



**Department of Computer Engineering**  
**Faculty of Engineering**  
**Prince of Songkla University**

**240-319**

# **Embedded System Developer Module**



March 23, 2023

*Associate Prof. Dr. Panyayot Chaikan*  
*[panyayot@coe.psu.ac.th](mailto:panyayot@coe.psu.ac.th)*



# Chapter 7

Timer/Counter :

fast PWM and phase correct PWM



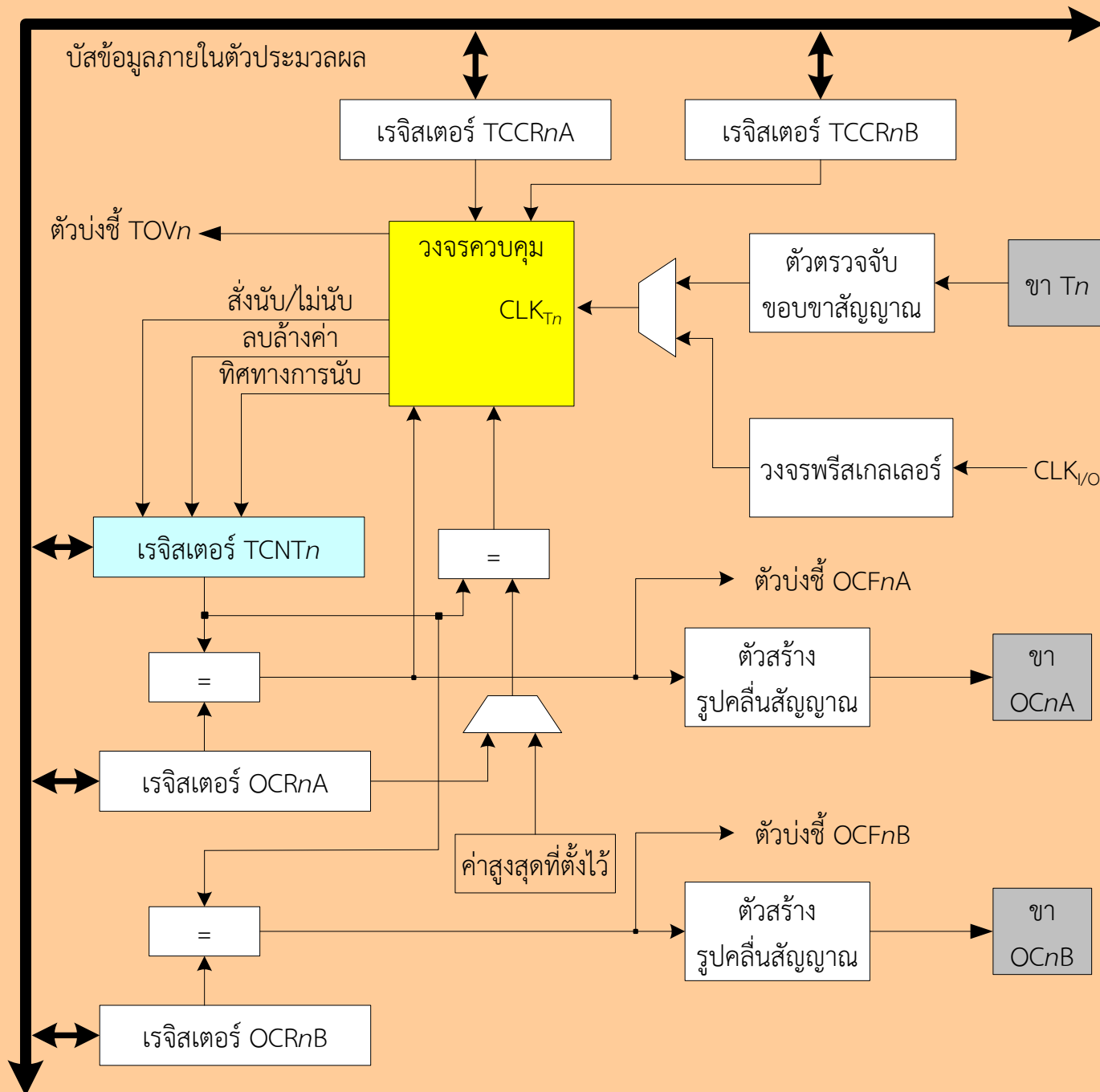


# เนื้อหา

การมอดูเลตความกว้างของพัลส์ด้วยวงจรรีบ/จับเวลา  
แบบวิธีการมอดูเลตความกว้างของพัลส์และตัวสร้างรูปคลื่นสัญญาณ



# วงจรนับ/จับเวลาหมายเลข 0 (Timer/Counter0)



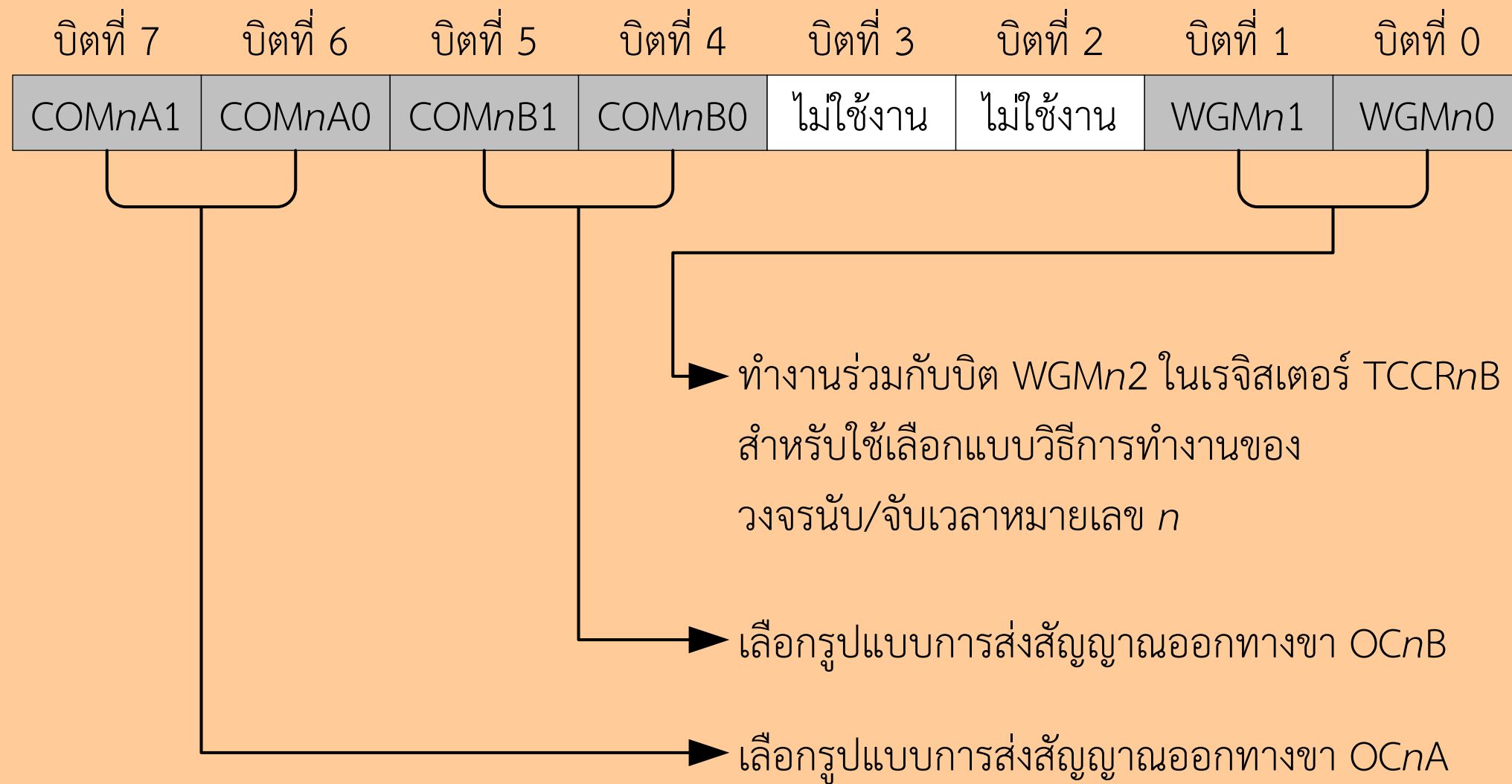


# เรจิสเตอร์ที่เกี่ยวข้องกับ Timer/Counter0

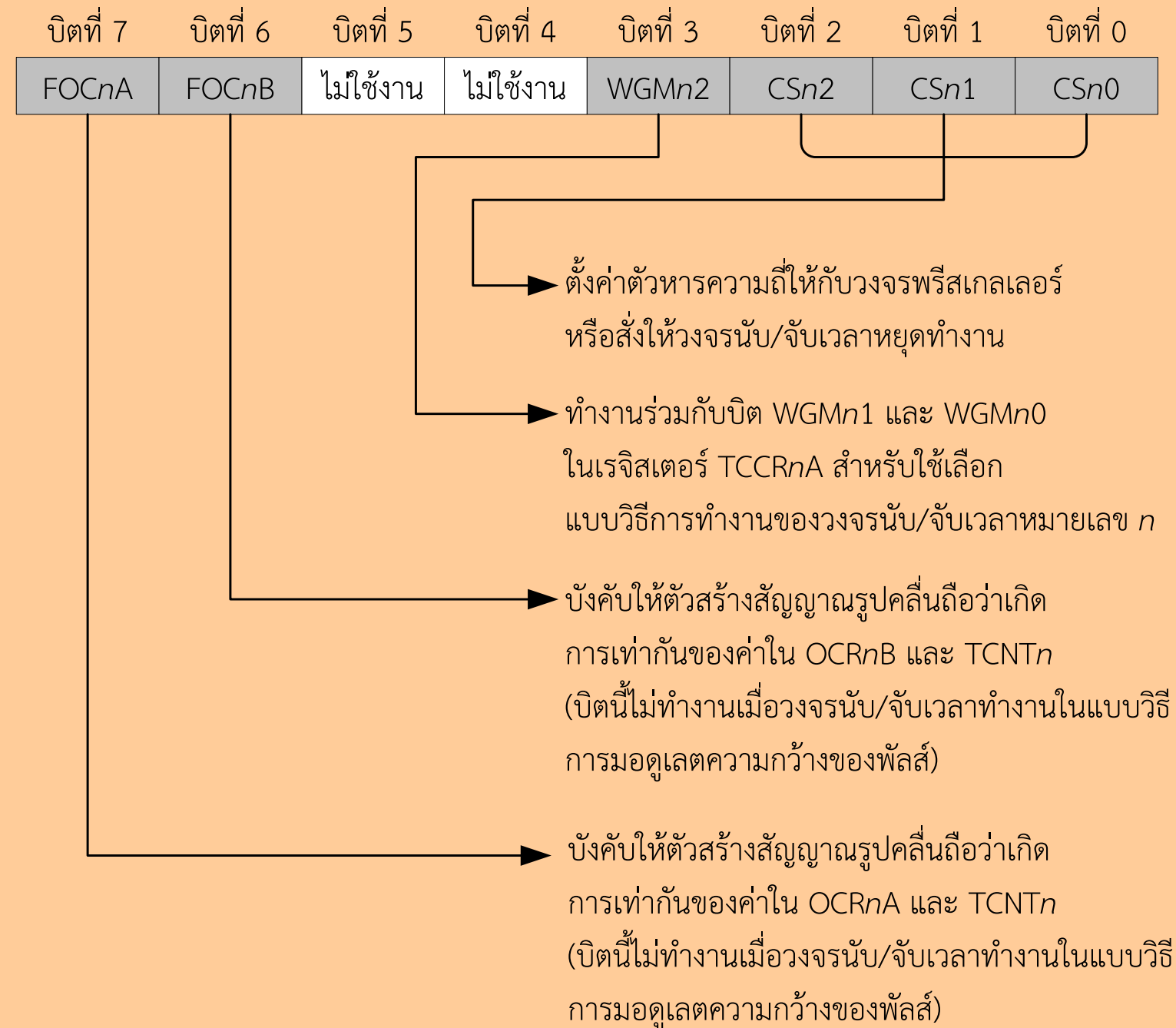
- ◆ TCNT0 Timer/Counter0
- ◆ OCR0A Output Compare Register 0 A
- ◆ OCR0B Output Compare Register 0 B
- ◆ TIFR0 Timer Interrupt Flag Register 0
- ◆ TIMSK0 Timer Interrupt Mask Register 0



# TCCR<sub>n</sub>A – Timer/Counter Control Register A



# TCCR<sub>n</sub>B – Timer/Counter Control Register B







# แบบวิธีการทำงานวงจรนับ/จับเวลาหมายเลข 0, 2

บิตควบคุมการทำงาน			ชื่อแบบวิธีการทำงาน	ค่าสูงสุดที่นับได้
WGMn2	WGMn1	WGMn0		
0	0	0	แบบวิธีปกติ	0xFF
0	0	1	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟส ถูกต้อง	0xFF
0	1	0	แบบวิธี CTC	กำหนดใน OCRnA
0	1	1	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว	0xFF
1	0	0	ไม่มีการใช้งาน	-
1	0	1	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟส ถูกต้อง	กำหนดใน OCRnA
1	1	0	ไม่มีการใช้งาน	-
1	1	1	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว	กำหนดใน OCRnA







# การเลือกสัญญาณนาฬิกาของวงจรนับ/จับเวลา

CSn2	CSn1	CSn0	สัญญาณสำหรับป้อนวงจรนับ/จับเวลา
0	0	0	ไม่มีสัญญาณนาฬิกาถูกป้อนให้ ส่งผลให้วงจรนับ/จับเวลาหมายเลข $n$ หยุดทำงาน
0	0	1	$CLK_{I/O}$
0	1	0	$CLK_{I/O}/8$
0	1	1	$CLK_{I/O}/64$
1	0	0	$CLK_{I/O}/256$
1	0	1	$CLK_{I/O}/1024$
1	1	0	วงจรนับ/จับเวลารับสัญญาณนาฬิกาที่ขอบขาของสัญญาณที่ขา $Tn$
1	1	1	วงจรนับ/จับเวลารับสัญญาณนาฬิกาที่ขอบขาขึ้นของสัญญาณที่ขา $Tn$

# TIMSK $n$ – Timer/Counter $n$ Interrupt Mask Register

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	OCIE $n$ B	OCIE $n$ A	TOIE $n$

- ◆ OCIE $n$ A (Timer/Counter  $n$  Compare Match A Interrupt Enable) ใช้เพื่อเปิดทาง/ปิดทางการตอบรับการขัดจังหวะเมื่อ TCNT $n$  = OCR $n$ A
- ◆ OCIE $n$ B (Timer/Counter  $n$  Compare Match B Interrupt Enable) ใช้เพื่อเปิดทาง/ปิดทางการตอบรับการขัดจังหวะเมื่อ TCNT $n$  = OCR $n$ B
- ◆ บิต TOIE $n$  (Timer/Counter  $n$  Overflow Interrupt Enable) ใช้เพื่อเปิดทาง/ปิดทางการตอบรับการขัดจังหวะเมื่อเกิดการล้นของค่าใน TCNT $n$





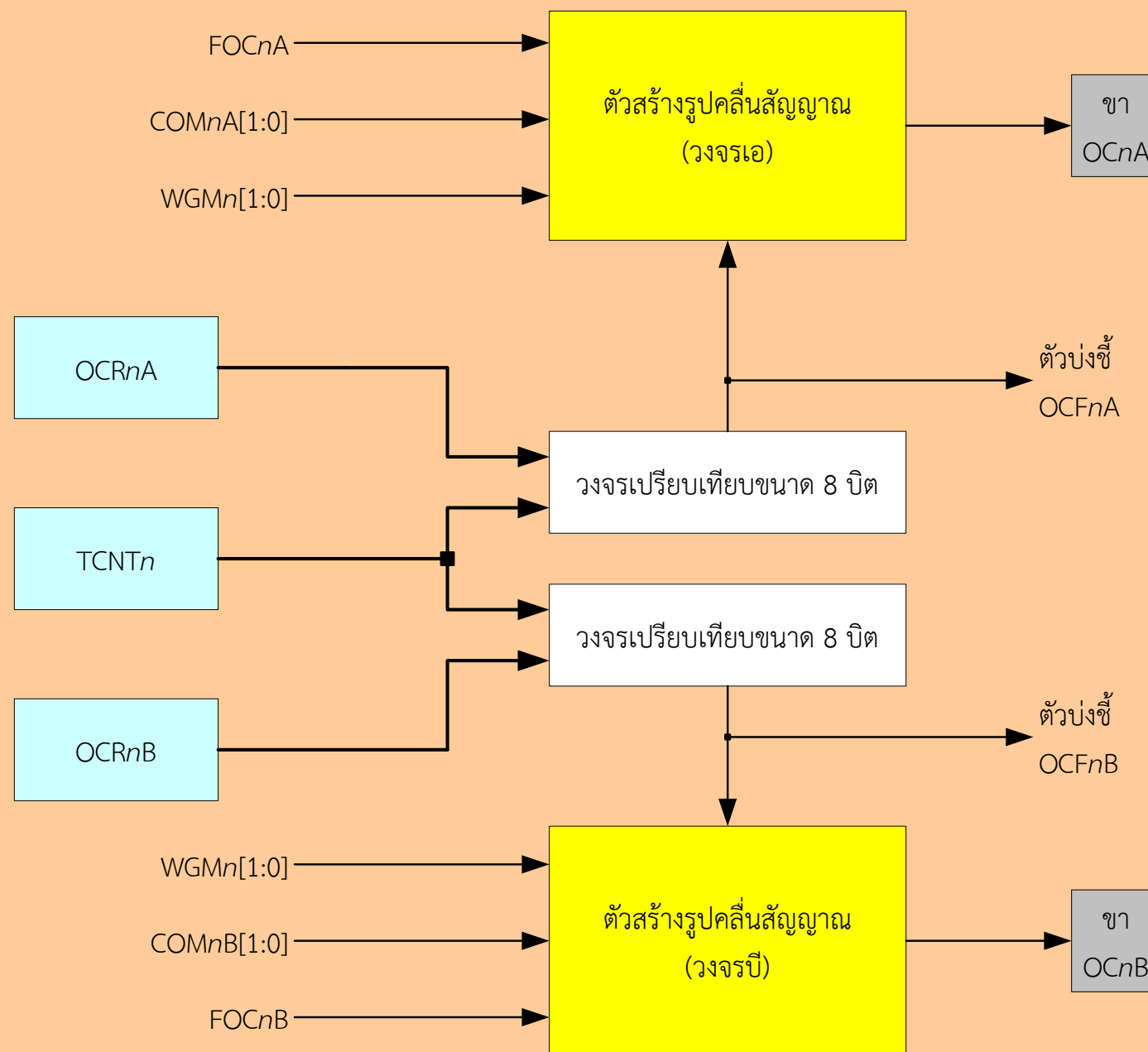
# TIFR<sub>n</sub> – Timer/Counter *n* Interrupt Flag Register

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	ไม่ใช้งาน	OCF <sub>n</sub> B	OCF <sub>n</sub> A	TOV <sub>n</sub>

- ◆ ตัวบ่งชี้ OCF<sub>n</sub>A ย่อมาจาก Timer/Counter *n* Output Compare Match A Flag ใช้บอกว่าค่าในเรจิสเตอร์ OCR<sub>n</sub>A มีค่าเท่ากับค่าในเรจิสเตอร์ TCNT<sub>n</sub>
- ◆ ตัวบ่งชี้ OCF<sub>n</sub>B ย่อมาจาก Timer/Counter *n* Output Compare Match B Flag ใช้บอกว่าค่าในเรจิสเตอร์ OCR<sub>n</sub>B มีค่าเท่ากับค่าในเรจิสเตอร์ TCNT<sub>n</sub>
- ◆ ตัวบ่งชี้ TOV<sub>n</sub> ย่อมาจาก Timer/Counter *n* Overflow Flag ใช้ในการบอกว่าค่าในเรจิสเตอร์ TCNT<sub>n</sub> เกิดการล้นและมีการวกกลับจากค่า 0xFF มาเป็นค่า 0x00



# ตัวสร้างรูปคลื่นสัญญาณ



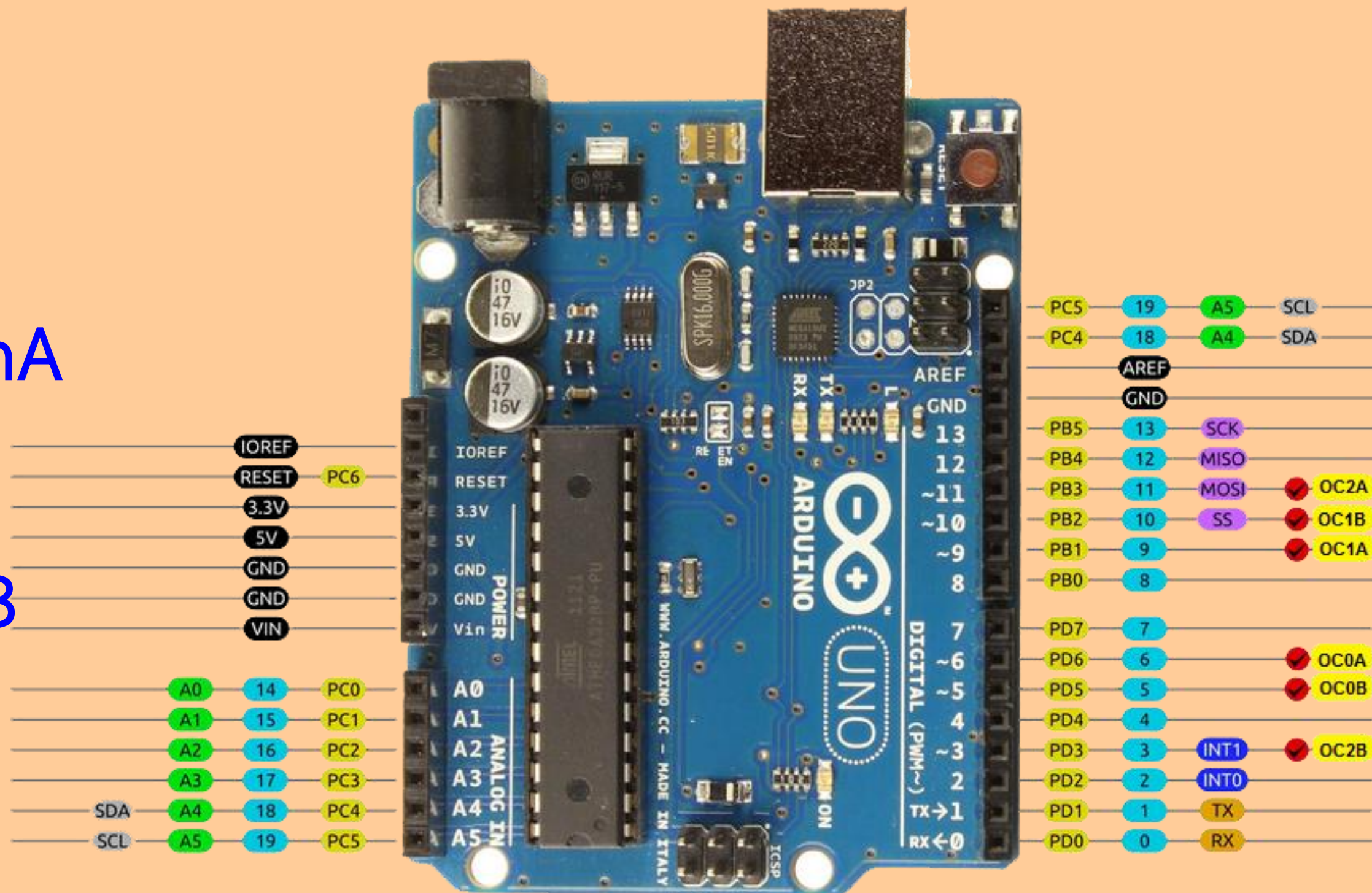




ขา OCnA

และ

OCnB



240-319

AVR

DIGITAL

ANALOG

POWER

SERIAL

SPI

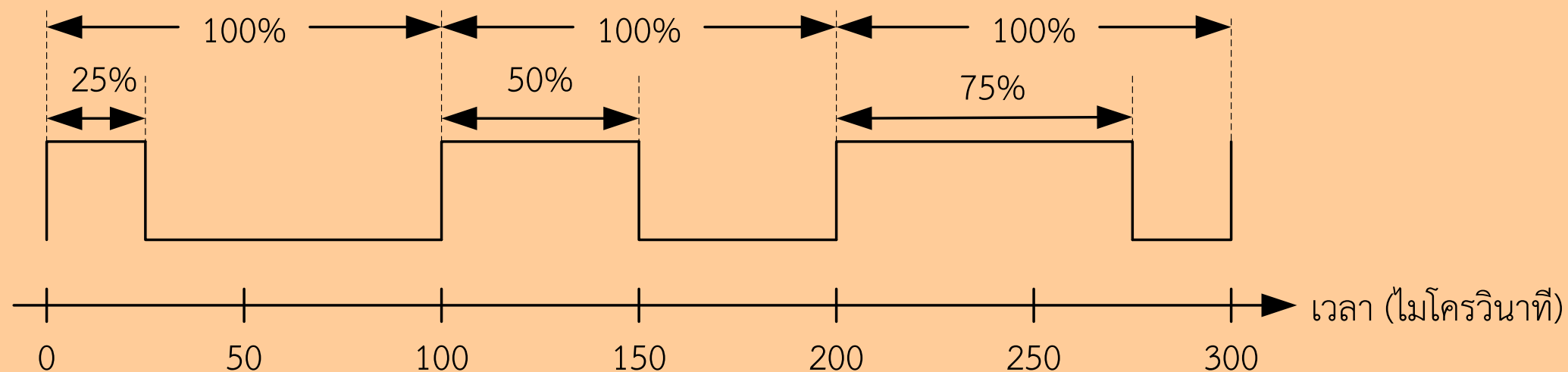
I2C

PWM

INTERRUPT

PWM

# การมอดูเลตความกว้างของพัลส์แบบเร็ว

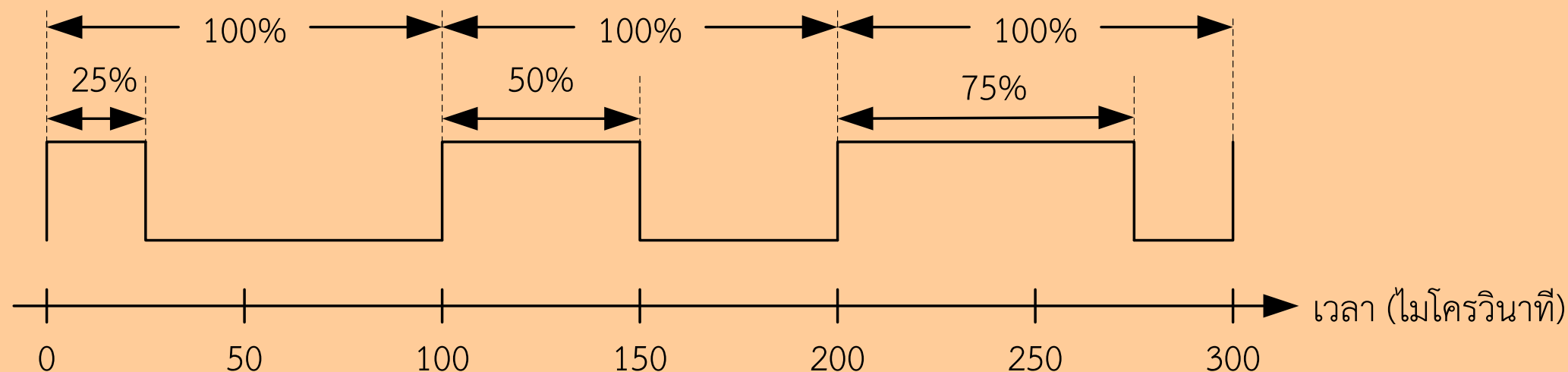


$$\text{Duty Cycle} = \frac{T_{\text{HIGH}}}{T_{\text{HIGH}} + T_{\text{LOW}}} * 100$$





# การมอดูเลตความกว้างของพัลส์แบบเร็ว



- ◆ วงจรนับ/จับเวลาขนาด 8 บิตสำหรับสนับสนุนการมอดูเลตความกว้างของพัลส์ให้  
เลือกใช้งานสองแบบ
  - ◆ การมอดูเลตความกว้างของพัลส์แบบเร็ว (Fast PWM)
  - ◆ การมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง (Phase Correct PWM)





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว ของวงจรรนับ/จับเวลาขนาด 8 บิต


บิตควบคุม		ความหมาย
COMnA1	COMnA0	
0	0	ไม่มีการใช้งานขา OCnA ทำให้ขาของพอร์ตที่ตรงกับขา OCnA ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	หากบิต WGMn2 ใน TCCRnB เท่ากับ 0 จะหมายถึงไม่มีการใช้งานขา OCnA หากบิต WGMn2 มีค่าเป็น 1 ให้ทำการกลับตรรกะของขา OCnA เป็นตรงข้ามทุกครั้งที่มีค่าใน OCnA มีค่าเท่ากับ TCNTn
1	0	ตั้งให้ขา OCnA เป็นตรรกะต่ำเมื่อค่าใน TCNTn และ OCRnA เท่ากัน และตั้งให้ OCnA กลับเป็นตรรกะสูงเมื่อ TCNTn มีค่าเท่ากับศูนย์
1	1	ตั้งให้ขา OCnA เป็นตรรกะสูงเมื่อค่าใน TCNTn และ OCRnA เท่ากัน และตั้งให้ OCnA กลับลงเป็นตรรกะต่ำเมื่อ TCNTn มีค่าเท่ากับศูนย์



# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว ของวงจรรนับ/จับเวลาขนาด 8 บิต

บิตควบคุม		ความหมาย
COM $n$ B1	COM $n$ B0	
0	0	ไม่มีการใช้งานขา OC $n$ B ทำให้ขาของพอร์ตที่ตรงกับขา OC $n$ B ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	ไม่มีการใช้งาน
1	0	ตั้งให้ขา OC $n$ B เป็นตรรกะต่ำเมื่อค่าใน TCNT $n$ และ OCR $n$ B เท่ากัน และตั้งให้ OC $n$ B กลับเป็นตรรกะสูงเมื่อ TCNT $n$ มีค่าเท่ากับศูนย์
1	1	ตั้งให้ขา OC $n$ B เป็นตรรกะสูงเมื่อค่าใน TCNT $n$ และ OCR $n$ B เท่ากัน และตั้งให้ OC $n$ B กลับลงเป็นตรรกะต่ำเมื่อ TCNT $n$ มีค่าเท่ากับศูนย์






# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว

## กรณีตั้ง $COMnx[1:0]$ มีค่าเป็น $10_2$ หรือ $11_2$

$$F_{OCnx\_Fast\_PWM} = \frac{CLK_{CPU}}{N * 256}$$

เมื่อ

$F_{OCnx\_Fast\_PWM}$	คือ ความถี่ของสัญญาณพัลส์ที่ออกทางขา $OCnx$ เมื่อวงจรรนับ/จับเวลา หมายเลข $n$ ทำงานในแบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว
$CLK_{CPU}$	คือ สัญญาณนาฬิกาที่ไม่โครคอนโทรลเลอร์ใช้ในการทำงาน
$N$	คือ ค่าคงที่สำหรับป้อนให้วงจรพรีสเกลเลอร์ใช้ในการหารความถี่ ซึ่งสามารถเลือกค่าได้ 5 ค่า ได้แก่ 1, 8, 64, 256 และ 1024



# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว

## กรณีตั้ง $COMnx[1:0]$ มีค่าเป็น $10_2$

$$Duty\_Cycle_{Fast\_PWM\_COMnx\_eq\_10} = \frac{OCRnx * 100}{256}$$

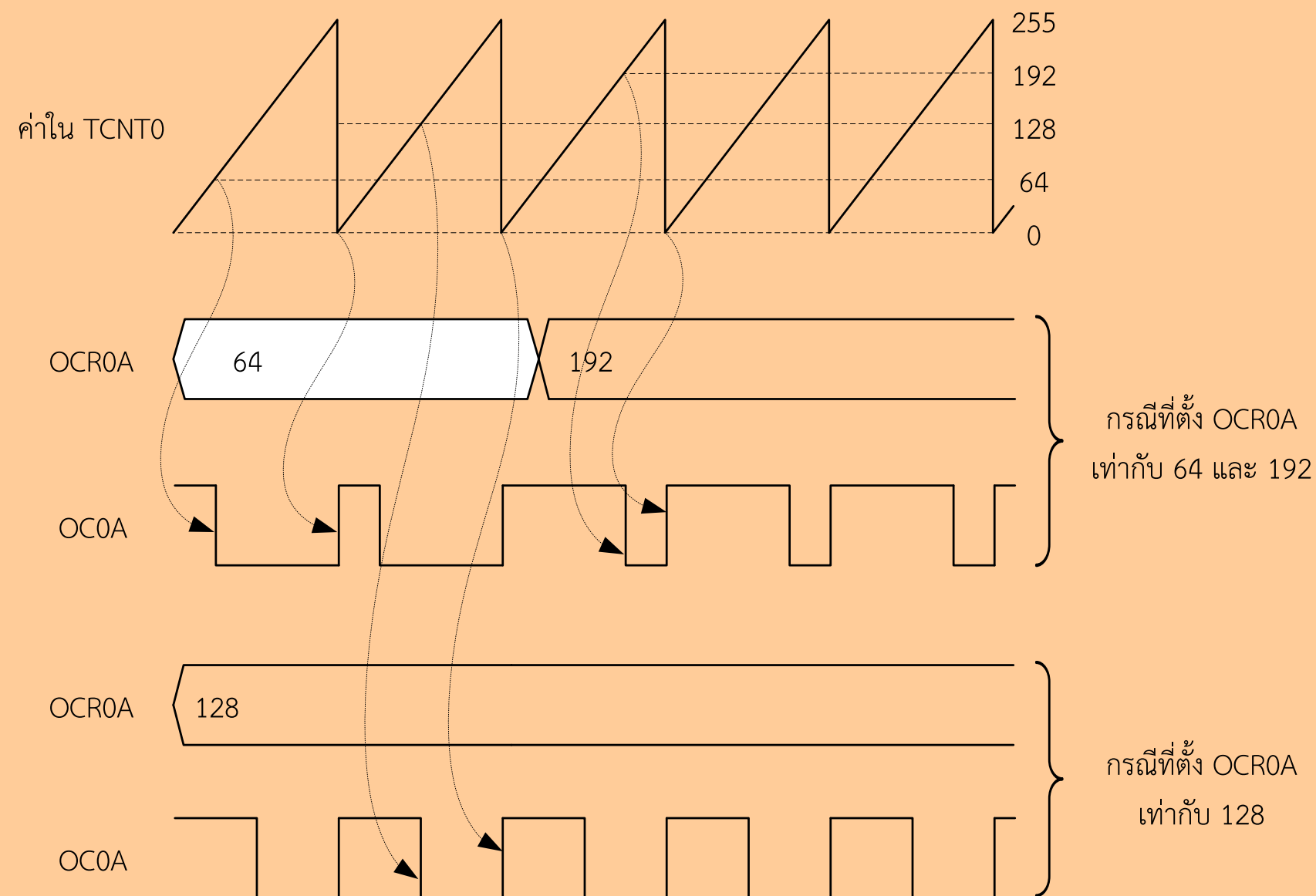
- ◆ TCNTn นับเพิ่มค่าขึ้น เริ่มจาก 0-255
- ◆ เมื่อค่าใน TCNTn เท่ากับค่าใน OCRnx จะส่งผลให้ขา OCnx มีค่าเป็น ตรรกะต่ำ และตั้งให้ OCnx มีค่าตรรกะสูงเมื่อค่าใน TCNTn เท่ากับ 0






# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว

## กรณีตั้ง $COMnx[1:0]$ มีค่าเป็น $10_2$







# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว

## กรณีตั้ง $COMnx[1:0]$ มีค่าเป็น $11_2$

$$\text{Duty\_Cycle}_{\text{Fast\_PWM\_COM}nx\_eq\_11} = \left( 1 - \frac{OCRnx}{256} \right) * 100$$

## ตัวอย่างที่ 4.6

- ◆ จงเขียนโปรแกรมบน ATmega 328P เพื่อส่งพัลส์ความถี่ 31,250 เฮิรตซ์ ซึ่งมีค่าวัฏจักรหน้าที่เท่ากับ 20 และ 70 เปอร์เซ็นต์ออกทางขา OC0A และ OC0B ตามลำดับ
- ◆ กำหนดให้ไมโครคอนโทรลเลอร์ทำงานที่ความถี่ 8 MHz





## ตัวอย่างที่ 4.6

$$F_{\text{OC0A\_Fast\_PWM}} = \frac{\text{CLK}_{\text{CPU}}}{N * 256}$$

เลือกใช้ค่า N เท่ากับ 1 จะได้

$$= \frac{8 * 10^6}{1 * 256}$$
$$= 31,250$$

◆ คำนวณหาค่า OCR0A เพื่อให้ได้วัฏจักรหน้าที่ 20 %

$$\text{OCR0A} = \frac{\text{Duty\_Cycle}_{\text{Fast\_PWM\_COMnx\_eq\_10}} * 256}{100}$$
$$= \frac{20}{100} * 256$$
$$= 51.2$$





## ตัวอย่างที่ 4.6

- ◆ กำหนดค่า OCR0A เพื่อให้ได้วัฏจักรหน้าที่ 70 %

$$\begin{aligned}\text{OCR0B} &= \frac{\text{Duty\_Cycle}_{\text{Fast\_PWM\_COMnx\_eq\_10}} * 256}{100} \\ &= \frac{70}{100} * 256 \\ &= 179.2\end{aligned}$$



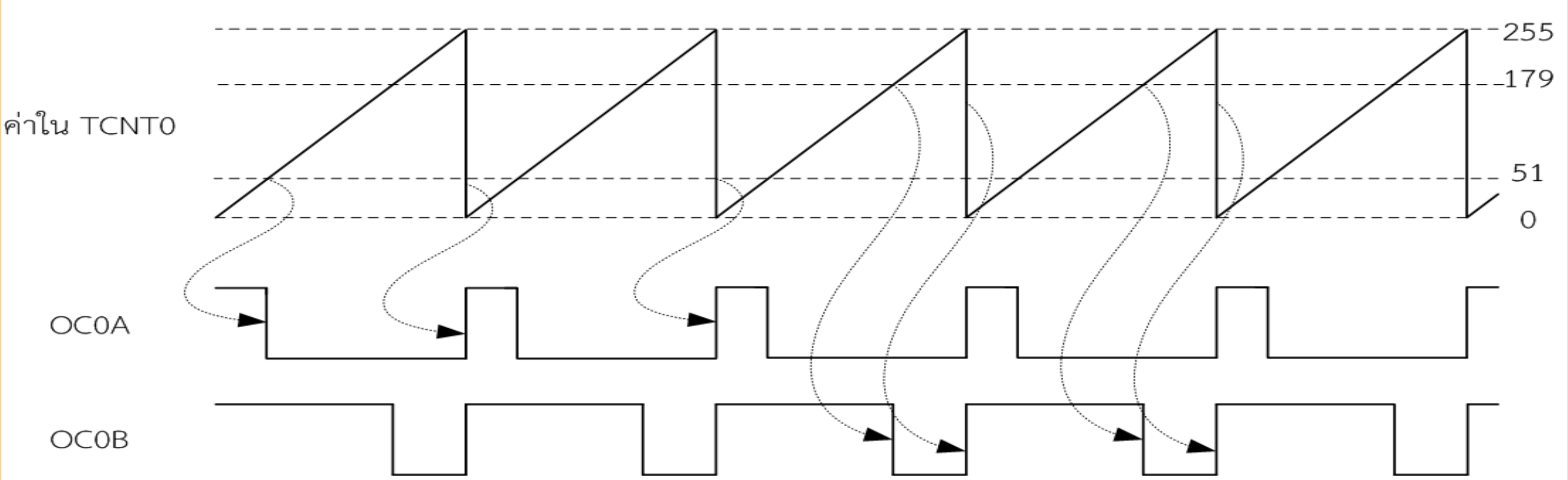
## ตัวอย่างที่ 4.6

```
1  #include <avr/io.h>           //ใช้คลังโปรแกรม io.h
2  int main(void)                //ฟังก์ชันหลักของโปรแกรม
3  {                             //เริ่มต้นขอบเขตของฟังก์ชันหลักของโปรแกรม
4      DDRD |= (1<<DDD6 | 1<<DDD5); //กำหนดให้พอร์ต D บิตที่ 5-6 เป็นเอาต์พุต
5      TCCR0A = (1<<WGM01 | 1<<WGM00); //ตั้งค่าบิตควบคุม WGM0[1:0] = 112
6      TCCR0A |= (1<<COM0B1 | 0<<COM0B0); //ตั้งค่าให้บิตควบคุม COM0A[1:0] = 102
7      TCCR0A |= (1<<COM0A1 | 0<<COM0B0); //ตั้งค่าให้บิตควบคุม COM0B[1:0] = 102
8      TCCRB = (0<<WGM02 | 1<<CS00); //CS0[2:0] = 0012 (หารความถี่ด้วยค่า N = 1 )
9      TCNT0 = 0;                //ตั้งค่าให้ TCNT0 มีค่าเริ่มต้นเป็นศูนย์
10     OCR0A = 51;                //ค่าจากการคำนวณเพื่อให้ได้วัฏจักรหน้าที่ 20 เปอร์เซ็นต์
11     OCR0B = 179;              //ค่าจากการคำนวณเพื่อให้ได้วัฏจักรหน้าที่ 70 เปอร์เซ็นต์
12     while(1)                  //วนซ้ำการทำงานไม่รู้จบ
13     {
14
15         //ไม่มีการทำงานใด ๆ ในส่วนนี้
16     }
16 }
```



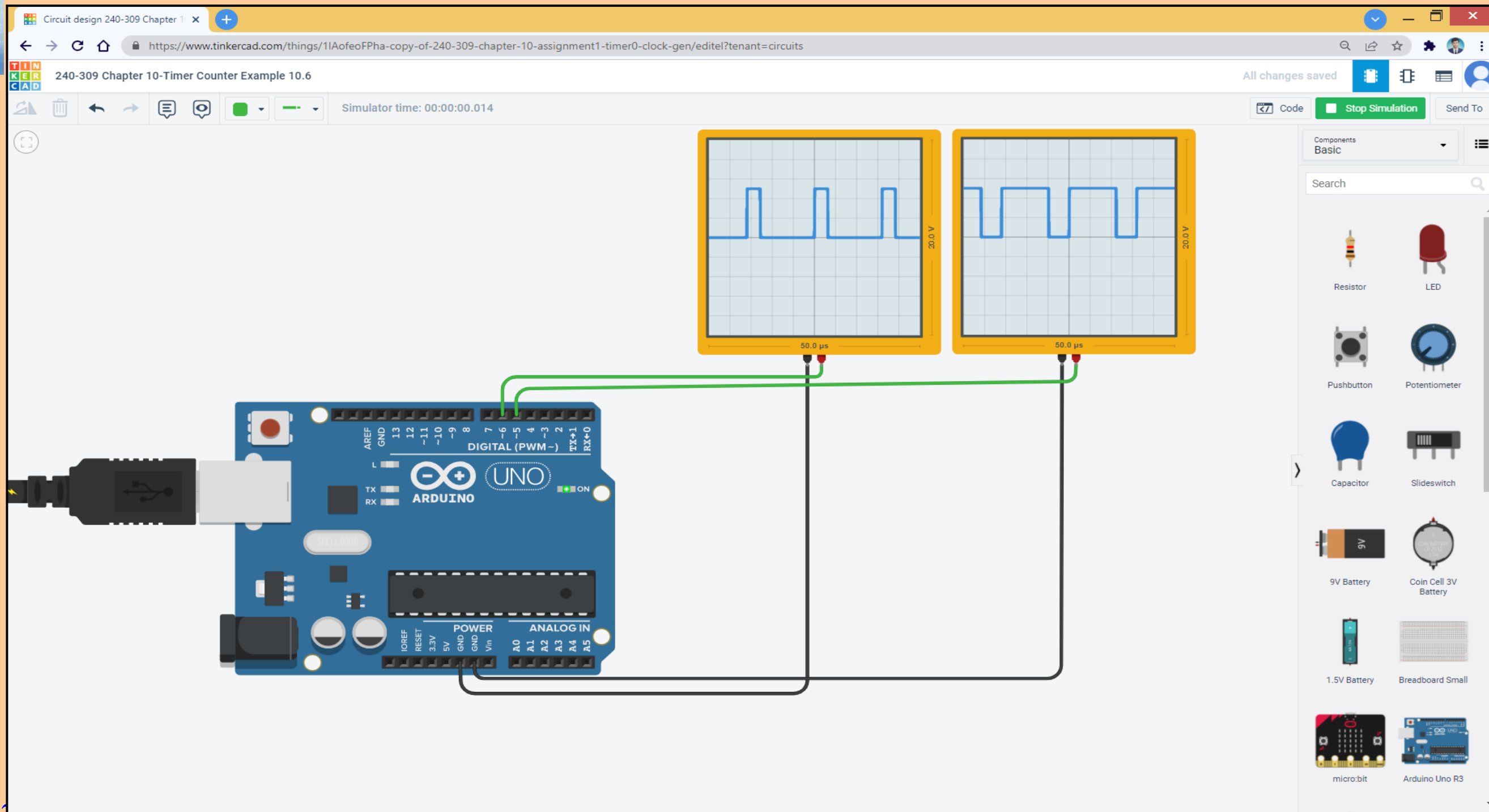


# ตัวอย่างที่ 4.6



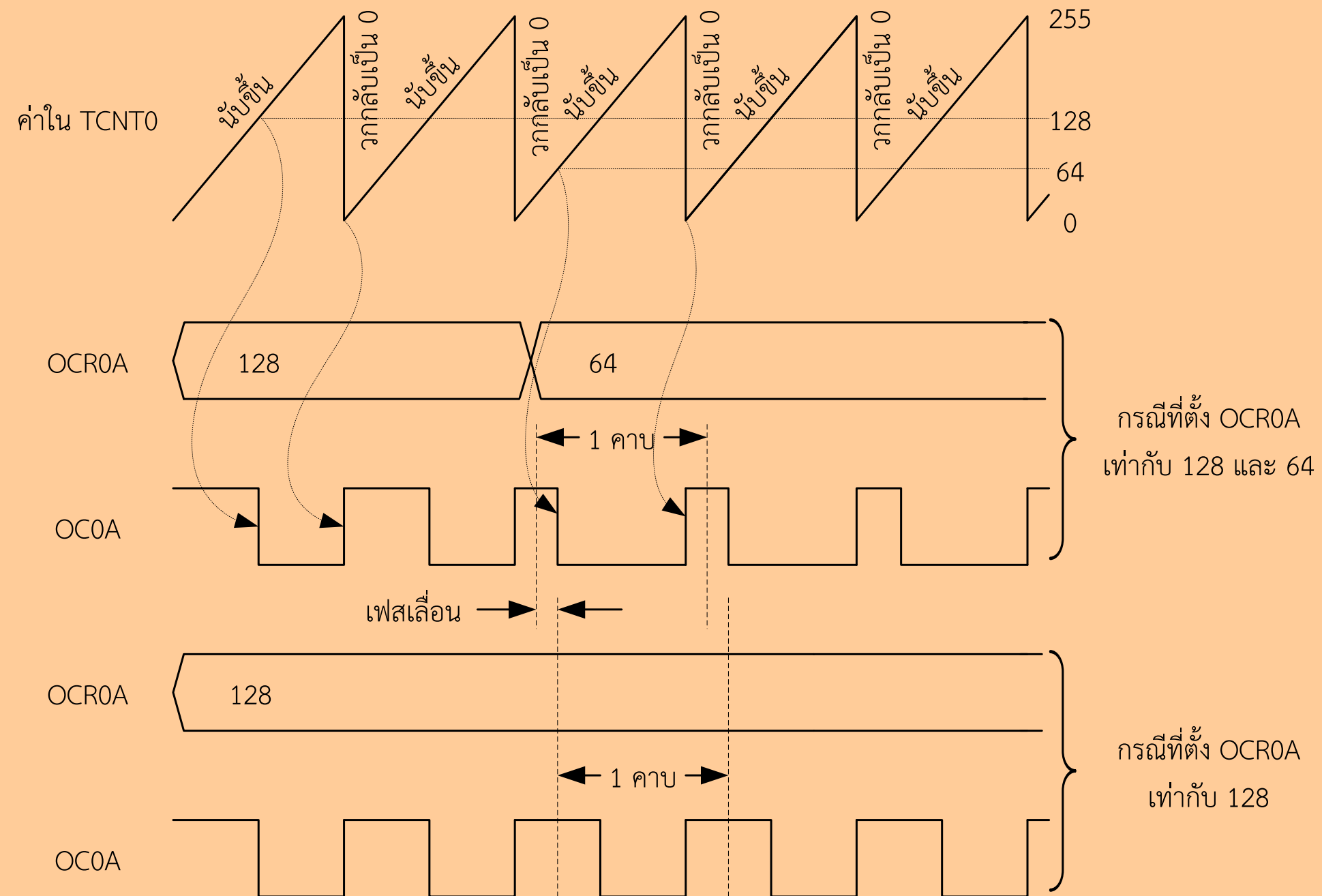


# ตัวอย่างที่ 4.6





# การเลื่อนเฟสของการมอดูเลตความกว้างของพัลส์แบบเร็ว





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

$$F_{\text{OCnx\_Phase\_Correct\_PWM}} = \frac{\text{CLK}_{\text{CPU}}}{N * 510}$$

- ◆ ไม่มีการเลื่อนของเฟสเมื่อปรับค่าวัฏจักรหน้าที่
- ◆ TCNT $n$  นับขึ้นและนับลง
  - ◆ นับขึ้นจาก 0-255
  - ◆ นับลงจาก 255-0
  - ◆ การนับขึ้นและลง 1 รอบจะเท่ากับ  $255 * 2$  หรือ 510 ทั้งนี้เนื่องจากการนับในแต่ละรอบ คือ 0, 1, 2, ..., 253, 254, 255, 254, 253, ..., 2, 1 ซึ่งมีการนับแค่ 510 ครั้งในแต่ละรอบเท่านั้น





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

บิตควบคุม		ความหมาย
COMnA1	COMnA0	
0	0	ไม่มีการใช้งานขา OCnA ทำให้พอร์ตบิตที่ต่อกับขา OCnA ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	หากบิต WGMn2 ใน TCCRnB เท่ากับ 0 จะหมายถึงไม่มีการใช้งานขา OCnA หากบิต WGMn2 มีค่าเป็น 1 ให้ทำการกลับตรรกะของขา OCnA เป็นตรงข้ามทุกครั้งที่มีค่าใน OCRnA มีค่าเท่ากับ TCNTn
1	0	ตั้งให้ขา OCnA เป็นตรรกะต่ำเมื่อนับขึ้นและค่าใน TCNTn และ OCRnA เท่ากัน และตั้งให้ OCnA กลับเป็นตรรกะสูงเมื่อนับลงและค่าใน TCNTn และ OCRnA เท่ากัน
1	1	ตั้งให้ขา OCnA เป็นตรรกะสูงเมื่อนับขึ้นและค่าใน TCNTn และ OCRnA เท่ากัน และตั้งให้ OCnA กลับเป็นตรรกะต่ำเมื่อนับลงและค่าใน TCNTn และ OCRnA เท่ากัน





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

บิตควบคุม		ความหมาย
COMnB1	COMnB0	
0	0	ไม่มีการใช้งานขา OCnB ทำให้พอร์ตบิตที่ต่อกับขา OCnB ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	ไม่มีการใช้งาน
1	0	ตั้งให้ขา OCnB เป็นตรรกะต่ำเมื่อนับขึ้นและค่าใน TCNTn และ OCRnB เท่ากัน และตั้งให้ OCnB กลับเป็นตรรกะสูงเมื่อนับลงและค่าใน TCNTn และ OCRnB เท่ากัน
1	1	ตั้งให้ขา OCnB เป็นตรรกะสูงเมื่อนับขึ้นและค่าใน TCNTn และ OCRnB เท่ากัน และตั้งให้ OCnB กลับเป็นตรรกะต่ำเมื่อนับลงและค่าใน TCNTn และ OCRnB เท่ากัน





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

กรณีตั้ง  $COMnx[1:0]$  มีค่าเป็น  $10_2$

$$Duty\_Cycle_{PhaseCorrect\_PWM\_COMnx\_eq\_10} = \frac{OCRnx * 100}{255}$$

กรณีตั้ง  $COMnx[1:0]$  มีค่าเป็น  $11_2$

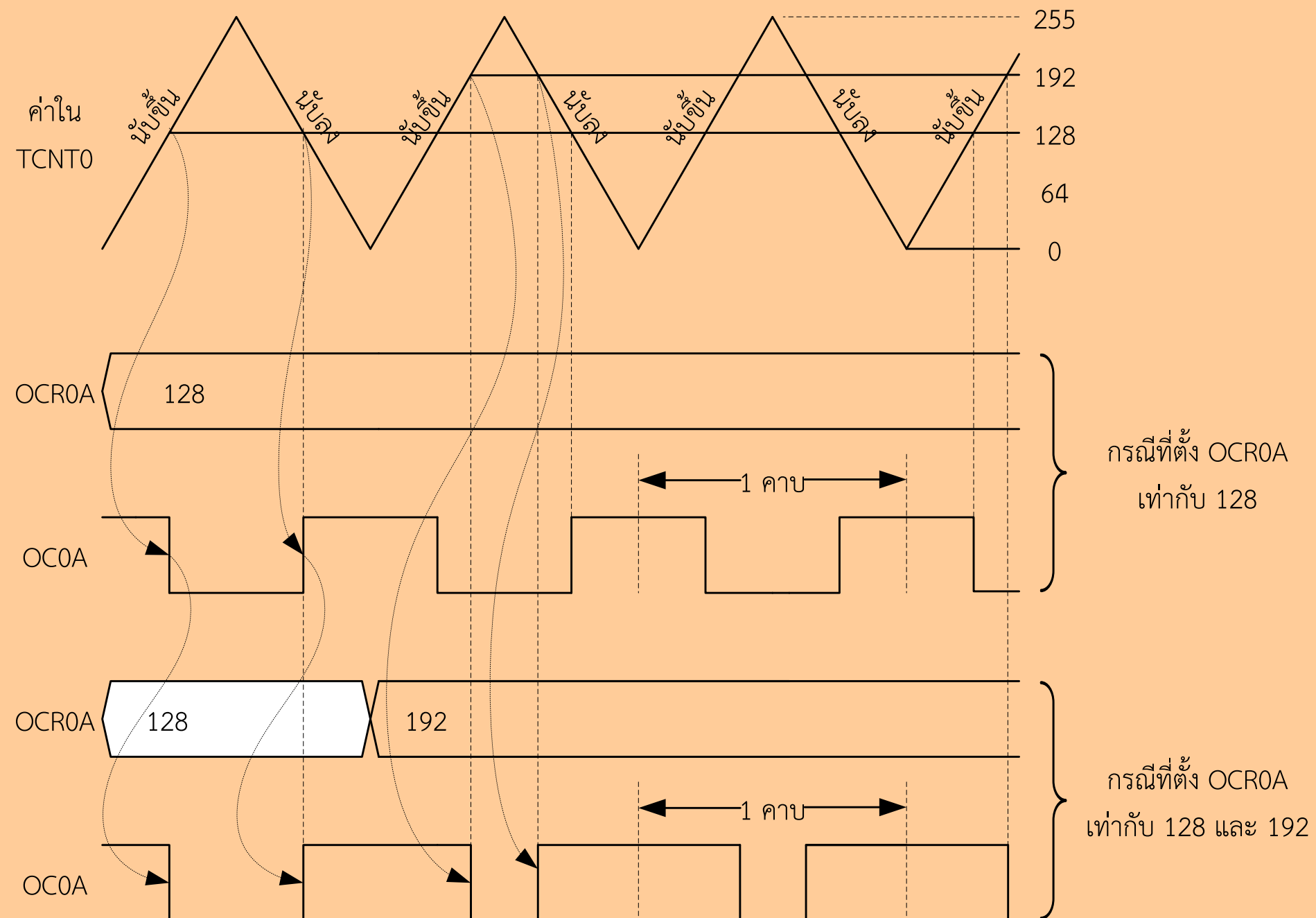
$$Duty\_Cycle_{PhaseCorrect\_PWM\_COMnx\_eq\_11} = \left( 1 - \frac{OCRnx}{255} \right) * 100$$





# แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

## กรณีตั้ง $COMnx[1:0]$ มีค่าเป็น $10_2$

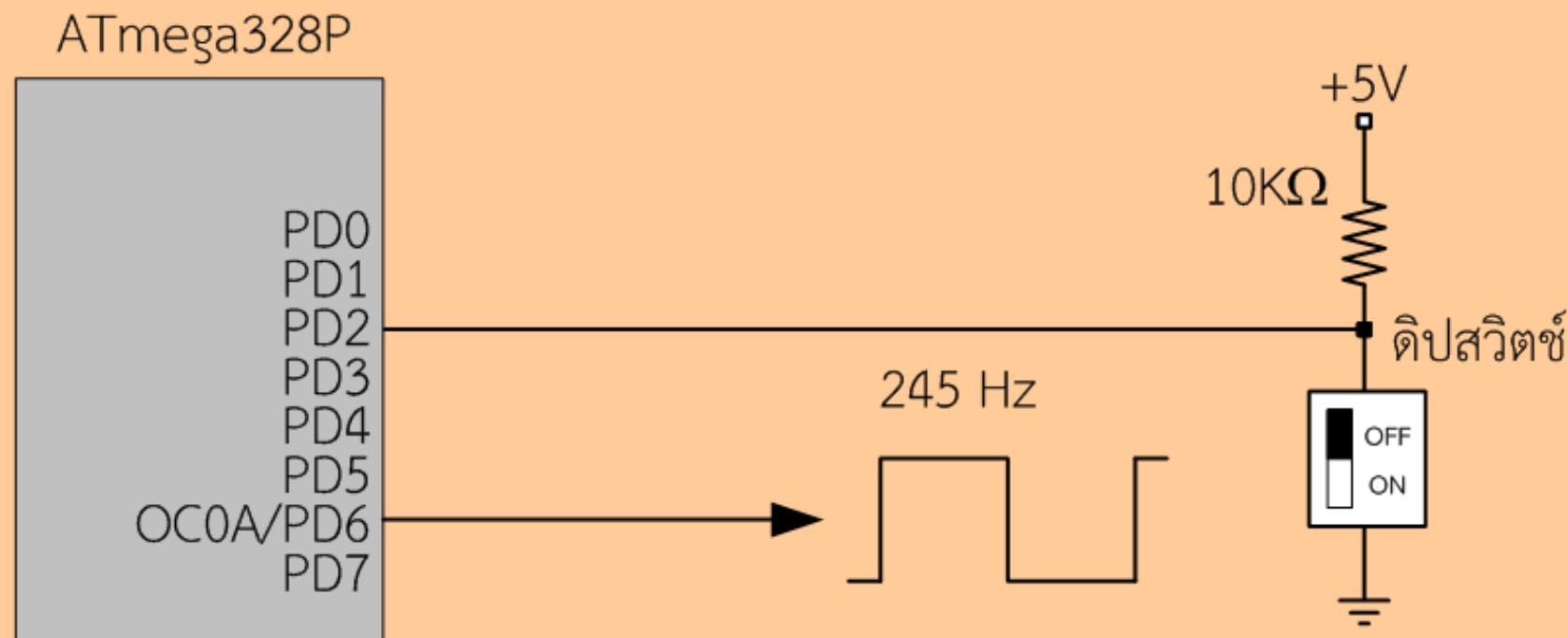


♦ จะเห็นว่าไม่มีการเลื่อนเฟสแม้จะเปลี่ยนความกว้างของพัลส์





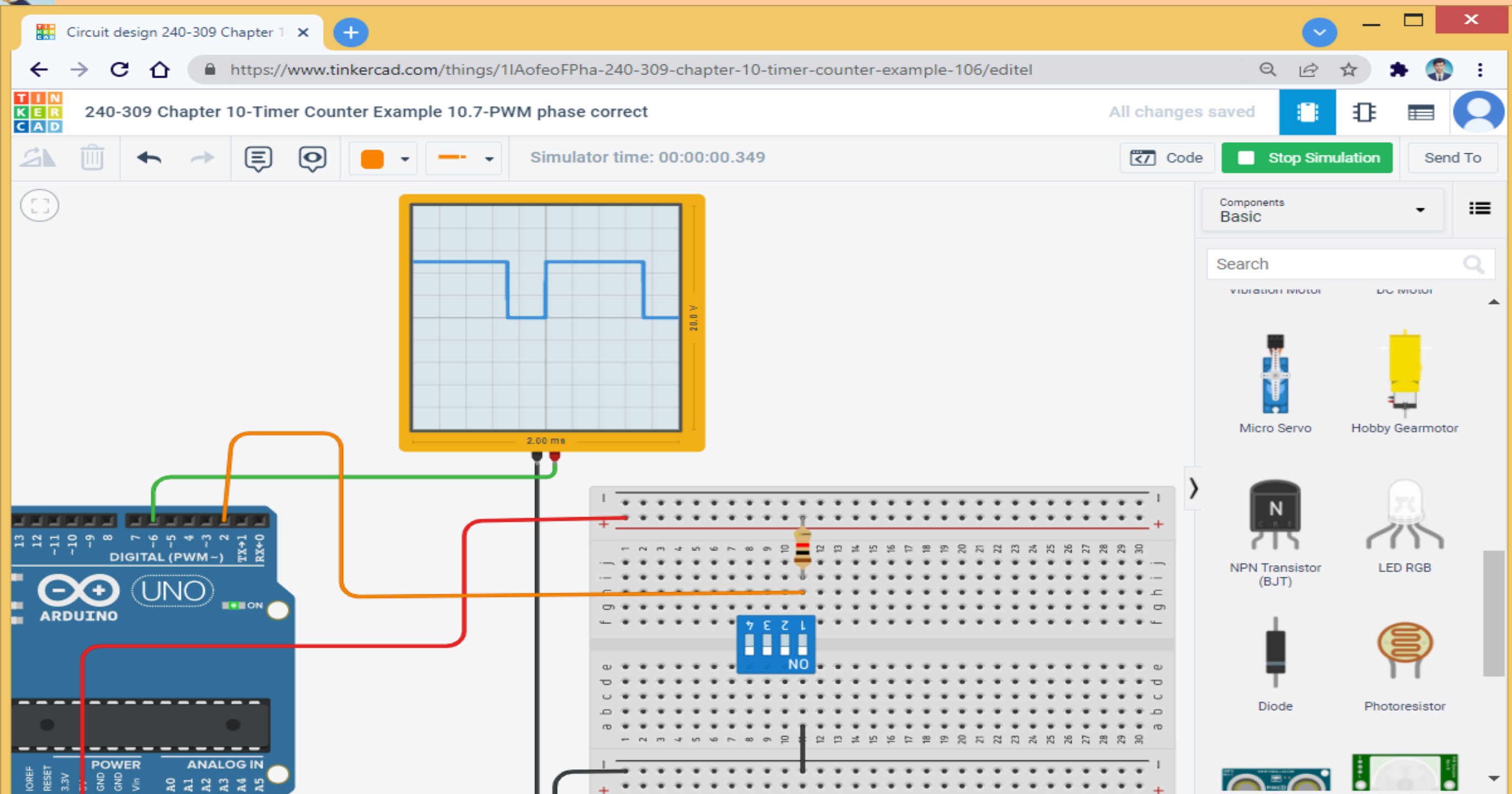
## ตัวอย่างที่ 4.7



- ◆ จงเขียนโปรแกรมบน ATmega 328P เพื่อส่งพัลส์ความถี่ 245 เฮิรตซ์ ออกทางขา OC0A
- ◆ โดยสามารถเลือกค่าวัฏจักรหน้าที่ได้จากการตั้งค่าที่ดีปสวิตช์
  - ◆ ดีปสวิตช์มีค่าตรรกะสูงให้สัญญาณขาออกมีค่าวัฏจักรหน้าที่เท่ากับ 25 %
  - ◆ ดีปสวิตช์มีค่าตรรกะต่ำให้สัญญาณขาออกมีค่าวัฏจักรหน้าที่เท่ากับ 72 %
- ◆ ตัวประมวลผลทำงานที่ความถี่ 1 เมกะเฮิรตซ์



# ตัวอย่างที่ 4.7





## ตัวอย่างที่ 4.7

- ◆ กำหนดให้วงจรนับ/จับเวลาหมายเลข 0 ทำงานในแบบวิธีมอดูเลต  
ความกว้างของพัลส์แบบเฟสถูกต้อง

$$\begin{aligned} F_{\text{OC0A\_Phase\_Correct\_PWM}} &= \frac{\text{CLK}_{\text{CPU}}}{N * 510} \\ \text{เลือกใช้ค่า } N \text{ เท่ากับ } 8 \text{ จะได้} &= \frac{1 * 10^6}{8 * 510} \\ &= 245.098 \end{aligned}$$





## ตัวอย่างที่ 4.7

$$\begin{aligned}\text{OCR0A} &= \frac{\text{Duty\_Cycle}_{\text{PhaseCorrect\_PWM\_COM0A\_eq\_10}} * 255}{100} \\ &= 25 * 255 / 100 \\ &= 63.75\end{aligned}$$

$$\begin{aligned}\text{OCR0A} &= \frac{\text{Duty\_Cycle}_{\text{PhaseCorrect\_PWM\_COM0A\_eq\_10}} * 255}{100} \\ &= 72 * 255 / 100 \\ &= 183.6\end{aligned}$$



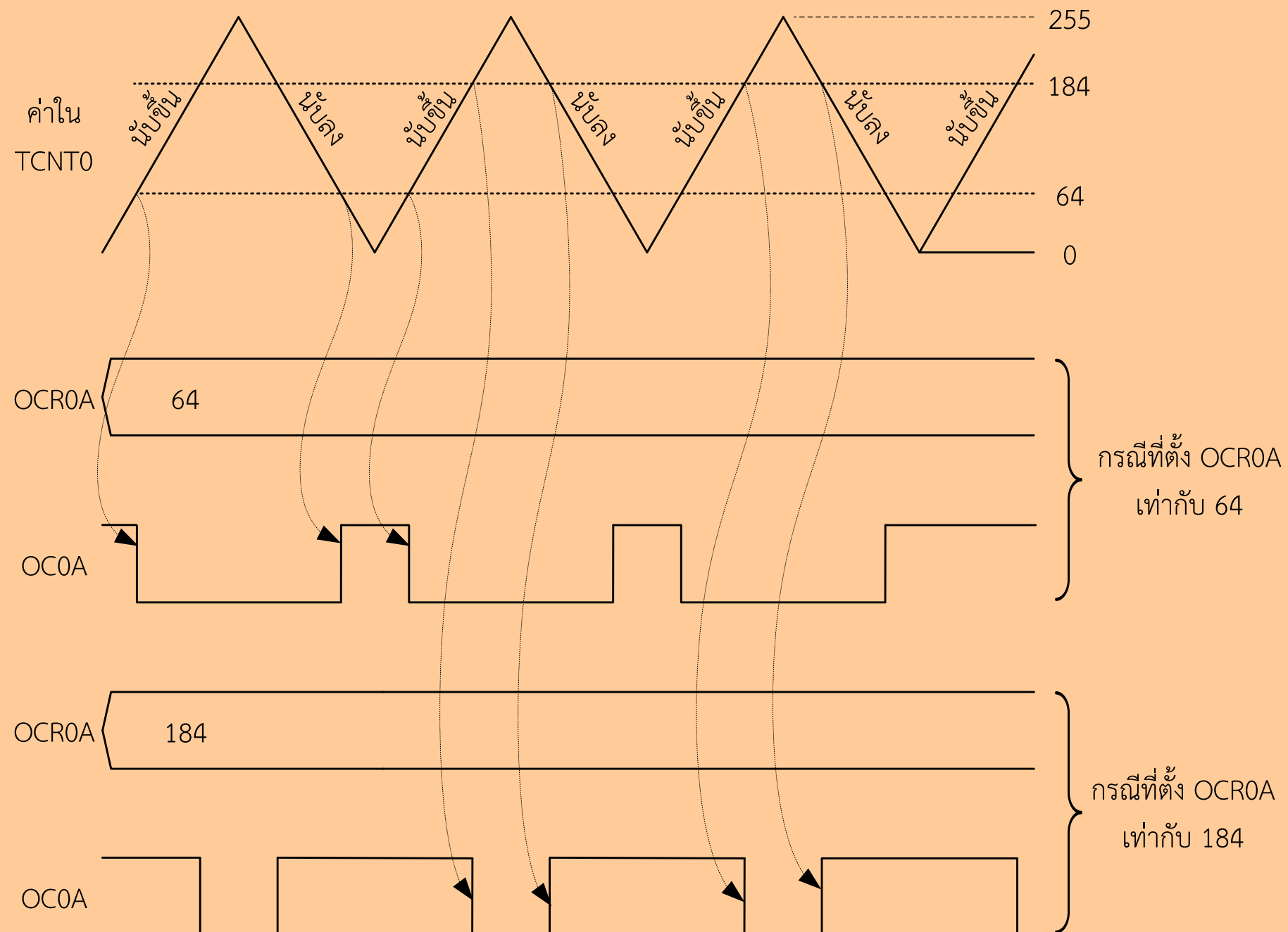


# ตัวอย่างที่ 4.7

1    #include <avr/io.h> 2    int main(void) 3    { 4        uint8_t tmp; 5        DDRD = (1<<DDD6   0<<DDD2); 6        TCCR0A = (0<<WGM01   1<<WGM00); 7        TCCR0A  = (1<<COM0A1   0<<COM0A0); 8        TCCR0B = (0<<WGM02   1<<CS01); 9        TCNT0 = 0; 10      while(1) 11      { 12          tmp = PIND; 13          tmp &= 0x04; 14          if (tmp) 15              OCR0A = 64; 16          else 17              OCR0A = 184; 18      } 19 }	//ใช้คลังโปรแกรม io.h //ฟังก์ชันหลักของโปรแกรม //เริ่มต้นขอบเขตของฟังก์ชันหลักของโปรแกรม //ประกาศตัวแปรชั่วคราวสำหรับอ่านค่าจากสวิตช์ //ตั้งทิศทางพอร์ต D ให้บิต 6 ส่งออก บิต 2 รับเข้า //ตั้งค่าบิตควบคุม WGM0[2:0] = 001 <sub>2</sub> //ตั้งค่าให้บิตควบคุม COM0A[1:0] = 10 <sub>2</sub> //CS0[2:0] = 010 <sub>2</sub> (หารความถี่ด้วยค่า N = 8 ) //ค่าเริ่มต้นในเรจิสเตอร์ TCNT0 เท่ากับศูนย์ //วนซ้ำการทำงานไม่รู้จบ //เริ่มต้นขอบเขตของการวนซ้ำไม่รู้จบ //อ่านค่าจากพอร์ต D //พรางบิตอื่นทิ้งให้เหลือเพียงบิตที่ต่อกับดิปสวิตช์ //หากค่าจากสวิตช์เป็นตรรกะสูง //ตั้งค่าเพื่อให้ได้วัฏจักรหน้าที่ 25 เปอร์เซ็นต์ //แต่หากค่าจากสวิตช์เป็นตรรกะต่ำ //ให้ตั้งค่าเพื่อให้ได้วัฏจักรหน้าที่ 75 เปอร์เซ็นต์ //สิ้นสุดขอบเขตของการวนซ้ำไม่รู้จบ //สิ้นสุดขอบเขตของฟังก์ชันหลักของโปรแกรม
---	--



# ตัวอย่างที่ 4.7





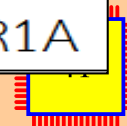
# เรจิสเตอร์ควบคุมวงจรรนับ/จับเวลาหมายเลข 1

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0	
COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	เรจิสเตอร์ TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	เรจิสเตอร์ TCCR1B
FOC1A	FOC1B	-	-	-	-	-	-	เรจิสเตอร์ TCCR1C
-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1	เรจิสเตอร์ TIMSK1
-	-	ICIF1	-	-	OCF1B	OCF1A	TOV1	เรจิสเตอร์ TIFR1



# แบบวิธีการทำงานของวงจรรนับ/จับเวลาหมายเลข 1

บิตควบคุมการทำงาน WGM1[3:0]	ชื่อแบบวิธีการทำงาน	ค่าสูงสุดที่นับได้
0000 <sub>2</sub>	แบบวิธีปกติ	0xFFFF
0001 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้องชนิด 8 บิต	0x00FF
0010 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้องชนิด 9 บิต	0x01FF
0011 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้องชนิด 10 บิต	0x03FF
0100 <sub>2</sub>	แบบวิธี CTC	กำหนดใน OCR1A
0101 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็วชนิด 8 บิต	0x00FF
0110 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็วชนิด 9 บิต	0x01FF
0111 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็วชนิด 10 บิต	0x03FF
1000 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสและความถี่ถูกต้อง	กำหนดใน ICR1
1001 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสและความถี่ถูกต้อง	กำหนดใน OCR1A
1010 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง	กำหนดใน ICR1
1011 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง	กำหนดใน OCR1A
1100 <sub>2</sub>	แบบวิธี CTC	กำหนดใน ICR1
1101 <sub>2</sub>	ไม่มีการใช้งาน	-
1110 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว	กำหนดใน ICR1
1111 <sub>2</sub>	แบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว	กำหนดใน OCR1A







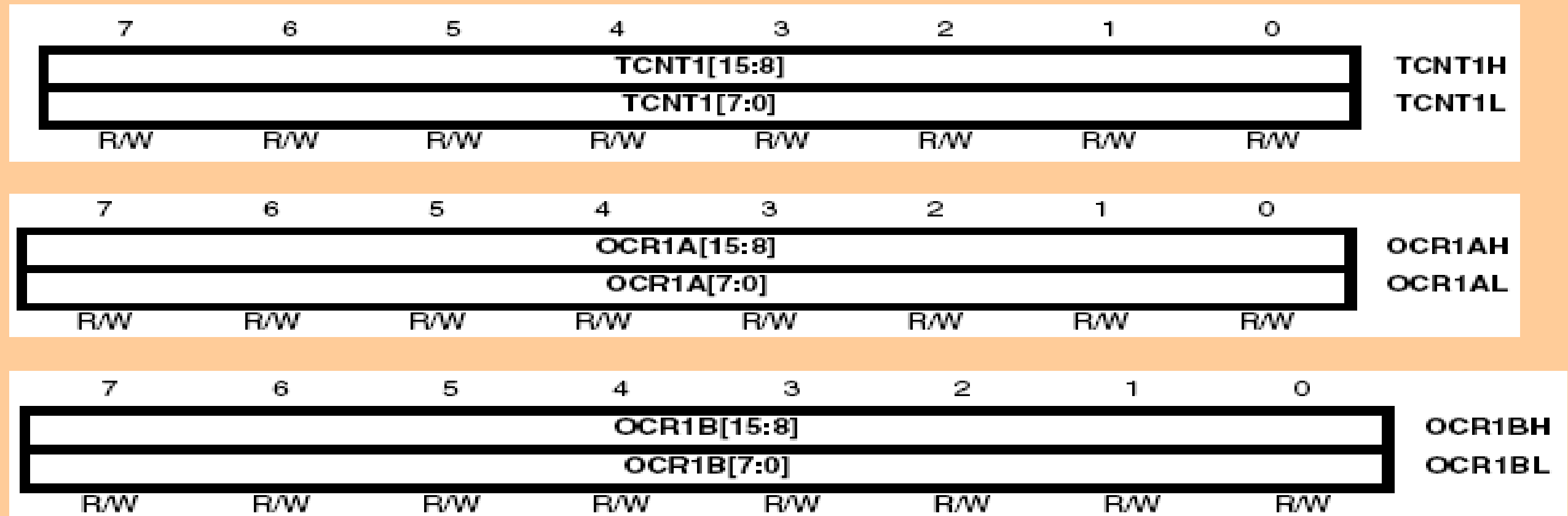
# CS12:CS10

CS12	CS11	CS10	ความหมาย
0	0	0	ไม่มีสัญญาณนาฬิกาถูกป้อนให้ ส่งผลให้วงจรนับ/จับเวลาหมายเลขหนึ่งหยุดทำงาน
0	0	1	วงจรนับ/จับเวลารับสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์โดยตรง ไม่มีการหารความถี่โดยวงจรพรีสเกลเลอร์
0	1	0	วงจรพรีสเกลเลอร์หารความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ด้วยค่า 8 แล้วส่งค่าความถี่ที่ได้ให้เป็นสัญญาณนาฬิกาของวงจรนับ/จับเวลาหมายเลขหนึ่ง
0	1	1	วงจรพรีสเกลเลอร์หารความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ด้วยค่า 64 แล้วส่งค่าความถี่ที่ได้ให้เป็นสัญญาณนาฬิกาของวงจรนับ/จับเวลาหมายเลขหนึ่ง
1	0	0	วงจรพรีสเกลเลอร์หารความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ด้วยค่า 256 แล้วส่งค่าความถี่ที่ได้ให้เป็นสัญญาณนาฬิกาของวงจรนับ/จับเวลาหมายเลขหนึ่ง
1	0	1	วงจรพรีสเกลเลอร์หารความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ด้วยค่า 1024 แล้วส่งค่าความถี่ที่ได้ให้เป็นสัญญาณนาฬิกาของวงจรนับ/จับเวลาหมายเลขหนึ่ง
1	1	0	วงจรนับ/จับเวลารับสัญญาณนาฬิกาที่ขอบขาของขา T1
1	1	1	วงจรนับ/จับเวลารับสัญญาณนาฬิกาที่ขอบขาขึ้นของขา T1





# เรจิสเตอร์นับของ Timer/Counter1



◆ ก่อนจะเข้าถึงเรจิสเตอร์ดังกล่าวจะต้องปิดทางการขัดจังหวะเสียก่อน





# วงจรสร้างสัญญาณรูปคลื่นของวงจรรนับ/จับเวลาหมายเลข 1

## การทำงานแบบวิธี Fast PWM

บิตควบคุม		ความหมาย
COM1x1	COM1x0	
0	0	ไม่มีการใช้งานขา OC1x ทำให้ขาของพอร์ตที่ตรงกับขา OC1x ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	หากบิต WGM1[3:0] มีค่าเท่ากับ $1110_2$ หรือ $1111_2$ ให้ทำการกลับตรรกะของขา OC1A เป็นตรงข้ามทุกครั้งที่ค่าใน OCR1A มีค่าเท่ากับ TCNT1 ส่วนขา OC1B ไม่มีการใช้งาน แต่หากบิตควบคุม WGM1[3:0] มีค่าอื่นจะหมายถึงไม่มีการใช้งานทั้งขา OC1A และ OC1B
1	0	ตั้งให้ขา OC1x เป็นตรรกะต่ำเมื่อค่าใน TCNT1 และ OCR1x เท่ากัน และตั้งให้ OC1x กลับเป็นตรรกะสูงเมื่อ TCNT1 มีค่าเท่ากับศูนย์
1	1	ตั้งให้ขา OC1x เป็นตรรกะสูงเมื่อค่าใน TCNTn และ OCR1x เท่ากัน และตั้งให้ OC1x กลับลงเป็นตรรกะต่ำเมื่อ TCNT1 มีค่าเท่ากับศูนย์



# วงจรสร้างสัญญาณรูปคลื่นของวงจรรนับ/จับเวลาหมายเลข 1

## กรณีทำงานแบบวิธี PWM เฟสถูกต้อง

บิตควบคุม		ความหมาย
COMnB1	COMnB0	
0	0	ไม่มีการใช้งานขา OC1x ทำให้ขาของพอร์ตที่ตรงกับขา OC1x ทำหน้าที่รับส่งข้อมูลตามปกติ
0	1	หากบิต WGM1[3:0] มีค่าเท่ากับ $1110_2$ หรือ $1111_2$ ให้ทำการกลับตรรกะของขา OC1A เป็นตรงข้ามทุกครั้งที่มีค่าใน OCR1A มีค่าเท่ากับ TCNT1 ส่วนขา OC1B ไม่มีการใช้งาน แต่หากบิตควบคุม WGM1[3:0] มีค่าอื่นจะหมายถึงไม่มีการใช้งานทั้งขา OC1A และ OC1B
1	0	ตั้งให้ขา OC1x เป็นตรรกะต่ำเมื่อนับขึ้นและค่าใน TCNT1 และ OCR1x เท่ากัน และตั้งให้ OC1x กลับเป็นตรรกะสูงเมื่อนับลงและค่าใน TCNT1 และ OCR1x เท่ากัน
1	1	ตั้งให้ขา OC1x เป็นตรรกะสูงเมื่อนับขึ้นและค่าใน TCNT1 และ OCR1x เท่ากัน และตั้งให้ OC1x กลับเป็นตรรกะต่ำเมื่อนับลงและค่าใน TCNT1 และ OCR1x เท่ากัน





# Timer/Counter1 เมื่อทำงานในแบบวิธี Fast PWM

$$F_{OC1x\_Fast\_PWM} = \frac{CLK_{CPU}}{N * (1 + MAX_{Fast\_PWM\_TCNT1})}$$

เมื่อ $F_{OC1x\_Fast\_PWM}$	คือ ความถี่ของสัญญาณพัลส์ที่ออกทางขา OC1x เมื่อวงจรนับ/จับเวลา หมายเลขหนึ่งทำงานในแบบวิธีการมอดูเลตความกว้างของพัลส์แบบเร็ว
$CLK_{CPU}$	คือ สัญญาณนาฬิกาที่ไม่โครคอนโทรลเลอร์ ใช้ในการทำงาน
N	คือ ค่าคงที่สำหรับป้อนให้วงจรพรีสเกลเลอร์ใช้ในการหารความถี่ ซึ่งสามารถเลือกค่าได้ 5 ค่า ได้แก่ 1, 8, 64, 256 และ 1024




# ค่าสูงสุดที่ TCNT1 นับได้ เมื่อทำงานในแบบวิธี Fast PWM

บิตควบคุม WGM1[3:0]	$MAX_{Fast\_PWM\_TCNT1}$	หมายเหตุ
$0101_2$	255	-
$0110_2$	511	-
$0111_2$	1,023	-
$1110_2$	กำหนดในเรจิสเตอร์ ICR1	ค่าสูงสุดที่ตั้งได้ คือ 65,535
$1111_2$	กำหนดในเรจิสเตอร์ OCR1A	ค่าสูงสุดที่ตั้งได้ คือ 65,535








# Timer/Counter 1 แบบวิธี Fast PWM

กรณีตั้ง  $COMnx[1:0]$  มีค่าเป็น  $10_2$

$$\text{Duty\_Cycle}_{\text{Fast\_PWM\_COM1x\_eq\_10}} = \frac{\text{OCR1x} * 100}{\text{MAX}_{\text{Fast\_PWM\_TCNT1}} + 1}$$



# Timer/Counter 1 แบบวิธี Fast PWM

กรณีตั้ง  $COMnx[1:0]$  มีค่าเป็น  $11_2$

$$\text{Duty\_Cycle}_{\text{Fast\_PWM\_COM1x\_eq\_11}} = \left( 1 - \frac{OCRnx}{MAX_{\text{Fast\_PWM\_TCNT1}} + 1} \right) * 100$$



# Timer/Counter1 เมื่อทำงานในแบบวิธี การมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

$$F_{OC1x\_PhaseCorrectPWM} = \frac{CLK_{CPU}}{2 * N * MAX_{PhaseCorrectPWM\_TCNT1}}$$

เมื่อ  $F_{OC1x\_PhaseCorrectPWM}$

คือ ความถี่ของสัญญาณดิจิทัลที่ออกทางขา OC1x เมื่อวงจรนับ/จับเวลาหมายเลขหนึ่ง ทำงานในแบบวิธีการมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

$CLK_{CPU}$

คือ ความถี่สัญญาณนาฬิกาที่ตัวประมวลผลใช้ในการทำงาน

N

คือ ค่าคงที่สำหรับป้อนให้วงจรพรีสเกลเลอร์ ใช้ในการหารความถี่ ซึ่งสามารถเลือกค่าได้ 5 ค่า ได้แก่ 1, 8, 64, 256 และ 1024





# Timer/Counter1 เมื่อทำงานในแบบวิธี การมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

- ◆ กรณีที่ตั้งบิตควบคุม COM1x[1:0] ให้มีค่าเท่ากับ  $10_2$

$$\text{Duty\_Cycle}_{\text{PhaseCorrectPWM\_COM1x\_eq\_10}} = \frac{\text{OCR1x} * 100}{\text{MAX}_{\text{PhaseCorrectPWM\_TCNT1}}}$$

- ◆ กรณีที่ตั้งบิตควบคุม COM1x[1:0] ให้มีค่าเท่ากับ  $11_2$

$$\text{Duty\_Cycle}_{\text{PhaseCorrectPWM\_COM1x\_eq\_11}} = \left( 1 - \frac{\text{OCRnx}}{\text{MAX}_{\text{PhaseCorrectPWM\_TCNT1}}} \right) * 100$$



# Timer/Counter1 เมื่อทำงานในแบบวิธี

## การมอดูเลตความกว้างของพัลส์แบบเฟสถูกต้อง

บิตควบคุม WGM1[3:0]	$MAX_{\text{PhaseCorrectPWM\_TCNT1}}$	หมายเหตุ
$0001_2$	255	-
$0010_2$	511	-
$0011_2$	1,023	-
$1010_2$	กำหนดในเรจิสเตอร์ ICR1	ค่าสูงสุดที่ตั้งได้ คือ 65,535
$1011_2$	กำหนดในเรจิสเตอร์ OCR1A	ค่าสูงสุดที่ตั้งได้ คือ 65,535



# ข้อสังเกตของเรจิสเตอร์ Timer Interrupt Flag

◆ เรจิสเตอร์ของชิพ AVR ต่างเบอร์กันจะมีชื่อไม่เหมือนกัน

◆ ATMEGA32 ใช้ TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

◆ ATMEGA328P ใช้ TIFR0 และ TIFR1

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	—	—	—	—	—	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	—	—	ICF1	—	—	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	





# ฉบับที่ 7

