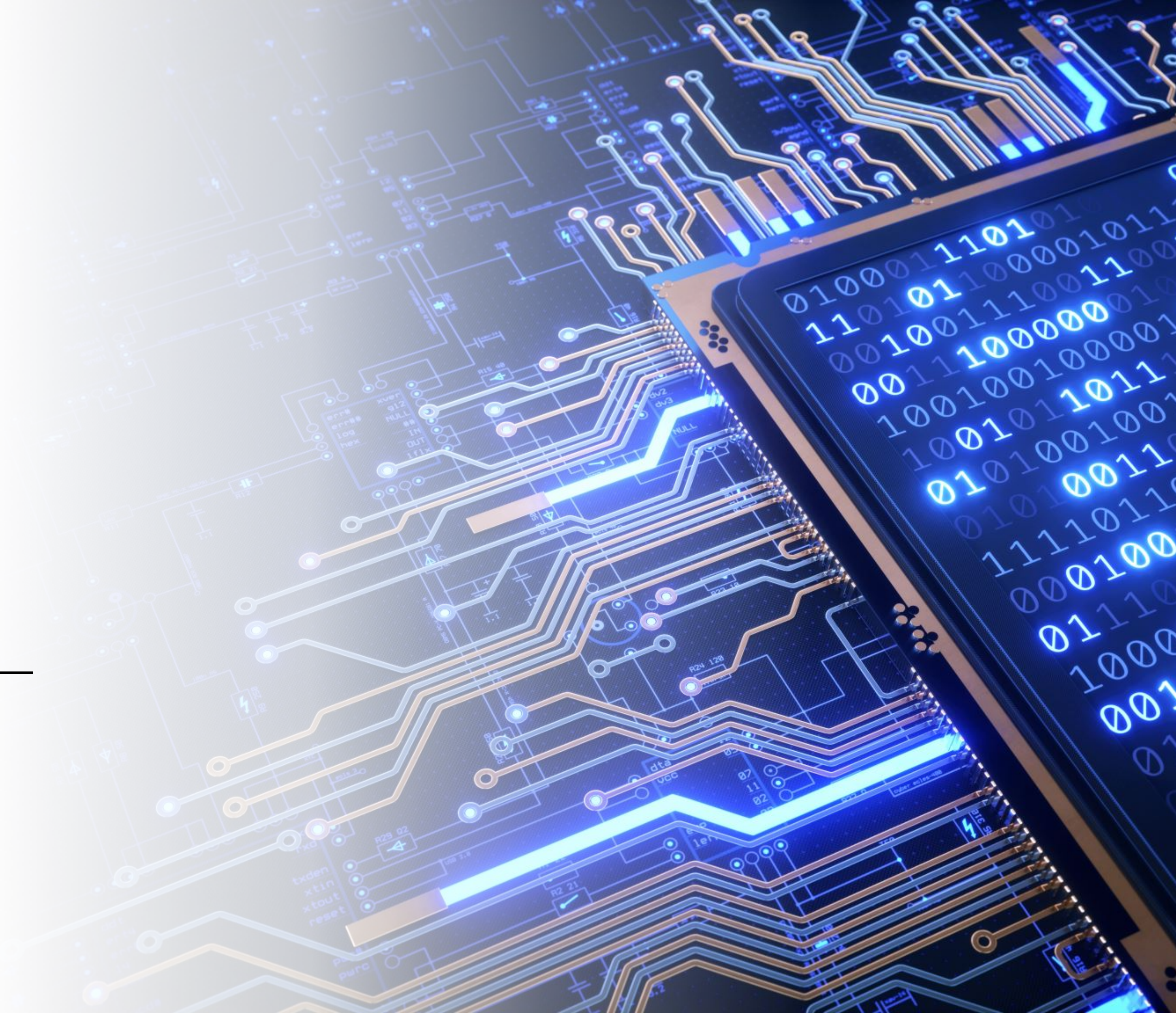
Timers and Counters



Lecture Outline

- Counters and Timer Basics
- Timers in the AVR Device Family

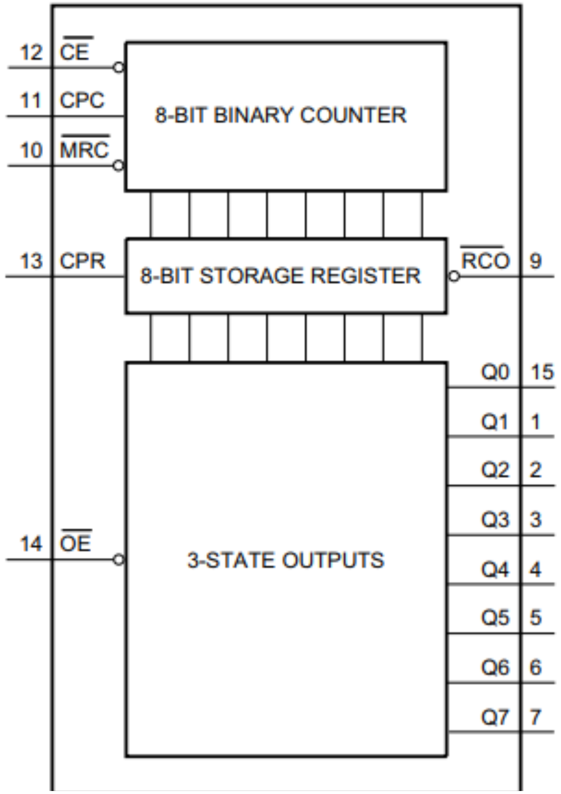
Timers and Counters in Embedded Systems

- **Counter:** A circuit that increments or decrements a number based on an input pulse
- **Timer:** A counter that is driven by a constant pulse so the time it takes to count from one number to the next is constant
- Embedded systems make extensive use of timers and counters
  - Measure duration of events
  - Count the number of events
  - Signal / clock generation

An Example Counter IC

74HC590 Counter

- Basic Operation Overview
  - Signal to be counted is sent to the CPC pin the count is incremented
  - When the CPR pin is set to high, the counter value is transferred to the storage register
  - Assuming the output is enabled (OE=0), the register value appears on the output pins Q0-Q7



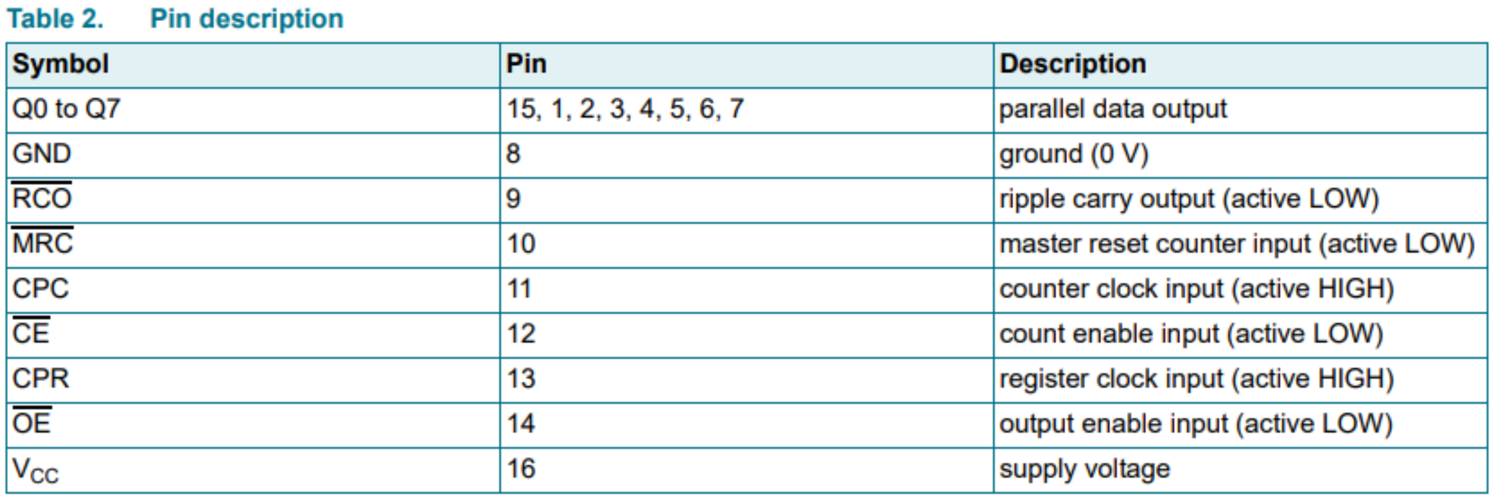
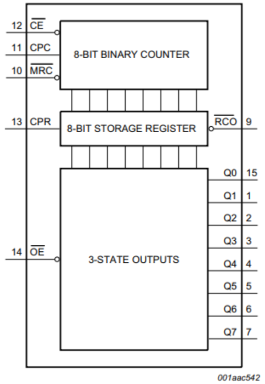


Table 2. Pin description

Symbol	Pin	Description
Q0 to Q7	15, 1, 2, 3, 4, 5, 6, 7	parallel data output
GND	8	ground (0 V)
$\overline{RCO}$	9	ripple carry output (active LOW)
$\overline{MRC}$	10	master reset counter input (active LOW)
CPC	11	counter clock input (active HIGH)
$\overline{CE}$	12	count enable input (active LOW)
CPR	13	register clock input (active HIGH)
$\overline{OE}$	14	output enable input (active LOW)
V_{CC}	16	supply voltage

Example Counter Timing Diagram

- Typical timing diagram for the 74HC590 Chip
- Sequence starts with RCO going low, indicating that the counter is starting over from 0
- NOTE: Cycles are positive edge triggered



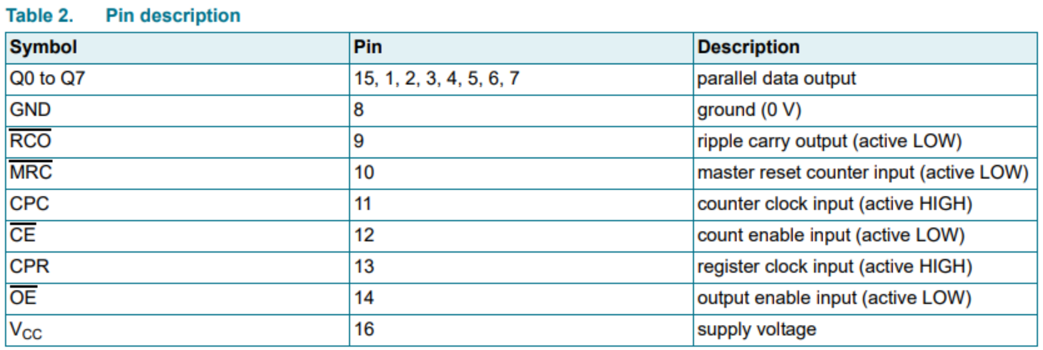
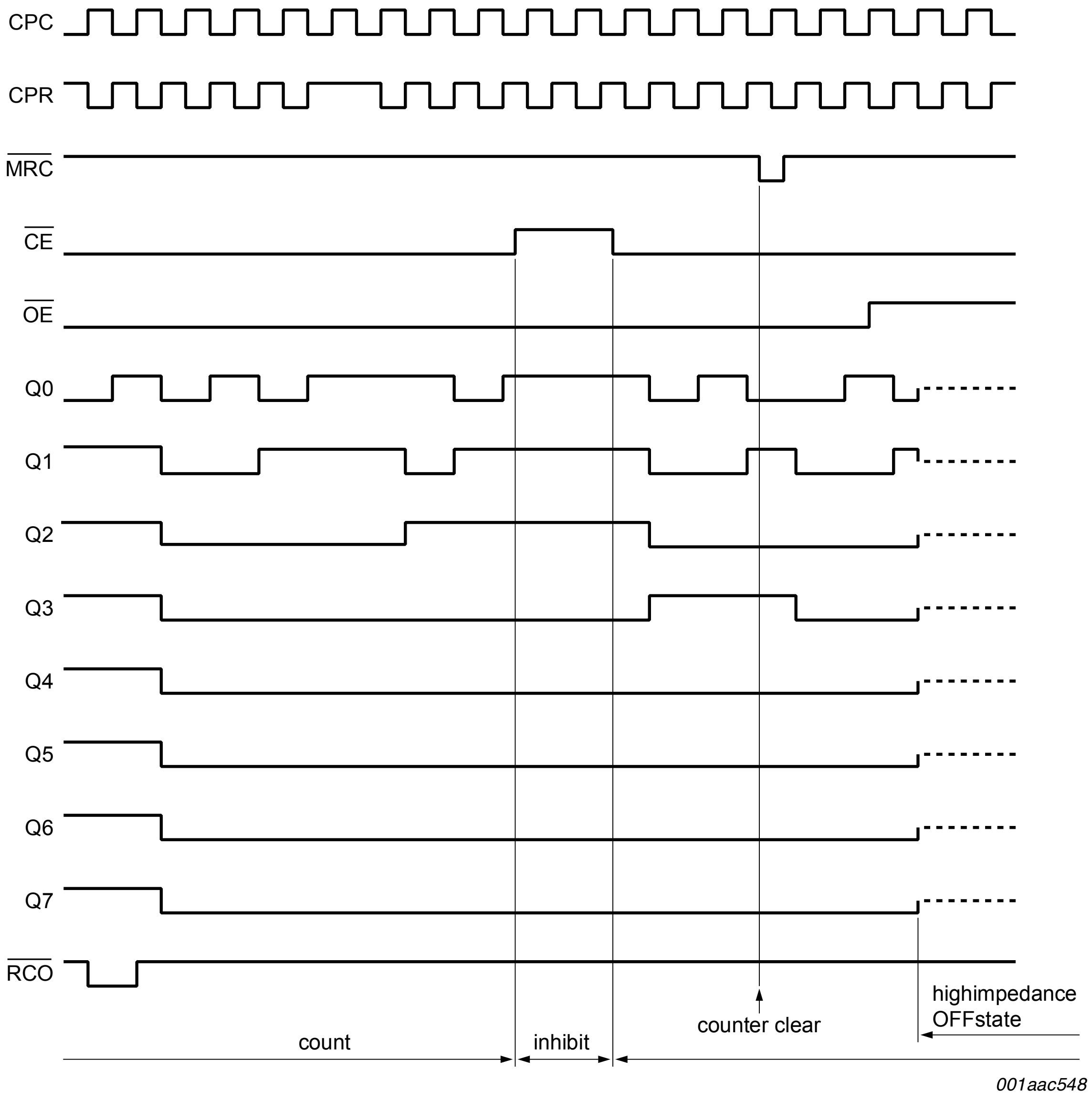


Table 2. Pin description

Symbol	Pin	Description
Q0 to Q7	15, 1, 2, 3, 4, 5, 6, 7	parallel data output
GND	8	ground (0 V)
$\overline{\text{RCO}}$	9	ripple carry output (active LOW)
$\overline{\text{MRC}}$	10	master reset counter input (active LOW)
CPC	11	counter clock input (active HIGH)
$\overline{\text{CE}}$	12	count enable input (active LOW)
CPR	13	register clock input (active HIGH)
$\overline{\text{OE}}$	14	output enable input (active LOW)
V_{CC}	16	supply voltage



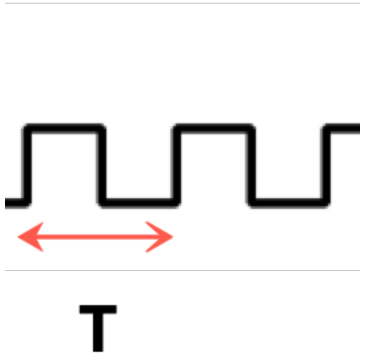
Source: https://assets.nexperia.com/documents/data-sheet/74HC590.pdf

Timers

- Timers are just counters where the counted input is a clock
- Since we know the frequency of the clock, the elapsed time for a given number of ticks can be determined
- Example:

$$f_c = 1 \text{ MHz}$$

$$T = \frac{1}{f_c} = 1\mu s$$



- If we start our timer at 123 and an event happens at count 231, the time elapsed would be (231-123)* 1 μs = 108 μs

The AVR Timers

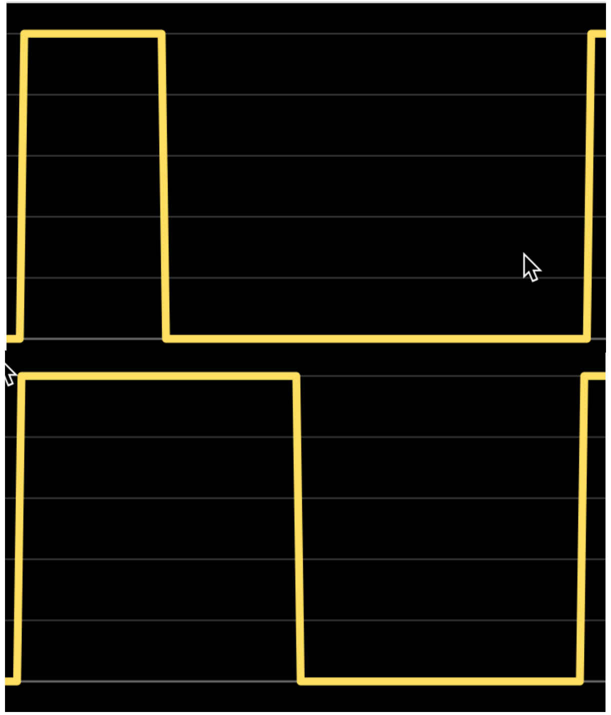
- Timers are hardware peripherals that are embedded in the microcontroller
    - In micro*processor* systems, timers would be in separate chips/systems
- The AVR family has from 1 to 6 timers
- In the 2560, there are 2 8-bit and 4 16-bit counters labelled 0 through 5
    - Timers 0 and 2 are 8-bit, timers 1 and 3-5 are 16 bit

MCU Timer Overview

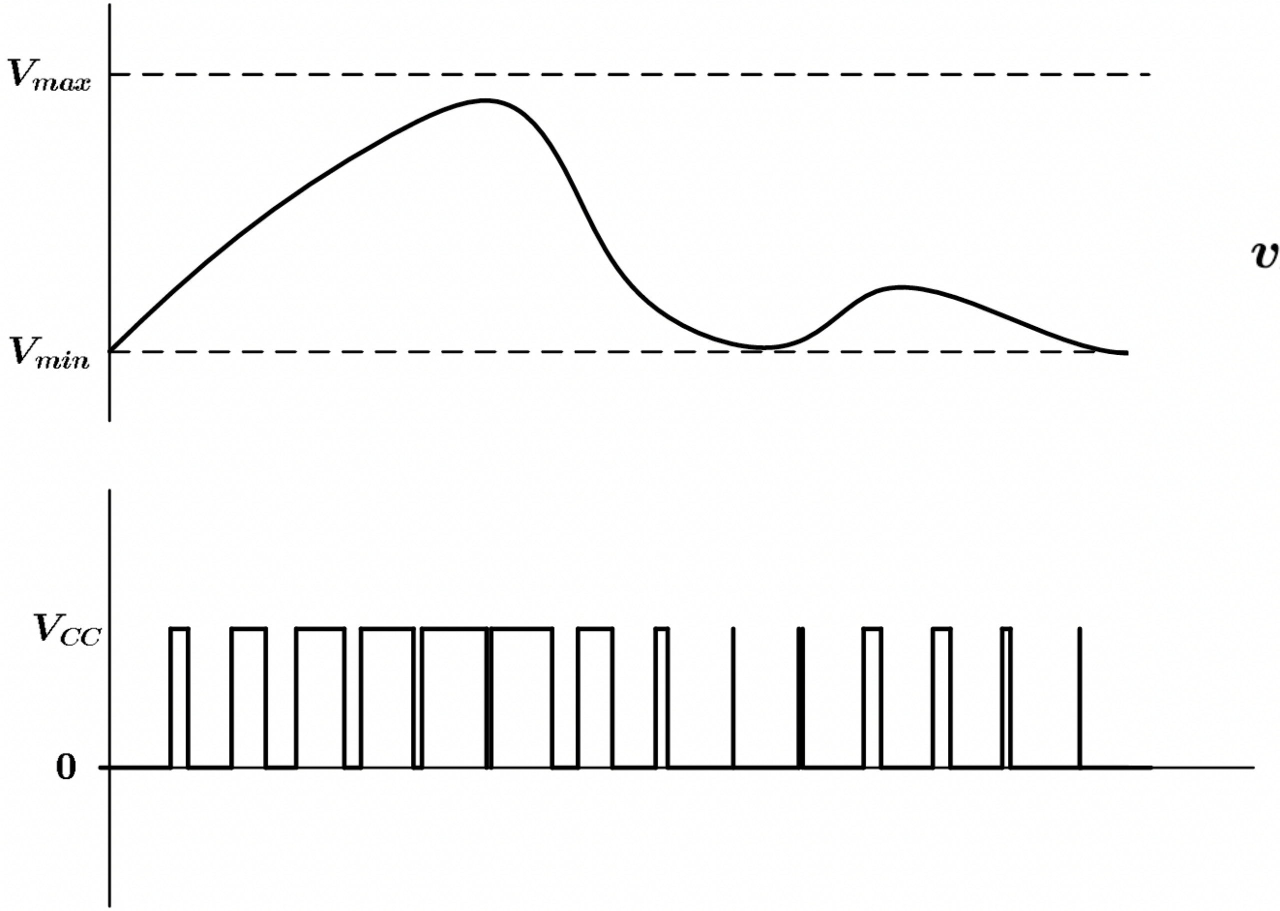
- A timer counts up (or up and down) based on a clock
- When the timer reaches its maximum value, it *rolls over*
    - When this happens, a *flag* is set
- Common approach for creating a delay:
    1. Set an initial value in the counter
    2. Start the counter
    3. Poll for the overflow flag (TOVn)
    4. Stop the counter
    5. Go to (1)

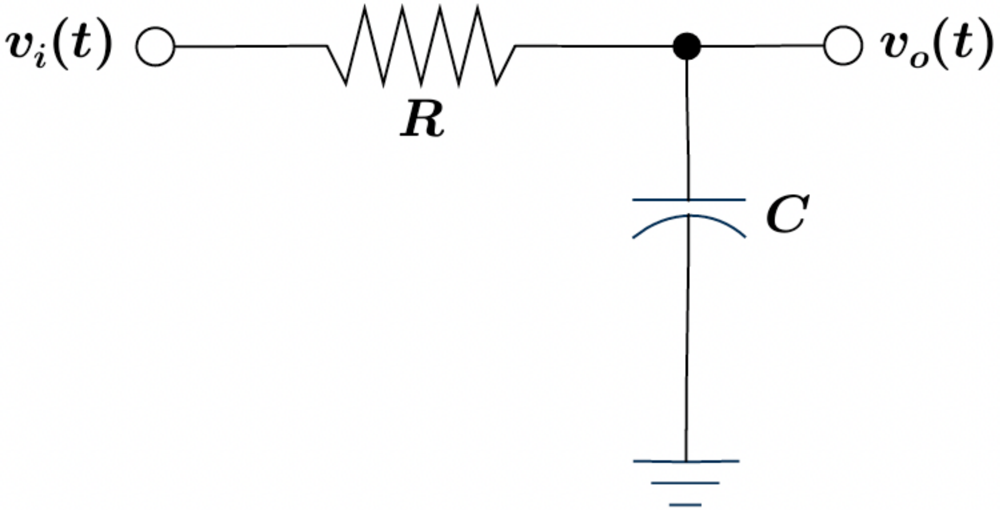
AVR Timer Details

- Each timer has two *channels*: A and B
- The output of these channels are at the same frequency but can have different pulse widths
- Timers have multiple *modes*
  - Modes determine the behavior of the counter/timer, affecting the pulse width, frequency, and relationship to the system clock



Pulse Width Modulation



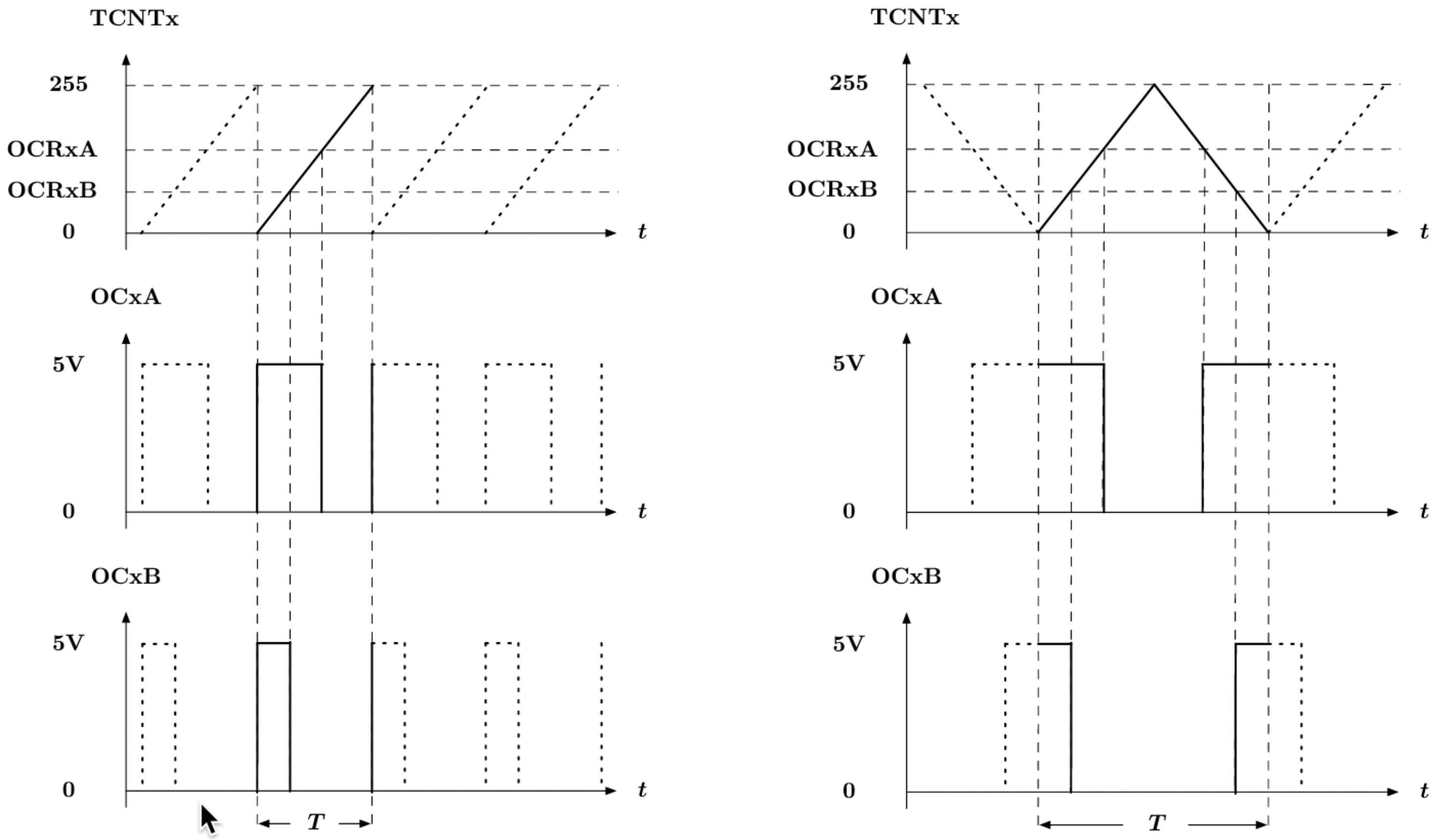


Timer Modes

- Normal mode
  - The timer always counts to its maximum value and then starts over
- Clear Timer on Compare Match (CTC) Mode
  - The OCR0A Register is used to manipulate the counter resolution.
  - In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A.
  - OC0A register can be used as output (toggles when reset)
- Fast PWM Mode
  - Used to generate fast PWM signals
  - The timer counts to its maximum value or to a predefined value based on a register settings
    - These settings effect both the frequency and the pulse width
- Phase-Correct PWM
  - The counter counts up and then back down.
  - Register settings control the pulse width
  - The generated pulses are centered on the timer counter
    - In fast PWM modes, the generated pulses are based on the leading edge of the clock

Timer Modes – TOP = $FF

- Setting the appropriate bits in the timer/counter control register (TCCR) causes the counter to its max value (called TOP)
- The output compare registers (OCRnA and B) control the width of the pulse

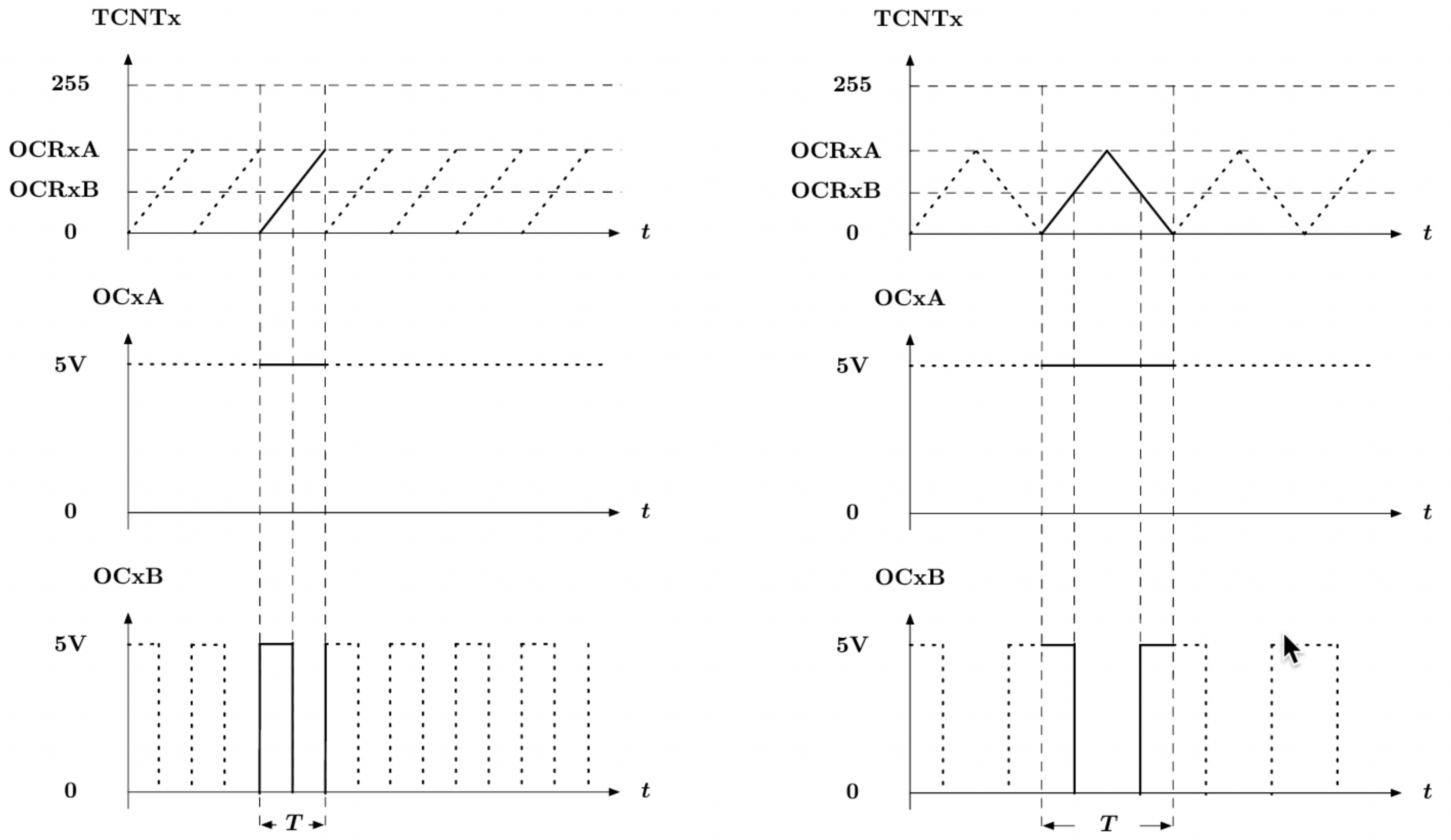


(a) Fast PWM

(b) Phase-Correct PWM

Timer Modes – TOP as OCRA Register

- Setting the appropriate bits in the timer/counter control register (TCCR) causes the counter to reset at the value stored in the output compare register A (OCRnA)
- Output compare register B (OCRnB)controls the pulse width

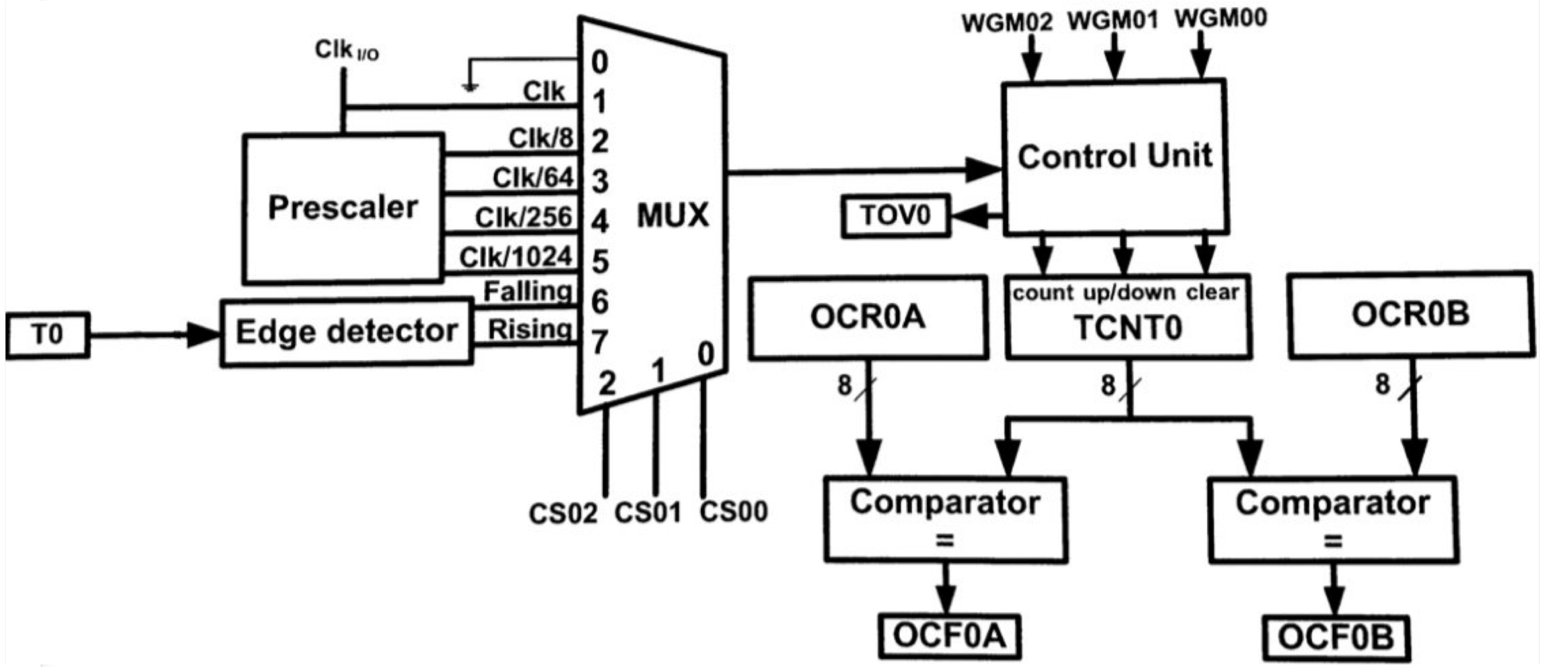


(a) Fast PWM

(b) Phase-Correct PWM

Timer 0 in Detail

- TOV0: Timer 0 overflow flag
- OCR0n: Output compare registers
- TCNT0: Timer 0 counter
- OCF0n: Output comparator flags
- WGM0n: Waveform generator mode bits



The AVR Microcontroller and Embedded Systems Using Assembly and C, Mazadi et al.

Controlling the Output Frequency

- The Arduino Mega 2560 clock is 16 MHz
- By default, the system clock is applied directly to the timers
- To gain better control of the clock frequencies applied to the timers, a *prescaler* can be used
- The scaling is enabled by setting the appropriate bits in the lower 3 bits of the TCCRnB register

CS02-0	Description
000	No clock (timer is stopped)
001	No pre-scaling
010	Clock / 8
011	Clock / 64
100	Clock / 256
101	Clock / 1024
110	External clock on T0 pin (falling edge)
111	External clock on T0 pin (rising edge)

Next Time

- Dive into the math and code settings required to use the timers on the Arduino
- Read Chapter 9 and 15 in Mazidi, with focus on the C code
- Read Chapter 7 sections 7.4 through 7.4.3 in Jimenez