

The background is a close-up, slightly blurred image of a green printed circuit board (PCB). A large, square, tan-colored integrated circuit (IC) is the central focus, with its pins visible. Other components like smaller chips, capacitors, and a circular component are also visible on the board.

MCT 4334

Embedded System Design

Week 06 Timer Ports

Outline

- Pulse Width Modulation (PWM)
- Timer ports on ATmega328p
- Timer registers
- Programming examples

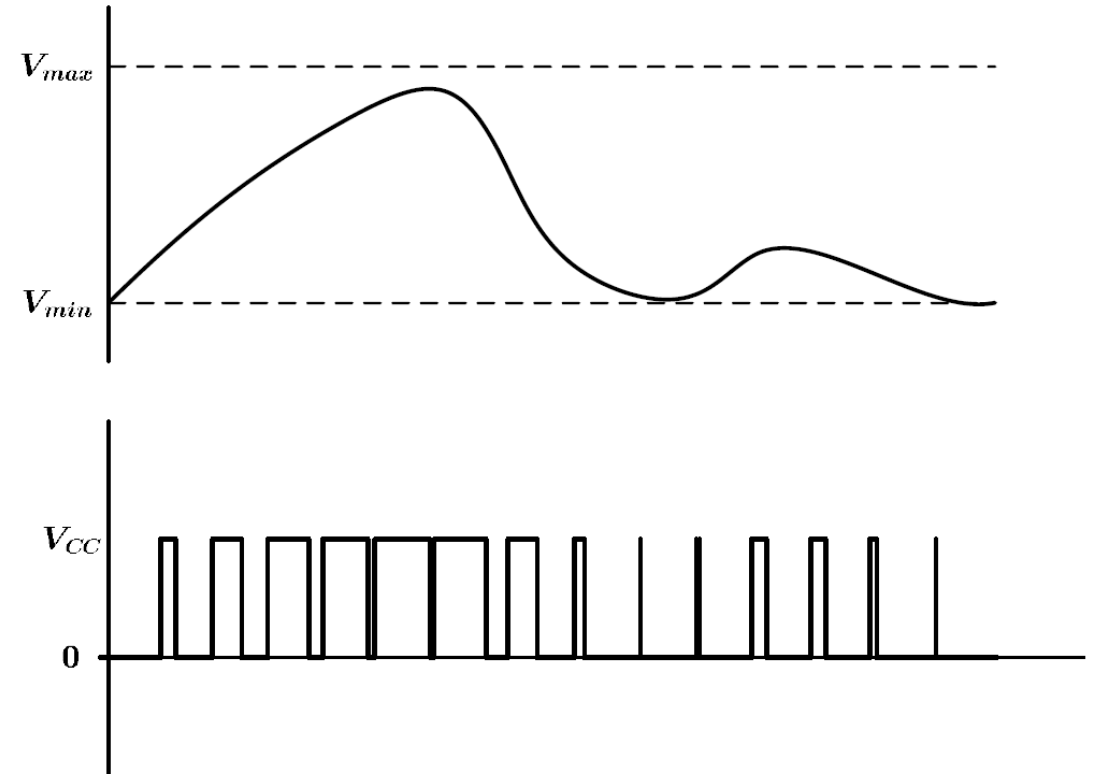
PWM

- Pulse-width modulation is a technique in communication engineering to allow transmission of analog signals digitally.
- Amplitude information is encoded in form of duty cycle.

$$\text{Duty cycle, } D = \frac{\text{High duration in a single period}}{\text{Period}}$$

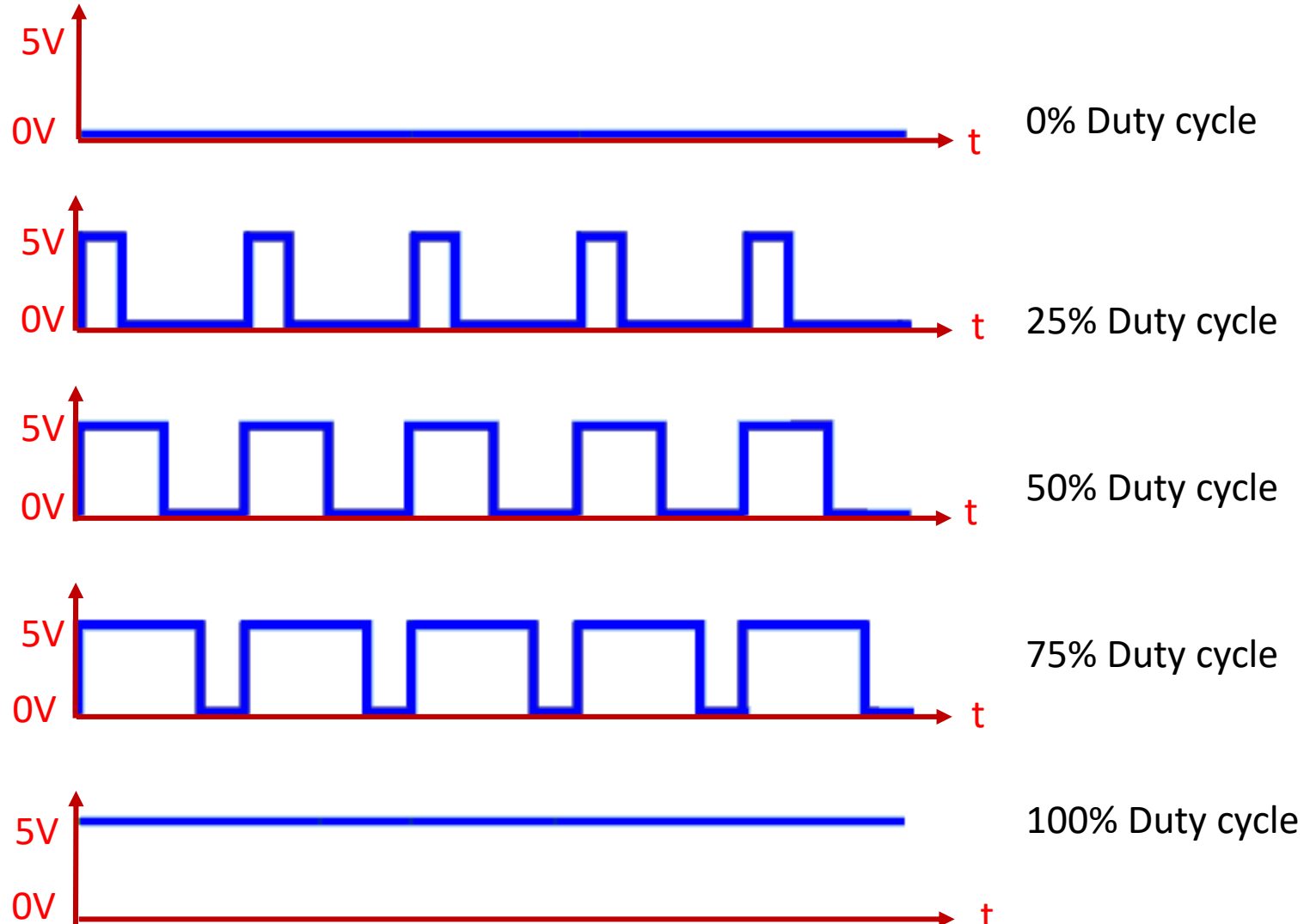
= Percentage of ON time

$$D = \frac{V_{sig} - V_{min}}{V_{max} - V_{min}}$$



PWM

- PWM is commonly utilized in microcontrollers to **emulate** “analog” output to actuators.



Timer ports on ATmega328p

ATmega328p has 3 counters each of which has output two channels (A and B)

Timer	Resolution (bits)	Channel	Pin	Output name
timer0	8	A	PD6	OC0A
timer0	8	B	PD5	OC0B
timer1	16	A	PB1	OC1A
timer1	16	B	PB2	OC1B
timer2	8	A	PB3	OC2A
timer2	8	B	PD3	OC2B

PWM Modes

Two common PWM modes supported by ATmega328p are:

- **Fast PWM**
- **Phase-correct PWM**

Fast PWM Mode

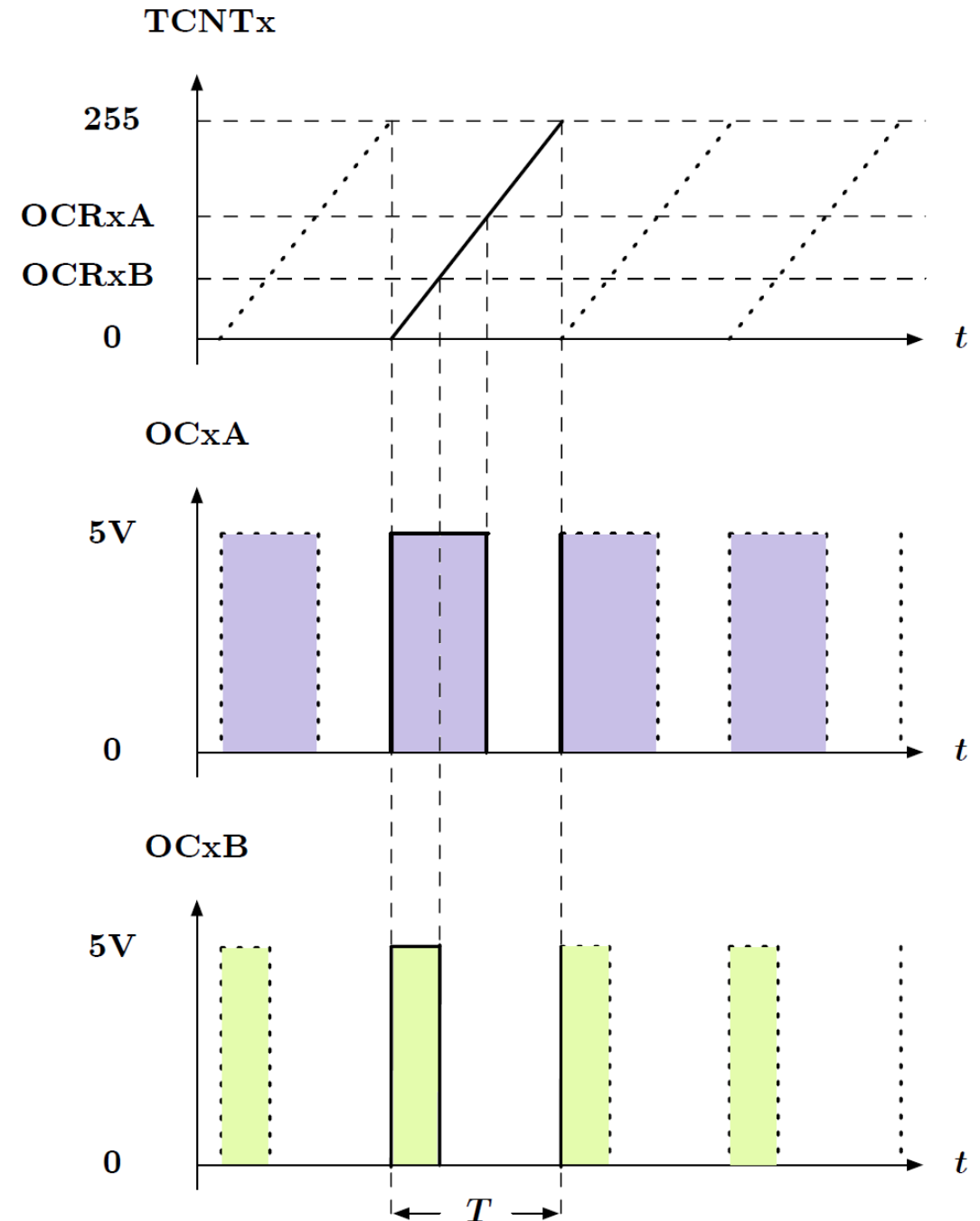
An 8-bit timer has a default maximum value of 255 and a 16-bit timer has a maximum value of 65,535.

The counter increments its value (TCNT) in every clock cycle.

As soon as the TCNT breaks the maximum value, it resets to 0.

OCR controls the duty cycle of the output waveform.
For example:

- If the value of the counter is **less** than OCRxA, the channel A output is **HIGH**
- If the value of the counter is **greater** than OCRxA, the channel A output is **LOW**.

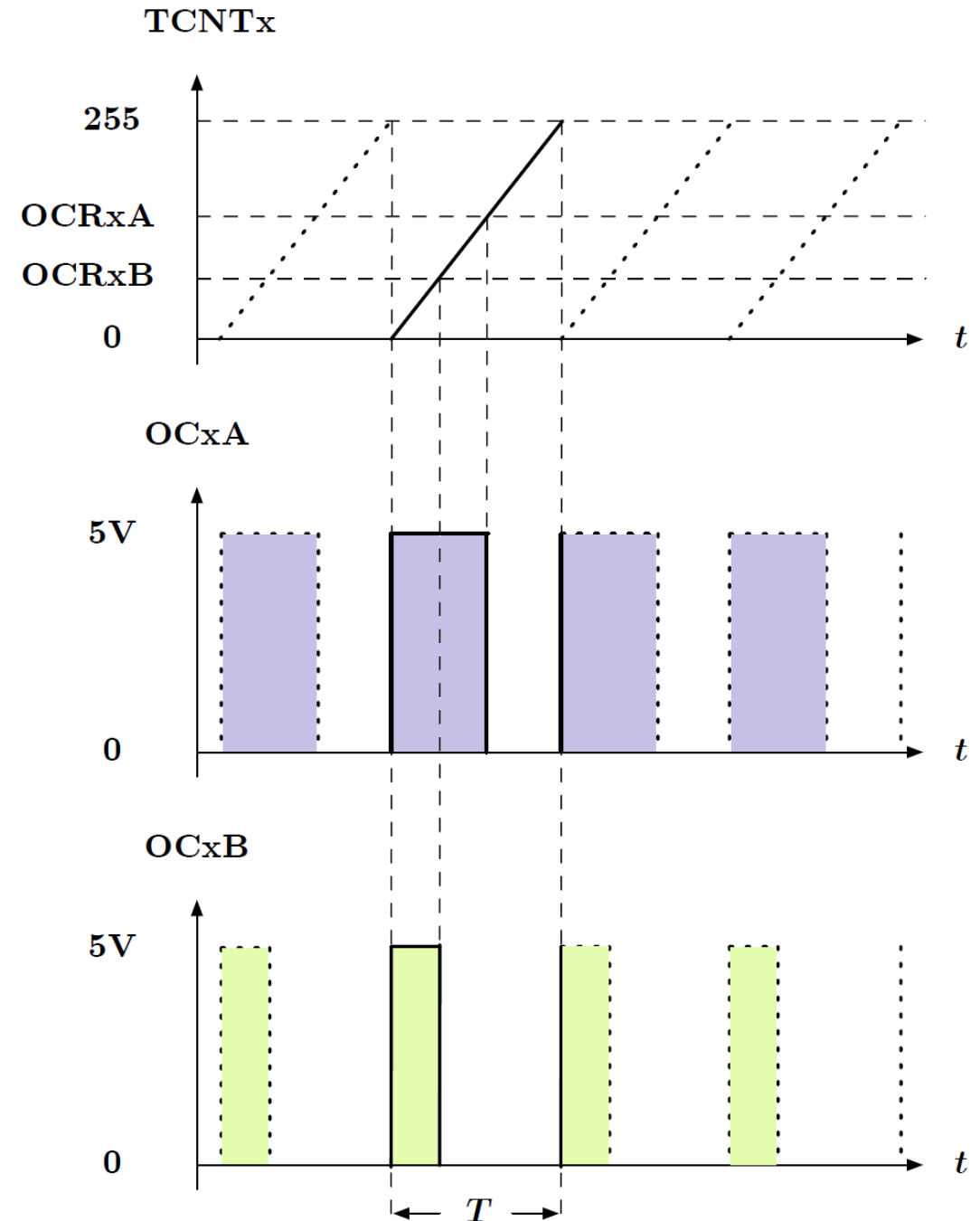


If $OCR = 255$, we have 100% duty cycle

If $OCR = 0$, we have 0% duty cycle.

Using an **8-bit timer**, we can have **256** possible levels of duty cycle.

Using a **16-bit timer**, we can have **65,536** possible levels of duty cycle.
(i.e. better resolution)



The frequency of the output PWM waveform is

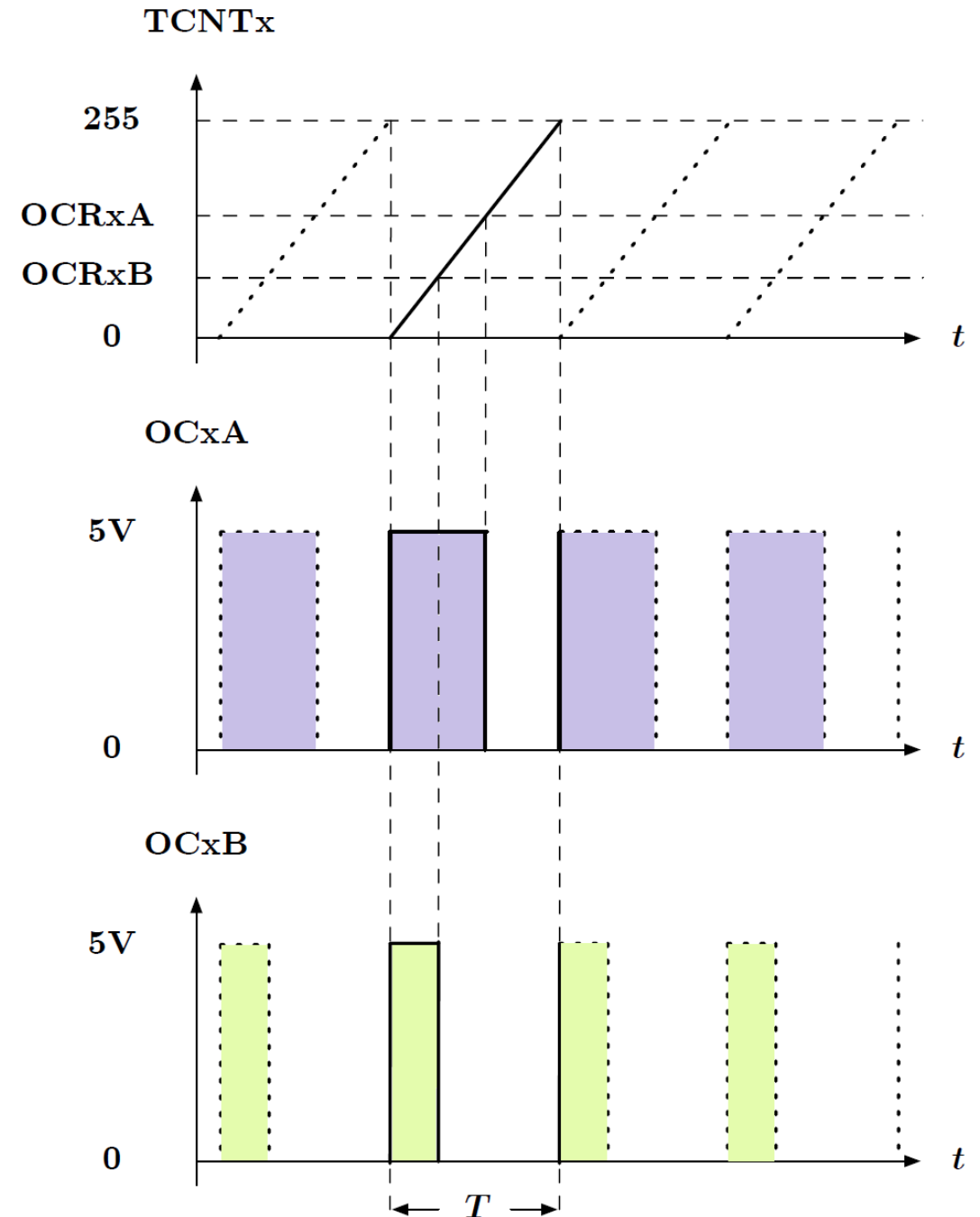
$$f_{pwm} = \frac{f_{clk}}{N}$$

where N is the total number of states (values) the counter experiences in one cycle

N = 256 for 8 bit

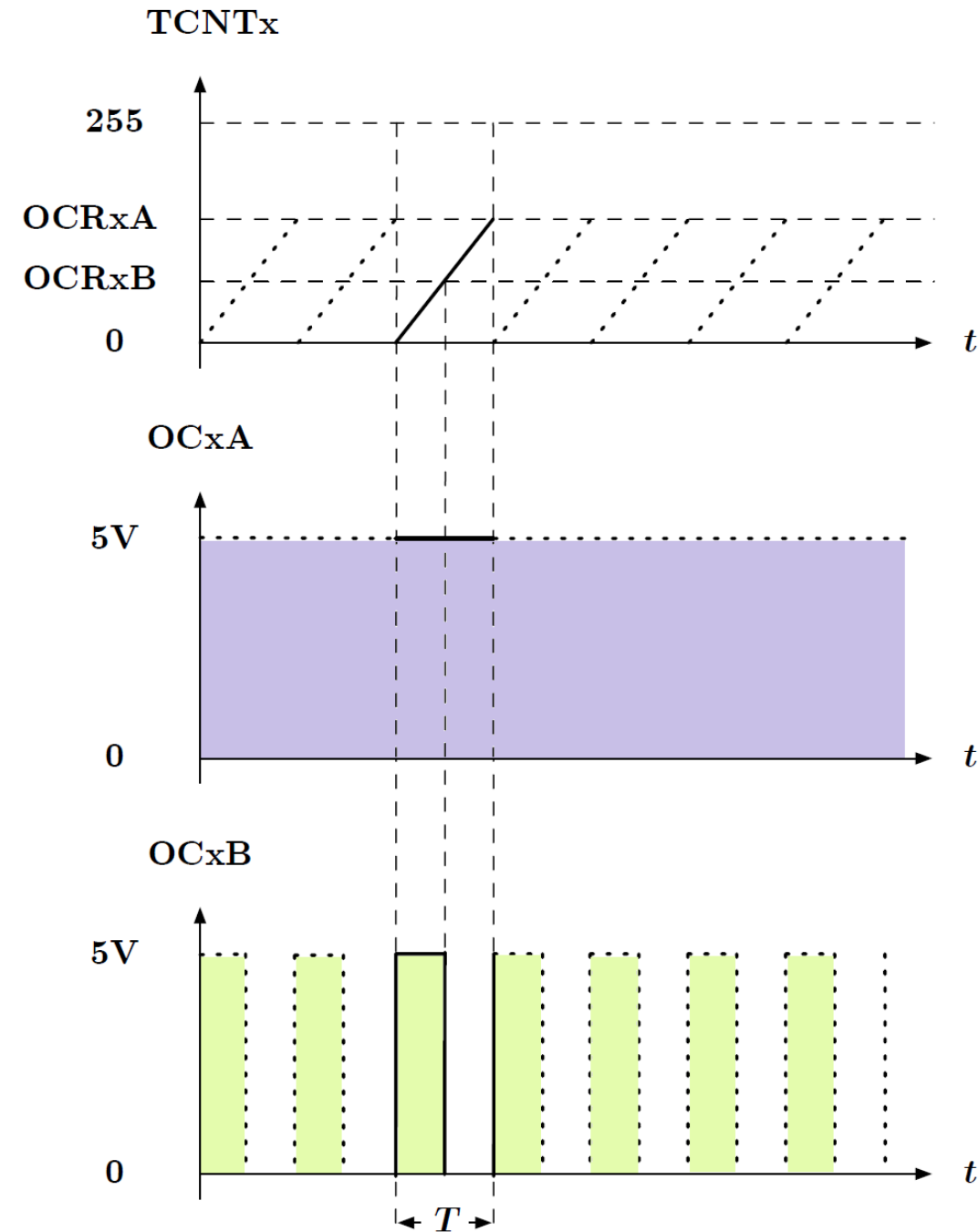
N = 65,536 for 16 bit

There is a frequency-vs-resolution trade-off



Fast PWM Mode with TOP value

- In this mode, the maximum value of the counter is OCRxA value (no more 255 or 65,535 as before)
- This mode effectively makes channel A unusable (HIGH all the time)
- The counter resets as soon as TCNT breaks the maximum value (OCRxA)



Fast PWM Mode with TOP value

Example:

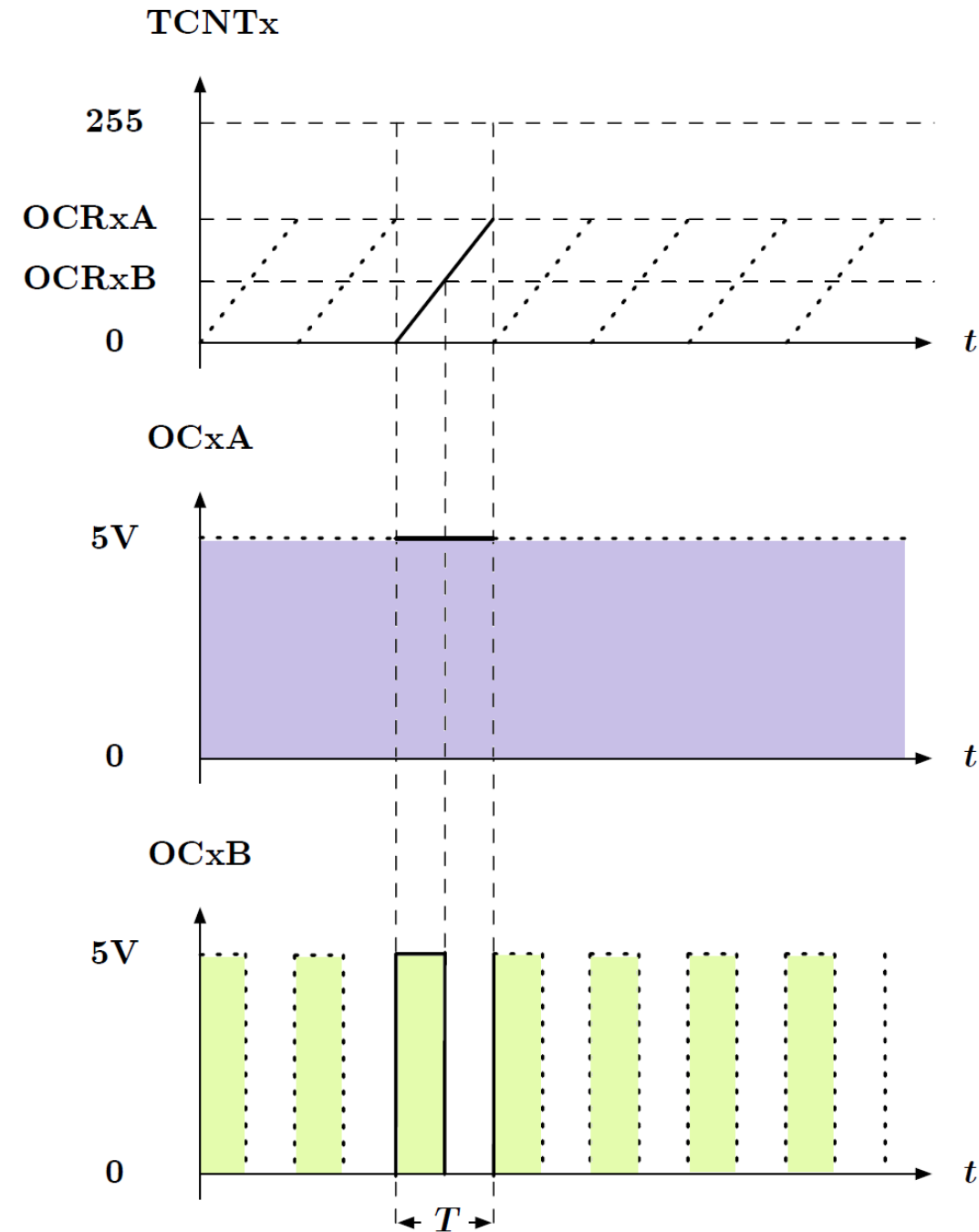
If $OCRxA = 150$

Setting $OCRxB = 150$, we have 100% duty cycle

Setting $OCRxB = 0$, we have 0% duty cycle

- We have only 156 possible levels of duty cycles
- But the PWM frequency has improved.

This mode improves the PWM frequency as the expense of PWM resolution.



Phase-correct Mode

In this mode, the counter counts-up to the maximum value and then counts-down to 0.

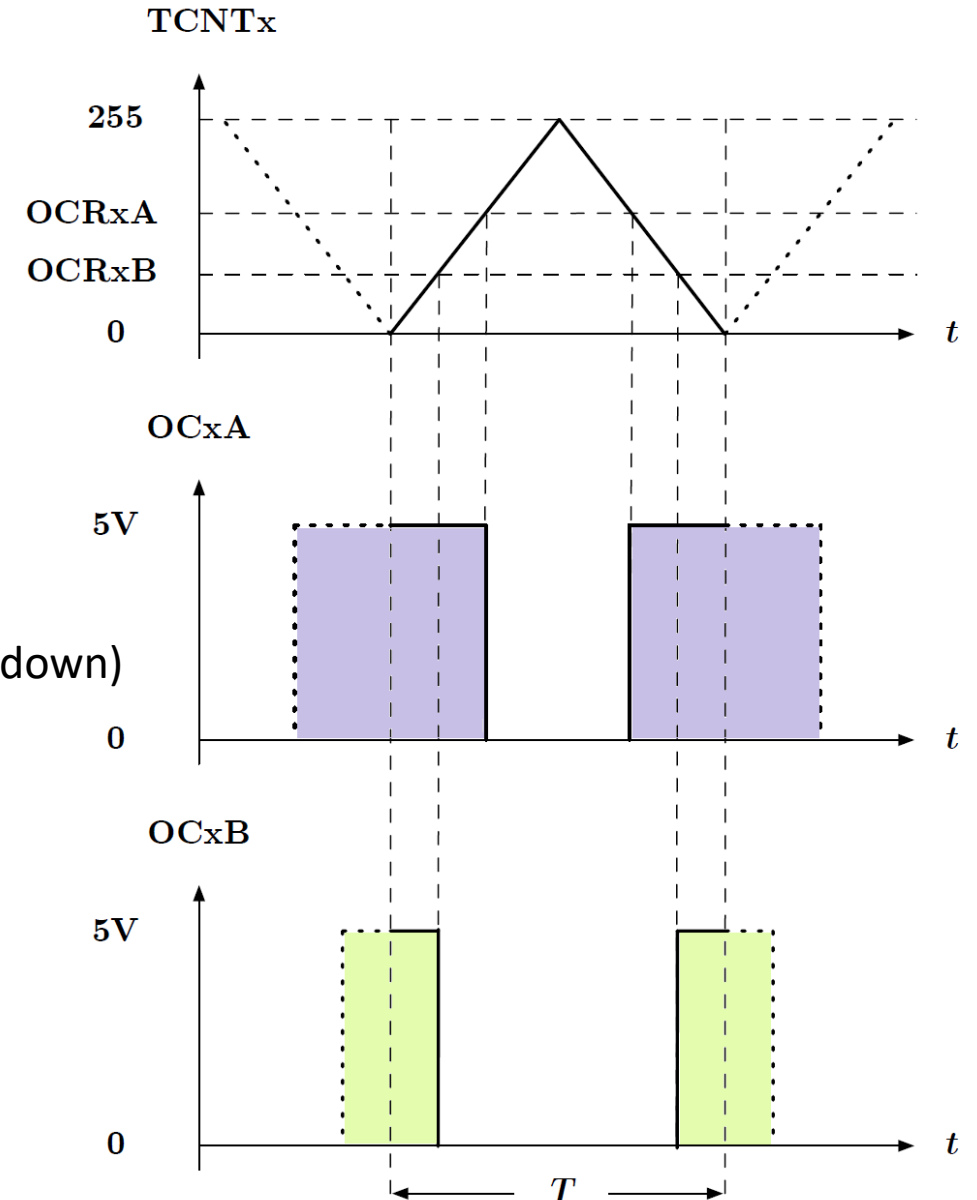
If maximum value = 255

Counting sequence = {0,1,2,.....253,254,255,254,253,.....}

Total number of states the counter experiences in one cycle
= 256 (count up) + 254 (count down)
= 510

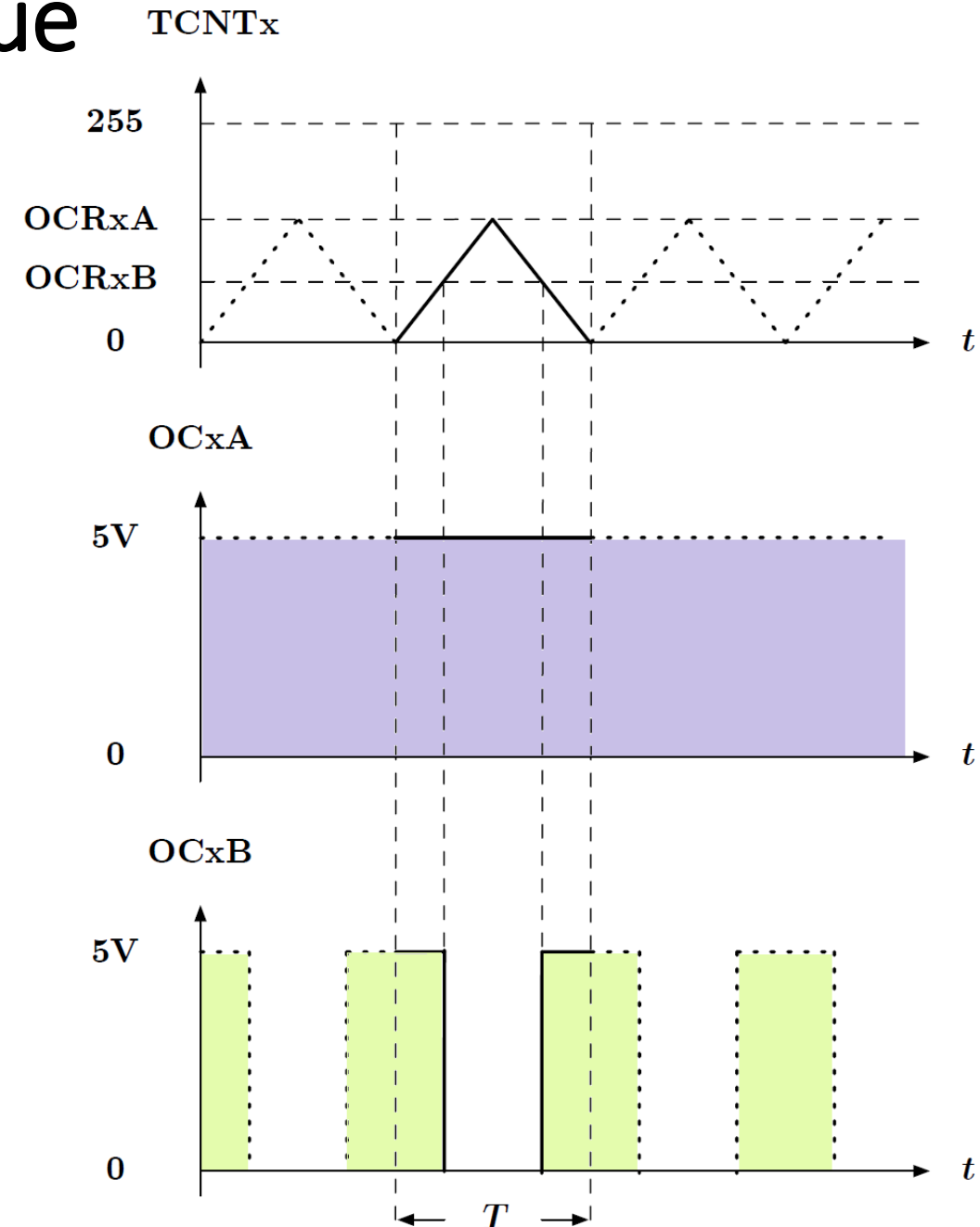
$$f_{pwm} = \frac{f_{clk}}{N}$$

The PWM frequency gets reduced by approximately a factor of 2.

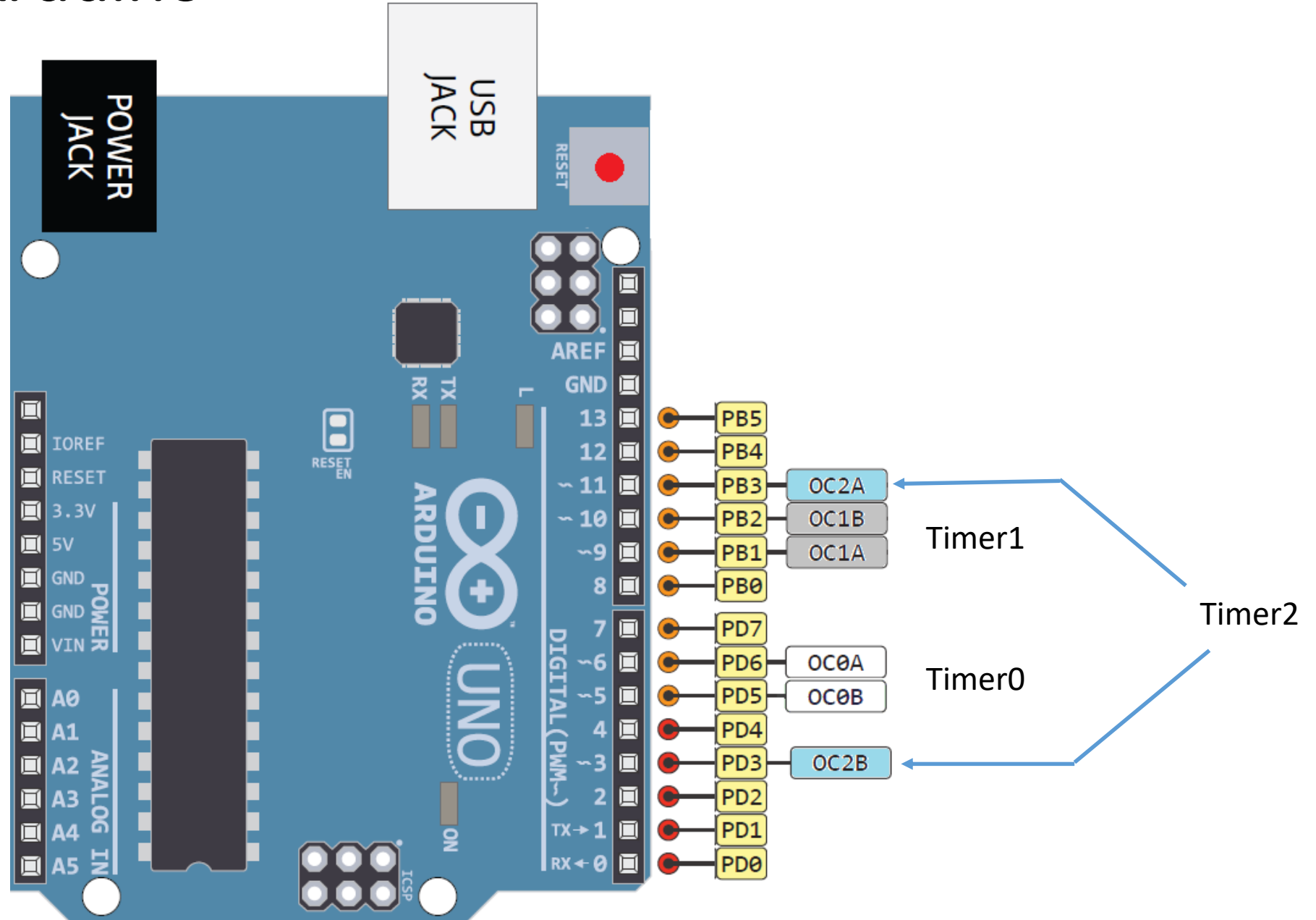


Phase-correct Mode with TOP value

- In this mode, the maximum value of the counter is OCRxA value (no more 255 or 65,535 as before)
- This mode effectively makes channel A unusable (HIGH all the time)



Timers on Arduino



Timers on Arduino

Timer	Default Arduino setting		Effective PWM frequency
timer0	Mode:	Fast PWM	976.56 Hz
	Top:	255	
	Pre-scaler:	64	
timer1 (16-bit)	Mode:	Phase-correct (only 8-bit used)	490.20 Hz
	Top:	255	
	Pre-scaler:	64	
timer2	Mode:	Phase-correct	490.20 Hz
	Top:	255	
	Pre-scaler:	64	

Note:

timer0 is utilized by millis() and micros() functions

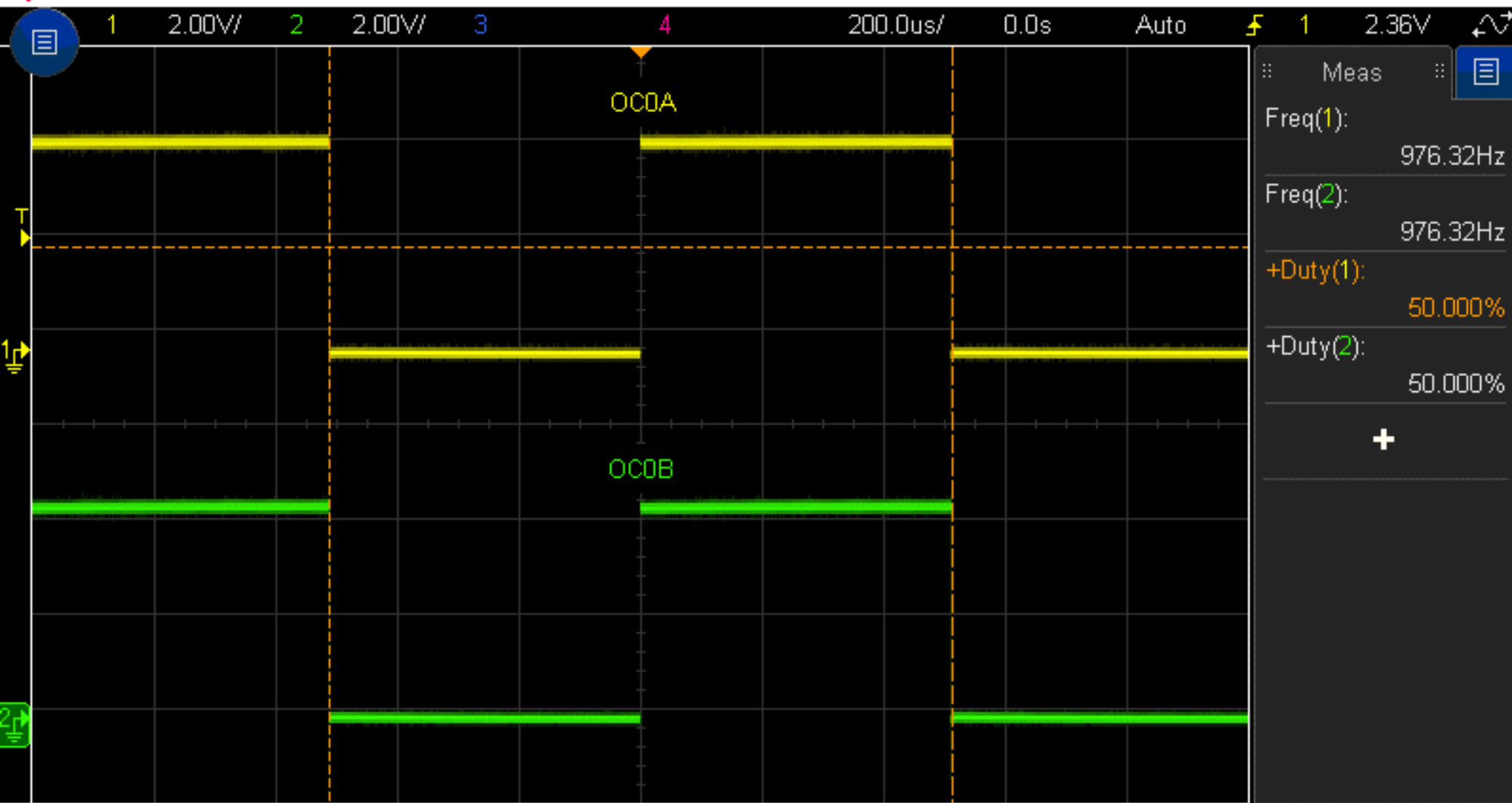
Arduino Example

```
void setup()  
{  
    analogWrite(3, 127);  
    analogWrite(5, 127);  
    analogWrite(6, 127);  
    analogWrite(9, 127);  
    analogWrite(10, 127);  
    analogWrite(11, 127);  
}
```

The value 127 gets stored in OCR0A, OCR0B,
OCR1A, OCR1B
OCR2A, OCR2B

analogWrite() does not have to be placed in the main loop.

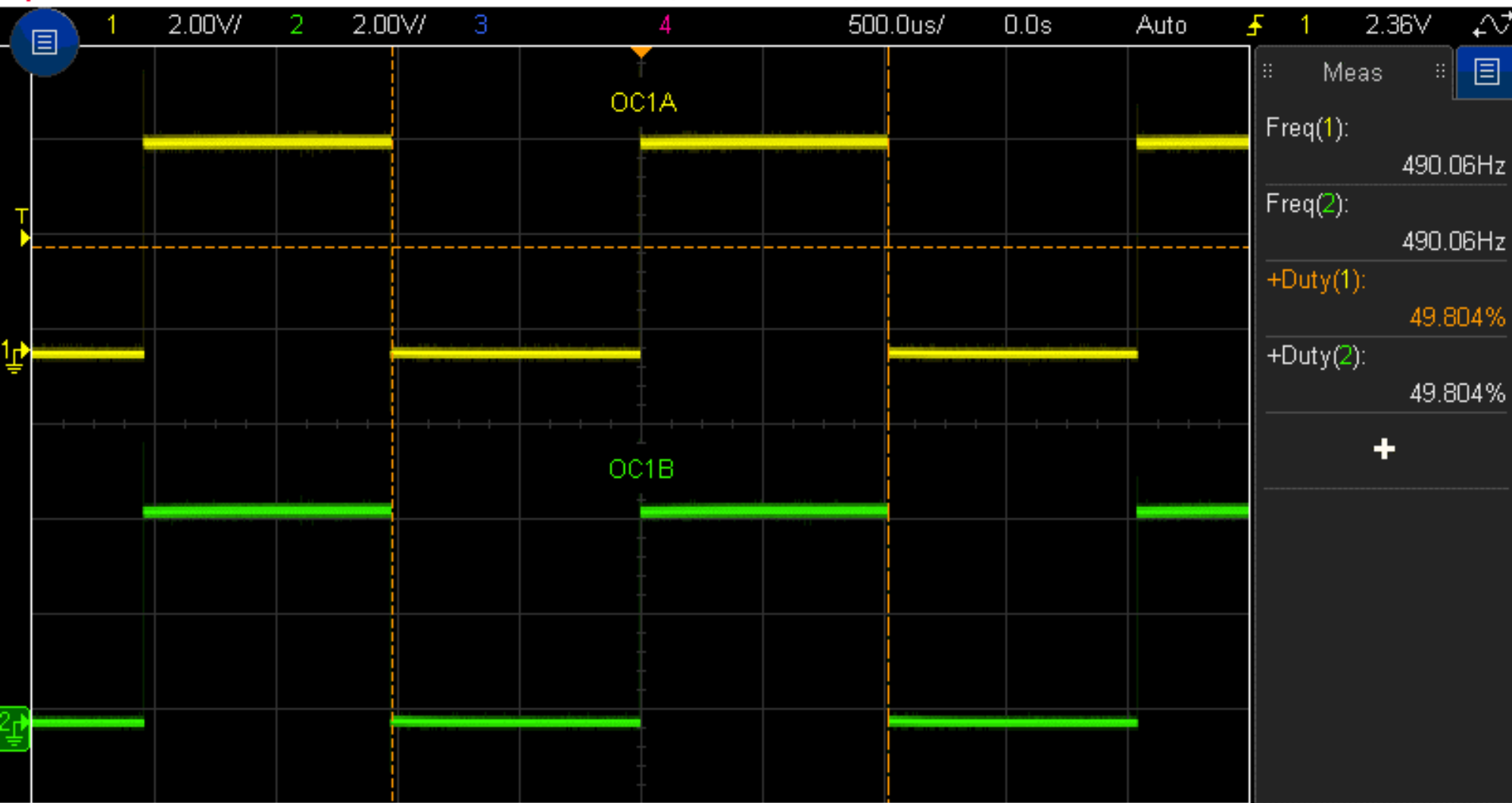
Hardware PWM circuitry is independent of the CPU



Meas	
Freq(1):	976.32Hz
Freq(2):	976.32Hz
+Duty(1):	50.000%
+Duty(2):	50.000%
+	

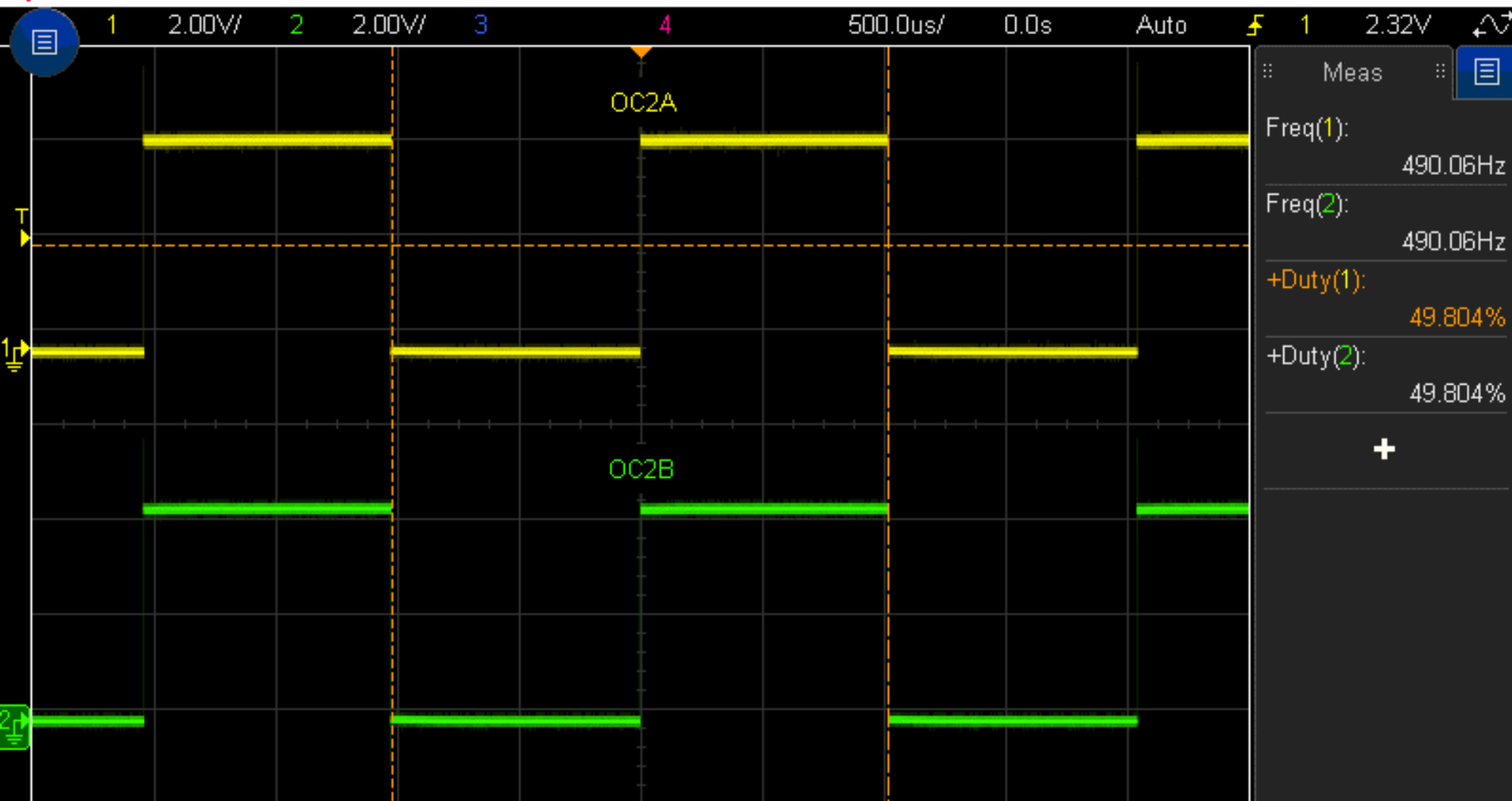
Output waveform of
both channels of
timer0

Measured frequency
= 976.32Hz



Output waveform of
both channels of
timer1

Measured frequency
= 490.06Hz



Output waveform of
both channels of
timer2

Measured frequency
= 490.06Hz

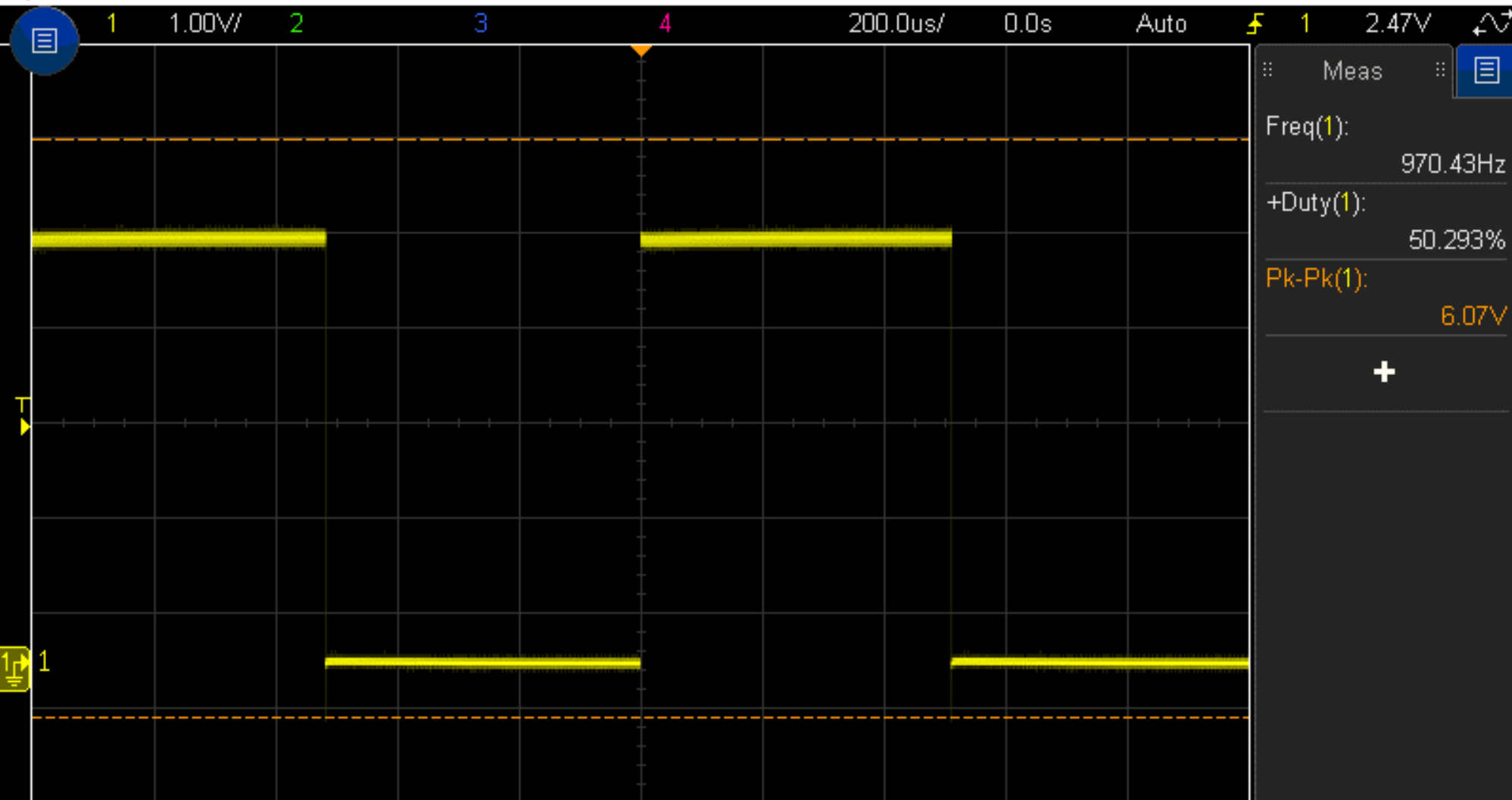
Software PWM

- CPU can also generate PWM on any GPIO pin.

```
void setup()  
{  
  
    *ddrd = 255;  
  
    while(1)  
    {  
        *portd = 255;           //Turn ON all the PORT D pins  
        delayMicroseconds(512);  
        *portd = 0;             //Turn OFF all the PORT D pins  
        delayMicroseconds(512);  
    }  
}
```

Expected duty cycle = 50%

Expected frequency = ?



Output waveform of
one of the Port D
pins

Measured frequency
= 970.43Hz

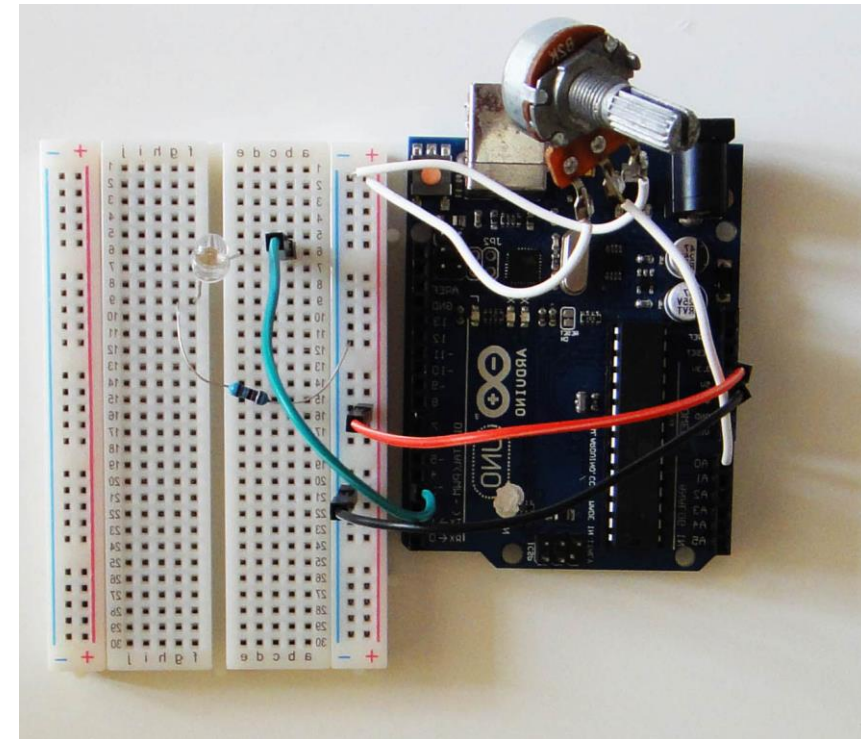
PWM applications – LED Dimmer

```
int main()    //To run on Arduino, change to void setup()
{

    Serial.begin(9600);

    while(1)
    {
        int value = analogRead(A0);
        value = value / 4;
        analogWrite(3, value);
        Serial.println(value);
    }
}
```

This program reads 10-bit analog value from PC0 pin and then converts to 8-bit and write it to PD3.



LED connected to PD3.

Potentiometer connected to PC0



Reading on PD3 when duty cycle = 100%



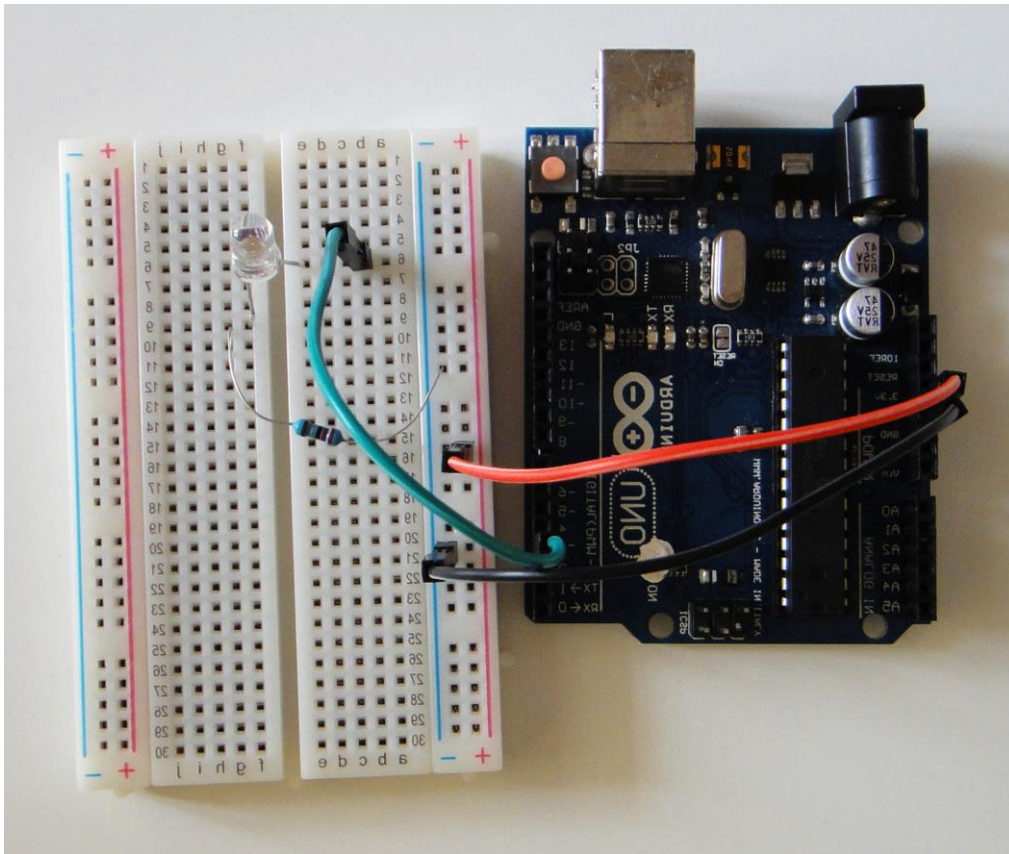
Reading on PD3 when duty cycle = 50%



490Hz is too fast for multimeter. Only the “average” gets registered

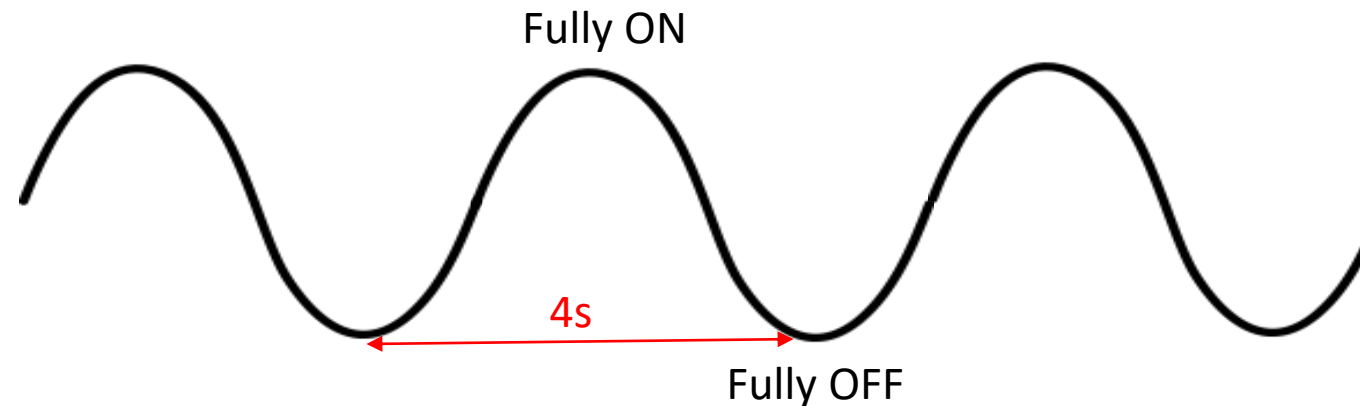
PWM applications – Auto-fading LED (sinusoidal pattern)

LED connected to PD3.



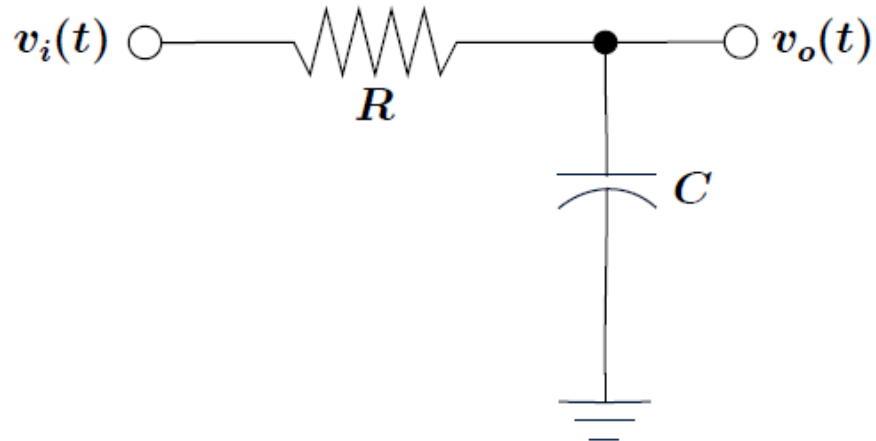
This program controls the brightness of the LED in a sinusoidal pattern with a period of 4 seconds.

Code hidden

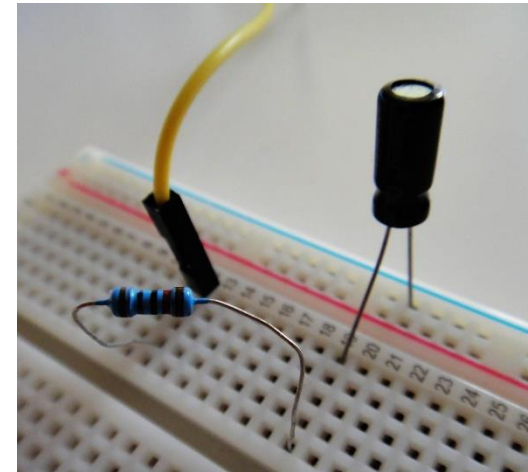


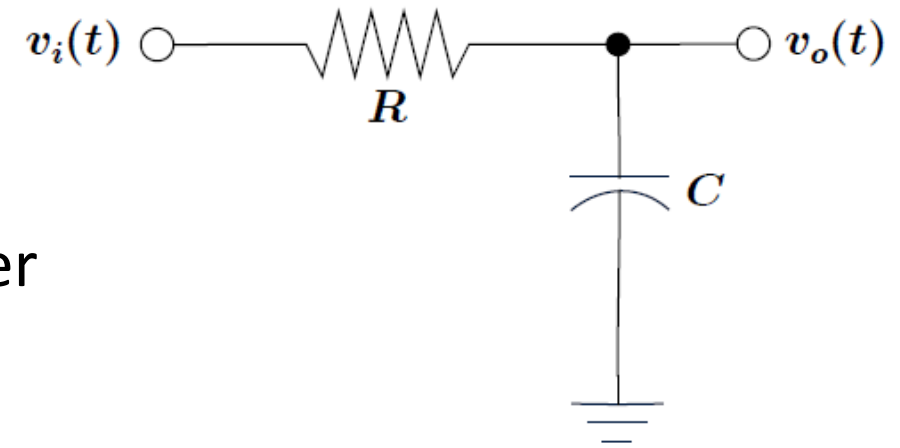
Demodulation

- Converting PWM signal to the true analog signal is called **demodulation**.
- A low-pass filter can be utilized.



First-order RC low-pass filter





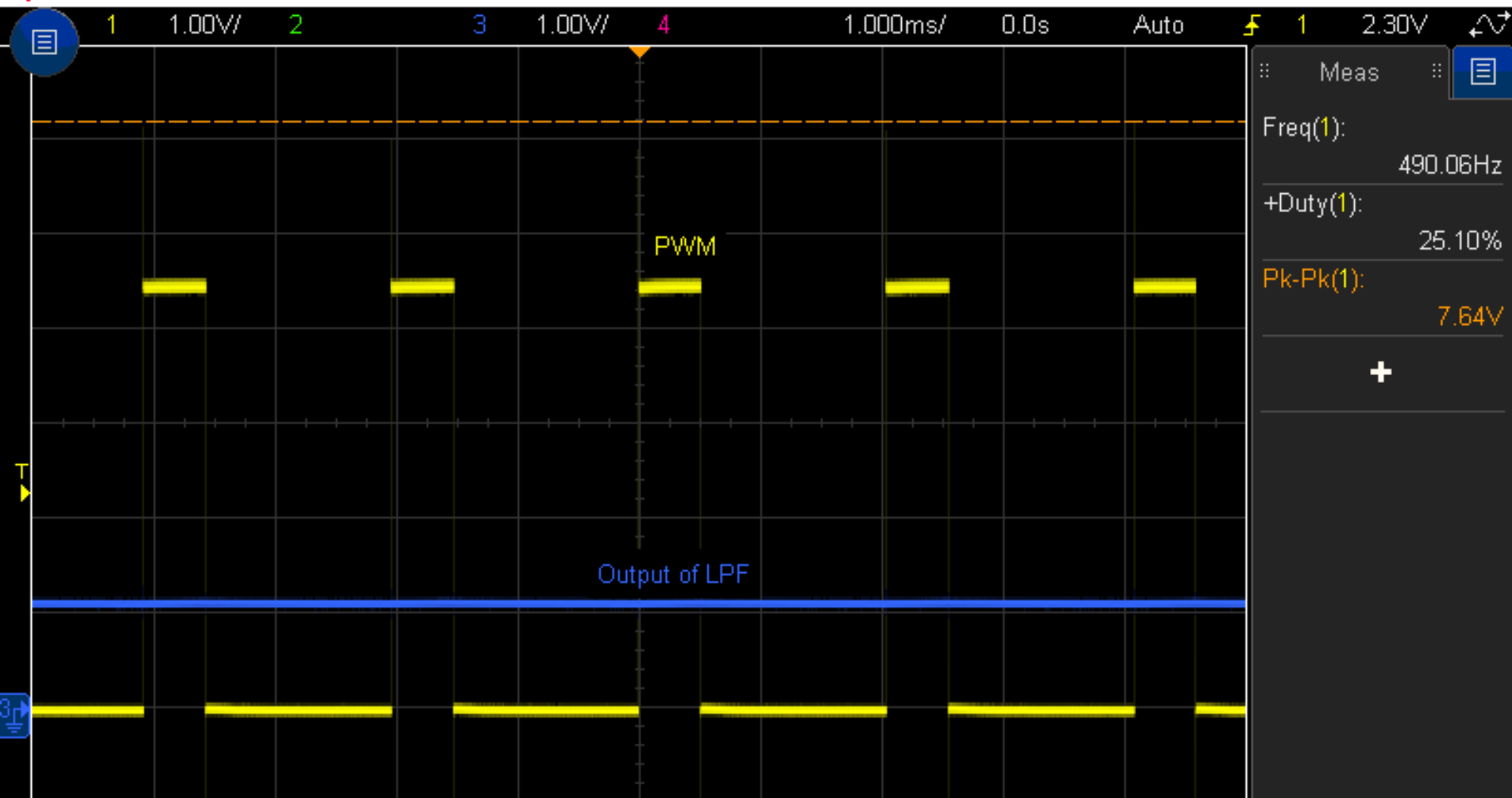
The cutoff frequency of a first-order RC low-pass filter
Is given by

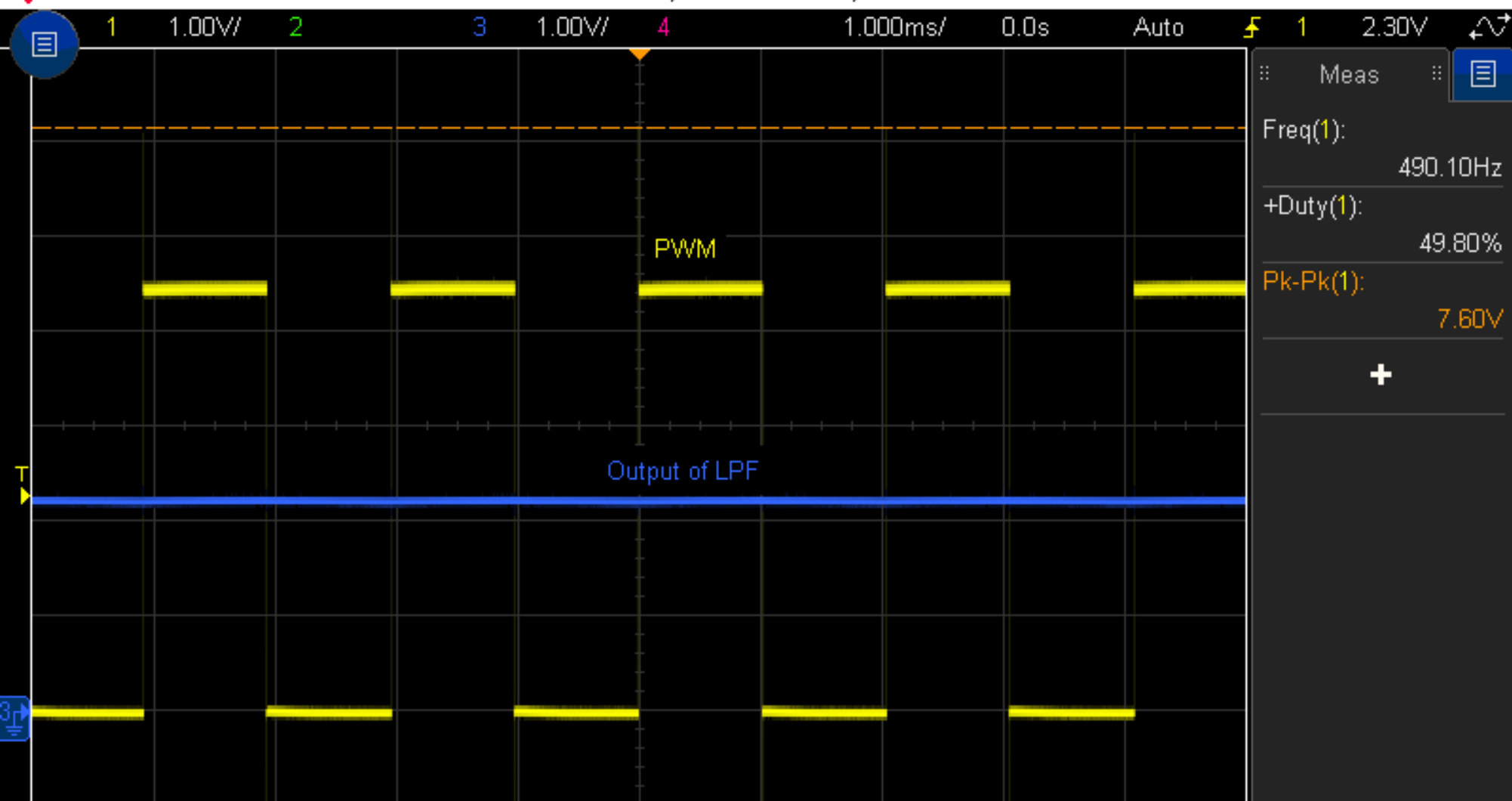
$$f_c = \frac{1}{2\pi RC}$$

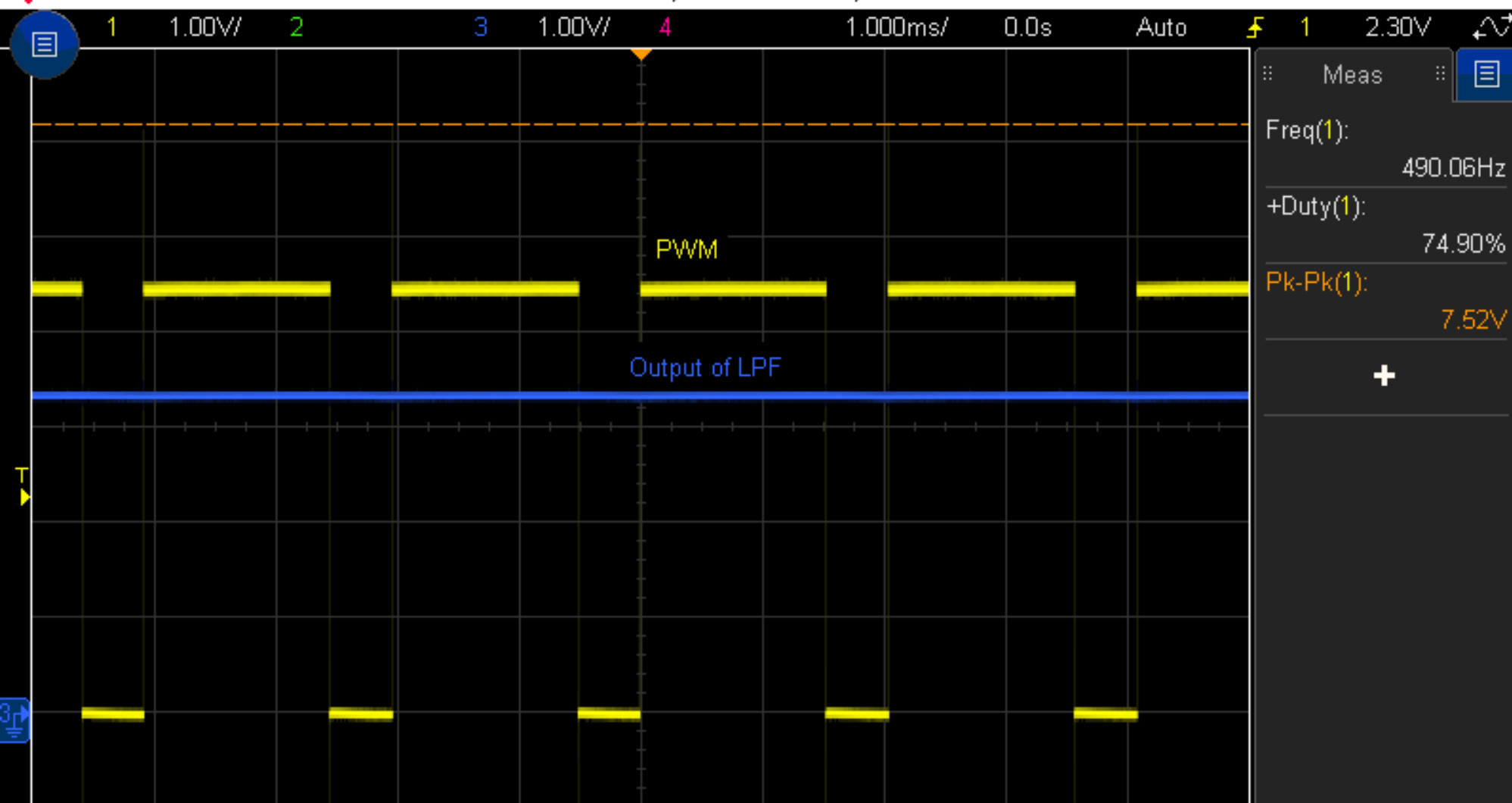
The cutoff frequency has to satisfy the following constraint

$$f_m \ll f_c \ll f_{pwm}$$


where f_m is the maximum frequency of the analog message signal.







Timer Registers

timer0 (8-bit)	timer1 (16-bit)	timer2 (8-bit)
TCCR0A TCCR0B	TCCR1A TCCR1B TCCR1C  Requires 3 control registers because of many functions this timer provides	TCCR2A TCCR2B
TCNT0	TCNT1H and TCNT1L	TCNT2
OCR0A OCR0B	OCR1AH and OCR1AL OCR1BH and OCR1BL	OCR2A OCR2B

TCCR: **T**imer/**C**ounter **C**ontrol **R**egister (Sets PWM mode and pre-scaler)

TCNT: **T**imer **Cou****N****T** (Stores raw value of the counter)

OCR: **O**utput **C**ompare **R**egister (Affects duty cycle)

TCCR0A

Timer/Counter0 Control Register A(x44)

Bit	7	6	5	4	3	2	1	0
0x44	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Default	0	0	0	0	0	0	0	0

WGM	Mode	TOP
000	*	*
001	Phase Correct PWM	0xFF
010	*	*
011	Fast PWM	0xFF
100	Reserved	-
101	Phase Correct PWM	OCR0A
110	Reserved	-
111	Fast PWM	OCR0A

COMx	Description
00	PWM disabled
01	*
10	Normal output
11	Output inverted

* Out of the scope of this course.
Refer to the datasheet

Highlighted row = default on Arduino

TCCROB

Timer/Counter0 Control Register B (x45)

Bit	7	6	5	4	3	2	1	0
0x45	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

CS	Pre-scaler
000	0
001	1
010	8
011	64
100	256
101	1024
110	*
111	*

FOC0A and FOC0B: *

CS: Clock source

WGM02 is the MSB of WGM defined in TCCR0A

* Out of the scope of this course

TCNT0

Timer/Counter0 Register (x46)

Bit	7	6	5	4	3	2	1	0
0x46	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Stores raw value of the counter (8-bit)

OCR0A

Output Compare0 Register A (x47)

Bit	7	6	5	4	3	2	1	0
0x47	OCR0A [7 : 0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register affects the duty cycle of Channel A of timer0.

OCROB

Output Compare0 Register B (x48)

Bit	7	6	5	4	3	2	1	0
0x48	OCROB [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register affects the duty cycle of Channel B of timer0.

TCCR1A

Timer/Counter1 Control Register A (x80)

Bit	7	6	5	4	3	2	1	0
0x80	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Default	0	0	0	0	0	0	0	0

WGM	Mode	TOP
0001	Phase correct PWM 8-bit	0x00FF
0010	Phase correct PWM 9-bit	0x01FF
0011	Phase correct PWM 10-bit	0x03FF
0101	Fast PWM 8-bit	0x00FF
0110	Fast PWM 9-bit	0x01FF
0111	Fast PWM 10-bit	0x03FF
1011	Phase correct PWM	OCR1A
1111	Fast PWM	OCR1A

COMx	Description
00	PWM disabled
01	*
10	Normal output
11	Output inverted

TCCR1B

Timer/Counter1 Control Register B (x81)

Bit	7	6	5	4	3	2	1	0
0x81	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

CS	Pre-scaler
000	0
001	1
010	8
011	64
100	256
101	1024
110	*
111	*

1CNC1, ICES1: *

CS: Clock source

* Out of the scope of this course

TCCR1C

Timer/Counter1 Control Register C (x82)

Bit	7	6	5	4	3	2	1	0
0x82	FOC1A	FOC1B	-	-	-	-	-	-
Read/Write	W	W	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

FOC1A and FOC1B: Out of the scope of this course

TCNT1H and TCNT1L

Timer/Counter1 Register – High byte and low byte (x84-85)

Bit	7	6	5	4	3	2	1	0
0x85	TCNT1[15:8]							
0x84	TCNT1[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Stores raw value of the counter.

Because timer1 is 16-bit, two bytes are required.

OCR1AH and OCR1AL

Output Compare1 Register A – High byte and low byte (x88-89)

Bit	7	6	5	4	3	2	1	0
0x89	OCR1A[15:8]							
0x88	OCR1A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These registers affects the duty cycle of Channel A of timer1.
Because timer1 is 16-bit, two bytes are required.

OCR1BH and OCR1BL

Output Compare1 Register B – High byte and low byte (x8A-8B)

Bit	7	6	5	4	3	2	1	0
0x8B	OCR1B[15:8]							
0x8A	OCR1B[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These registers affects the duty cycle of Channel B of timer1.
Because timer1 is 16-bit, two bytes are required.

TCCR2A

Timer/Counter2 Control Register A (xB0)

Bit	7	6	5	4	3	2	1	0
0xB0	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W
Default	0	0	0	0	0	0	0	0

WGM	Mode	TOP
000	*	*
001	Phase Correct PWM	0xFF
010	*	*
011	Fast PWM	0xFF
100	Reserved	-
101	Phase Correct PWM	OCR2A
110	Reserved	-
111	Fast PWM	OCR2A

COMx	Description
00	PWM disabled
01	*
10	Normal output
11	Output inverted

TCCR2B

Timer/Counter2 Control Register B (xB1)

Bit	7	6	5	4	3	2	1	0
0xB1	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

CS	Pre-scaler
000	0
001	1
010	8
011	32
100	64
101	128
110	256
111	1024

FOC2A and FOC2B: Out of scope of this course

TCNT2

Timer/Counter2 Register (xB2)

Bit	7	6	5	4	3	2	1	0
0xB2	TCNT2[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Stores raw value of the counter (8-bit)

OCR2A

Output Compare2 Register A(xB3)

Bit	7	6	5	4	3	2	1	0
0xB3	OCR2A[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register affects the duty cycle of Channel A of timer2.

OCR2B

Output Compare2 Register B (xB4)

Bit	7	6	5	4	3	2	1	0
0xB4	OCR2B [7 : 0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register affects the duty cycle of Channel B of timer2.

Example 1

Write a program to produce a PWM signal with 50% duty cycle on Channel B of timer0. Use Fast PWM mode and pre-scaler value of 64.

Determine the theoretical frequency of the generated PWM signal.

Style #1

```
int main()
{
    unsigned char* ddrd = (unsigned char*) 0x2A;
    unsigned char* ocr0b = (unsigned char*) 0x48;
    unsigned char* tccr0a = (unsigned char*) 0x44;
    unsigned char* tccr0b = (unsigned char*) 0x45;

    *ddrd = 1 << 5;
    *ocr0b = 127;
    *tccr0a = 0b00100011;
    *tccr0b = 0b00000011;
}
```

Style #1 is the preferred option for this course

Style #2 can be used for learning and experimentation

Same



As mentioned before, these two pieces of code achieve the same thing.

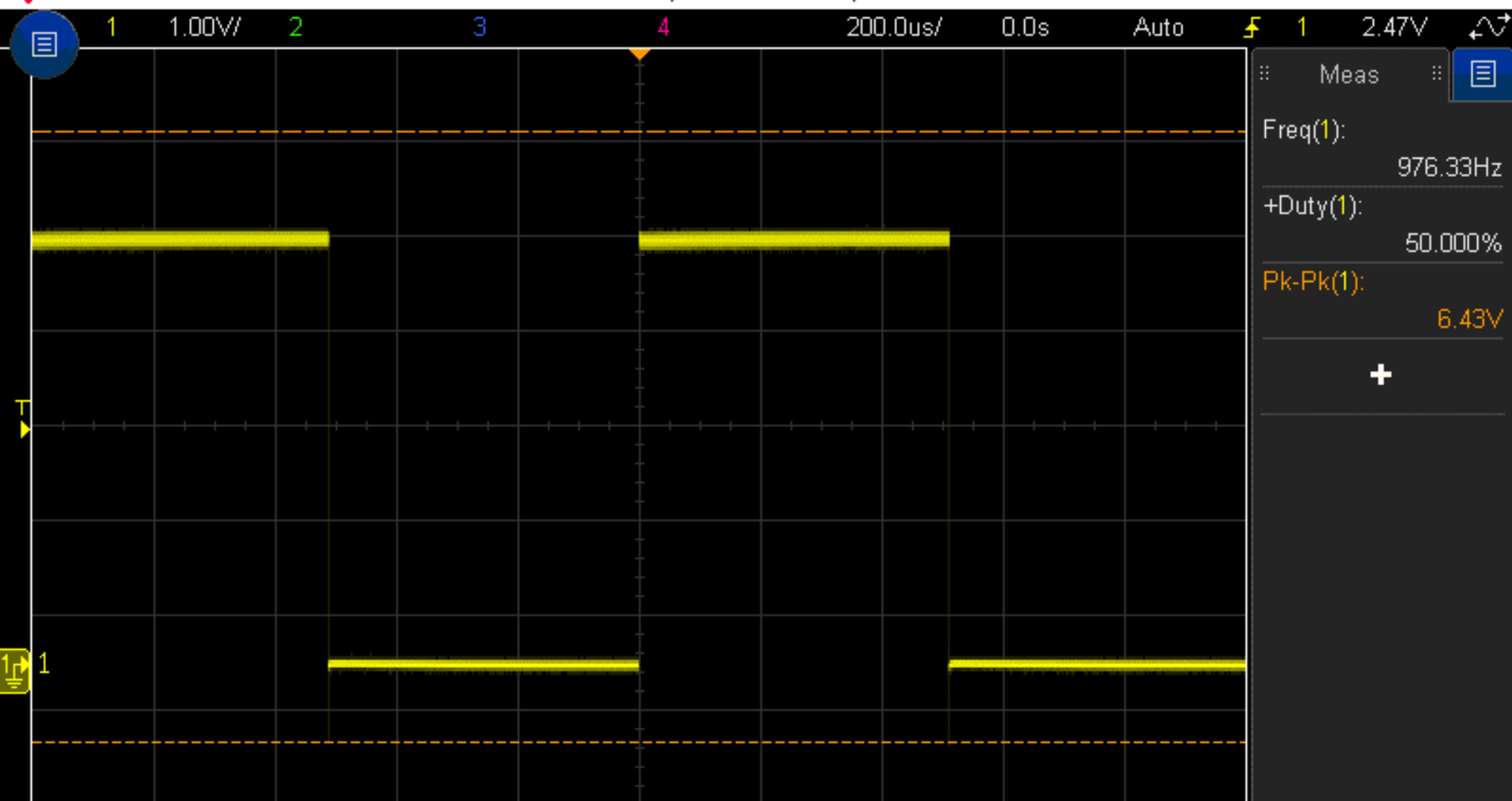
The Arduino library uses AVR library which predefines variables that have the same names as the registers.

Style #2 uses Arduino/AVR library.

In this course, we try to make the program as library-free as possible

Style #2

```
int main()
{
    DDRB = 1 << 5;
    OCR1AL = 127;
    TCCR1A = 0b10000011;
    TCCR1B = 0b00000010;
}
```

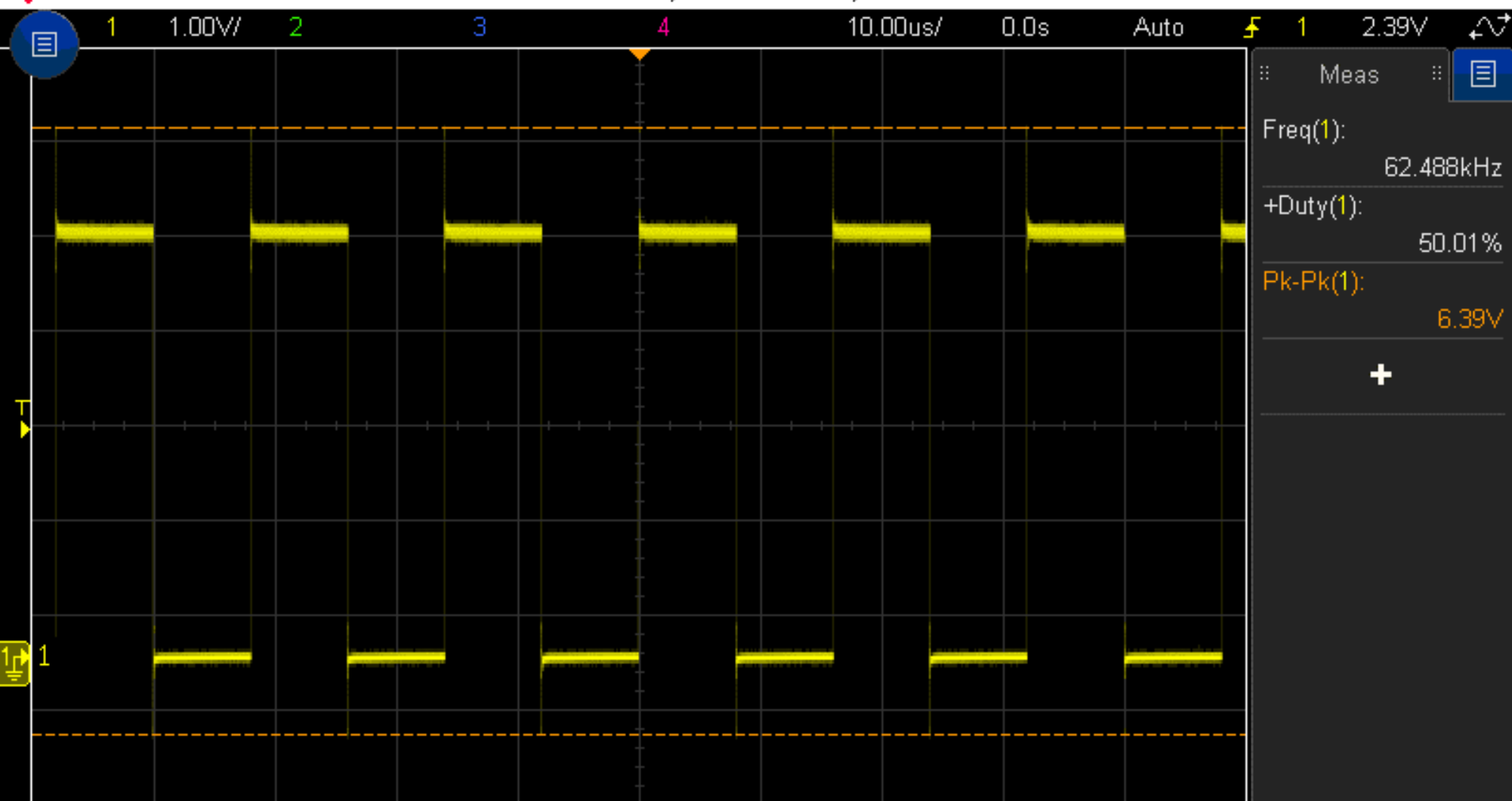



Example 2

Write a program to produce a PWM signal with 50% duty cycle on Channel B of timer0. Use Fast PWM mode and pre-scaler value of 1.

Determine the theoretical frequency of the generated PWM signal.

```
int main()  
{  
    unsigned char* ddrd = (unsigned char*) 0x2A;  
    unsigned char* ocr0b = (unsigned char*) 0x48;  
    unsigned char* tccr0a = (unsigned char*) 0x44;  
    unsigned char* tccr0b = (unsigned char*) 0x45;  
  
    *ddrd = 1 << 5;  
    *ocr0b = 127;  
    *tccr0a = 0b00100011;  
    *tccr0b = 0b00000001;  
}
```



Example 3

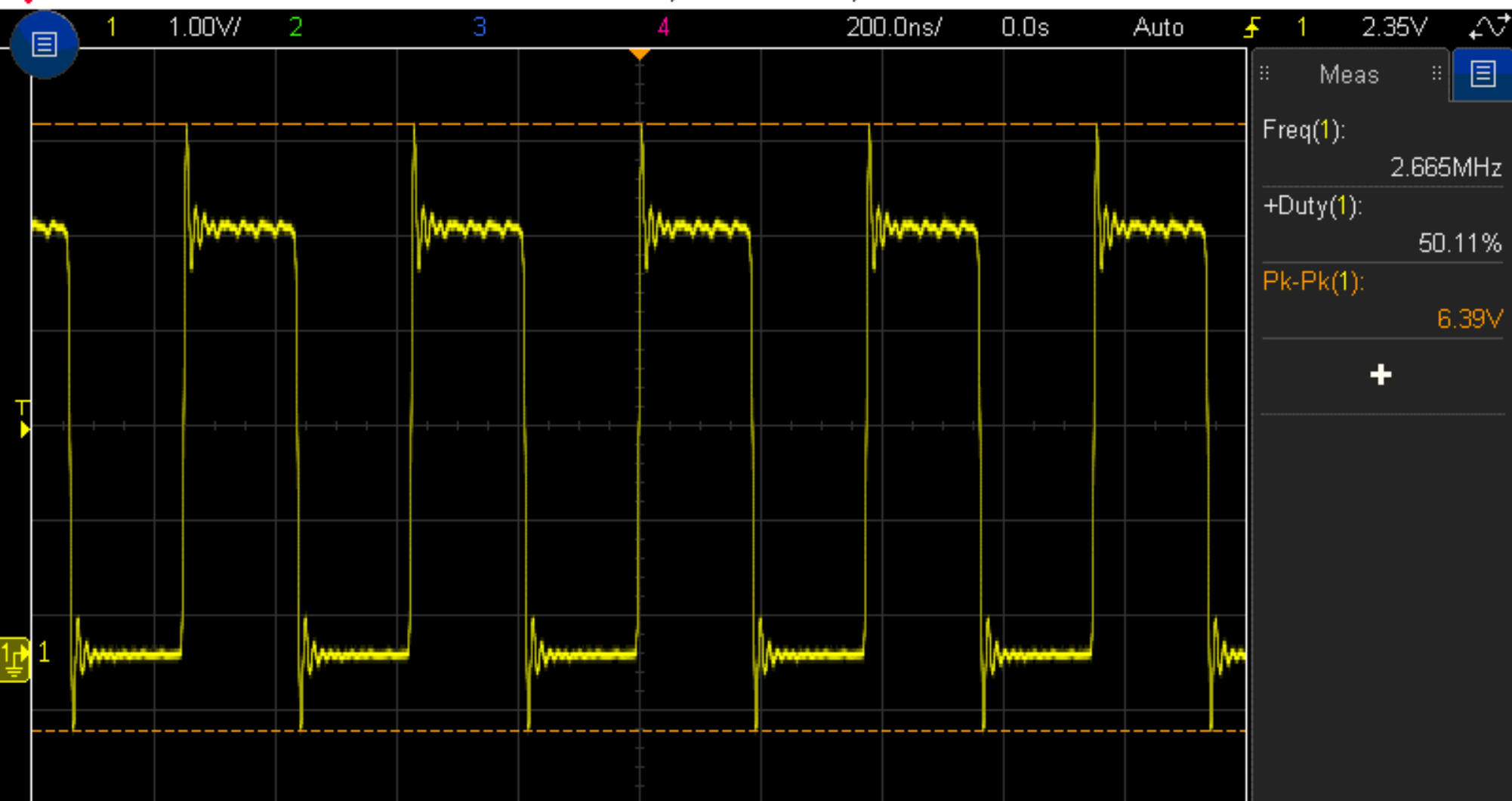
Write a program to produce a PWM signal with 50% duty cycle on Channel B of timer0. Use Fast PWM mode, pre-scaler value of 1, top value of 5.

Determine the theoretical frequency of the generated PWM signal.

List **all** the PWM duty cycles that can be generated using the above settings.

0% 25% 50% 75% 100%

```
int main()  
{  
    unsigned char* ddrd = (unsigned char*) 0x2A;  
    unsigned char* ocr0a = (unsigned char*) 0x47;  
    unsigned char* ocr0b = (unsigned char*) 0x48;  
    unsigned char* tccr0a = (unsigned char*) 0x44;  
    unsigned char* tccr0b = (unsigned char*) 0x45;  
  
    *ddrd = 1 << 5;  
    *ocr0a = 5;  
    *ocr0b = 2;  
    *tccr0a = 0b00100011;  
    *tccr0b = 0b00001001;  
}
```



Example 4

Write a program to produce a PWM signal with 25% duty cycle on Channel A of timer1. Use 10-bit Phase Correct PWM-mode and pre-scaler value of 8.

Determine the theoretical frequency of the generated PWM signal.


```
int main()  
{  
    unsigned char* ddrb = (unsigned char*) 0x24;  
    unsigned char* ocr1a1 = (unsigned char*) 0x88;  
    unsigned char* tccr1a = (unsigned char*) 0x80;  
    unsigned char* tccr1b = (unsigned char*) 0x81;  
  
    *ddrb = 2;  
    *ocr1a1 = 255;  
    *tccr1a = 0b10000011;  
    *tccr1b = 0b00000010;  
}
```

