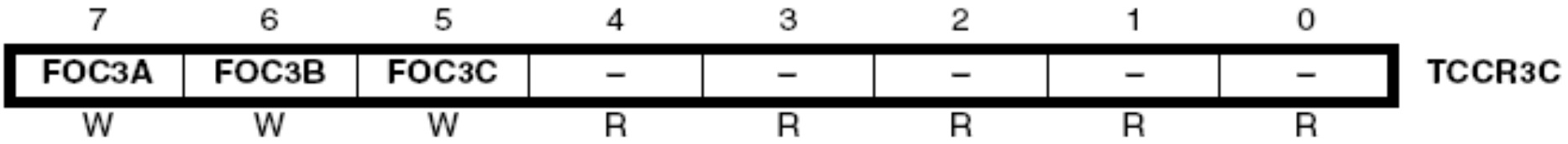
Waveform Generation Mode Bit: Data sheet page 135

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| ICNC3 | ICES3 | – | WGM33 | WGM32 | CS32 | CS31 | CS30 | TCCR3B |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |

**CS3x**: **Clock Select** bits (same as for Interrupt Capture)
Table in ATmega128 User Guide, page 137

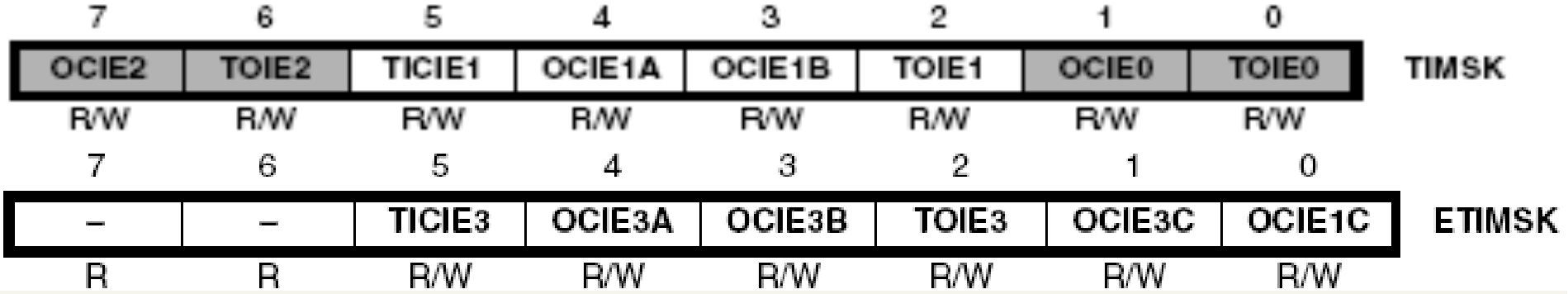
| CSn2 | CSn1 | CSn0 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source. (Timer/Counter stopped) |
| 0 | 0 | 1 | $clk_{I/O}$/1 (No prescaling |
| 0 | 1 | 0 | $clk_{I/O}$/8 (From prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (From prescaler) |
| 1 | 0 | 0 | $clk_{I/O}$/256 (From prescaler) |
| 1 | 0 | 1 | $clk_{I/O}$/1024 (From prescaler) |
| 1 | 1 | 0 | External clock source on Tn pin. Clock on falling edge |
| 1 | 1 | 1 | External clock source on Tn pin. Clock on rising edge |

Note: ICNC3, ICES3 are For Input Capture Noise Canceller and Edge Select




FOC: Force output compare

- When writing a logical one to a FOC bit, an immediate compare match is forced on the waveform generation unit. The OC output is changed according to the corresponding COM bits setting.
- In other words, writing 1 to a FOC bit causes an artificial match event on that channel.



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| OCIE2 | TOIE2 | TICIE1 | OCIE1A | OCIE1B | TOIE1 | OCIE0 | TOIE0 | TIMSK |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| – | – | TICIE3 | OCIE3A | OCIE3B | TOIE3 | OCIE3C | OCIE1C | ETIMSK |
| R | R | R/W | R/W | R/W | R/W | R/W | R/W | |

Interrupt enable bits for Timer/Counter 1 and 3

TICIEn: Input Capture Interrupt Enable

OCIEnA, OCIEnB, OCIEnC: Output Compare Match Interrupt Enable for channel A, B, and C, respectively

TOIEn: Overflow Interrupt Enable

# Programing Example: Normal Mode

Initialize Timer/Counter 3's OC unit as normal mode, 64 prescalar, toggle-on-match, and uses channel A only, interrupt enabled on output compare match

```
timer_init()
{
  DDRE |= _BV(3);   // OC3A is pin 3
  // COM3A=01, WGM=0000, CS3=011, FOC3A=0
  // OCIE3A=1
  TCCR3A = _BV(COM3A0);
  TCCR3B = _BV(CS31)|_BV(CS30);
  TCCR3C = 0;
  ETIMSK = _BV(OCIE3A);
}
```

# CTC Mode: How It Works

## CTC: Clear Timer on Compare Match

In CTC mode, the **OCRnA** or **ICRn** Register are used to manipulate the counter resolution (# of steps)

The **TCNTn** counter increments to **TOP** and then resets to zero, repeatedly

- WGM = 4 (0100): Use OCRnA as TOP
- WGM = 12 (1100): Use ICRn as TOP

What is the trade-off of using OCRnA as TOP?

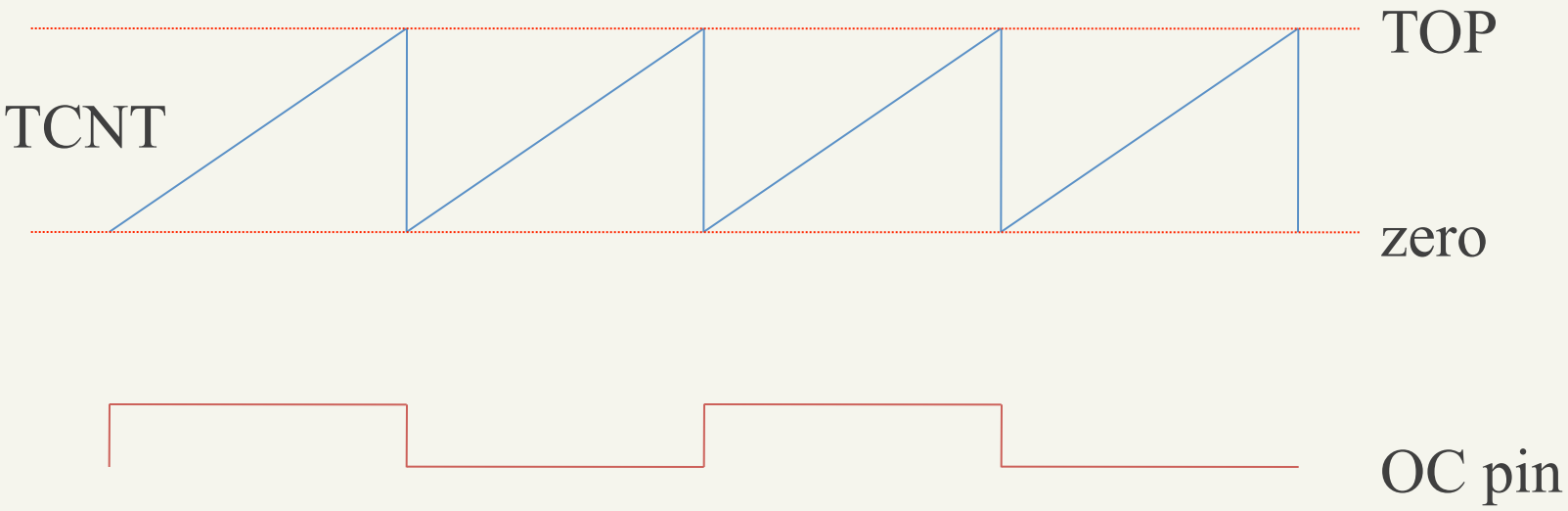
How about ICRn?

# CTC Mode: How It Works

The OCRnA or ICRn defines the **top value** for the counter. This mode allows greater control of the compare match output frequency.

# CTC Mode: Match Events

Generate square waveform continuously




Cycle time: 2 × (TOP + 1)

Why is CTC mode better than Normal mode for this purpose?

# CTC Mode: Frequency

Generate periodic square waveform

Time interval: 1 + OCRnA

- TCNTn: 0, 1, 2, …, OCRnA, 0, 1, 2, …
- One pulse is two events

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

# CTC Mode: Square Waveform

Generate a square waveform: Initialize Timer/Counter 3's OC unit as **CTC** mode 4 (OCRnA as TOP), 64 prescalar, toggle-on-match, and uses channel A only, with 200-cycle period. **No interrupt is needed**.

```
timer_init()
{
  DDRE |= _BV(OC3A);
  OCR3A = (100-1);
  //COM3A=01,WGM=0100,FOC3A=0,CS3=011,OCIE3A=0
  TCCR3A = _BV(COM3A0);
  TCCR3B = _BV(CS31)|_BV(CS30)|_BV(WGM32);
  TCCR3C = 0;
}
```

# CTC Mode: Square Waveform

No interrupt processing overhead, which includes

- Save and restore CPU registers
- Save and restore PC and other special registers
- Find out the interrupt source
- Call and execute the Interrupt Service Routine (ISR) function

**Overall Interrupt Overhead = Interrupt Frequency × Average Overhead per Interrupt**

# CTC Mode Programming Example

**Example**: Use CTC mode to generate timer interrupt (lab 4)

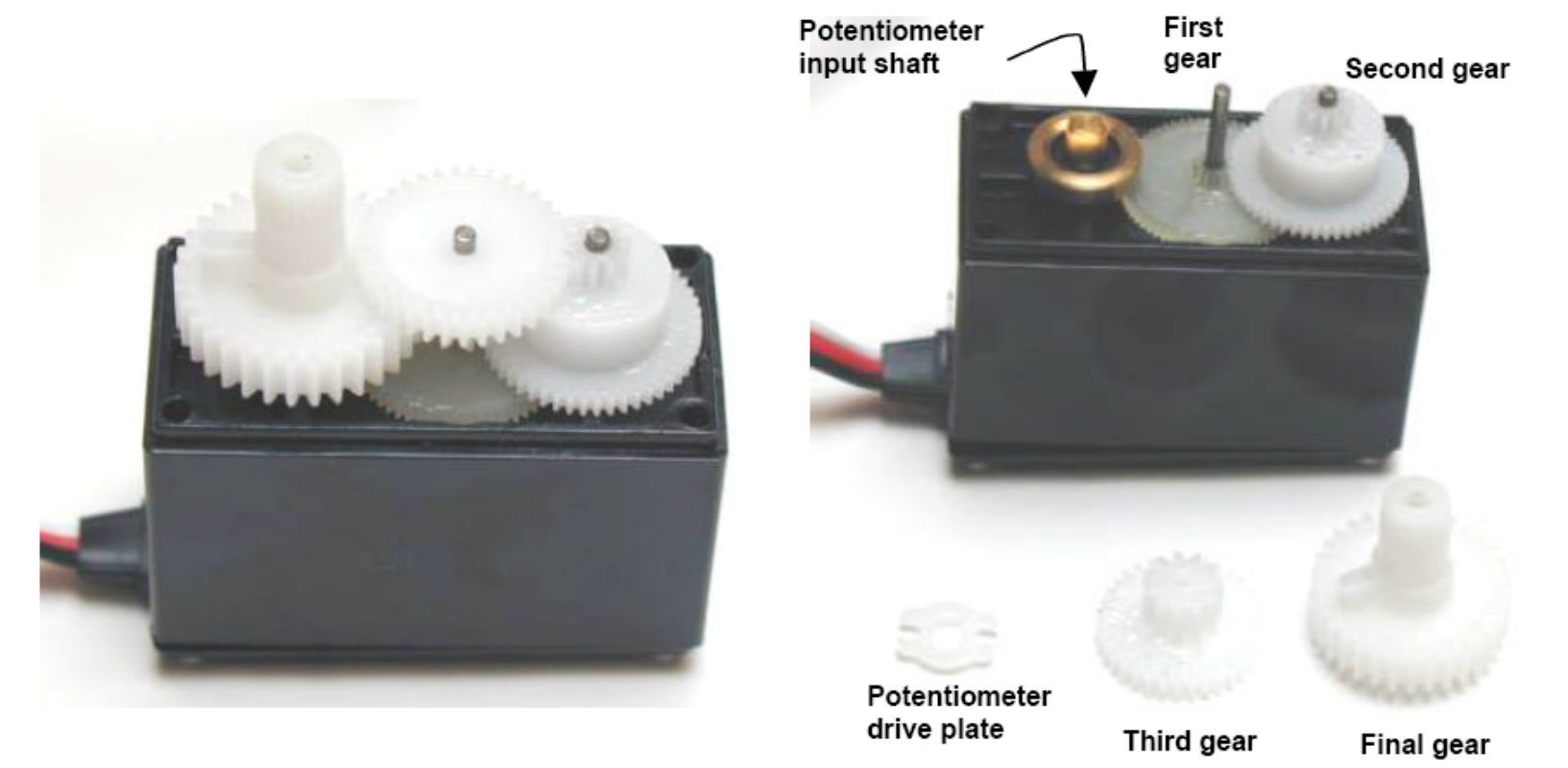
```
void timer_init(void)
{
  TCCR1A = 0b00000000; // WGM1[1:0]=00
  TCCR1B = 0b00001101; // WGM1[3:2]=01, CS=101
  TCCR1C = 0b10000000; // FOC1A=1
  OCR1A = CLOCK_COUNT - 1;
  // enable OC interrupt, timer 1, channel A
  TIMSK = _BV(OCIE1A);
}
```

# Servo Control

A **servo** is a special motor with built-in position feedback

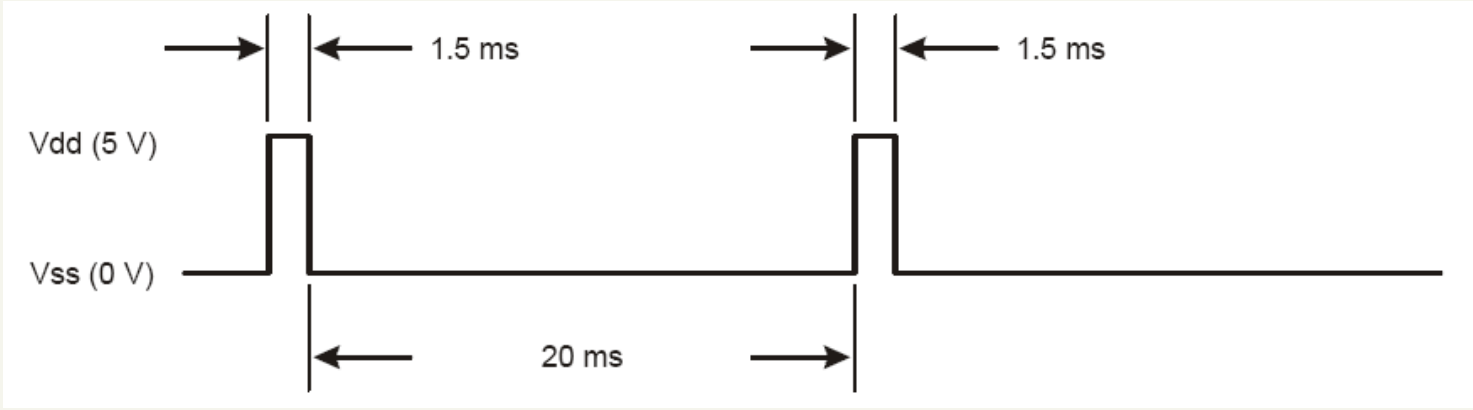
- Can stop the shaft at a given position
- Relatively precise
- Need calibration

# Servo Control




Source: Parallax Robotics Student guide, V1.4

# Servo Control




Digital waveform to put the servo shaft at the center position

# Servo Control

Lab 8 draft

1. Send pulses to the servo to make it move
2. Make the servo stop at the center position
3. Make the servo stop at different degrees of angle, do calibration

   0, 45, 90 (center), 135, 180

# Servo Control

Use digital waveform to inform the servo the target position

1ms pulse – clockwise far end

1.5ms pulse – center position

2ms pulse – counterclockwise far end

20ms interval between pulses (doesn't have to be precise)

***Must repeat until shaft arrives the target position***

# Servo Control

Programming tasks: Generate a periodic waveform with a certain pulse width and a fixed period

1.0~2.0ms corresponds to 0~180 degree counterclockwise

***NOTE: Suggested by servo's document; calibration IS necessary***

# Servo Control

Programming Approaches

- Use Output Compare normal model
- Use Output Compare CTC model
- Use PWM (to be discussed)

Are the first two good choices?

- Is interrupt overhead acceptable?
- When can overhead be a problem?

# Pulse Width Modulation

Parameters: Period Length and Pulse Width

**Duty Cycle** = Pulse width / Period Length

Programming: How to set the two parameters?

*Note: Duty cycle is not important to Lab 8, but it is to most applications*

# ATmega128 PWM

Three types of PWM

- **Fast mode PWM**
- Phase correct PWM
- Phase and Frequency Correct PWM

The other two modes: Waveform is different, the desired duty cycle is the same

- Not required to learn in this course

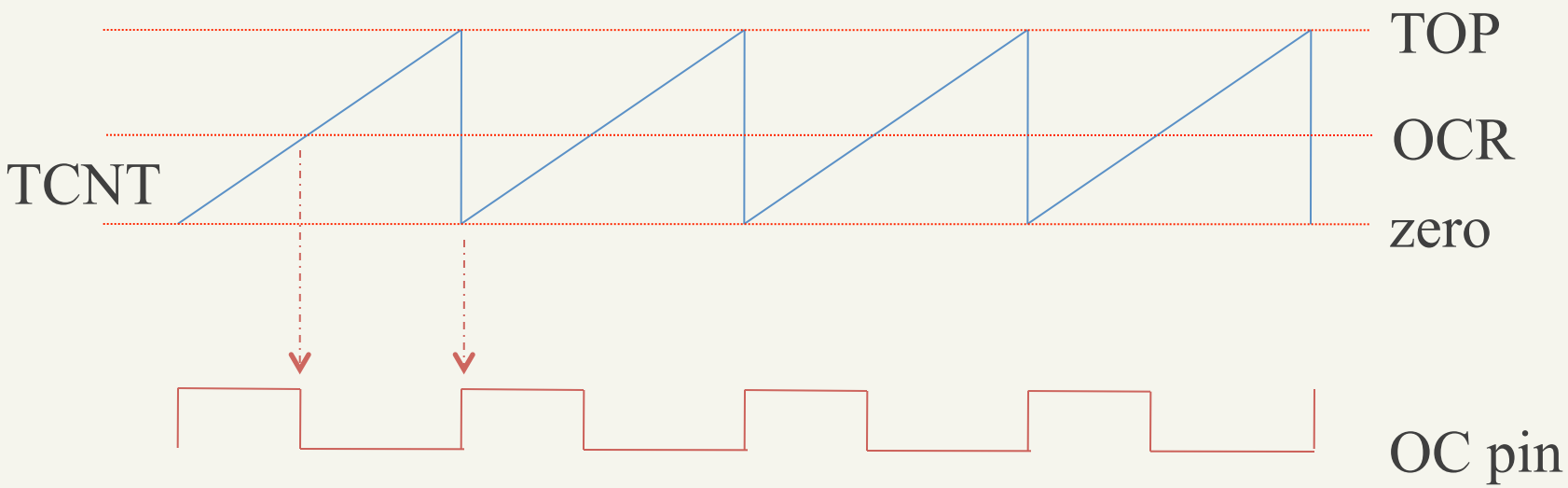
# ATMmega128 PWM

Fast PWM:

- TCNTn increments every cycle
- First compare match event occurs when TCNTn is reaches OCRnX.
- Second compare match event occurs when TCNTn is reset (after reaches a TOP and is reset to BOTTOM)

# Fast Mode PWM

Generate single-slope PWM waveform continuously



# ATmega128 PWM

ATmega128 has WGMs that use fixed TOP

In two WGMs, either ICRn or OCRnA is used as the TOP

In lab 8, the *suggestion* is to

- Use WGM 1111
- Use OCRnA to set the TOP (pulse interval – 1)
- Use OCRnB or OCRnC to store (pulse width-1) (depends on whether channel B or C is used)

*WGM: Waveform Generation Mode*

# Programming COM Bits

COM bits: When to generate events, and what event to generate.

***Their meanings change with WGM***

# COM for Fast PWM

**Table 59.** Compare Output Mode, Fast PWM

| COMnA1/COMnB1/ COMnC1 | COMnA0/COMnB0/ COMnC0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OCnA/OCnB/OCnC disconnected. |
| 0 | 1 | WGMn3:0 = 15: Toggle OCnA on Compare Match, OCnB/OCnC disconnected (normal port operation).<br>For all other WGMn settings, normal port operation, OCnA/OCnB/OCnC disconnected. |
| 1 | 0 | Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM, (non-inverting mode) |
| 1 | 1 | Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM, (inverting mode) |

# PWM Mode: Programming Example

```
unsigned pulse_interval = …;                    //  pulse interval in cycles

void timer3_init()
{
  OCR3A = pulse_interval-1;          // number of cycles in the interval
  OCR3B = mid_point-1;               // if you want to move servo to the middle
  TCCR3A = …;                        // set COM and WGM (bits 3 and 2)
  TCCR3B = …;                        // set WGM (bits 1 and 0) and CS
  TCCR3C = 0;

  // it's necessary to set the OC3B (PE4) pin as the output
  DDRE |= _BV(4);                    // set Port E pin 4 (OC3B) as output
}
```

```
void move_servo(unsigned degree)
{
  unsigned pulse_width;              // pulse width in cycles

  …                                  // calculate pulse width

  OCR3B = pulse_width-1;             // set pulse width

  // you need to call wait_ms() here to enforce a delay for the servo to
  /// move to the position
}
```

# Other Fast PWM Modes

WGM bits

- 0101: 8-bit fast PWM (TOP = 256)
- 0110: 9-bit fast PWM (TOP = 512)
- 0111: 10-bit fast PWM (TOP = 1024)

What are pros and cons of using any of them in lab 8?

# Summary of OC Programming

WGM: Decides **the timing of events**

- All channels shared the same WGM setting

COM: Decides **the action upon events**

- Each channel has its own COM setting
- The exact meaning of COM setting is dependent on WGM setting

# Summary of OC Programming

Other control bits

**CS**: Clock Select (prescalar)

**TIMSK/ETIMSK**: Contain interrupt enable/disable bits for Timer/Counter 1 and 3

FOC: Force output compare

- When writing a logical one to a FOC bit, an immediate compare match is forced on the waveform generation unit. The OC output is changed according to the corresponding COM bits setting.
- In other words, writing 1 to a FOC bit causes an artificial match event on that channel.

# Summary of OC Programming Interface

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| COM3A1 | COM3A0 | COM3B1 | COM3B0 | COM3C1 | COM3C0 | WGM31 | WGM30 | TCCR3A |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| ICNC3 | ICES3 | – | WGM33 | WGM32 | CS32 | CS31 | CS30 | TCCR3B |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| FOC3A | FOC3B | FOC3C | – | – | – | – | – | TCCR3C |
| W | W | W | R | R | R | R | R | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| OCIE2 | TOIE2 | TICIE1 | OCIE1A | OCIE1B | TOIE1 | OCIE0 | TOIE0 | TIMSK |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| – | – | TICIE3 | OCIE3A | OCIE3B | TOIE3 | OCIE3C | OCIE1C | ETIMSK |
| R | R | R/W | R/W | R/W | R/W | R/W | R/W | |

**Plus: ICRn, OCRnA, OCRnB, OCRnC, TCNTn**

**Table 61.** Waveform Generation Mode Bit Description

| Mode | WGMn3 | WGMn2 (CTCn) | WGMn1 (PWMn1) | WGMn0 (PWMn0) | Timer/Counter Mode of Operation[1] | TOP | Update of OCRnX at | TOVn 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 | OCRnA | 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 | ICRn | BOTTOM | BOTTOM |
| 9 | 1 | 0 | 0 | 1 | PWM, Phase and Frequency Correct | OCRnA | BOTTOM | BOTTOM |
| 10 | 1 | 0 | 1 | 0 | PWM, Phase Correct | ICRn | TOP | BOTTOM |
| 11 | 1 | 0 | 1 | 1 | PWM, Phase Correct | OCRnA | TOP | BOTTOM |
| 12 | 1 | 1 | 0 | 0 | CTC | ICRn | Immediate | MAX |
| 13 | 1 | 1 | 0 | 1 | (Reserved) | – | – | – |
| 14 | 1 | 1 | 1 | 0 | Fast PWM | ICRn | BOTTOM | TOP |
| 15 | 1 | 1 | 1 | 1 | Fast PWM | OCRnA | BOTTOM | TOP |

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Waveform Generation Mode Bit: Data sheet page 135

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| COM3A1 | COM3A0 | COM3B1 | COM3B0 | COM3C1 | COM3C0 | WGM31 | WGM30 | TCCR3A |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

**COMn*Xm* bits: Compare Output Mode – what to do when a *match event* happens**

**n – timer index, X – channel index, m – bit index**

COM code for **Normal and CTC Modes**

| COMnA1/COMnB1/ COMnC1 | COMnA0/COMnB0/ COMnC0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OCnA/OCnB/OCnC disconnected. |
| 0 | 1 | Toggle OCnA/OCnB/OCnC on compare match. |
| 1 | 0 | Clear OCnA/OCnB/OCnC on compare match (set output to low level). |
| 1 | 1 | Set OCnA/OCnB/OCnC on compare match (set output to high level). |

# COM for Fast PWM

**Table 59.** Compare Output Mode, Fast PWM

| COMnA1/COMnB1/ COMnC1 | COMnA0/COMnB0/ COMnC0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OCnA/OCnB/OCnC disconnected. |
| 0 | 1 | WGMn3:0 = 15: Toggle OCnA on Compare Match, OCnB/OCnC disconnected (normal port operation).<br>For all other WGMn settings, normal port operation, OCnA/OCnB/OCnC disconnected. |
| 1 | 0 | Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM, (non-inverting mode) |
| 1 | 1 | Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM, (inverting mode) |

# Summary of OC Normal Mode

Good for generating **waveforms of any shapes**

Programming: Use ISR to pre-set the timing of the next event

Cons:

- Interrupt overhead can be high
- Cannot generate high-frequency waveforms
- CPU cannot sleep into deep power-saving modes

# Summary of CTC Mode

Good for generating **square waveform**

Programming

- WGM=0100: Put TOP into OCRnA
- WGM=1100: Put TOP into ICRn

Cycle time is 2 × (TOP + 1), one event for every TOP +1 cycles

Caveats:

- *Borrow either OCRnA or ICRn* to set TOP
- TCNT reset to zero after reaching TOP

# Summary of Fast PWM

Good for generating **Pulse Width Modulation Waveforms**

Two parameters: **Pulse Width** and **Period Length** (they decide the timing of two events)

Programming with WGM = 1111

- Top is OCRnA, stores Period_Length-1
- OCRnB/OCRnC stores Pulse_Width-1
- Caveats: OCRnA is borrowed to set TOP; TCNT resent to zero after reaching TOP