

# Motion control using Pulse Width Modulation in Firebird V

e-Yantra Team  
Embedded Real-Time Systems Lab  
Indian Institute of Technology-Bombay

IIT Bombay  
January 15, 2016



# Agenda for Discussion

- 1 Pulse Width Modulation
  - Duty Cycle
  - Motion Control Using Pulse Width Modulation in Firebird V
- 2 Registers
  - Timer/Counter 5(TCNT5)
  - Output Compare Register 5
  - Timer/Counter Control Register (TCCR5A and TCCR5B)
    - TCCR5A
    - TCCR5B
- 3 Summary
- 4 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

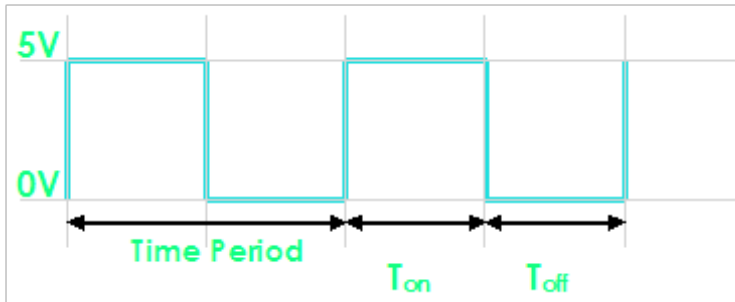




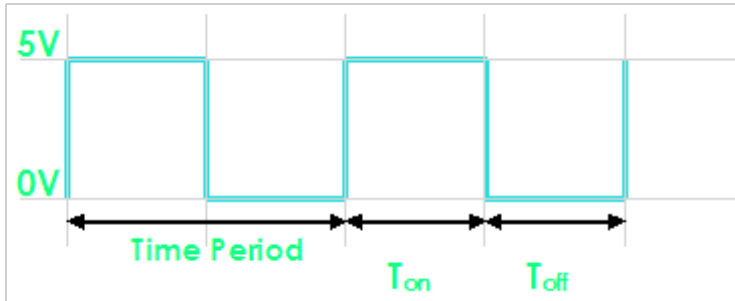
# Duty Cycle



# Duty Cycle



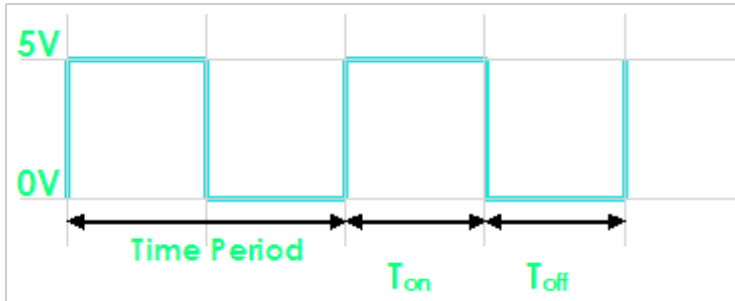
# Duty Cycle



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



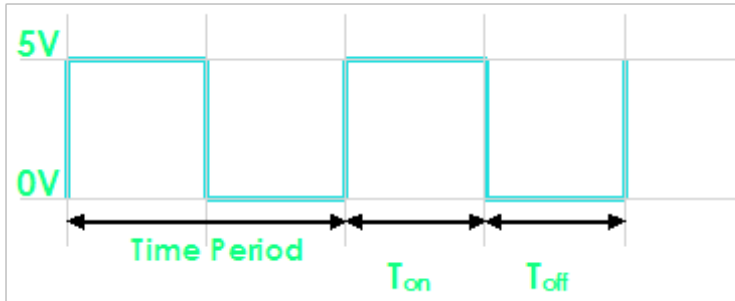
# Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.



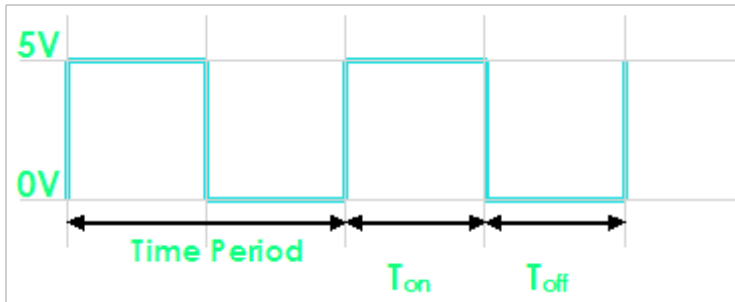
# Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.



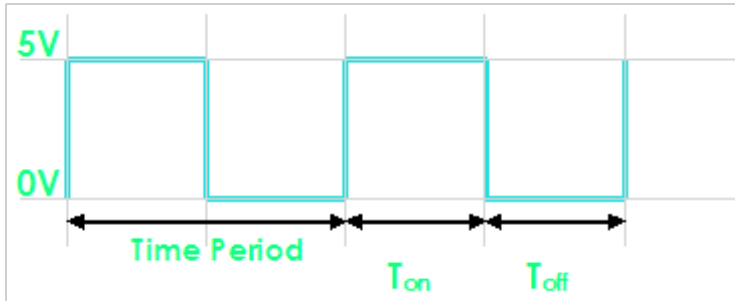
# Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.
- ✓ When output is high the voltage is 5v



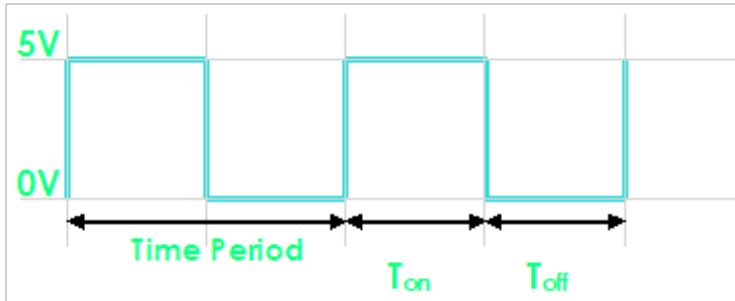
# Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v



# Duty Cycle

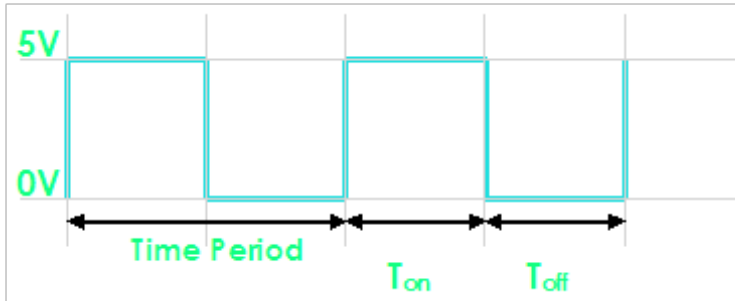


- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓  $\text{Time Period}(T) = T_{on} + T_{off}$





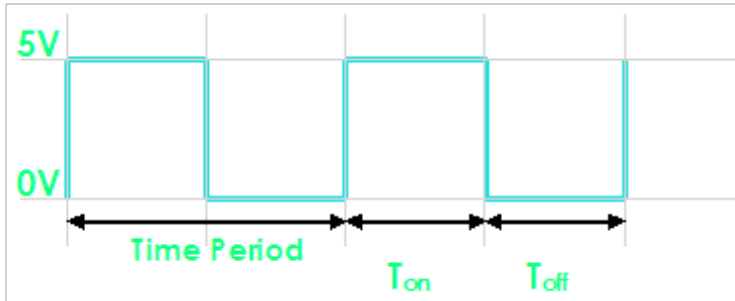
# Duty Cycle



- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓ Time Period( $T$ ) =  $T_{on} + T_{off}$
- ✓ Duty Cycle =  $T_{on} / (T_{on} + T_{off})$



# Duty Cycle



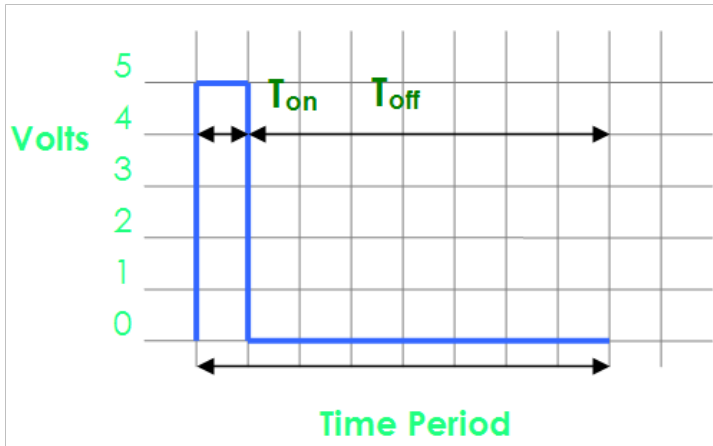
- ✓ The signal remains "ON" for some time and "OFF" for some time.
- ✓  $T_{on}$  = Time the output remains high.
- ✓  $T_{off}$  = Time the output remains Low.
- ✓ When output is high the voltage is 5v
- ✓ When output is low the voltage is 0v
- ✓ Time Period( $T$ ) =  $T_{on} + T_{off}$
- ✓ Duty Cycle =  $T_{on} / (T_{on} + T_{off})$
- ✓ Duty Cycle = 50%



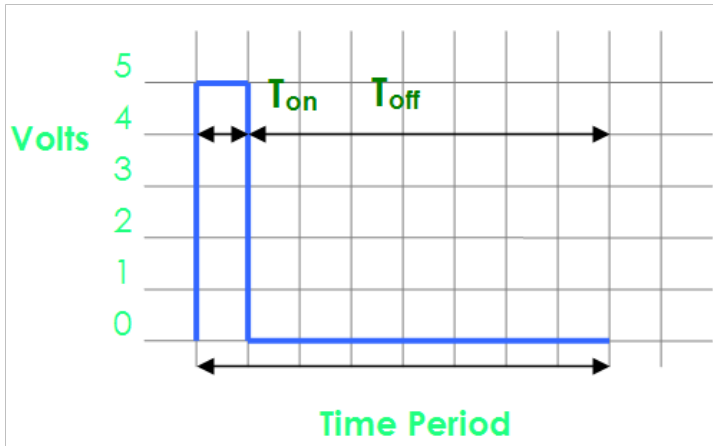
# Duty Cycle (Contd..)



# Duty Cycle (Contd..)



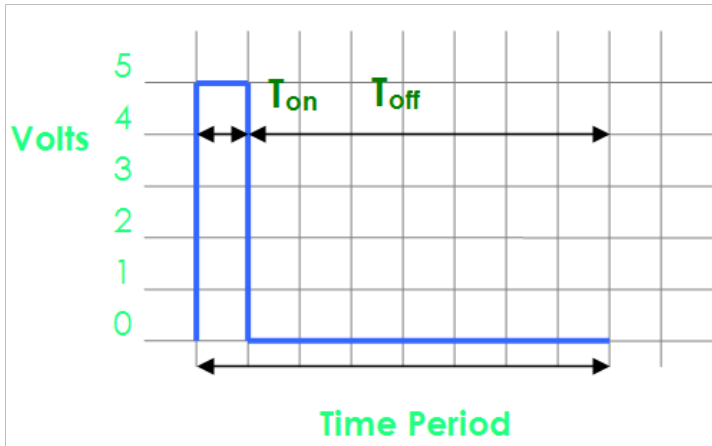
## Duty Cycle (Contd..)



✓  $T_{on}$  = Time the output remains high = 1



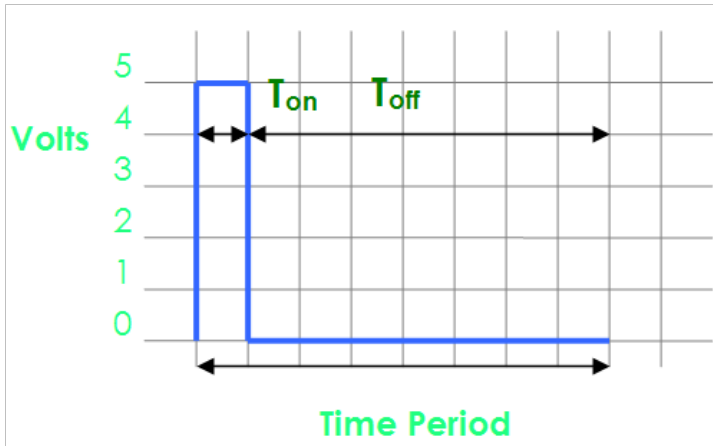
## Duty Cycle (Contd..)



- ✓  $T_{on}$  = Time the output remains high = 1
- ✓  $T_{off}$  = Time the output remains Low = 7



## Duty Cycle (Contd..)



- ✓  $T_{on}$  = Time the output remains high = 1
- ✓  $T_{off}$  = Time the output remains Low = 7
- ✓ Duty Cycle = 12.5%

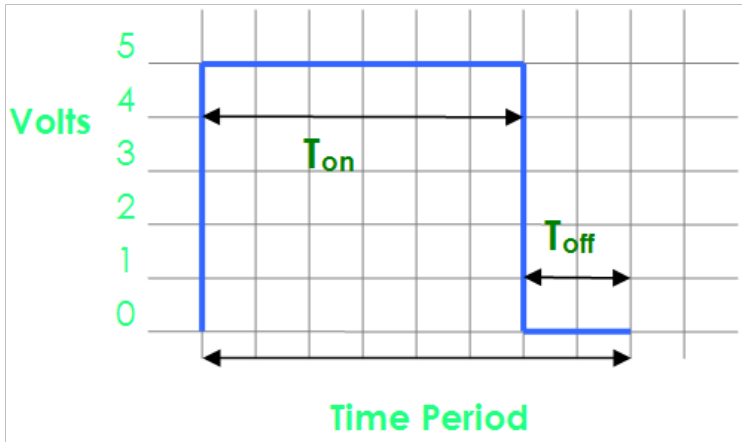


# Duty Cycle (Contd..)

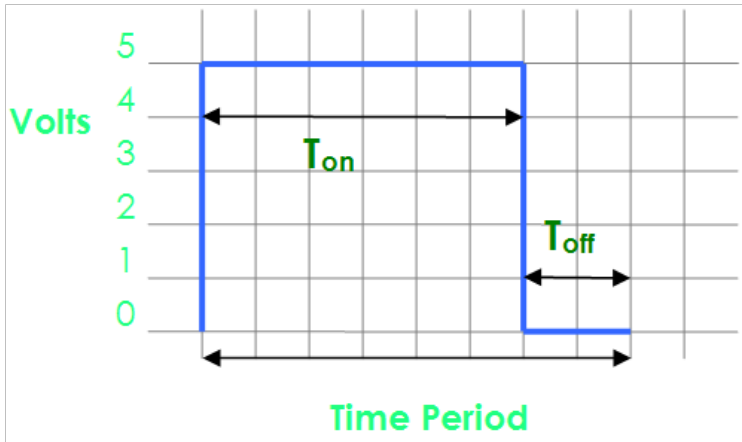




## Duty Cycle (Contd..)



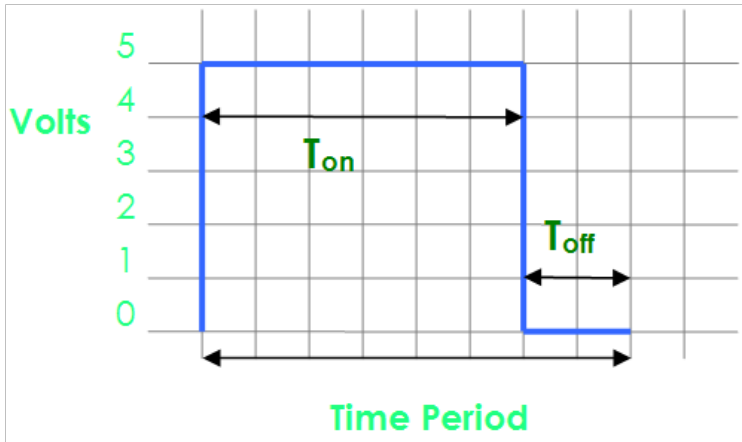
## Duty Cycle (Contd..)



✓  $T_{on}$  = Time the output remains high = 6



## Duty Cycle (Contd..)

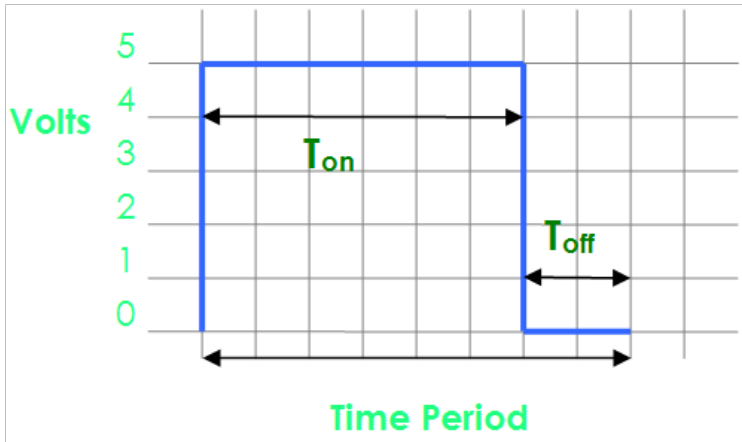


✓  $T_{on}$  = Time the output remains high = 6

✓  $T_{off}$  = Time the output remains Low = 2



## Duty Cycle (Contd..)



- ✓  $T_{on}$  = Time the output remains high = 6
- ✓  $T_{off}$  = Time the output remains Low = 2
- ✓ Duty Cycle = 75%



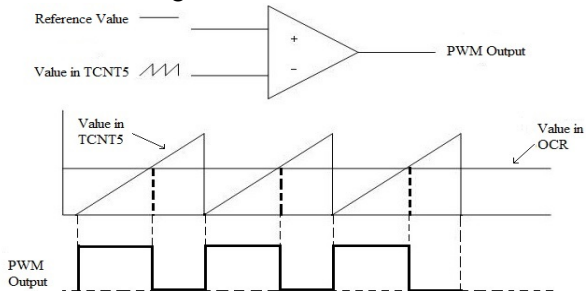
# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:



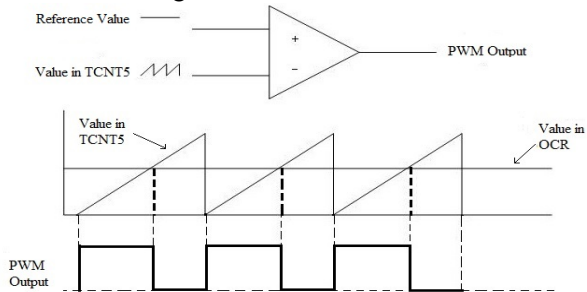
# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:



# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:

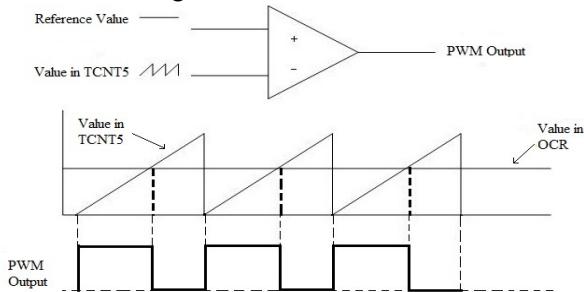


- Its generation involves the use of following registers:



# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:



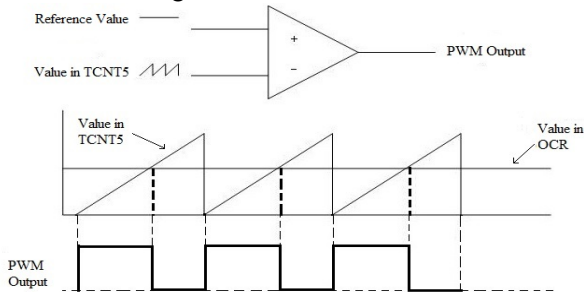
- Its generation involves the use of following registers:
  - ✓ Timer/Counter register 5(TCNT5)





# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:

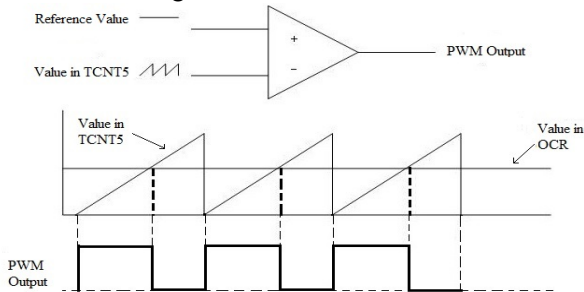


- Its generation involves the use of following registers:
  - ✓ Timer/Counter register 5(TCNT5)
  - ✓ Output Comparator register 5(OCR5A and OCR5B)



# Pulse Width Modulation

- Pulse width waveform generated for motion control of Firebird V is:



- Its generation involves the use of following registers:
  - ✓ Timer/Counter register 5(TCNT5)
  - ✓ Output Comparator register 5(OCR5A and OCR5B)
  - ✓ Timer Counter Comparator register(TCCR5A and TCCR5B)



# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.



# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.



# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.



## Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.
- For example, a 3 bit counter will have 8 values (i.e. 0-7). Its waveform will be seen as follows:



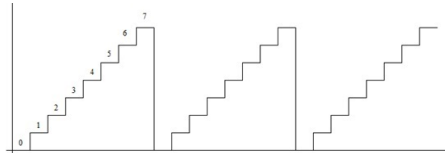
## Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.
- For example, a 3 bit counter will have 8 values (i.e. 0-7). Its waveform will be seen as follows:



# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.
- For example, a 3 bit counter will have 8 values (i.e. 0-7). Its waveform will be seen as follows:



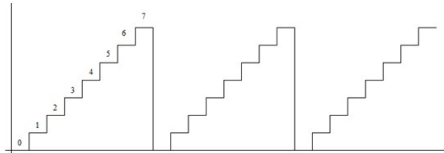
- For n-bit counter, maximum value =  $2^n - 1$ .





# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.
- For example, a 3 bit counter will have 8 values (i.e. 0-7). Its waveform will be seen as follows:

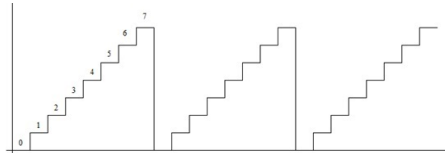


- For n-bit counter, maximum value =  $2^n - 1$ .
- The Timer/Counter 5 is a 16 bit register.



# Timer/Counter 5 (TCNT5)

- The Timer/Counter is a register that increments its value after every clock cycle.
- The value in the timer/counter is compared with a reference value to generate PWM.
- This value depends upon the resolution of Timer.
- For example, a 3 bit counter will have 8 values (i.e. 0-7). Its waveform will be seen as follows:



- For n-bit counter, maximum value =  $2^n - 1$ .
- The Timer/Counter 5 is a 16 bit register.
- We use it in 8-bit mode, for PWM generation.



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation:



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation:



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation: OCR5A, OCR5B and OCR5C.



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation: OCR5A, OCR5B and OCR5C.
- For motion control of Firebird V, we use OCR5A and OCR5B registers





# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation: OCR5A, OCR5B and OCR5C.
- For motion control of Firebird V, we use OCR5A and OCR5B registers
- OCR5A is associated with the OC5A pin (PORTL.3), which is the enable(EN1) pin connected with the left motor.



# Output Compare Register (OCR5A, OCR5B and OCR5C)

- The value of the Timer/Counter 5 is constantly compared with a reference value.
- This reference value is given in the Output Compare Register(OCR).
- Output Compare Registers associated with Timer 5 for PWM generation: OCR5A, OCR5B and OCR5C.
- For motion control of Firebird V, we use OCR5A and OCR5B registers
- OCR5A is associated with the OC5A pin (PORTL.3), which is the enable(EN1) pin connected with the left motor.
- Similarly, OCR5B is associated with the OC5B pin (PORTL.4), which is the enable(EN2) pin connected with the right motor.



# Timer/Counter Control Register (TCCR5A and TCCR5B)

- Bits in TCCR5A and TCCR5B registers configure the Timer 5 module for generating PWM



# Timer/Counter Control Register (TCCR5A and TCCR5B)

- Bits in TCCR5A and TCCR5B registers configure the Timer 5 module for generating PWM
- Each bit in these registers determine the kind of signal to be generated.



# Timer/Counter Control Register (TCCR5A and TCCR5B)

- Bits in TCCR5A and TCCR5B registers configure the Timer 5 module for generating PWM
- Each bit in these registers determine the kind of signal to be generated.
- TCCR5A is a control register and is used to set COM bits and WGM bits.



# Timer/Counter Control Register (TCCR5A and TCCR5B)

- Bits in TCCR5A and TCCR5B registers configure the Timer 5 module for generating PWM
- Each bit in these registers determine the kind of signal to be generated.
- TCCR5A is a control register and is used to set COM bits and WGM bits.
- TCCR5B is also a control register,used to select clock frequency for the Timer 5 and for PWM generation.



# Timer/Counter Control Register 5A(TCCR5A)



# Timer/Counter Control Register 5A(TCCR5A)

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

- It has 2 types of bits: Compare output mode bit & waveform generation mode bit.





# Timer/Counter Control Register 5A(TCCR5A)

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

- It has 2 types of bits: Compare output mode bit & waveform generation mode bit.
- Compare Output Mode bits decide the action to be taken when counter(TCNT5) value matches reference value in Output Compare Register(OCR5).



# Timer/Counter Control Register 5A (TCCR5A) (...contd)

- The Waveform Generation Mode bits are used to generate the type of PWM signal needed.



# Timer/Counter Control Register 5A (TCCR5A) (...contd)

- The Waveform Generation Mode bits are used to generate the type of PWM signal needed.
- In the given table:



# Timer/Counter Control Register 5A (TCCR5A) (...contd)

- The Waveform Generation Mode bits are used to generate the type of PWM signal needed.
- In the given table:
  - ✓ COM5A1 AND COM5A0 bits are used to control the output on left motor.



# Timer/Counter Control Register 5A (TCCR5A) (...contd)

- The Waveform Generation Mode bits are used to generate the type of PWM signal needed.
- In the given table:
  - ✓ COM5A1 AND COM5A0 bits are used to control the output on left motor.
  - ✓ COM5B1 and COM5B0 bits are used to control the output on right motor.



# TCCR5A: Compare Output Mode Bits

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).

① We are using non-inverting mode for PWM generation.



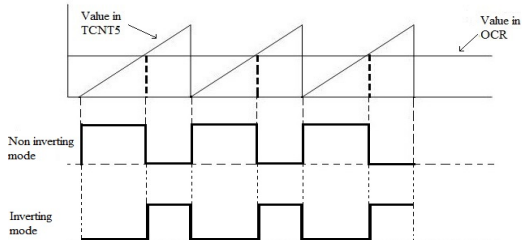
# TCCR5A: Waveform Generation Mode Bits

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

- WGM bits determine, type of waveform to be generated. We will be using Fast PWM, 8-bit mode.



# Inverting and Non-inverting mode

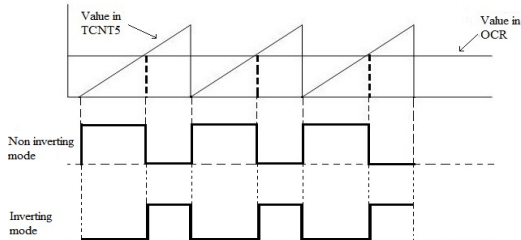


- There are two modes of pwm waveform generation:





# Inverting and Non-inverting mode



- There are two modes of pwm waveform generation:
- Non-inverting mode and inverting mode



# Timer/Counter Control Register 5B (TCCR5B)

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



# Timer/Counter Control Register 5B (TCCR5B)

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

- In the above Table:



# Timer/Counter Control Register 5B (TCCR5B)

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

- In the above Table:
- Input Capture Noise Canceller and Input Capture Edge Select bits are not being used here.



# Timer/Counter Control Register 5B (TCCR5B)

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

- In the above Table:
- Input Capture Noise Canceller and Input Capture Edge Select bits are not being used here.
- WGM bits (WGM52 and WGM53), are used for PWM generation.



# Timer/Counter Control Register 5B (TCCR5B)

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

- In the above Table:
- Input Capture Noise Canceller and Input Capture Edge Select bits are not being used here.
- WGM bits (WGM52 and WGM53), are used for PWM generation.
- CS52, CS51, CS50 (Clock select) bits are used to select a frequency at which timer/counter Register will increment its value.



# TCCR5B: Clock Select Bits

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{I/O}$ /(No prescaling)
0	1	0	$\text{clk}_{I/O}/8$ (From prescaler)
0	1	1	$\text{clk}_{I/O}/64$ (From prescaler)
1	0	0	$\text{clk}_{I/O}/256$ (From prescaler)
1	0	1	$\text{clk}_{I/O}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

- Prescaler is used to reduce the frequency of the clock, suitable for the type of PWM being generated.



# TCCR5B: Clock Select Bits

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{\text{I/O}}$ /(No prescaling)
0	1	0	$\text{clk}_{\text{I/O}}/8$ (From prescaler)
0	1	1	$\text{clk}_{\text{I/O}}/64$ (From prescaler)
1	0	0	$\text{clk}_{\text{I/O}}/256$ (From prescaler)
1	0	1	$\text{clk}_{\text{I/O}}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

- Prescaler is used to reduce the frequency of the clock, suitable for the type of PWM being generated.
- Clock select bits decide the factor with which clock frequency will be divided.





# TCCR5B: Clock Select Bits

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{I/O}$ /(No prescaling)
0	1	0	$\text{clk}_{I/O}/8$ (From prescaler)
0	1	1	$\text{clk}_{I/O}/64$ (From prescaler)
1	0	0	$\text{clk}_{I/O}/256$ (From prescaler)
1	0	1	$\text{clk}_{I/O}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

- Prescaler is used to reduce the frequency of the clock, suitable for the type of PWM being generated.
- Clock select bits decide the factor with which clock frequency will be divided.
- We are using 64 as prescaler so, Clock select bits, we need is 011 .



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- $\text{Clockfrequency}(f_{clk_{I/O}}) = 14745600\text{Hz}$



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get
- $1000 = \frac{14745600}{2^8 \times \text{prescaler}}$



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get
- $1000 = \frac{14745600}{2^8 \times \text{prescaler}}$
- $1000 = \frac{14745600}{256 \times \text{prescaler}}$





# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get
- $1000 = \frac{14745600}{2^8 \times \text{prescaler}}$
- $1000 = \frac{14745600}{256 \times \text{prescaler}}$
- $\text{prescaler} = \frac{14745600}{256 \times 1000}$



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get
- $1000 = \frac{14745600}{2^8 \times \text{prescaler}}$
- $1000 = \frac{14745600}{256 \times \text{prescaler}}$
- $\text{prescaler} = \frac{14745600}{256 \times 1000}$
- $\text{prescaler} = 57.6$



# PWM Mode selection

- Component frequency =  $\frac{f_{clk_{I/O}}}{N \times \text{prescaler}}$  .....equation(1)
- Here the component is DC motor whose frequency is 1000Hz
- Clockfrequency( $f_{clk_{I/O}}$ ) = 14745600Hz
- Since we are using PWM in 8 bit mode  $N = 2^8 = 256$
- Putting all the above values in equation (1), we get
- $1000 = \frac{14745600}{2^8 \times \text{prescaler}}$
- $1000 = \frac{14745600}{256 \times \text{prescaler}}$
- $\text{prescaler} = \frac{14745600}{256 \times 1000}$
- $\text{prescaler} = 57.6$
- Closest value to 57.6 is 64. So, we chose 64 as a prescaler value in 8-bit Fast PWM mode.



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$





# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$
- $TCCR5B = 0x0B$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$
- $TCCR5B = 0x0B$
- $OCR5AH = 0x00$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$
- $TCCR5B = 0x0B$
- $OCR5AH = 0x00$
- $OCR5AL = 0xFF$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$
- $TCCR5B = 0x0B$
- $OCR5AH = 0x00$
- $OCR5AL = 0xFF$
- $OCR5BH = 0x00$



# Summary

- In order to use Fast PWM mode to control dc motors of Firebird V. We have to initialize following registers.
- $TCNT5H = 0xFF$
- $TCNT5L = 0x00$
- $TCCR5A = 0xA9$
- $TCCR5B = 0x0B$
- $OCR5AH = 0x00$
- $OCR5AL = 0xFF$
- $OCR5BH = 0x00$
- $OCR5BL = 0xFF$



# 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!

