Pulse Width Modulation (PWM)

ENCE361 Embedded Systems 1

Course Coordinator: Ciaran Moore (ciaran.moore@Canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

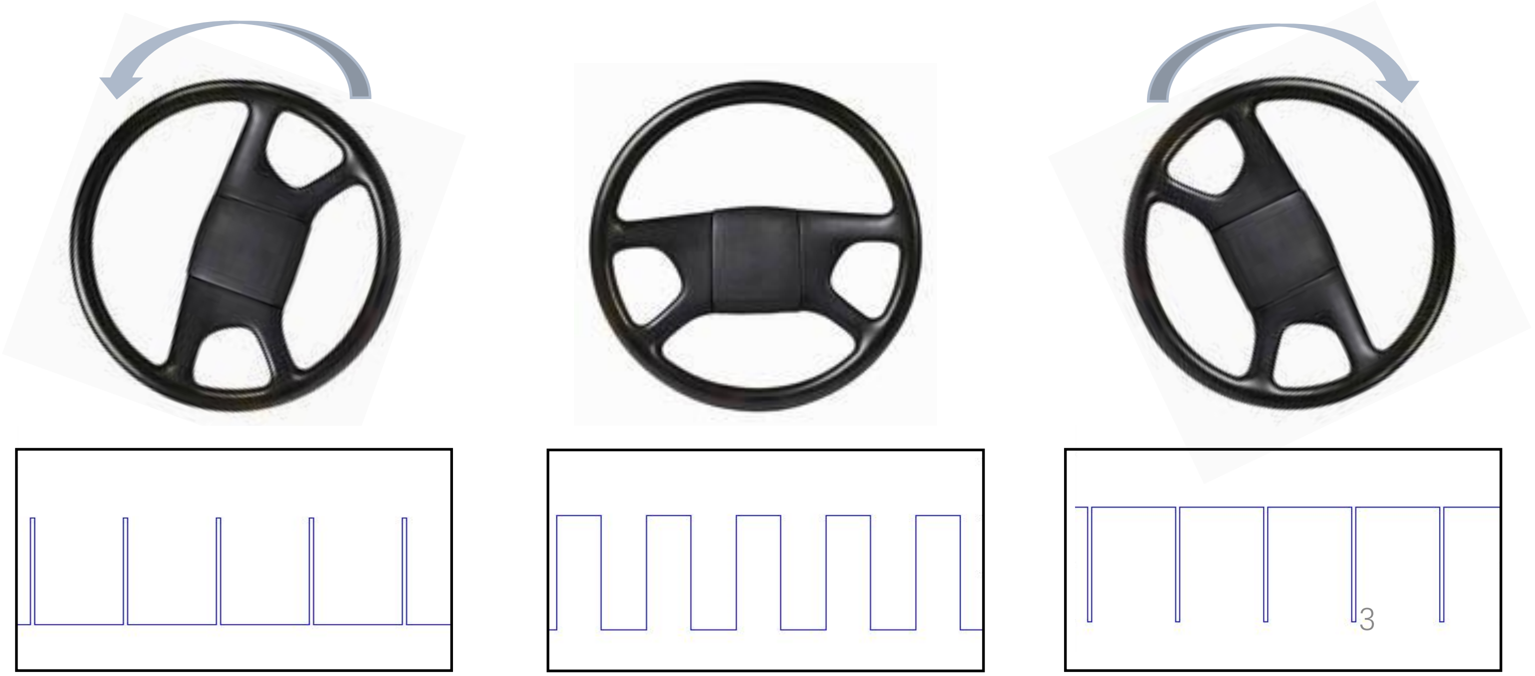
Department of Electrical and Computer Engineering

Where we’re going today

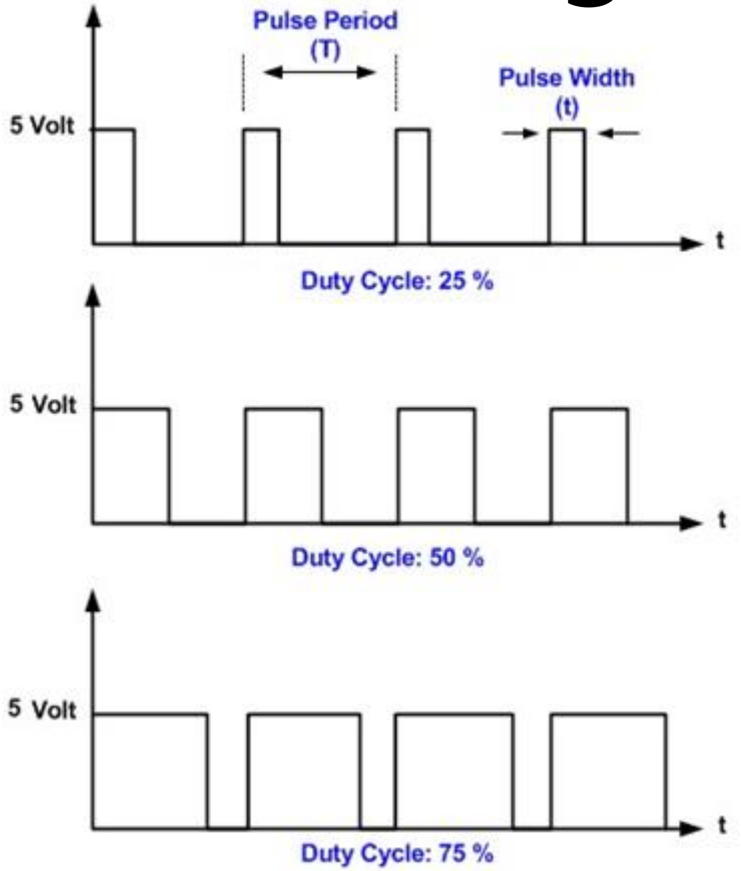
- **PWM overview**
- Generate PWM signals on Tiva C-series launchpad
- Example program in C
- Homework

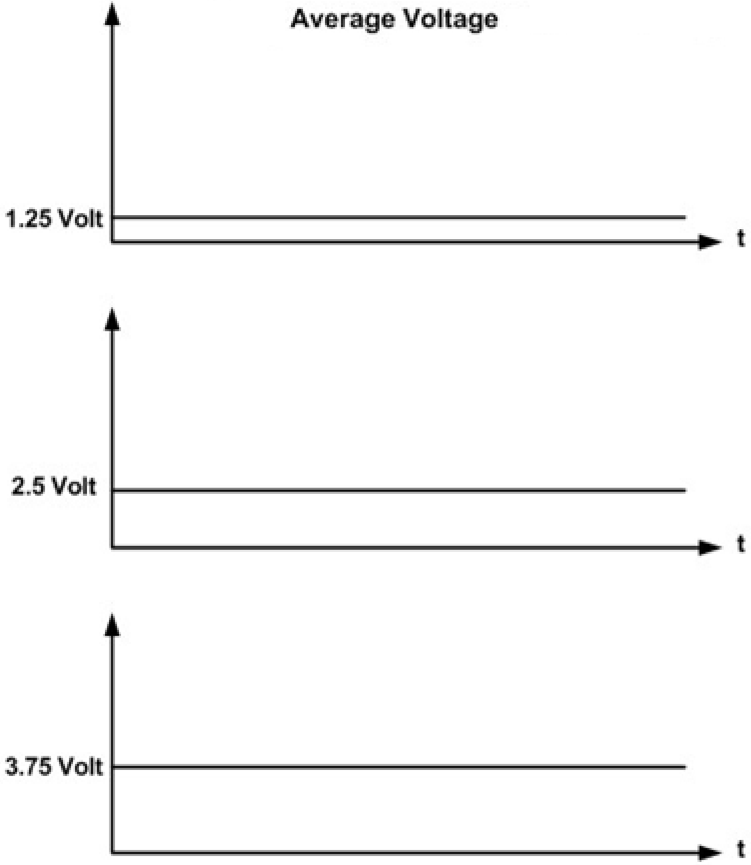
PWM Overview

- Pulse width modulation (PWM)
  - Modulate (change) the pulse width of a square wave to e.g., carry information
  - Square wave generated using programmable digital control
- Application example 1:
  - Encode steering wheel position
    - Right turn: increased pulse width
    - Left turn: reduced pulse width
    - Center: 50% duty cycle



PWM Timing Diagram





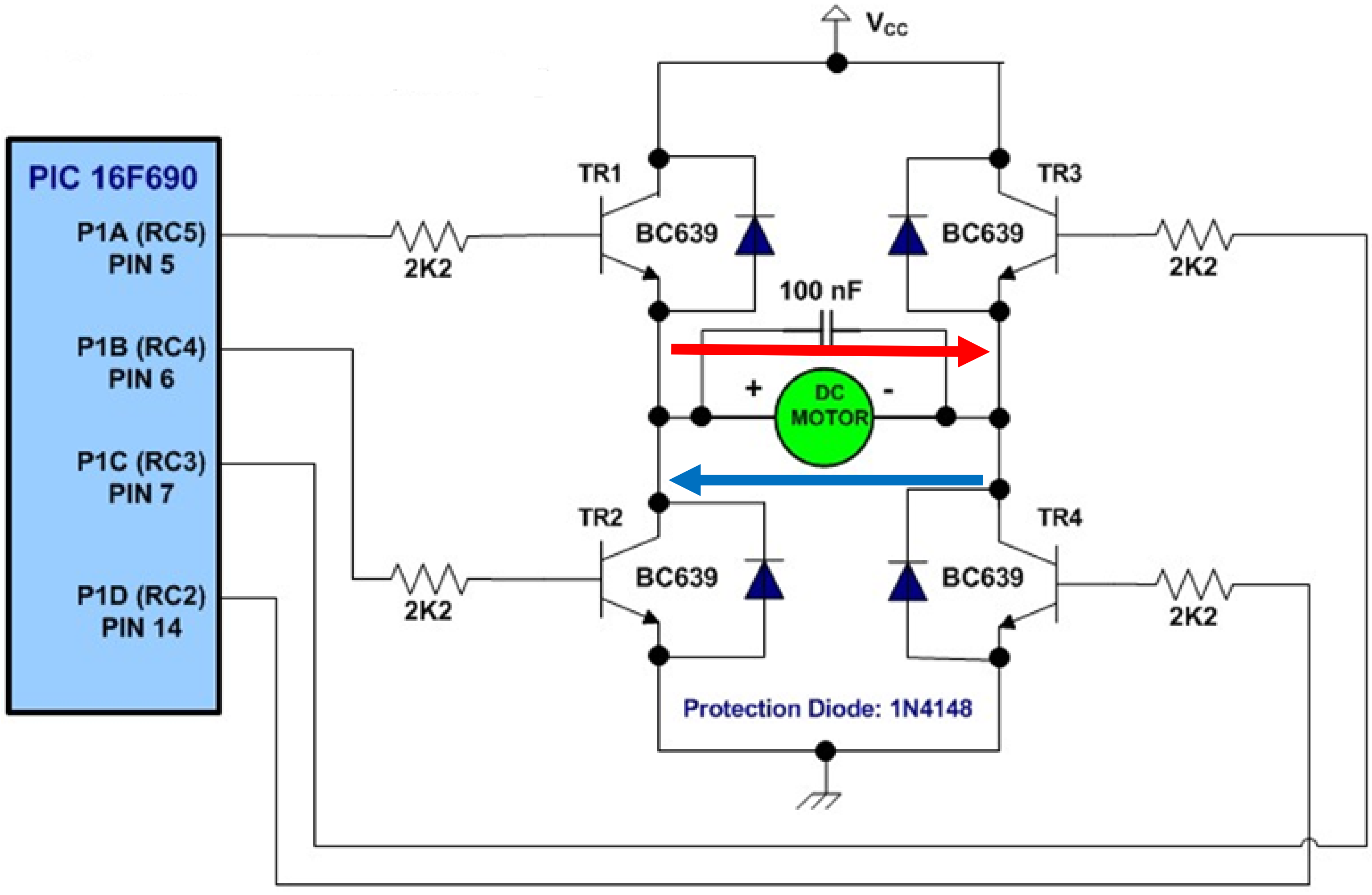
- Duty cycle = pulse width/pulse period
  - For steering wheel example, 'full lock' at 5% or 95% duty cycle

- Measure duty cycle via a DC voltmeter
  - Find average voltage (DC level)

From http://www.ermicro.com/blog/?p=706

PWM-based Motor Control (1)

- PIC 16F690 is a microcontroller
- H-bridge circuit allows clockwise or anti-clockwise rotation of DC motor
- DC motor speed relies on the received average DC voltage
  - Change PWM duty cycle to control motor speed

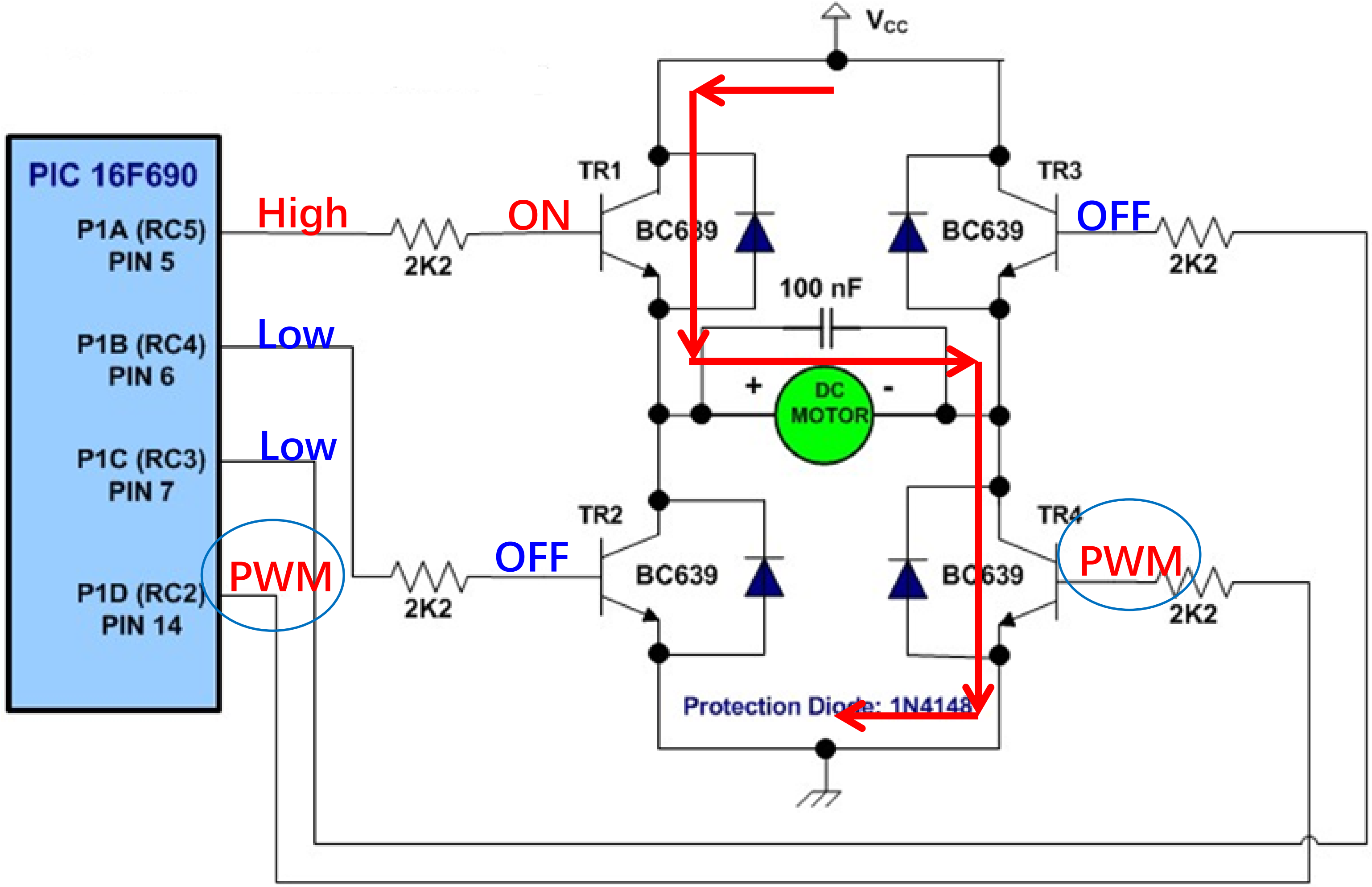


All NPN Transistors H-Bridge Circuit

From http://www.ermicro.com/blog/?p=706

PWM-based Motor Control (2)

- DC motor performs clockwise rotation
- PWM controls motor speed

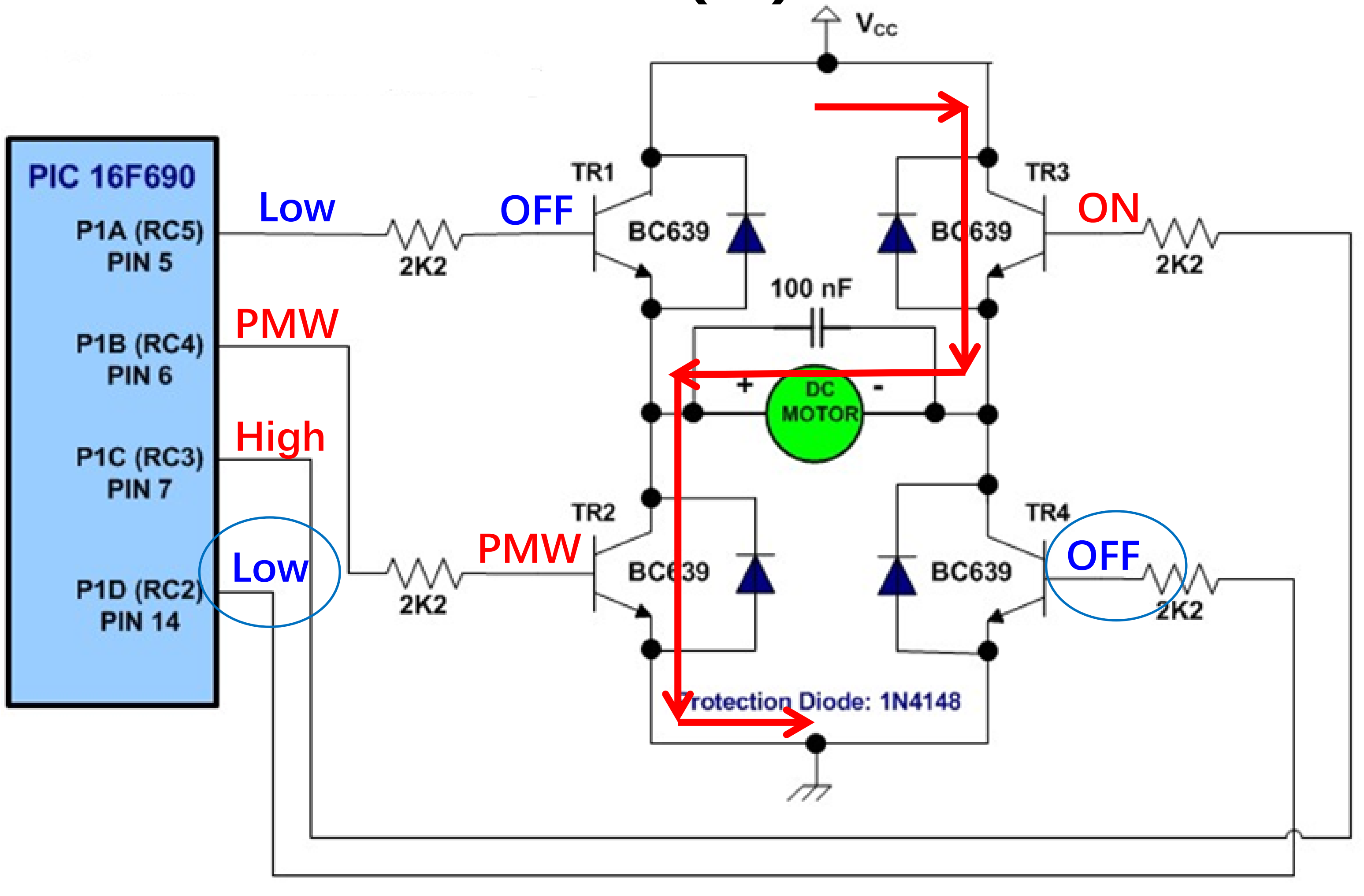


All NPN Transistors H-Bridge Circuit

From http://www.ermicro.com/blog/?p=706

PWM-based Motor Control (3)

- DC motor performs anti-clockwise rotation
- PWM controls motor speed



All NPN Transistors H-Bridge Circuit

From http://www.ermicro.com/blog/?p=706

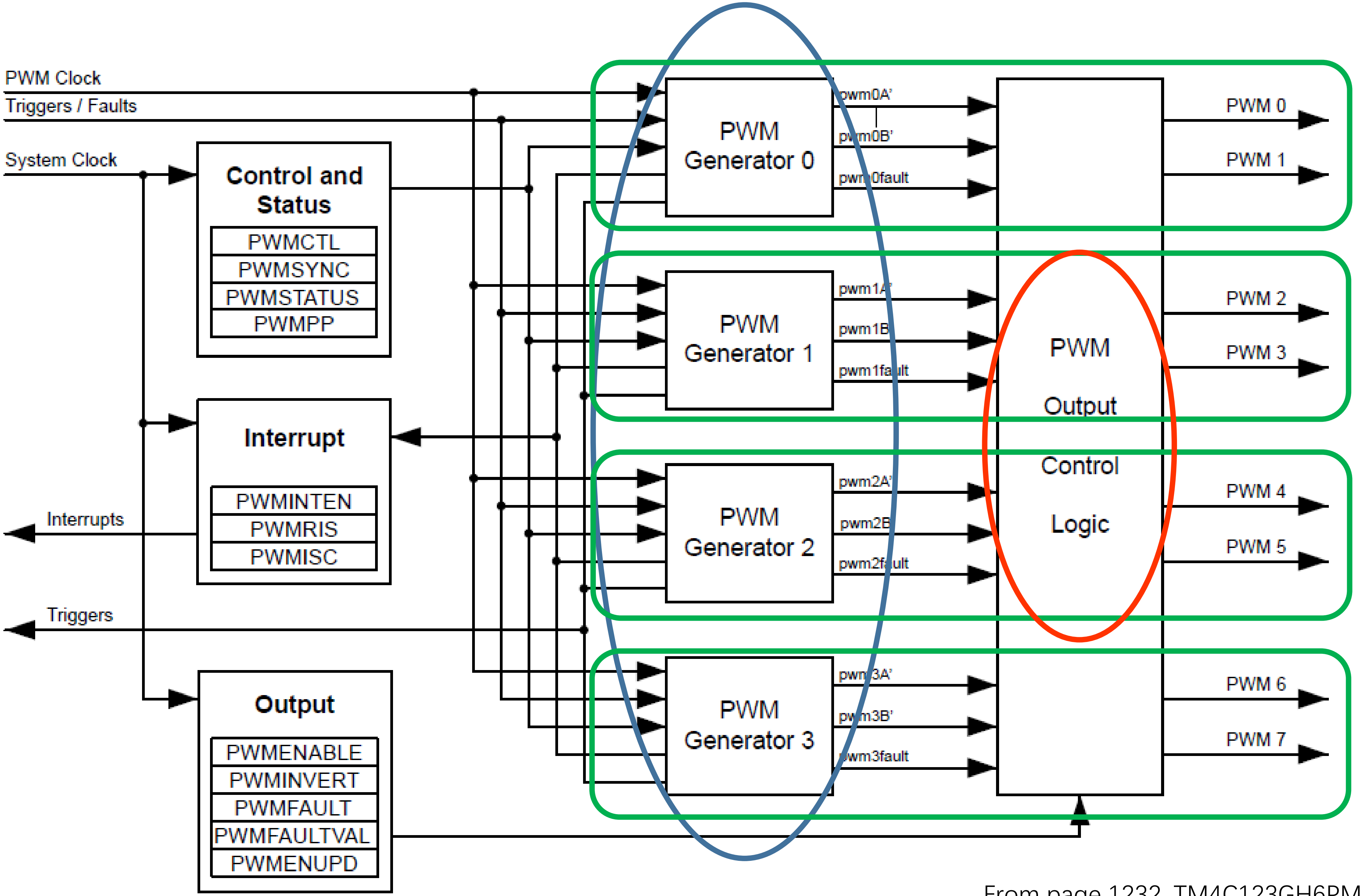
Where we’re going today

- PWM overview
- **Generate PWM signals on Tiva C-series launchpad**
- Example program in C
- Homework

Generate PWM Signals on Tiva Board

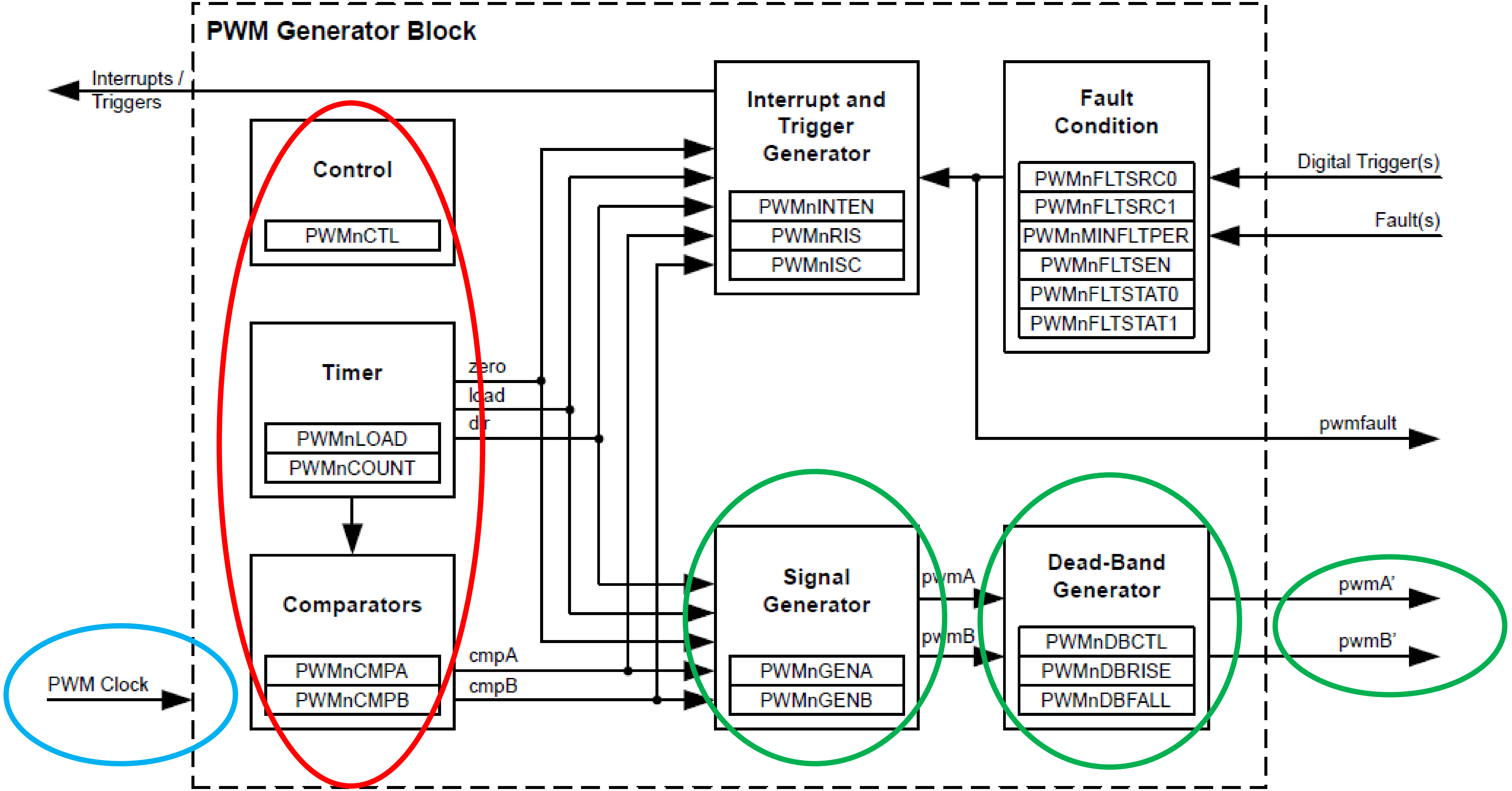
- PWM signal characteristics realized via digital means
    - Pulse period (signal frequency)
    - Pulse width (duty cycle)
- Tiva C-series Launchpad has 2 PWM modules, M0 and M1
    - Each module consists of 4 PWM generators (0, 1, 2, 3) and a control block
        - Each generator produces 2 PWM signals
        - 8 PWM signals from a module are called PWM0 – PWM7 (see next slide)
    - Control block determines
        - Mode of operation
        - Which pins PWM are passed to

Figure 20-1. PWM Module Diagram



From page 1232, TM4C123GH6PM Data Sheet.pdf

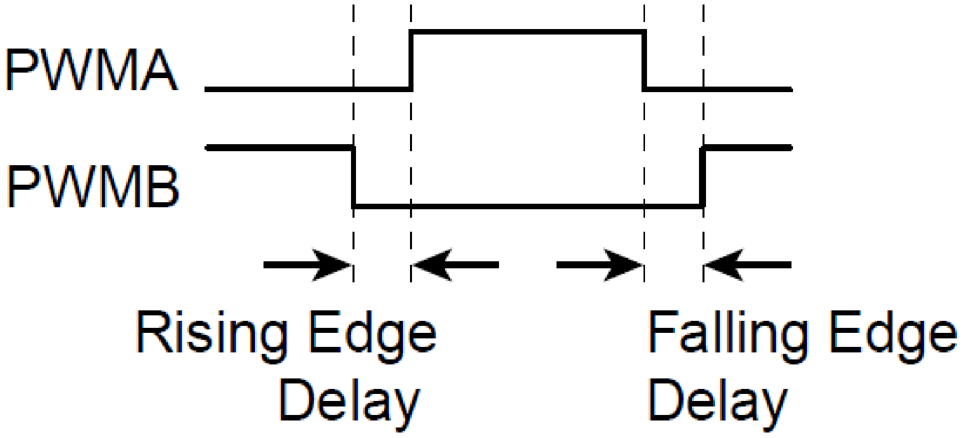
Figure 20-2. PWM Generator Block Diagram

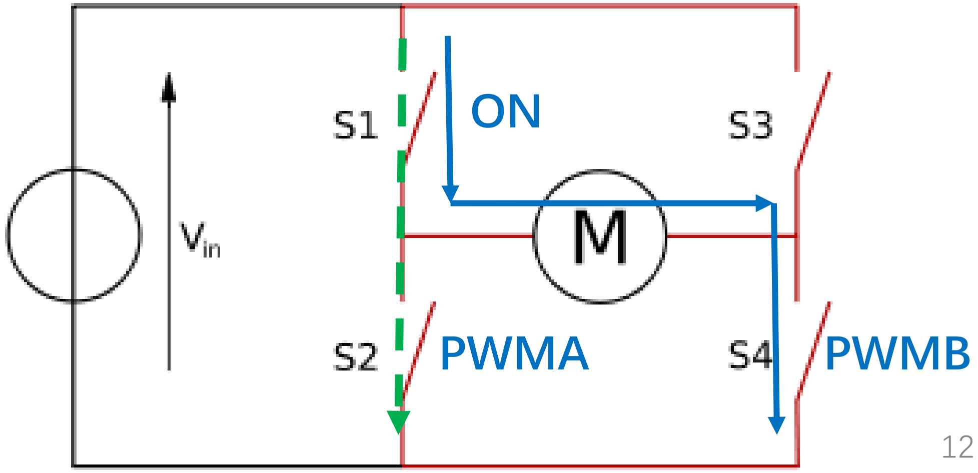


From page 1232, TM4C123GH6PM Data Sheet.pdf

PWM Generator

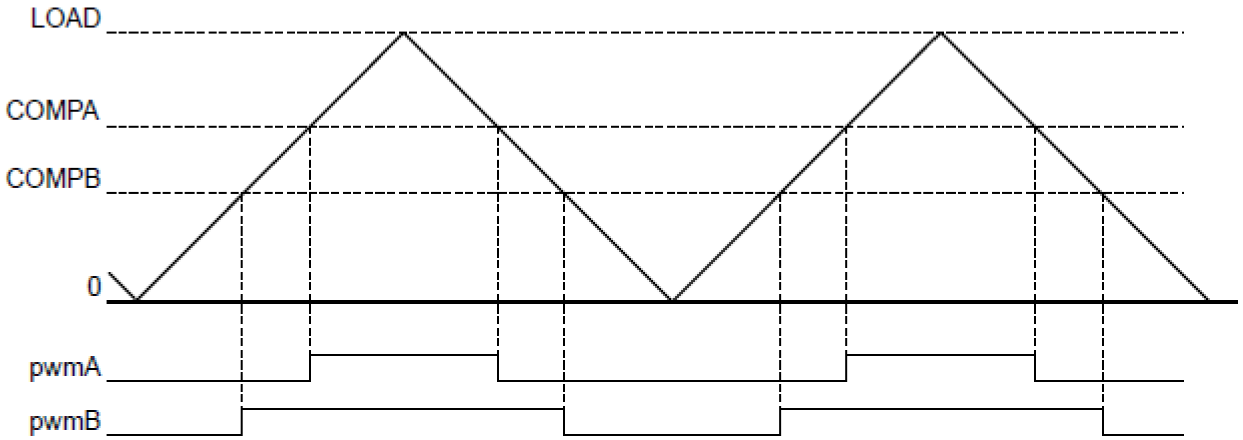
- PWM generator has only 1 PWM clock
  - PWM clock is derived from microprocessor clock via a pre-scaler (see example code)
    - /2, /4, /8, up to /64
- Produce 2 PWM signals with the same period
  - 2 PWM signals can have different duty cycles
  - 2 PWM signals can be complementary with dead-band delay inserted





Generate PWM Signals

- LOAD register: 16 bits
  - Together with PWM clock to determine PWM signal period
- COMP: digital comparator
  - Two thresholds, COMPA and COMPB, for specifying duty cycles of two PWM signals
- Count-up/down mode
  - Maximum signal period = 2×2^{16} PWM clock cycles
- Count-down mode
  - Page 1235, TM4C123GH6PM Data Sheet.pdf



From page 1236, TM4C123GH6PM Data Sheet.pdf

Where we’re going today

- PWM overview
- Generate PWM signals on Tiva C-series launchpad
- **Example program in C**
- Homework

Example PWM module use – “pwm100.c” on Learn (1)

```
/*************************************************************************
 * pwm100.c - Example program to output PWM on the main rotor pin.  Tiva version.
 * Author:  P.J. Bones            UC ECE
 * Last modified:       23.1.2019 by Le Yang UC ECE
 ************************************************************************/
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/pin_map.h"                    // Needed for pin configure
#include "driverlib/debug.h"
#include "driverlib/gpio.h“                       // Needed for using GPIO pins
#include " driverlib/pwm.h"                       // Needed for using PWM modules
#include "driverlib/sysctl.h"

/*********************************************
 *     PWM configuration details.
 *********************************************/
#define PWM_RATE_HZ  200                          // PWM signal frequency
#define PWM_START_PC  10                          // PWM signal duty cycle (in percentage: 10%)
#define PWM_DIVIDER_CODE  SYSCTL_PWMDIV_2         // PWM clock pre-scale (1/2)
#define PWM_DIVIDER  2
```

Example PWM module use – “pwm100.c” on Learn (2)

```
int main(void)
{
    // Set the processor clock rate to 20 MHz
    SysCtlClockSet (SYSCTL_SYSDIV_10 | SYSCTL_USE_PLL |
                          SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    // Set PWM clock rate to 10 MHz (which means 80Hz min rate) via pre-scale PWM_DIVIDER_CODE
    SysCtlPWMClockSet(PWM_DIVIDER_CODE);

    // As a precaution, make sure that the peripherals used are reset
    SysCtlPeripheralReset(PWM_MAIN_PERIPH_GPIO);        // Used for PWM output
    SysCtlPeripheralReset(PWM_MAIN_PERIPH_PWM);        // Main motor PWM

    initialisePWM();        // Initialize PWM module

    // Output PWM signal to PWM_MAIN_OUTBIT Pin
    PWMOutputState(PWM_MAIN_BASE, PWM_MAIN_OUTBIT, true);
                                        .
                                        .
                                        .
}
```

Example PWM module use – “pwm100.c” on Learn (3)

```
void initialisePWM(void)
{
    SysCtlPeripheralEnable(PWM_MAIN_PERIPH_PWM);
    SysCtlPeripheralEnable(PWM_MAIN_PERIPH_GPIO);

    // Configure PWM_MAIN_GPIO_PIN as PWM pin
    GPIOPinTypePWM(PWM_MAIN_GPIO_BASE, PWM_MAIN_GPIO_PIN);
    GPIOPinConfigure(PWM_MAIN_GPIO_CONFIG);

    // Calculate PWM period (in PWM clock cycles) using PWM_RATE_HZ
    uint32_t ui32Period = SysCtlClockGet() / PWM_DIVIDER / PWM_RATE_HZ;

    // Set generator PWM_MAIN_GEN to count-up/down mode and period ui32Period
    PWMGenConfigure(PWM_MAIN_BASE, PWM_MAIN_GEN,
                    PWM_GEN_MODE_UP_DOWN | PWM_GEN_MODE_NO_SYNC);
    PWMGenPeriodSet(PWM_MAIN_BASE, PWM_MAIN_GEN, ui32Period);

    // Set the pulse width based on PWM_START_PC % duty cycle.
    PWMPulseWidthSet(PWM_MAIN_BASE, PWM_MAIN_OUTNUM, ui32Period * PWM_START_PC / 100);
                                        .
                                        .
}
```

```
#define PWM_RATE_HZ  200
#define PWM_START_PC  10
#define PWM_DIVIDER  2
```

Number of PWM clock cycles

Homework

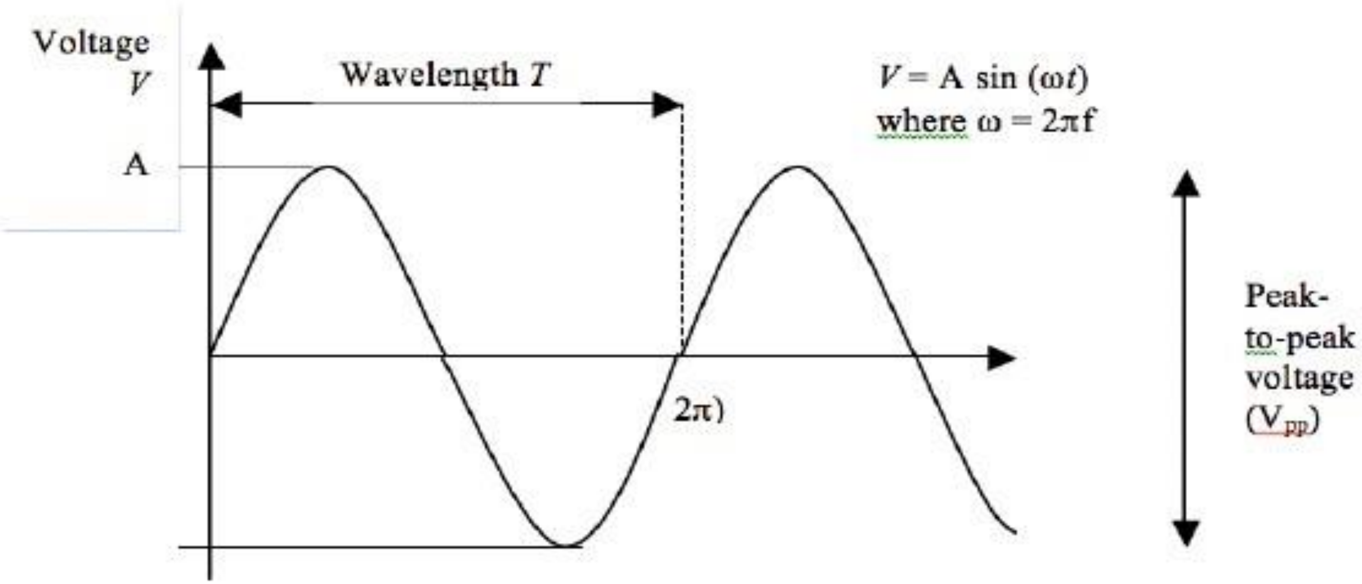
1. Why might it be preferable to use PWM to pass information about a continuous variable, such as the position of the steering wheel on slide 3, rather than using an analogue voltage directly?

2. Are the average voltages shown on the RHS of Slide 4 the same as the RMS voltages (see next slide) for the respective waveforms? Calculate the RMS values for the waveforms shown to corroborate your answer.

3. Show the calculation that leads to the conclusion that the minimum PWM frequency that can be achieved on the Tiva C-Series Launchpad with processor clock frequency 20 MHz and the default (no prescale) for the PWM clock is about 160 Hz.

4. What is the minimum PWM frequency that can be achieved on the Tiva C-Series Launchpad with processor clock frequency 20 MHz and with a PWM prescale of 32?

Root Mean Square (RMS) Voltage



- RMS voltage is the average power of a periodic signal when applied to a 1 Ohm resistor
  - *v(t)*: the signal with period *T*

$$\boxed{V_{RMS} = \sqrt{\frac{1}{T}\int_0^T v^2(t)dt}}$$

- A Sinewave with amplitude *A* (see the figure above) has a RMS voltage of $V_{RMS} = \frac{A}{\sqrt{2}}$