DC Motor Velocity Control Using Pulse Width Modulation (PWM)

e-Yantra Team

Embedded Real-Time Systems Lab
Indian Institute of Technology-Bombay

IIT Bombay
October 10, 2014

Agenda for Discussion

1. Introduction
   - Pulse Width Modulation
   - Duty Cycle
2. PWM Generation in AVR
   - Timer in AVR
   - Timer for PWM generation in Firebird
3. PWM Generation in Firebird V
   - Registers
   - TCCR5A
   - TCCR5B
   - TCNT5
   - OCR5
   - Block Diagram
   - Program

Pulse Width Modulation

Pulse Width Modulation

1. Pulse Width Modulation (PWM), is a method of transmitting information on a series of pulses

Pulse Width Modulation

1. Pulse Width Modulation (PWM), is a method of transmitting information on a series of pulses

2. The data that is being transmitted is encoded on the width of these pulses to control the amount of power being sent to a load

Pulse Width Modulation

1. Pulse Width Modulation (PWM), is a method of transmitting information on a series of pulses
2. The data that is being transmitted is encoded on the width of these pulses to control the amount of power being sent to a load
3. Examples: Electric stoves, Lamp dimmers, and Robotic Servos

Pulse Width Modulation

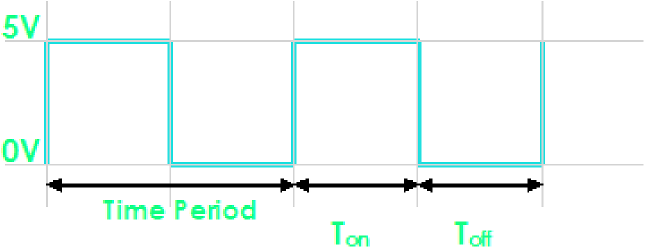
1. Pulse Width Modulation (PWM), is a method of transmitting information on a series of pulses
2. The data that is being transmitted is encoded on the width of these pulses to control the amount of power being sent to a load
3. Examples: Electric stoves, Lamp dimmers, and Robotic Servos

Pulse Width Modulation

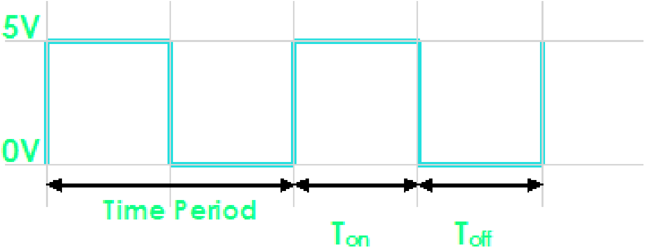
1. Pulse Width Modulation (PWM), is a method of transmitting information on a series of pulses
2. The data that is being transmitted is encoded on the width of these pulses to control the amount of power being sent to a load
3. Examples: Electric stoves, Lamp dimmers, and Robotic Servos

Duty Cycle

Duty Cycle

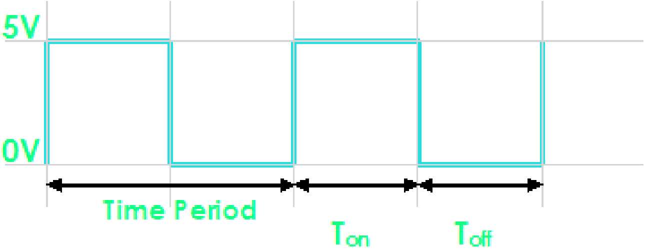


Duty Cycle



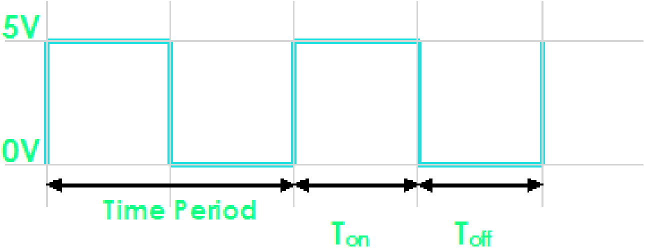
- ✓ The signal remains "ON" for some time and "OFF" for some time.

Duty Cycle



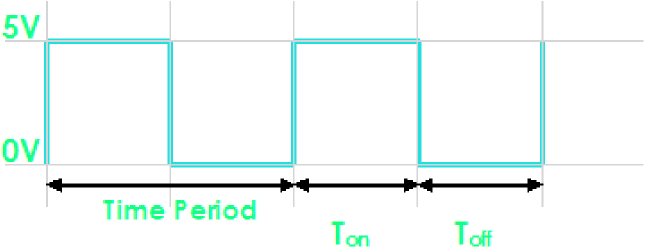
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.

Duty Cycle



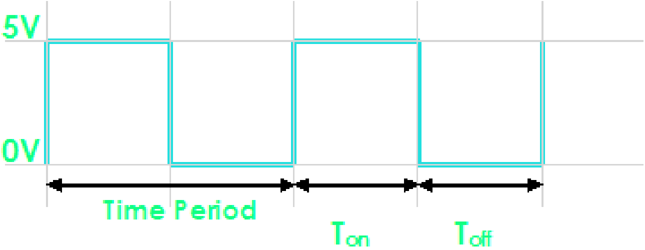
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.

Duty Cycle



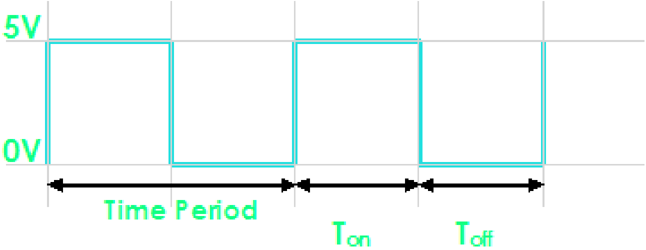
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.
- ✓ When output is high the voltage is 5v

Duty Cycle



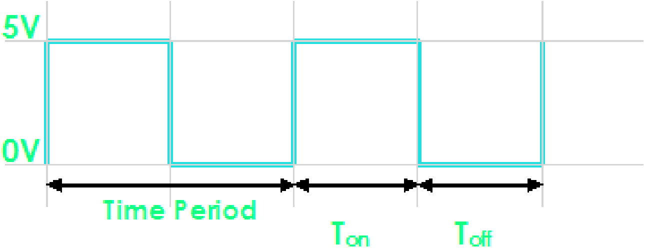
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v

Duty Cycle



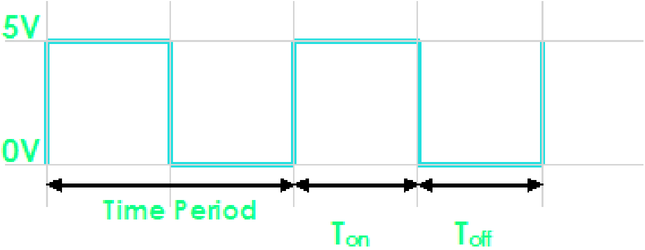
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓ Time Period(T) = Ton + Toff

Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓ Time Period(T) = Ton + Toff
- ✓ Duty Cycle = Ton*100/(Ton + Toff)

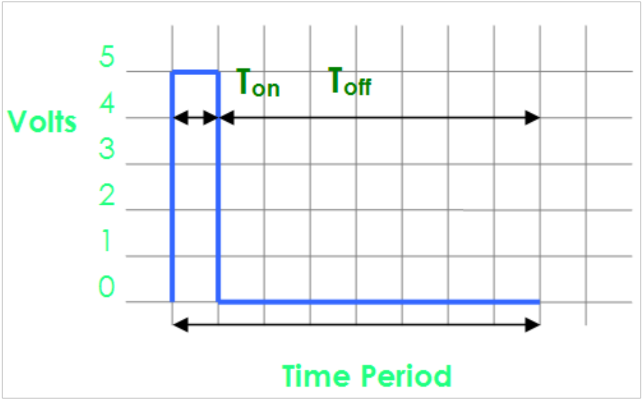
Duty Cycle



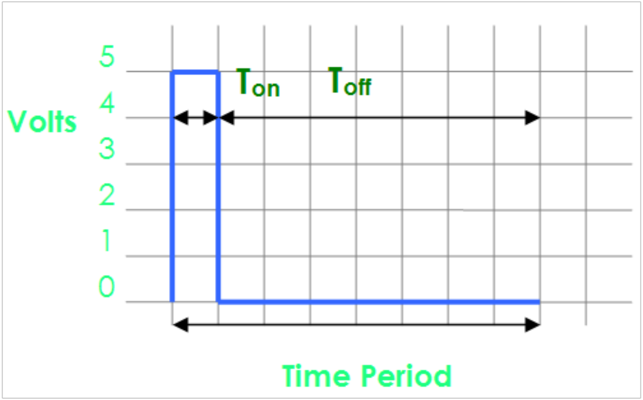
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓ Ton = Time the output remains high.
- ✓ Toff = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓ Time Period(T) = Ton + Toff
- ✓ Duty Cycle = Ton*100/(Ton + Toff)
- ✓ Duty Cycle = 50%

Duty Cycle (Contd..)

Duty Cycle (Contd..)

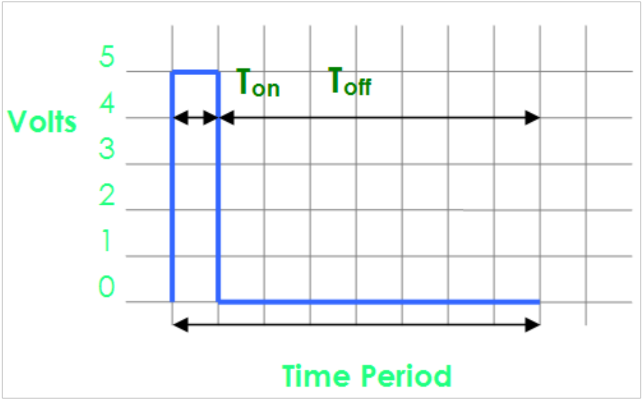


Duty Cycle (Contd..)



✓ Ton = Time the output remains high = 1

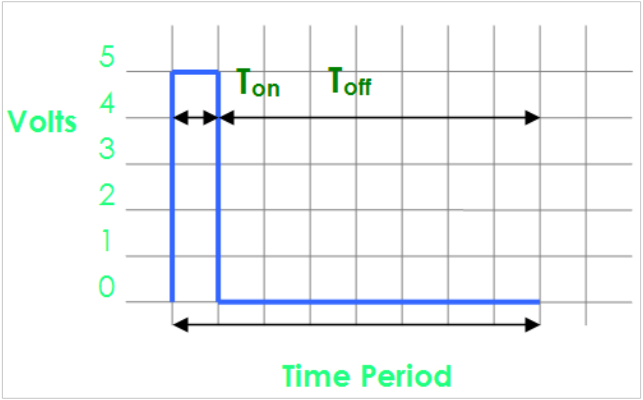
Duty Cycle (Contd..)



✓ Ton = Time the output remains high = 1

✓ Toff = Time the output remains Low = 7

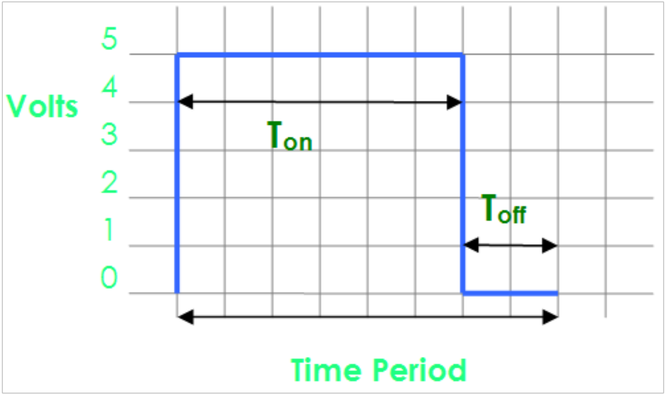
Duty Cycle (Contd..)



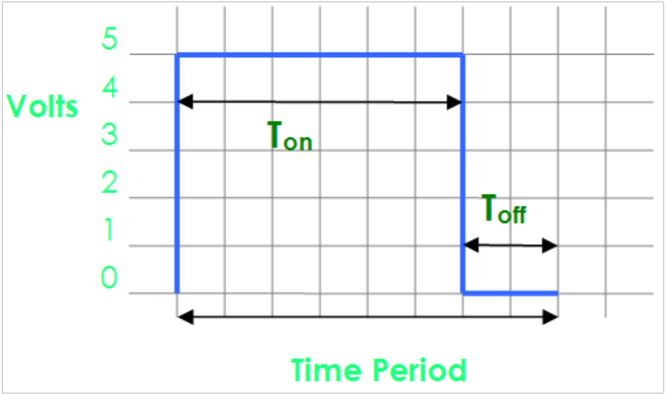
- ✓ Ton = Time the output remains high = 1
- ✓ Toff = Time the output remains Low = 7
- ✓ Duty Cycle = 12.5%

Duty Cycle (Contd..)

Duty Cycle (Contd..)

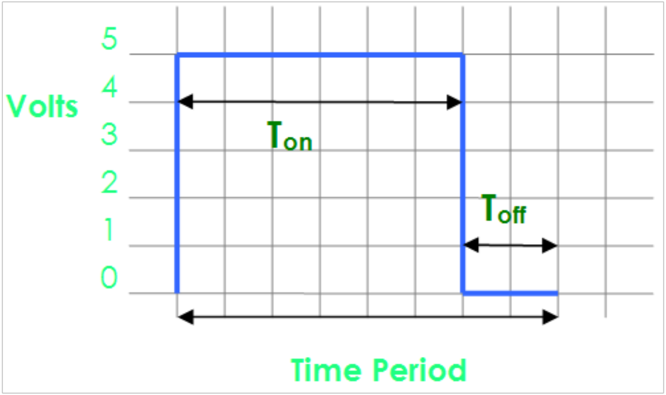


Duty Cycle (Contd..)



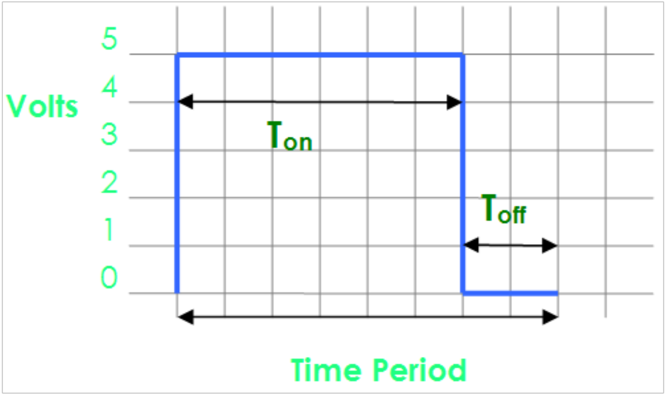
- ✓ Ton = Time the output remains high = 6

Duty Cycle (Contd..)



- ✓ Ton = Time the output remains high = 6
- ✓ Toff = Time the output remains Low = 2

Duty Cycle (Contd..)



- ✓ Ton = Time the output remains high = 6
- ✓ Toff = Time the output remains Low = 2
- ✓ Duty Cycle = 75%

Timer in AVR

Timer in AVR

1. The AVR microcontroller ATmega2560 has

Timer in AVR

1. The AVR microcontroller ATmega2560 has
    - two 8-bit timers (Timer0 and Timer2) and

Timer in AVR

1. The AVR microcontroller ATmega2560 has
   - two 8-bit timers (Timer0 and Timer2) and
   - four 16-bit timers (Timer 1, 3, 4 and 5)

Timer in AVR

1. The AVR microcontroller ATmega2560 has
    - two 8-bit timers (Timer0 and Timer2) and
    - four 16-bit timers (Timer 1, 3, 4 and 5)
2. When the counter reaches its maximum count, it rolls over and executes from the start

Timer in AVR

1. The AVR microcontroller ATmega2560 has
   - two 8-bit timers (Timer0 and Timer2) and
   - four 16-bit timers (Timer 1, 3, 4 and 5)
2. When the counter reaches its maximum count, it rolls over and executes from the start
   - For 8-bit counter, roll over occurs at 255 count and

Timer in AVR

1. The AVR microcontroller ATmega2560 has
   - two 8-bit timers (Timer0 and Timer2) and
   - four 16-bit timers (Timer 1, 3, 4 and 5)
2. When the counter reaches its maximum count, it rolls over and executes from the start
   - For 8-bit counter, roll over occurs at 255 count and
   - For 16-bit counter it occurs at 65535 count

Timer for PWM generation in Firebird

Timer for PWM generation in Firebird

1. Timer 5 can be used for PWM generation for controlling speed of motors

Timer for PWM generation in Firebird

1. Timer 5 can be used for PWM generation for controlling speed of motors

2. The duty cycle of square wave generated by the Timer5 can be varied to produce different average DC values for motors

Timer for PWM generation in Firebird

1. Timer 5 can be used for PWM generation for controlling speed of motors
2. The duty cycle of square wave generated by the Timer5 can be varied to produce different average DC values for motors
3. Using FAST PWM mode to vary speed of motors

Timer for PWM generation in Firebird

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it

2. Four registers are:

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it
2. Four registers are:
   - TCCR5A

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it
2. Four registers are:
   - TCCR5A
   - TCCR5B

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it

2. Four registers are:

   - TCCR5A
   - TCCR5B
   - TCNT5

Timer for PWM generation in Firebird

1. To Program PWM, we have to initialize some register before use it
2. Four registers are:
    - TCCR5A
    - TCCR5B
    - TCNT5
    - OCR5n

TCCR5A- Timer Counter Control Register A

This register is Used to Configure Timer for PWM generation

TCCR5A- Timer Counter Control Register A

This register is Used to Configure Timer for PWM generation

Bit	Symbol	Description	Bit Value
7	COM5A1	Compare Output Mode for Channel A bit 1	1
6	COM5A0	Compare Output Mode for Channel A bit 0	0
5	COM5B1	Compare Output Mode for Channel B bit 1	1
4	COM5B0	Compare Output Mode for Channel B bit 0	0
3	COM5C1	Compare Output Mode for Channel C bit 1	1
2	COM5C0	Compare Output Mode for Channel C bit 0	0
1	WGM11	Waveform Generation Mode bit 1	0
0	WGM10	Waveform Generation Mode bit 0	1

TCCR5A- Timer Counter Control Register A

This register is Used to Configure Timer for PWM generation

Bit	Symbol	Description	Bit Value
7	COM5A1	Compare Output Mode for Channel A bit 1	1
6	COM5A0	Compare Output Mode for Channel A bit 0	0
5	COM5B1	Compare Output Mode for Channel B bit 1	1
4	COM5B0	Compare Output Mode for Channel B bit 0	0
3	COM5C1	Compare Output Mode for Channel C bit 1	1
2	COM5C0	Compare Output Mode for Channel C bit 0	0
1	WGM11	Waveform Generation Mode bit 1	0
0	WGM10	Waveform Generation Mode bit 0	1

TCCR5A=0xA9

Compare Output Mode Fast PWM

Compare Output Mode Fast PWM

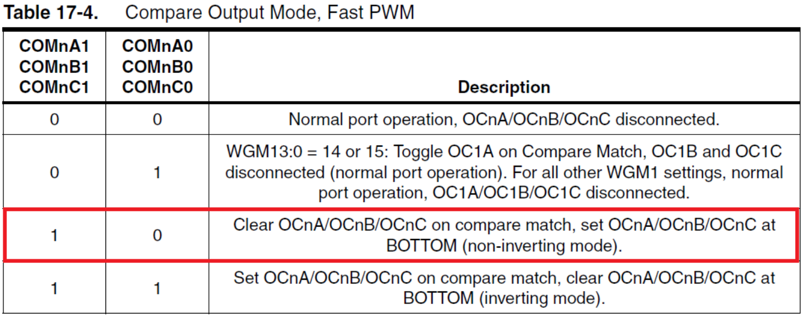


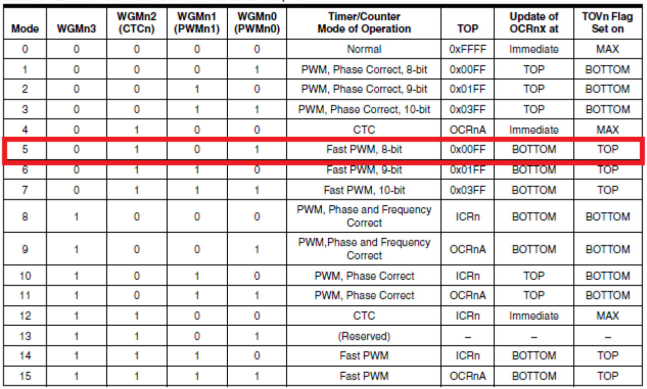
Table 17-4. Compare Output Mode, Fast PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected.
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM (non-inverting mode).
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM (inverting mode).

Waveform Generation Bit

Waveform Generation Bit

Table 17-2. Waveform Generation Mode Bit Description[1]



Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

TCCR5B- Timer Counter Control Register B

This register is Used to Configure Timer for PWM generation

TCCR5B- Timer Counter Control Register B

This register is Used to Configure Timer for PWM generation

Bit	Symbol	Description	Bit Value
7	ICNC5	Input Capture Noise Canceller	0
6	ICES5	Input Capture Edge Select	0
5	–	Reserved Bit	0
4	WGM53	Waveform Generation Mode bit 3	0
3	WGM52	Waveform Generation Mode bit 2	**1**
2	CS52	Clock Select	0
1	CS51	Clock Select	**1**
0	CS50	Clock Select	**1**

TCCR5B- Timer Counter Control Register B

This register is Used to Configure Timer for PWM generation

Bit	Symbol	Description	Bit Value
7	ICNC5	Input Capture Noise Canceller	0
6	ICES5	Input Capture Edge Select	0
5	–	Reserved Bit	0
4	WGM53	Waveform Generation Mode bit 3	0
3	WGM52	Waveform Generation Mode bit 2	1
2	CS52	Clock Select	0
1	CS51	Clock Select	1
0	CS50	Clock Select	1

TCCR5B = 0x0B

Clock Select Bit

Clock Select Bit

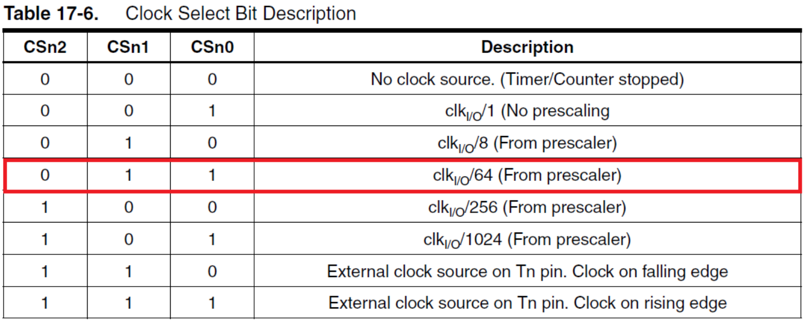


Table 17-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

TCNT5 : Timer/Counter5

TCNT5 : Timer/Counter5

- Purpose: Counts Up/down according to clock frequency

TCNT5 : Timer/Counter5

- Purpose: Counts Up/down according to clock frequency
- TCNT5 is a 16-bit Register

TCNT5 : Timer/Counter5

- Purpose: Counts Up/down according to clock frequency
- TCNT5 is a 16-bit Register
- Counts from 0 to 255 if used in 8-Bit Mode

TCNT5 : Timer/Counter5

- Purpose: Counts Up/down according to clock frequency
- TCNT5 is a 16-bit Register
- Counts from 0 to 255 if used in 8-Bit Mode
- Counts from 0 to 65535 if used in 16-Bit Mode

Output Compare Register 5

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs

- OCR5n: where n=A/B/C are three different 16-bit Register

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs
- OCR5n: where n=A/B/C are three different 16-bit Register
- Each can be used individually as separate PWM channel

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs
- OCR5n: where n=A/B/C are three different 16-bit Register
- Each can be used individually as separate PWM channel
- OCR5n is represented as two 8-bit register as OCR5nL and OCR5nH

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs
- OCR5n: where n=A/B/C are three different 16-bit Register
- Each can be used individually as separate PWM channel
- OCR5n is represented as two 8-bit register as OCR5nL and OCR5nH
- PWM generated is 8-bit, so only lower register is used

Output Compare Register 5

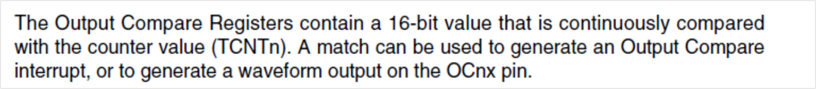
- Purpose: Compares itself from TCNT counter and set flags when match occurs
- OCR5n: where n=A/B/C are three different 16-bit Register
- Each can be used individually as separate PWM channel
- OCR5n is represented as two 8-bit register as OCR5nL and OCR5nH
- PWM generated is 8-bit, so only lower register is used
- OCR5AL (PortL3) is connected to Left Motor

Output Compare Register 5

- Purpose: Compares itself from TCNT counter and set flags when match occurs
- OCR5n: where n=A/B/C are three different 16-bit Register
- Each can be used individually as separate PWM channel
- OCR5n is represented as two 8-bit register as OCR5nL and OCR5nH
- PWM generated is 8-bit, so only lower register is used
- OCR5AL (PortL3) is connected to Left Motor
- OCR5BL (PortL4) is connected to Right Motor

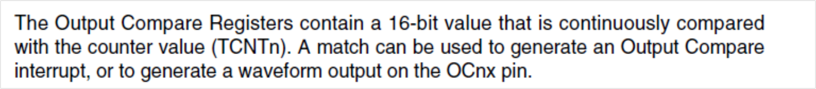
Output Compare Register 5 (Contd..)

Output Compare Register 5 (Contd..)

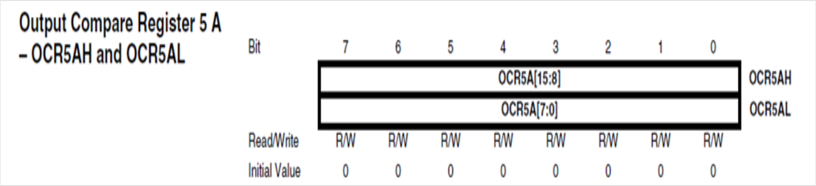


The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnx pin.

Output Compare Register 5 (Contd..)



The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnx pin.

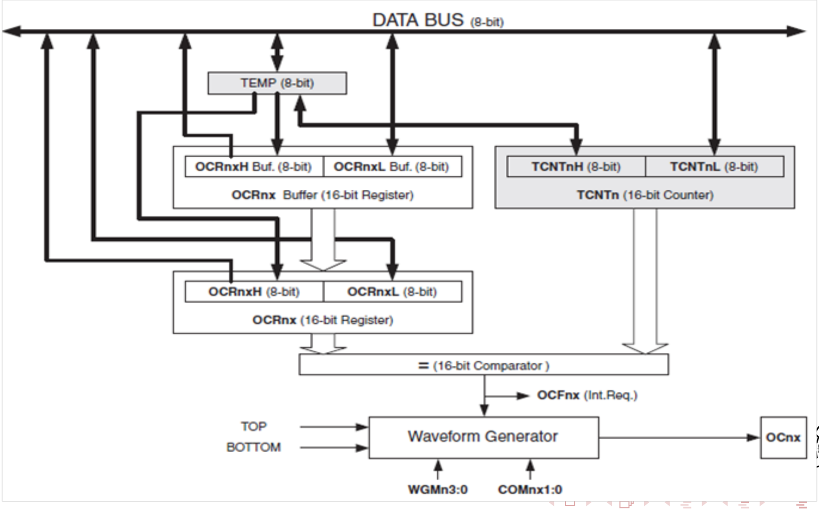


Output Compare Register 5 A – OCR5AH and OCR5AL

Bit	7	6	5	4	3	2	1	0	
	OCR5A[15:8]								OCR5AH
	OCR5A[7:0]								OCR5AL
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	

Block Diagram - Output Compare Unit

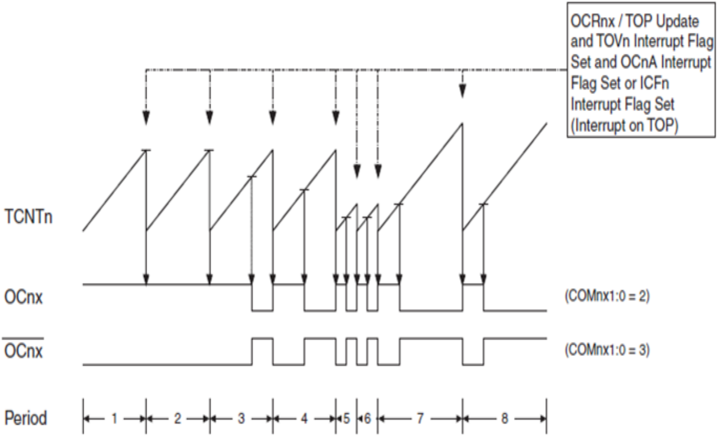
Block Diagram - Output Compare Unit



Timing Diagram Fast PWM

Timing Diagram Fast PWM

Figure 55. Fast PWM Mode, Timing Diagram


OCRnx / TOP Update and TOVn Interrupt Flag Set and OCnA Interrupt Flag Set or ICFn Interrupt Flag Set (Interrupt on TOP)
TCNTn
OCnx
(COMnx1:0 = 2)
$\overline{\text{OCnx}}$
(COMnx1:0 = 3)
Period
1
2
3
4
5
6
7
8

Syntax for C-Program

PWM Initialization

Syntax for C-Program

PWM Initialization

Port Pin Config

Syntax for C-Program

PWM Initialization

Port Pin Config

```
void motion_pin_config (void)  //Configure Pins as Output
{

 Port A for motion control and Port L for Velocity Control must be defined Output

}
```

Syntax for C-Program

PWM Initialization

Port Pin Config

```
void motion_pin_config (void)  //Configure Pins as Output
{

 Port A for motion control and Port L for Velocity Control must be defined Output

}
```

PWM Initialization

Syntax for C-Program

PWM Initialization

Port Pin Config

```
void motion_pin_config (void)  //Configure Pins as Output
{

 Port A for motion control and Port L for Velocity Control must be defined Output

}
```

PWM Initialization

```
void timer5_init()  //Set Register Values for starting Fast 8-bit PWM
{

   TCCR5A =
   TCCR5B =
   TCNT5H = 0xFF;
   TCNT5L = 0x00;
   OCR5AH = 0x00;
   OCR5AL = 0xFF;
   OCR5BH = 0x00;
   OCR5BL = 0xFF;
}
```

Syntax for C-Program

Program

Syntax for C-Program

Program

Main Program

Syntax for C-Program

Program

Main Program

```
int main(void)
{
  init_devices();
  forward();
  while(1)
  {
    velocity(100,100);
    _delay_ms(500);
    velocity(0,255);
    _delay_ms(500);
  }
}
```

Syntax for C-Program

Program

Main Program

```
int main(void)
{
  init_devices();
  forward();
  while(1)
  {
    velocity(100,100);
    _delay_ms(500);
    velocity(0,255);
    _delay_ms(500);
  }
}
```

Velocity Function

Syntax for C-Program

Program

Main Program

```
int main(void)
{
  init_devices();
  forward();
  while(1)
  {
    velocity(100,100);
    _delay_ms(500);
    velocity(0,255);
    _delay_ms(500);
  }
}
```

Velocity Function

```
void velocity (unsigned char left_motor, unsigned char right_motor)
{

   OCR5AL = (unsigned char)left_motor;
   OCR5BL = (unsigned char)right_motor;
}
```

Thank You!

Post your queries on: http://qa.e-yantra.org/