PWM Programming

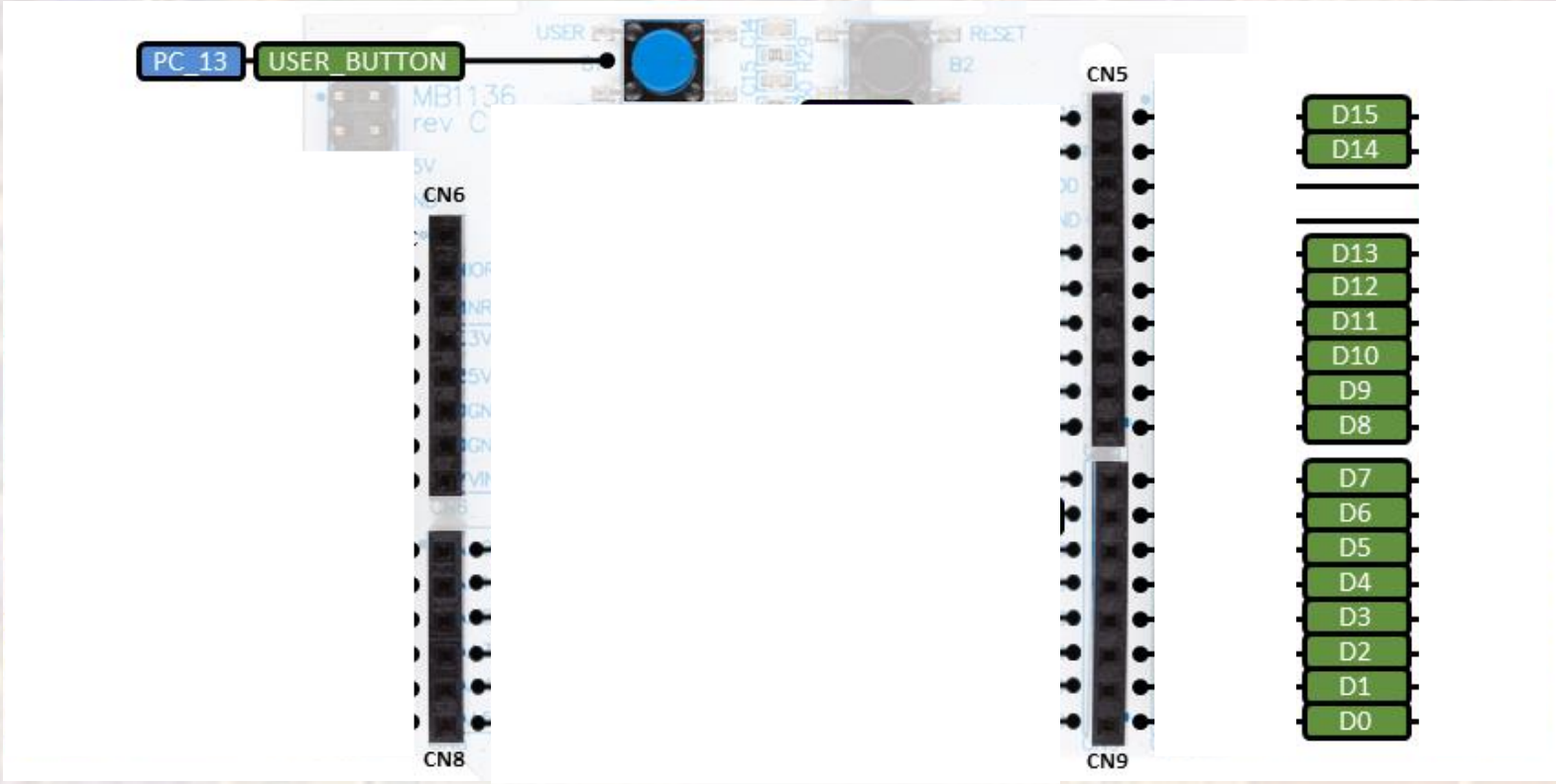
Last updated 6/7/21

PWM Programming

- PWM Operation
  - Nucleo-L476RG uses timers to manage the PWM feature
  - At least 7 timers can be used for PWM generation
  - Only a few of these are enabled on our implementation

PWM Programming

- PWM Connections
  - All the digital pins can be used for PWM generation ???
  - The compiler WILL NOT warn you if you use an incorrect pin



PWM Programming

- PwmOut Class

Public Member Functions	
	PwmOut (PinName pin)
	Create a PwmOut connected to the specified pin. More...
	PwmOut (const PinMap &pinmap)
	Create a PwmOut connected to the specified pin. More...
void	write (float value)
	Set the output duty-cycle, specified as a percentage (float) More...
float	read ()
	Return the current output duty-cycle setting, measured as a percentage (float) More...
void	period (float seconds)
	Set the PWM period, specified in seconds (float), keeping the duty cycle the same. More...
void	period_ms (int ms)
	Set the PWM period, specified in milliseconds (int), keeping the duty cycle the same. More...
void	period_us (int us)
	Set the PWM period, specified in microseconds (int), keeping the duty cycle the same. More...
int	read_period_us ()
	Read the PWM period. More...
void	pulsewidth (float seconds)
	Set the PWM pulsewidth, specified in seconds (float), keeping the period the same. More...
void	pulsewidth_ms (int ms)
	Set the PWM pulsewidth, specified in milliseconds (int), keeping the period the same. More...
void	pulsewidth_us (int us)
	Set the PWM pulsewidth, specified in microseconds (int), keeping the period the same. More...
int	read_pulsewitdth_us ()
	Read the PWM pulsewidth. More...
void	suspend ()
	Suspend PWM operation. More...
void	resume ()
	Resume PWM operation. More...
PwmOut &	operator= (float value)
	A operator shorthand for write() More...
PwmOut &	operator= (PwmOut &rhs)
	A operator shorthand for write() More...
	operator float ()
	An operator shorthand for read() More...

PWM Programming

- Constructors

Public Member Functions	
	PwmOut (PinName pin)
	Create a PwmOut connected to the specified pin. More...
	PwmOut (const PinMap &pinmap)
	Create a PwmOut connected to the specified pin. More...

```
// Create PWM object attached to D4
PwmOut Pwm_a(D4);
```

PWM Programming

- Member Functions (Methods)

void	write (float value) 0.0 – 1.0 : (0% - 100% duty cycle)
	Set the output duty-cycle, specified as a percentage (float) More...
float	read () 0.0 – 1.0 : (0% - 100% duty cycle)
	Return the current output duty-cycle setting, measured as a percentage (float) More...
void	period (float seconds)
	Set the PWM period, specified in seconds (float), keeping the duty cycle the same. More...
void	period_ms (int ms)
	Set the PWM period, specified in milliseconds (int), keeping the duty cycle the same. More...
void	period_us (int us)
	Set the PWM period, specified in microseconds (int), keeping the duty cycle the same. More...
int	read_period_us ()
	Read the PWM period. More...

void	pulsewidth (float seconds)
	Set the PWM pulsewidth, specified in seconds (float), keeping the period the same. More...
void	pulsewidth_ms (int ms)
	Set the PWM pulsewidth, specified in milliseconds (int), keeping the period the same. More...
void	pulsewidth_us (int us)
	Set the PWM pulsewidth, specified in microseconds (int), keeping the period the same. More...
int	read_pulsewitdth_us ()
	Read the PWM pulsewidth. More...
void	suspend ()
	Suspend PWM operation. More...
void	resume ()
	Resume PWM operation. More...

```
// Initialize the PWM object
// using 50KHz for the PWM frequency
// and a 50% duty cycle
Pwm_a.period_us(200);
Pwm_a.write(0.5);
```

PWM Programming

- Operator Overloads

PwmOut &	operator= (float value)
	A operator shorthand for write() More...
PwmOut &	operator= (PwmOut &rhs)
	A operator shorthand for write() More...
	operator float ()
	An operator shorthand for read() More...

```
// use the overload operator
Pwm_a = 0.5;
```

PWM Programming

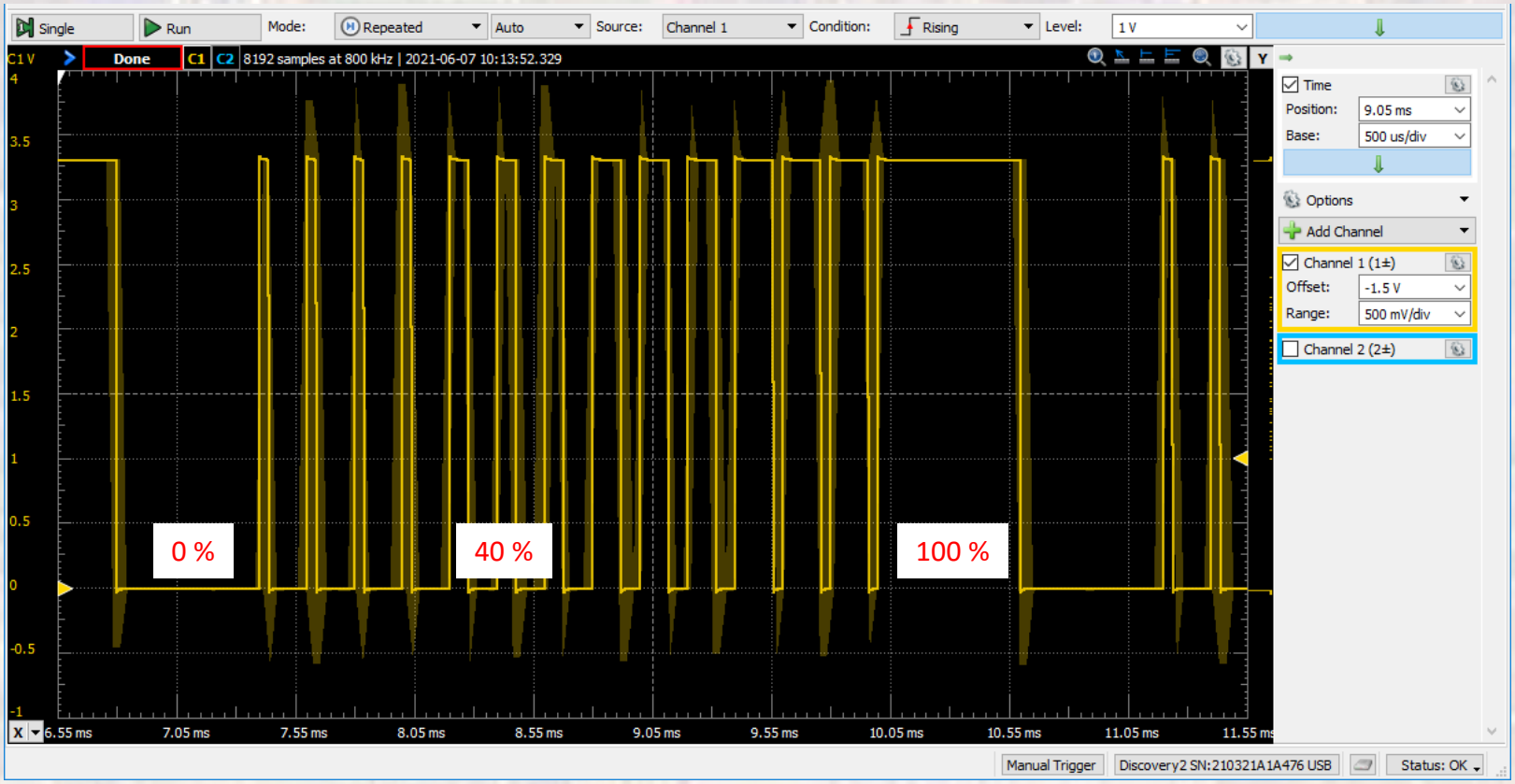
- Simple example 1
  - Generate a series of PWM output pulses
  - 3 pulses at 5 different pulse widths

```
////////////////////////////////////
//
// pwm_class_ex_1 project
//
// created 6/4/21 by tj
// rev 0
//
////////////////////////////////////
//
// PWM example file for class
//
// shows basic PWM operation
//
////////////////////////////////////

#include "mbed.h"
#include <stdio.h>

#define T_WAIT 600    // in us

// Global HARDWARE Objects
// Create PWM object attached to D4
PwmOut Pwm_a(D4);

int main(void){
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("pwm_class_ex_1 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // Initialize the PWM object
    // using 50KHz for the PWM frequency
    // and a 50% duty cycle
    Pwm_a.period_us(200);
    Pwm_a.write(0.5);
```

```
    // run through an endless series of duty cycles
    while(1){
        // generate some PWM outputs
        for(int i = 0; i <= 5; i++){
            Pwm_a.write(i/5.0);
            wait_us(T_WAIT);
        }// end for

//    // use the overload operator
//    Pwm_a = 0.5;

    }// end while

    return 0;
}// end main
```

PWM Programming

- Simple example 1
  - Generate a series of PWM output pulses
  - 3 pulses at 5 different pulse widths



PWM Programming

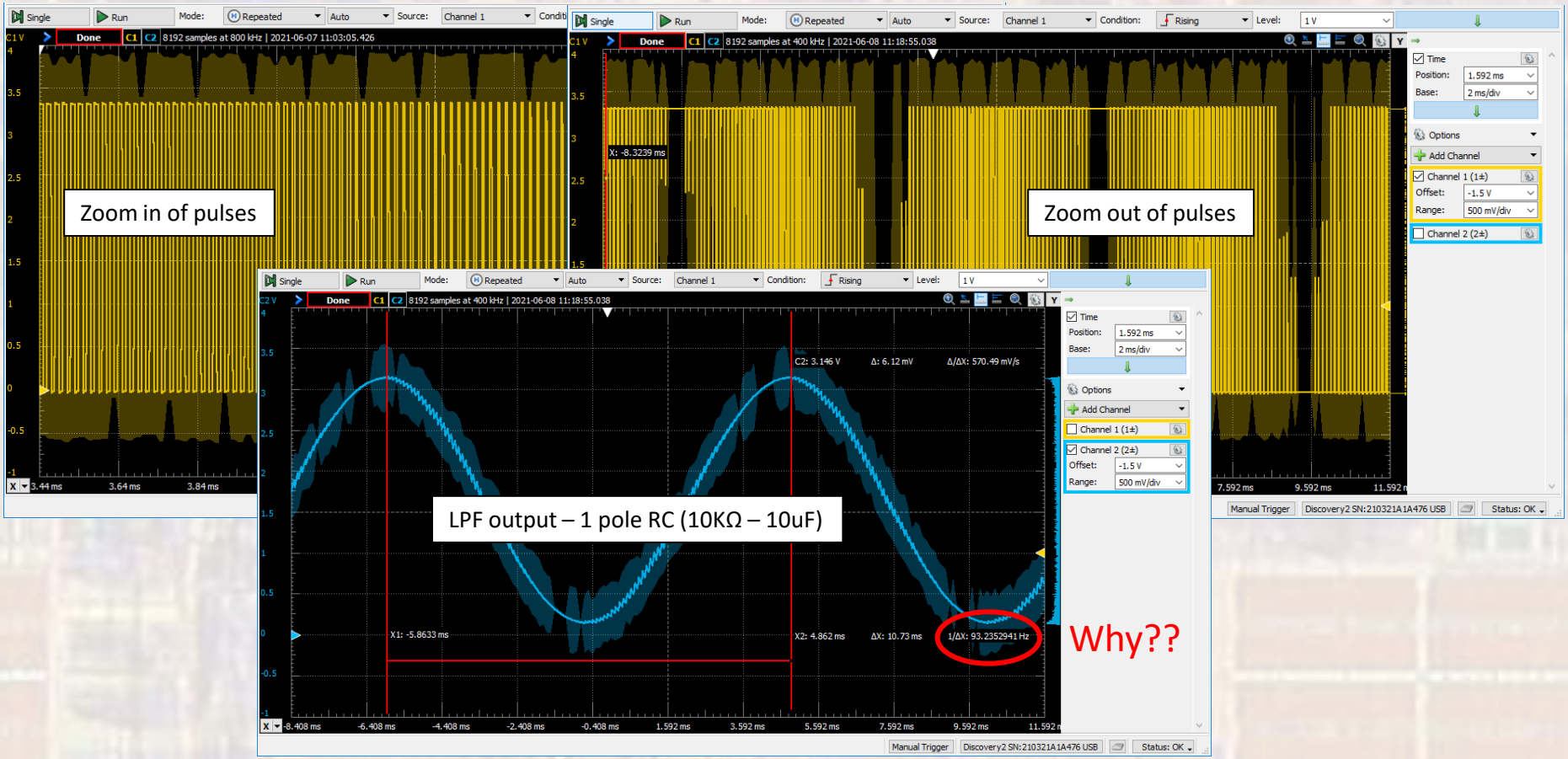
- Simple example 2
  - Generate a Sine wave
  - 10KHz PWM, 100Hz sinewave with 50 points

```
////////////////////////////////////
//
// pwm_class_ex_2 project
//
// created 6/4/21 by tj
// rev 0
//
////////////////////////////////////
//
// PWM example file for class
//
// shows a sine wave PWM
//
////////////////////////////////////

#include "mbed.h"
#include <stdio.h>

#define PI 3.14159
#define F_SINE 100 // Hz
#define NUM_PTS 50

// Global HARDWARE Objects
// Create PWM object attached to D4
PwmOut Pwm_a(D4);

int main(void){
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("pwm_class_ex_2 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // working variables
    float sine_ary[NUM_PTS];

    // Initialize the PWM object
    // using 10KHz for the PWM frequency
    Pwm_a.period_us(100);

    // array setup
    for (int i = 0; i < NUM_PTS; i++){
        sine_ary[i] = sin(i*(2*PI/NUM_PTS))*.5 + 0.5;
    }// end for
```

```
    // run through an endless series of conversions
    while(1){
        // write out a sine wave of variable frequency
        // the sin fn outputs -1 to +1 so we need to divide by 2 to get a
        // total swing of 1. Then need to offset by 0.5 to make everything
        // positive
        // Hardcoding 50 points for the sine wave
         for (int i = 0; i < NUM_PTS; i++){
            Pwm_a.write(sine_ary[i]);
            // 1000000 to convert to sec, 1/F_sine for period,
            // with NUM_PTS outputs / period
            wait_us(1000000*1.0/F_SINE/NUM_PTS);
        }// end for
    }// end while

    return 0;
}// end main
```

PWM Programming

- Simple example 2
  - Generate a Sine wave
  - 10KHz PWM, 100Hz sinewave with 50 points



PWM Programming

- Simple example 3
  - Estimate PWM programming time
  - Measure time for a known # of programming cycles

```
////////////////////////////////////
//
// pwm_class_ex_3 project
//
// created 6/4/21 by tj
// rev 0
//
////////////////////////////////////
//
// PWM example file for class
//
// PWM delay estimate
//
////////////////////////////////////

#include "mbed.h"
#include <stdio.h>

// Global HARDWARE Objects
// Create PWM object attached to D4
PwmOut Pwm_a(D4);

int main(void){
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("pwm_class_ex_3 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
            MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // Initialize the PWM object
    // using 250KHz for the PWM frequency
    // allows for 1us min pulse width at 20% duty cycle
    Pwm_a.period_us(4);

    // run through an endless series of conversions
    while(1){
        // Toggle the duty cycle back and forth
        Pwm_a.write(0.25);
        wait_us(12);
        Pwm_a.write(0.75);
        wait_us(12);
    }// end while

    return 0;
}// end main
```

PWM Programming

- Simple example 3
  - Estimate PWM programming time
  - Measure time for a known # of programming cycles

PWM Programming

- Limitations summary
  - It looks like the minimum pulse width is 1us
    - Need to account for this when setting frequency and duty cycle
  - PWM duty cycle programming time is approximately 14us
    - Checking Example 2:
      - 100Hz → 10ms period
      - 50 writes / period = 700us
      - New period = 10.7ms → 93.46Hz (93.21Hz measured)