CSSE2010

CSSE2010/CSSE7201
Learning Lab 14

AVR PWM
(Pulse Width Modulation)

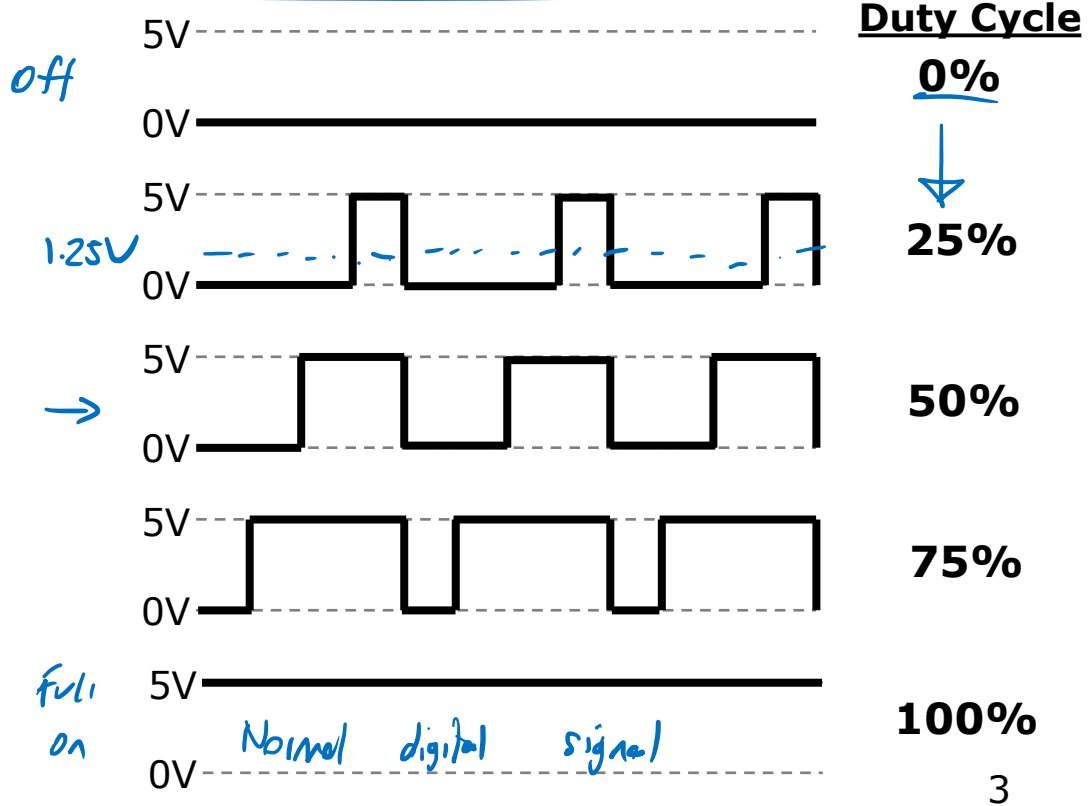
School of Information Technology and Electrical Engineering
The University of Queensland

Today

- AVR Pulse Width Modulation ✓
- This lab assumes knowledge of lab 13 (AVR Timers) ✓

[Recall from Lecture 16] Pulse Width Modulation (PWM)

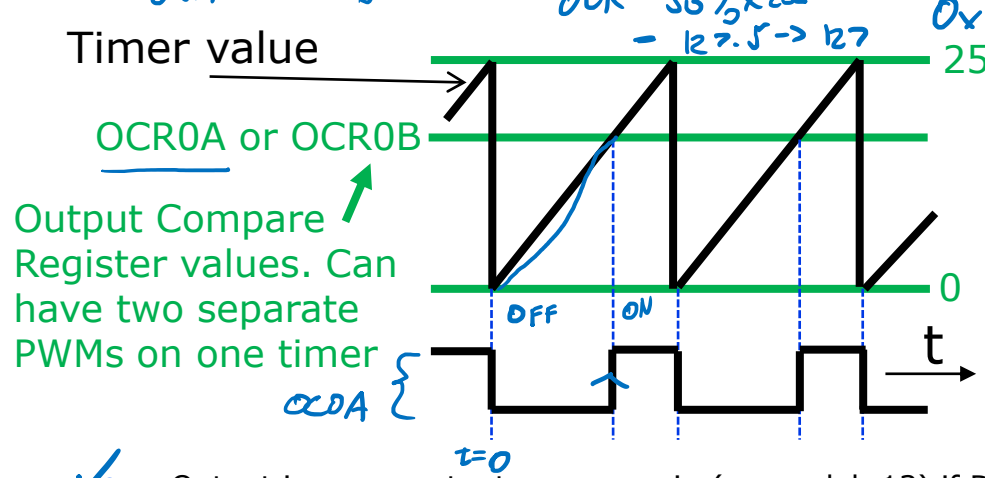
- Varying the duty cycle of a periodic pulse
- **Duty cycle** = % of time that signal is on (high)



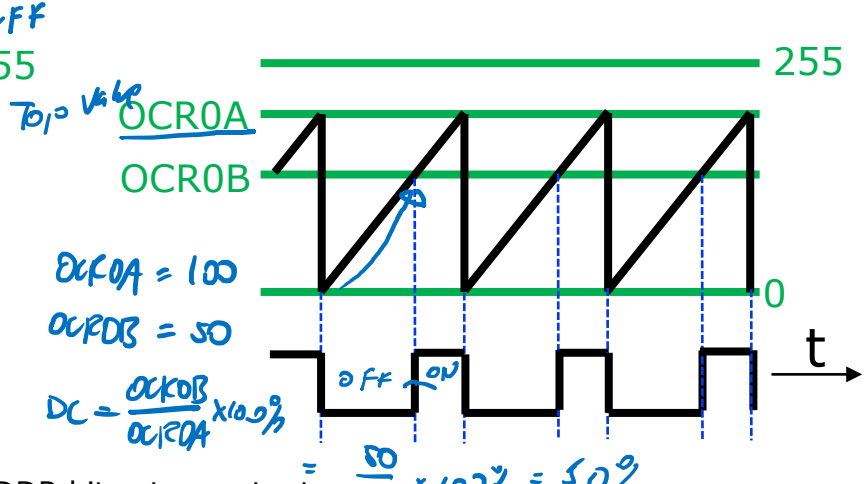
PWM on the AVR: Fast PWM Mode (Example with Timer/Counter 0)

- Two options – Waveform Generation Mode bits (recall other modes from lab 13)

WGM[2:0] = **011** (0x3)



WGM[2:0]= **111** (0x7)



- Output is on an output compare pin (as per lab 13) if DDR bit set as output.
- Output can be inverse of that shown (inverting compare mode shown)
- Timer count rate (slope of line above) is determined by prescaler, e.g. CLK, CLK/8, etc. (also seen in lab 13).
- Output compare registers are "double buffered" – any writes don't take effect until next cycle

Delay Macros

```
#define F_CPU 16000000UL
#include <util/delay.h>
```

- Macros `_delay_ms()` and `_delay_us()` are then available for use
  - Macros take a **constant** number and make the CPU do nothing for that number of milliseconds or microseconds
- Example:

```
_delay_ms(10);      //Do nothing for 10ms
```

- This is called "busy waiting"
- See AVR C Library documentation for details

Task 1

- Add code to lab14-1.c to fade two LEDs on/off using PWM
  - One connected to OC0A and one to OC0B
- You'll need to review pages 84 to 87 of the datasheet to determine register values
- Build the code and download it to the board and test it
  - Can you predict which LED starts on and which starts off?
- Try changing the clock prescaler to CLK/1024 and observe any differences
- (These slides and the code are available on Blackboard.)

Task 2

- Add code to lab14-2.c to generate sound using the piezo buzzer
  - Uses PWM on Timer/Counter 1
  - Piezo buzzer should be connected between output pin (OC1B) and ground.
    - Connect an LED to OC1B also – lets you better see what is going on
- Program allows both frequency and duty cycle to be varied using push buttons
- Make sure you understand all aspects of the code – ask if you're not sure
  - Note 32 bit constants (e.g. 105UL) in places – why do you think this is necessary?
- Challenge task: Add additional code to show the duty cycle (0-99) on the seven segment display