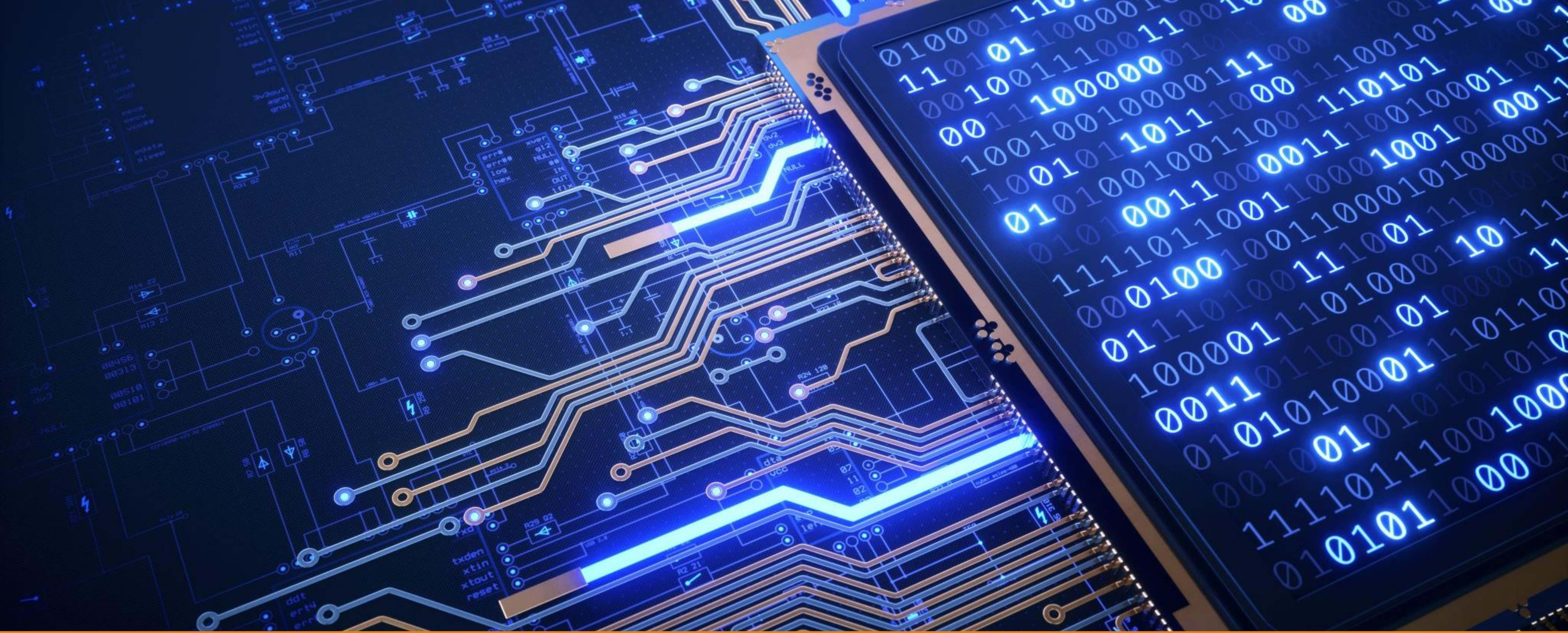


Embedded Systems

Dr. Abdelhay Ali

Lecture 5

PWM and External Hardware Interrupts

Outlines

Outlines

Introduction

Pulse Width Modulation (PWM) is a technique by which the width of a pulse is varied while keeping the frequency constant.

Why do we need to do this?

Let’s take an example of controlling DC motor speed, more the Pulse width more the speed. Also, there are applications like controlling light intensity by PWM.

Introduction

A period of a pulse consists of an **ON** cycle (5V) and an **OFF** cycle (0V). The fraction for which the signal is ON over a period is known as the **duty cycle**.

Duty Cycle (In %) = $\frac{T_{ON}}{Total\,Period} * 100$

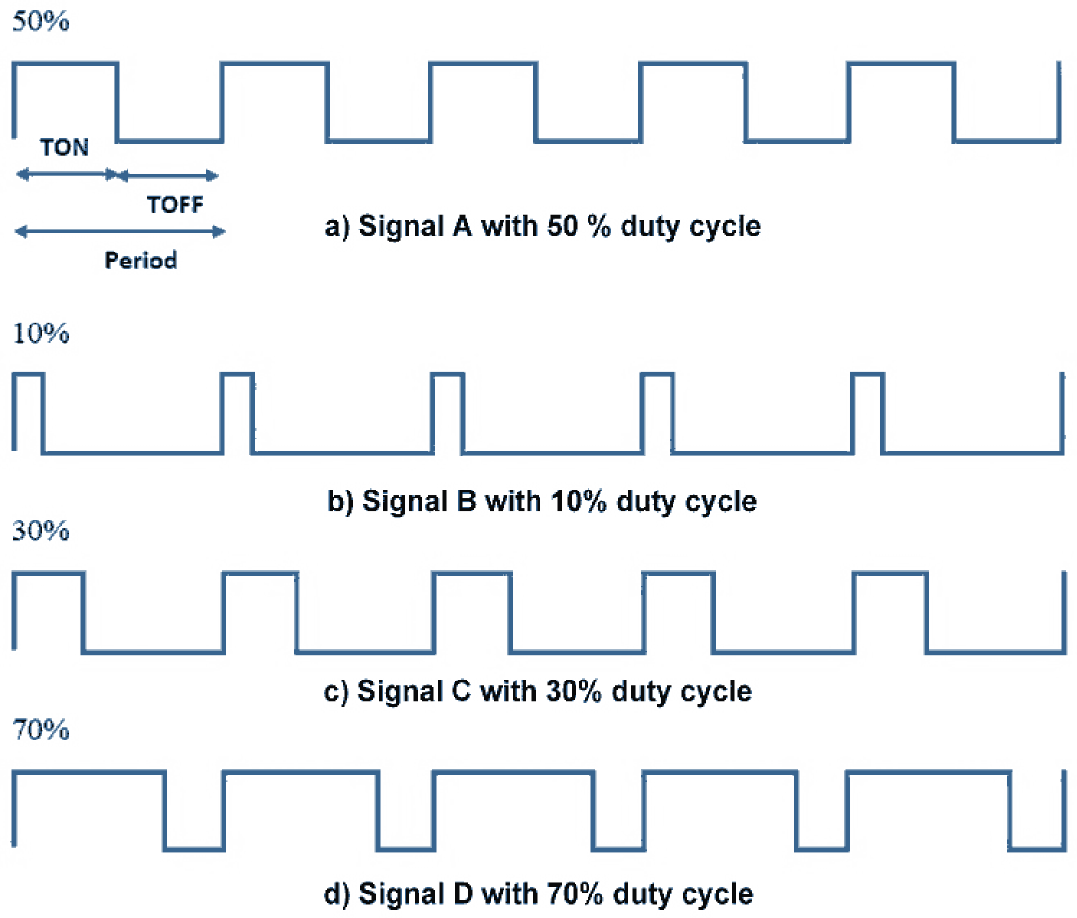
E.g. Consider a pulse with a period of 10ms which remains ON (high) for 2ms.The duty cycle of this pulse will be

D = 2ms / 10ms = 20%

Introduction

Through the PWM technique, we can control the power delivered to the load by using the ON-OFF signal

Introduction



a) Signal A with 50 % duty cycle

b) Signal B with 10% duty cycle

c) Signal C with 30% duty cycle

d) Signal D with 70% duty cycle

Outlines

Introduction

ATmega has an inbuilt PWM unit. As we know, ATmega has 3 Timers T0, T1, and T2 which can be used for PWM generation.

Mainly there are two modes in PWM.

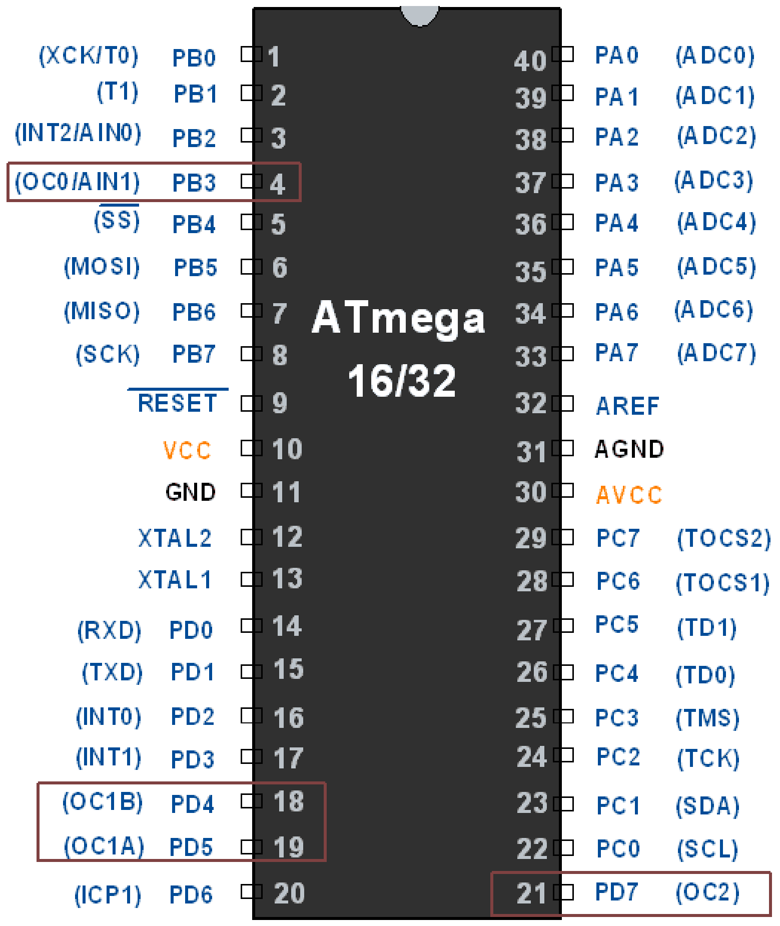
1. Fast PWM
2. Phase correct PWM

We need to configure the Timer Register for generating PWM.

PWM output will be generated on the corresponding Timer's output compare pin (OCx).

Introduction

PWM output will be generated on the corresponding Timer’s output compare pin (OCx).

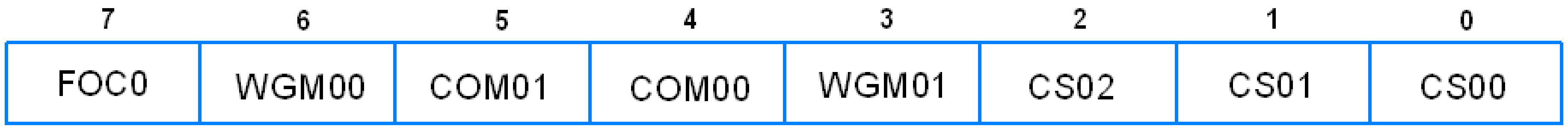


Outlines

Configuring Timer0 for PWM generation

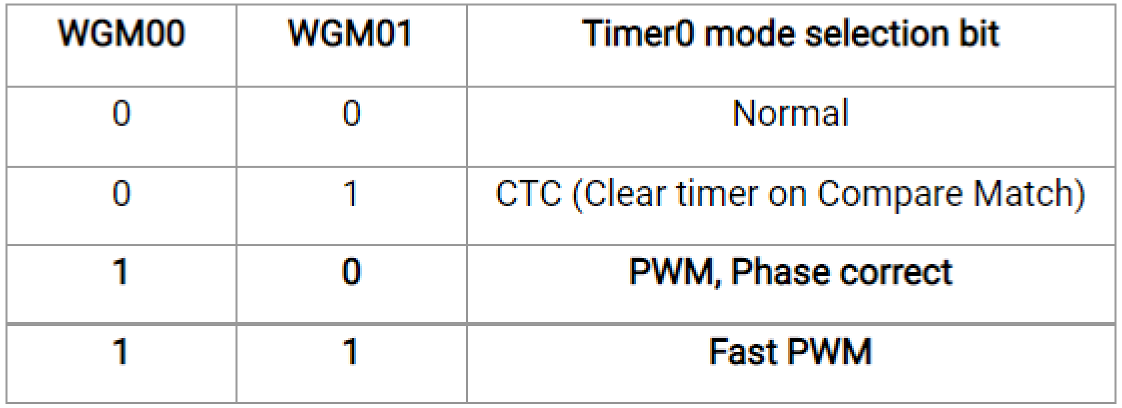
We just need to set some bits in the TCCR0 register.

TCCR0: Timer Counter Control Register 0



7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 6, 3 - WGM00, WGM01: Waveform Generation Mode

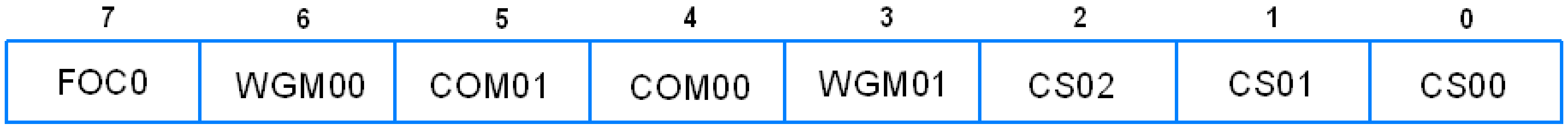


WGM00	WGM01	Timer0 mode selection bit
0	0	Normal
0	1	CTC (Clear timer on Compare Match)
1	0	PWM, Phase correct
1	1	Fast PWM

Configuring Timer0 for PWM generation

We just need to set some bits in the TCCR0 register.

TCCR0: Timer Counter Control Register 0



7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 5:4 - COM01:00:
1.When WGM00: WGM01= 11 i.e. **Fast PWM.** Compare Output Mode waveform generator on OC0 pin

COM01	COM00	Mode Name	Description
0	0	Disconnected	The normal port operation, OC0 disconnected
0	1	Reserved	Reserved
1	**0**	**Non-inverted**	Clear OC0 on compare match, set OC0 at TOP
1	**1**	**Inverted PWM**	Set OC0 on compare match, clear OC0 at TOP

Configuring Timer0 for PWM generation

We just need to set some bits in the TCCR0 register.

TCCR0: Timer Counter Control Register 0

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 5:4 - COM01:00:
2. When WGM00: WGM01= 10 i.e. **Phase correct PWM.** Compare Output Mode waveform generator on OC0 pin

COM01	COM00	Description
0	0	The normal port operation, OC0 disconnected
0	1	Reserved
1	**0**	Clear OC0 on compare match when up-counting, set OC0 on compare match when down-counting
1	**1**	Set OC0 on compare match when up-counting, Clear OC0 on compare match when down-counting

Configuring Timer0 for PWM generation

We just need to set some bits in the TCCR0 register.

TCCR0: Timer Counter Control Register 0

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 2:0 - CS02:CS00: Clock Source Select

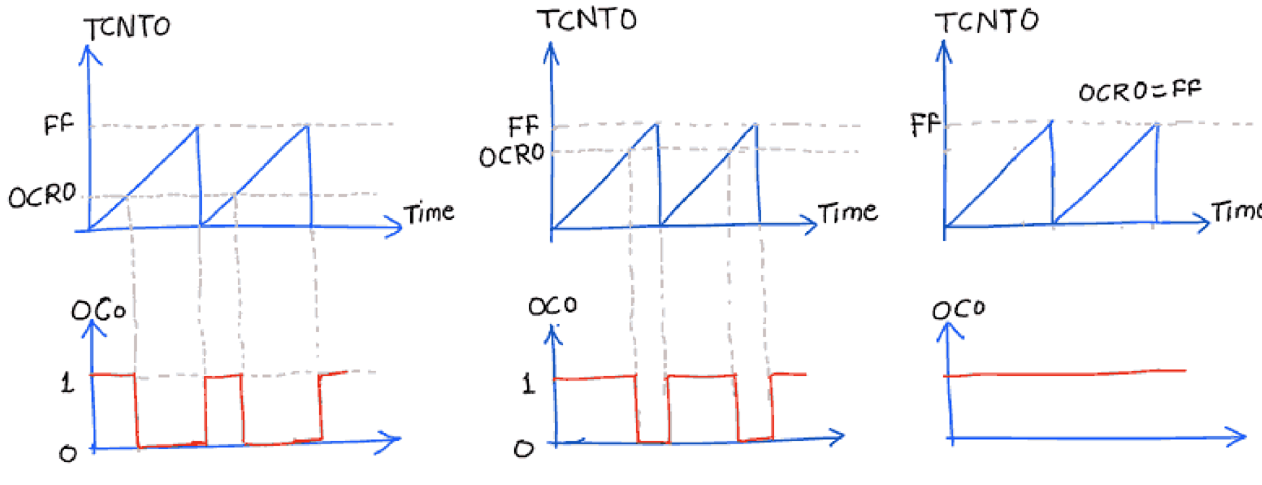
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer / Counter stopped)
0	0	1	clk (no pre-scaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. clock on falling edge
1	1	1	External clock source on T0 pin. clock on rising edge.

Outlines

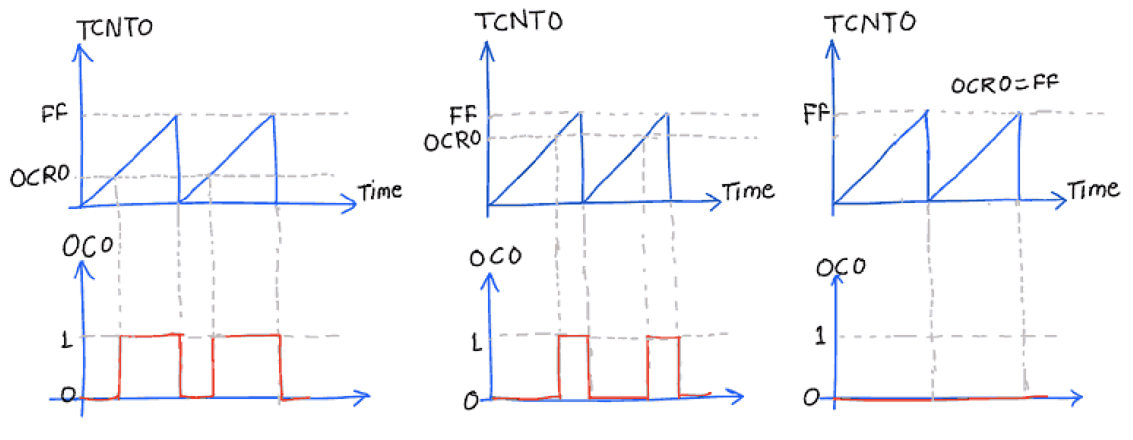
Fast PWM mode

To set Fast PWM mode, we have to set WGM00: 01= 11. To generate a PWM waveform on the OC0 pin, we need to set COM01:00= 10 or 11.

COM01:00= 10 will generate Noninverting PWM output waveform and COM01:00= 11 will generate Inverting PWM output waveform



Non-Inverted Fast PWM

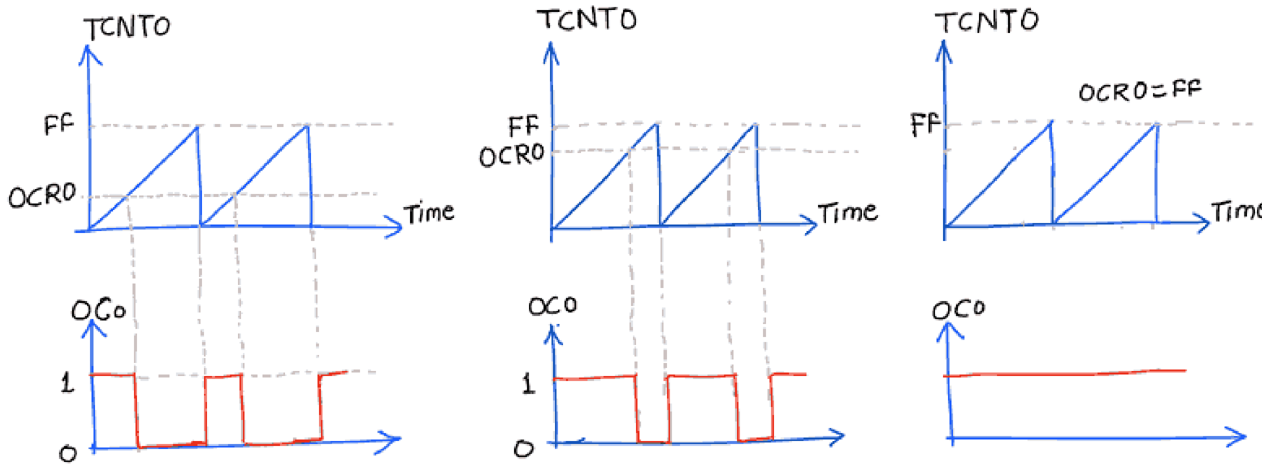


Inverted Fast PWM

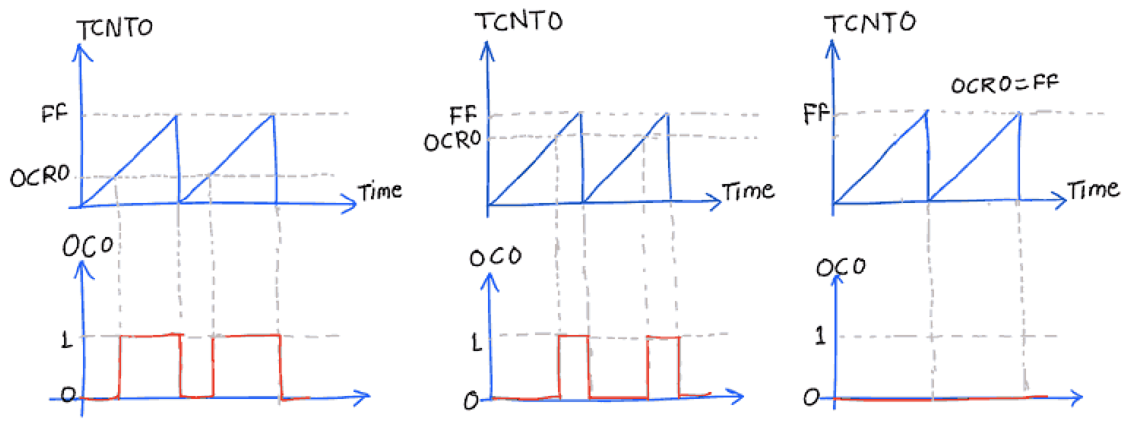
Fast PWM mode

Setting Duty cycle: we have to load value in the OCR0 register to set the duty cycle.

255 value for 100% duty cycle and 0 for 0% duty cycle. Accordingly, if we load value 127 in OCR0, the Duty cycle will be 50%.



Non-Inverted Fast PWM



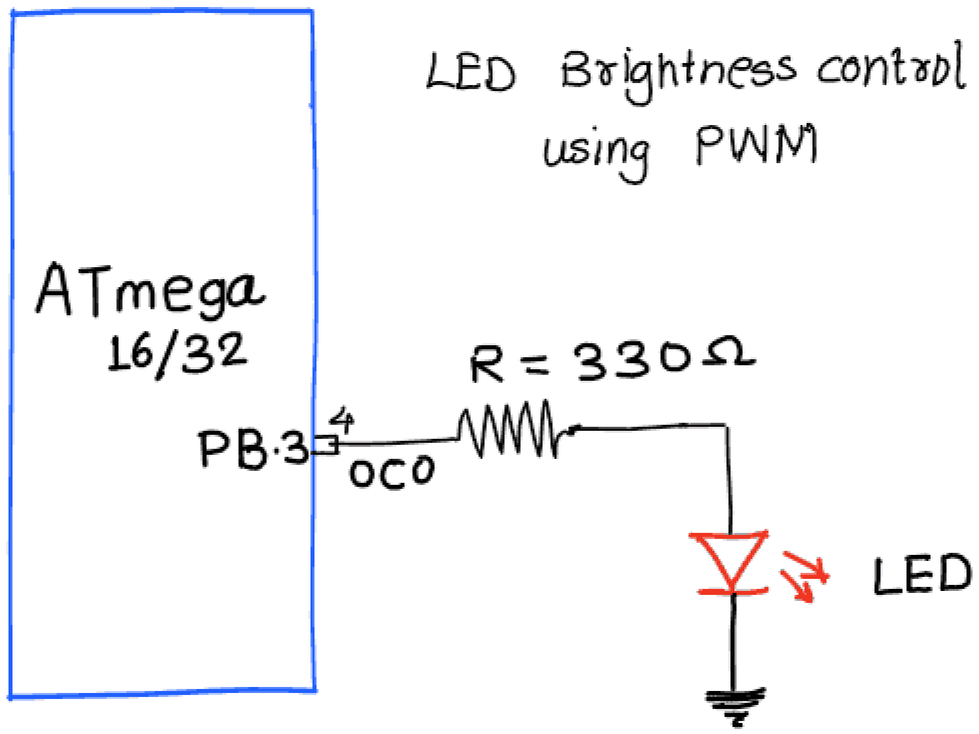
Inverted Fast PWM

Outlines

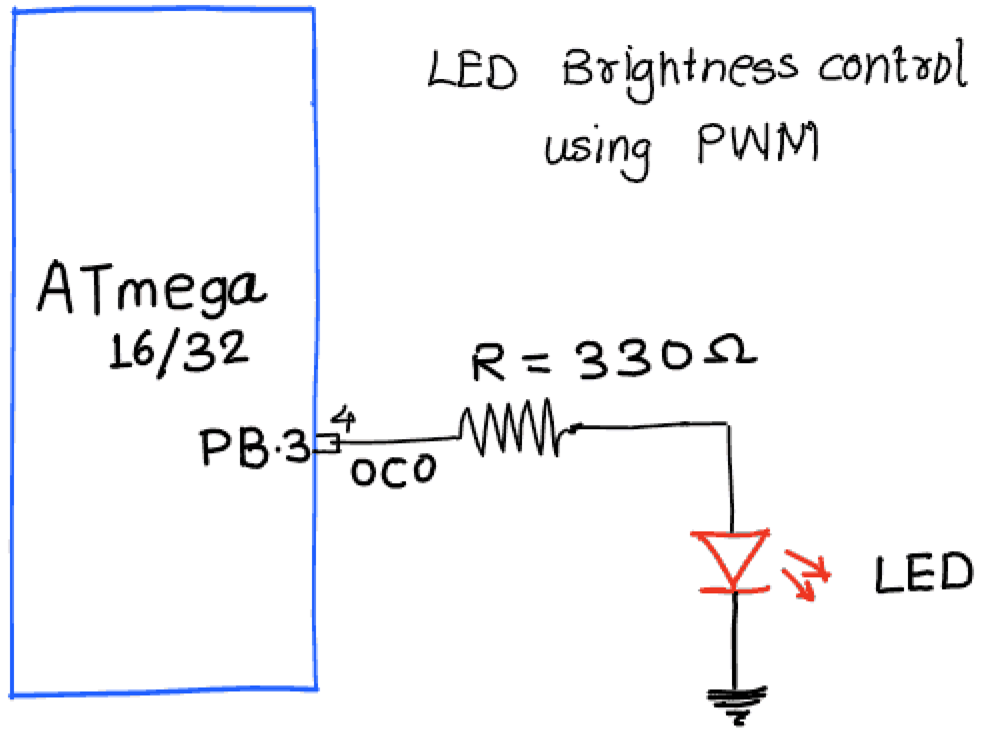
1. Introduction
2. AVR ATmega PWM
3. Configuring Timer0 for PWM generation
4. Fast PWM mode
5. Example
6. External Hardware Interrupts
7. External Hardware Interrupts:Example

Example

Control LED brightness using Fast PWM.



7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

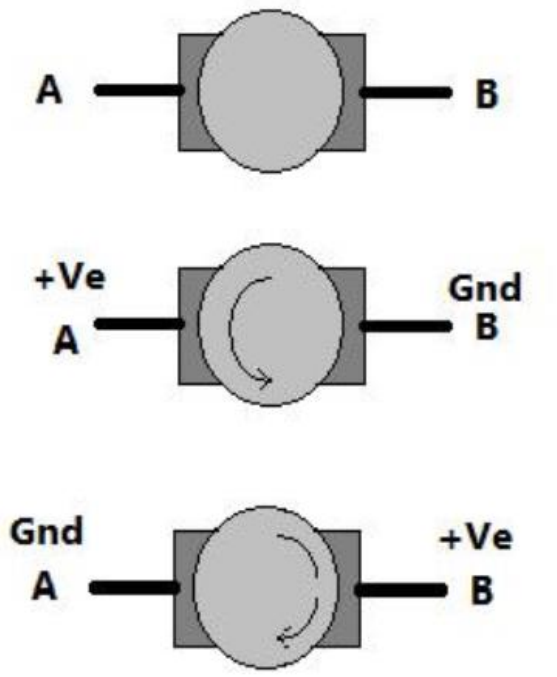


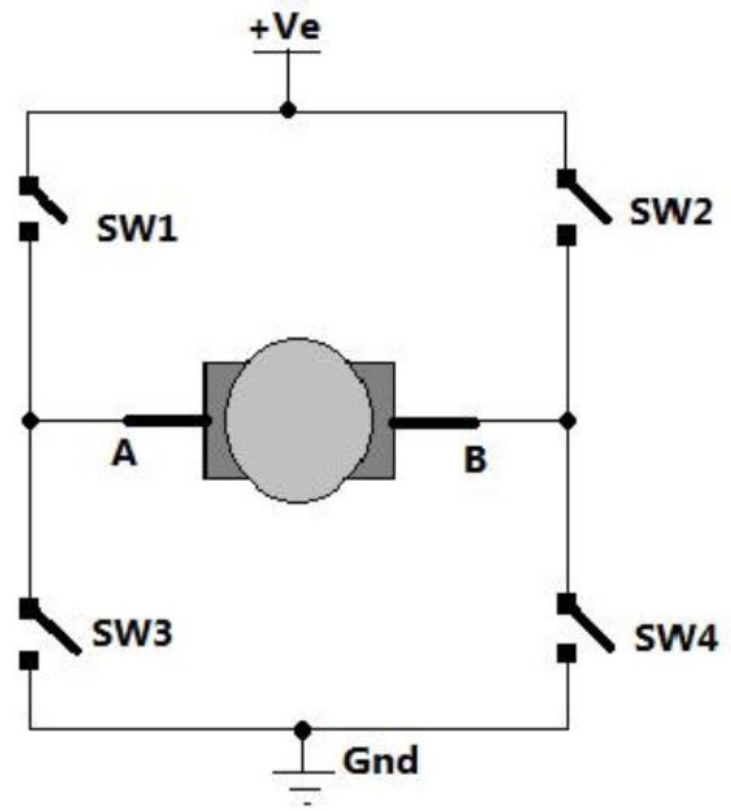
```
void PWM_init() {
/*set fast PWM mode with non-inverted output*/
TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) |
(1<<CS00);
DDRB|=(1<<PB3); /*set OC0 pin as output*/
 }

int main () {
unsigned char duty;
PWM_init();
while (1) {
for(duty=0; duty<255; duty++) {
      OCR0=duty;      /*increase the LED light intensity*/
       delay_ms(8);  }

for(duty=255; duty>1; duty--) {
 OCR0=duty; /*decrease the LED light intensity*/
_delay_ms(8); }


 }}
```

Example

Motor Drivers

H-Bridge



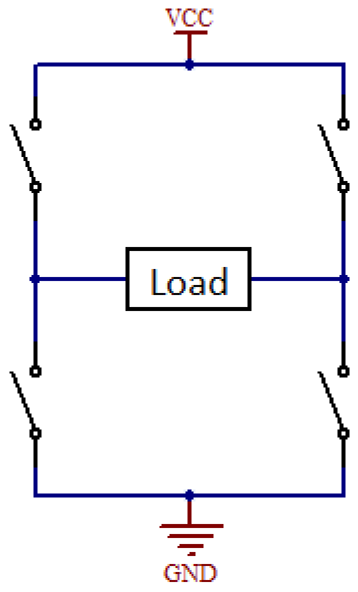


Example

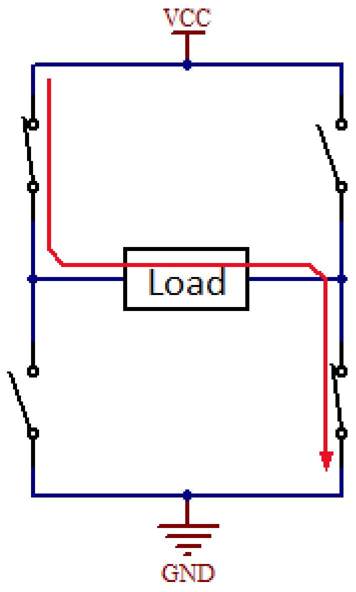
Motor Drivers

H-Bridge

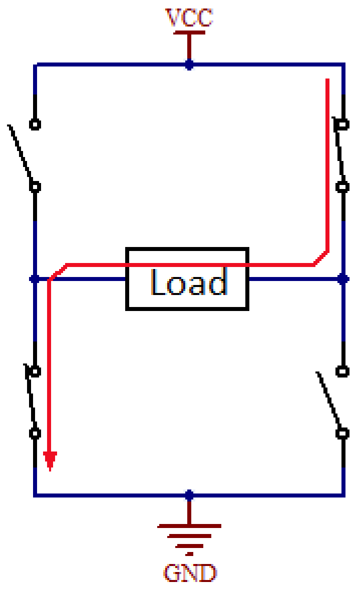
H bridge topology



Connecting the load in one direction



Connecting the load in the other direction



Outlines

External Hardware Interrupts

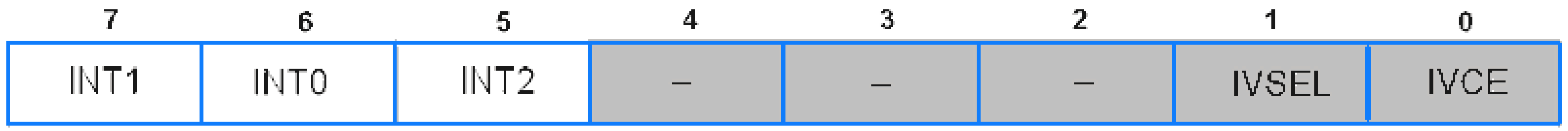
AVR ATmega16/ATmega32 has three external hardware interrupts on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively.

Upon activation of these interrupts, the ATmega controller gets interrupted in whatever task it is doing and jumps to perform the interrupt service routine.

External interrupts can be level-triggered or edge-triggered. We can program this triggering. INT0 and INT1 can be level-triggered and edge-triggered whereas INT2 can be only edge-triggered.

External Hardware Interrupts

We can enable/disable external interrupts by the **GICR register.**



7	6	5	4	3	2	1	0
INT1	INT0	INT2	–	–	–	IVSEL	IVCE

- **Bit 7 – INT1: External Interrupt Request 1 Enable**
  0: Disable external interrupt
  1: Enable external interrupt
- **Bit 6 – INT0: External Interrupt Request 0 Enable**
  0: Disable external interrupt
  1: Enable external interrupt
- **Bit 5 – INT2: External Interrupt Request 2 Enable**
  0: Disable external interrupt
  1: Enable external interrupt

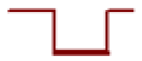
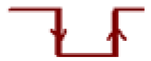
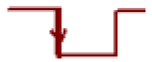
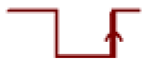
External Hardware Interrupts

MCU Control Register (MCUCR)

To define a level trigger or edge trigger on external INT0 and INT1 pins MCUCR register is used.

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC01, ISC00 (Interrupt Sense Control bits)
These bits define the level or edge that triggers the INT0 pin.

ISC01	ISC00		Description
0	0		The low level on the INT0 pin generates an interrupt request.
0	1		Any logical change on the INT0 pin generates an interrupt request.
1	0		The falling edge on the INT0 pin generates an interrupt request.
1	1		The rising edge on the INT0 pin generates an interrupt request.

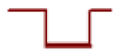
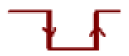
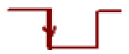
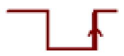
External Hardware Interrupts

MCU Control Register (MCUCR)

To define a level trigger or edge trigger on external INT0 and INT1 pins MCUCR register is used.

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC11, ISC10 (Interrupt Sense Control bits)
These bits define the level or edge that triggers the INT1 pin.

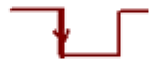
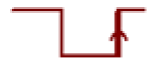
ISC01	ISC00		Description
0	0		The low level on the INT1 pin generates an interrupt request.
0	1		Any logical change on the INT1 pin generates an interrupt request.
1	0		The falling edge on the INT1 pin generates an interrupt request.
1	1		The rising edge on the INT1 pin generates an interrupt request.

External Hardware Interrupts

MCU Control and Status Register (MCUCSR)

To define the INT2 interrupt activity, bit 6 of MCUCSR is used.

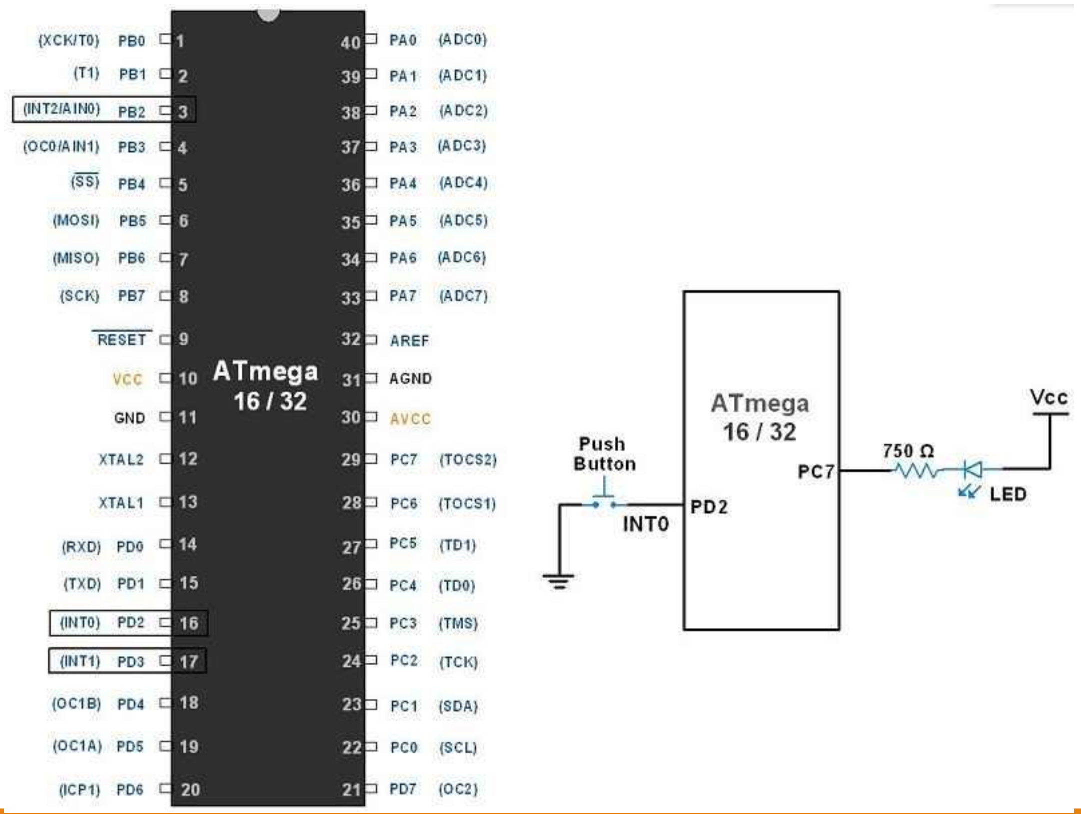
ISC2 bit defines the INT2 interrupt triggering.

ISC2		Description
0		The falling edge on the INT2 pin generates an interrupt request.
1		The rising edge on the INT2 pin generates an interrupt request.

Outlines

External Hardware Interrupts:Example

Toggle the LED connected on PORTC using external interrupt INT0 (PORTD 2).



External Hardware Interrupts:Example

```
#include <mega16.h>
#include <delay.h>

/*Interrupt Service Routine for INT0*/
 ISR(INT0_vect) {
PORTC=~PORTC; /* Toggle PORTC */
delay_ms(50); /* Software debouncing control delay */
 }

int main () {

DDRC=0xFF; /* Make PORTC as output PORT*/
PORTC=0; DDRD=0; /* PORTD as input */
PORTD=0xFF; /* Make pull up high */

GICR = 1<<INT0; /* Enable INT0*/

MCUCR = 1<<ISC01 | 1<<ISC00; /* Trigger INT0 on rising edge */

sei(); /* Enable Global Interrupt */

while(1);
 }
```

Thanks