ICT5307 : Embedded System Design

Lecture 10
Input Capture, Duty Cycle and Variable PWM

Professor S.M. Lutful Kabir

BUET

TCCR0 Register square Wave Generation

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

	WGM00	WGM01
NORMAL	0	0
CTC	0	1
-------	1	0
-------	1	1

	COM01	COM00
NORMAL	0	0
Toggles…	0	1
CLEARS….	1	0
SETS….	1	1

TCCR0 Register for PWM Generation

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

	WGM00	WGM01
-----	0	0
-----	0	1
P.C. PWM	1	0
FAST PWM	1	1

	COM01	COM00
NORMAL	0	0
Reserve	0	1
CLEARS….	1	0
SETS….	1	1

TCCR1A & TCCR1B (Timer Counter Control Registers)

TCCR1A Register

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

TCCR1B Register

ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

FOC1A, FOC1B – Related to force compare

ICNC1, ICES1 – Related to Input Capture

COM1A1, COM1A0, COM1B1, COM1B0 – Related to Waveform Generation

WGM13, WGM12, WGM11, WGM10 – Mode Selection

CS12, CS11, CS10 – Clock Selection

Modes in Timer 1

Mode	WGM13	WGM12	WGM11	WGM10	Timer/Counter Mode of Operation	Top	Update of OCR1x	TOV1 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	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

What is Input Capture?

- In input capture function is widely used for many applications. Among them, a few are
  - Recording the arrival time of an event,
  - Pulse width measurement and
  - Period measurement.
- In ATmega32, Timer1 can be used as the Input Capture to detect and measure the events happening outside the chip.
- Upon detection of an event, the TCNT1 register value is loaded into ICR1 register.
- And the ICF1 flag is set

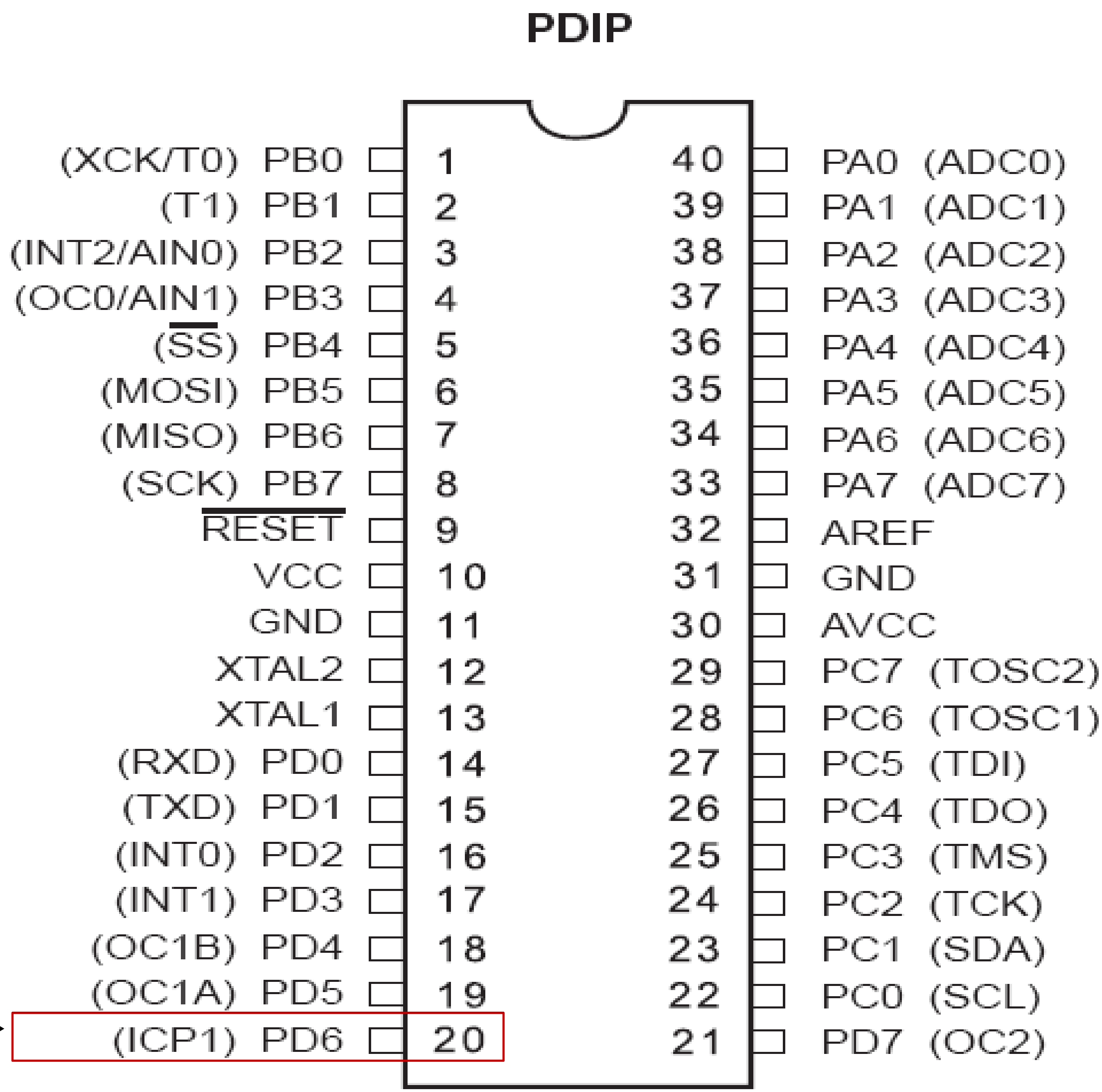
TIMSK Register

OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

TIFR Register

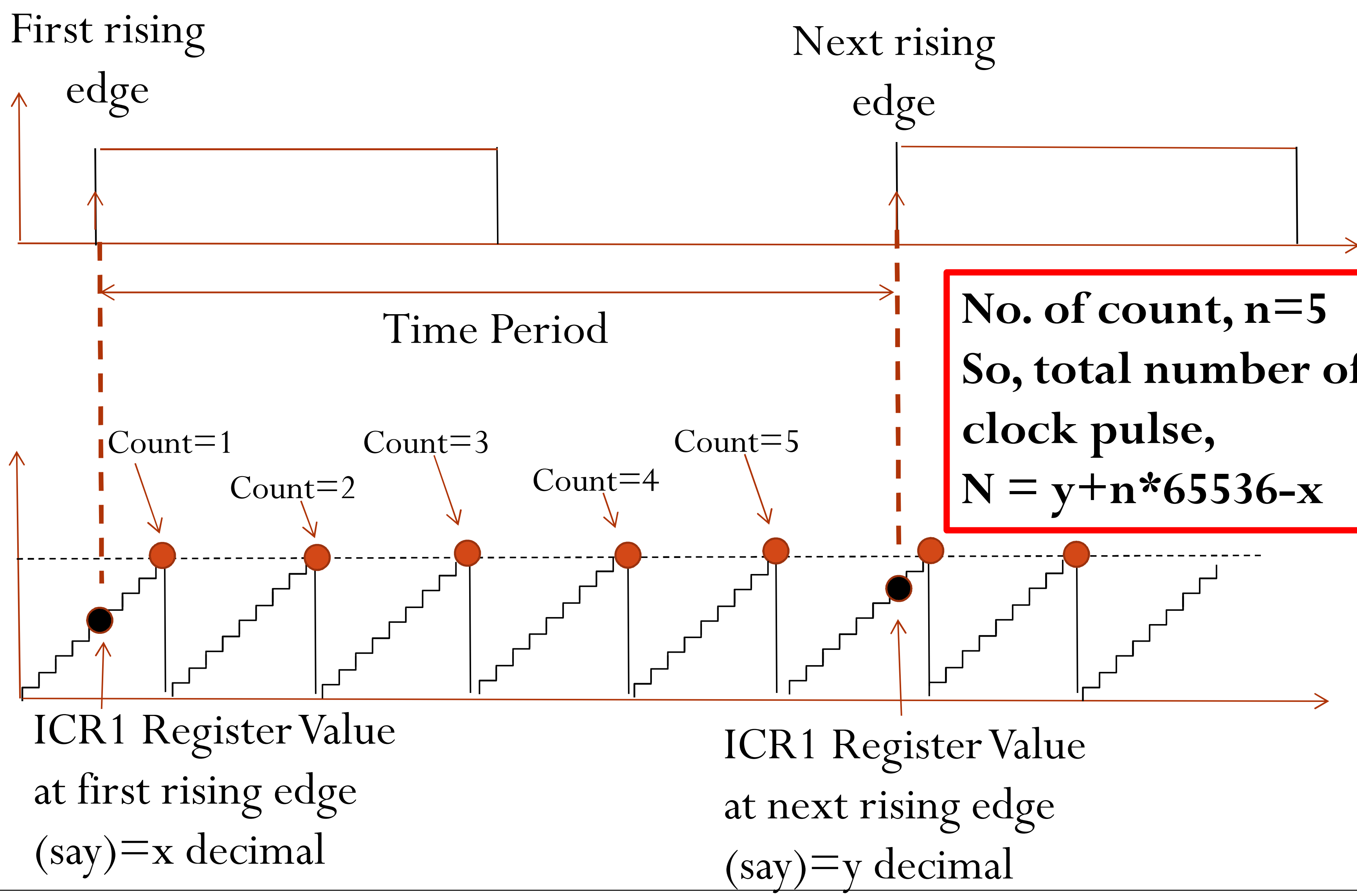
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

ICP1 Pin



Input Capture

Principle of Calculating Time Period of a Square Wave



No. of count, n=5

So, total number of clock pulse,

N = y+n*65536-x

Bits in TCCR1x Registers

TCCR1A Register

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

TCCR1B Register

ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

ICNC1 – Input Capture Noise Canceller , normally not set

ICES1 – Input Select Edge Select, 0-falling and 1- rising edge

Steps for the Calculation of the Period of a Square Wave

- **Step#1**: Enable both TOV1 and TICIE1 bit of TIMSK register.
- **Step#2**: In the ISR for Timer overflow increment a count, indicating number of overflow, n between two successive input edges.
- **Step#3**: In the ISR for input capture, store the number in ICR1 register as a present value, y.
- **Step#4**: Subtract this present value, y from the previous value, x stored in the previous input capture and add with number of ticks in the complete cycle of overflow. The number, N=y+n*256-x.
- **Step#5**: Calculate the time for the number of total ticks (N). This time is the time period.
- **Step#6**: Replace the previous value with the present value and reset the value of the count.

The Code (1st Part)

```
#include <mega32.h>
#include <stdlib.h>
#include <alcd.h>

long int new_value=0;
long int prv_value=0;
char disp[16];
int count=0;
float Tp=0;
float Ttick=1/(2000*1e3);
float Tov=65535*Ttick;
```

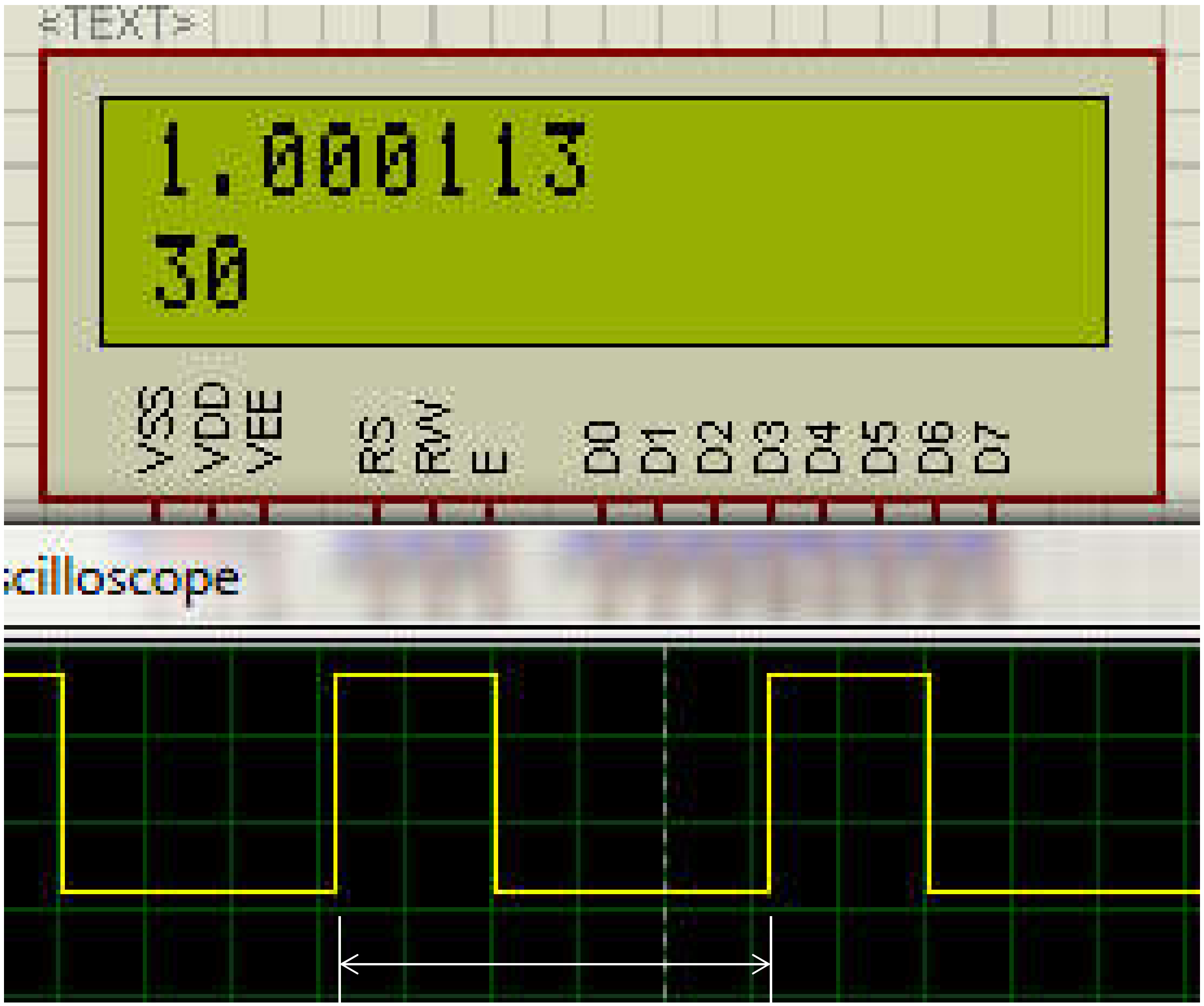
The Code (2nd Part)

```
// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
 count++;
}
// Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
new_value=ICR1H*256+ICR1L;
Tp=count*Tov+(new_value-prv_value)*Tpulse;
prv_value=new_value;
ftoa(Tp,6,disp);
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts(disp);
itoa(count,disp);
lcd_gotoxy(0,1);
lcd_puts(disp);
count=0;
}
```

The Code (3rd Part)

```
void main(void)
{
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (1<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (1<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (1<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (1<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1)
      {
      }
}
```

The Input in Virtual Oscilloscope and Output in LCD Display



5 div, 1 div = 0.2 s, Tp = 1s

Summary on Operation of Timer1

Can be obtained using Timer0 and Timer2 also

1. No auto output is required

2. Auto output in OC1A or OC1B pin, (Square Wave)
Timer Register Resets : (2a.Normal) or (2b. CTC)

3. Auto output in OC1A or OC1B pin,
(PWM Wave of discrete frequency)
Timer Register Resets : Normal

4. Auto output in OC1A or OC1B pin,
(PWM Wave of Variable frequency)
Timer Register Resets : Normal

Output

5. Input Capture in ICP1 pin, (Time Period)

6. Input Capture in ICP1 pin, (Time Period & DC)

Input

Different Cases (Discussed so far)

- Case #1- WGM: NORMAL & COM: Normal
- Case #2- 2a. WGM: NORMAL & COM: OC1A or OC1B

  toggles at Compare Match

  WGM: CTC & COM: OC1A or OC1B

  toggles at Compare Match
- Case #3. WGM: Fast PWM & COM: OC1A or OC1B

  output pin sets at Compare Match and clears at

  TOP or opposite
- Case #5. WGM: Normal & COM: Normal

  Input Capture & OVERFLOW Interrupt is enable and

  Edge mode is selected

TCCR1A & TCCR1B (Timer Counter Control Registers)

TCCR1A Register

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

TCCR1B Register

ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

FOC1A, FOC1B – Related to force compare

ICNC1, ICES1 – Related to Input Capture

COM1A1, COM1A0, COM1B1, COM1B0

– Related to Waveform Generation

WGM13, WGM12, WGM11, WGM10 – Mode Selection

CS12, CS11, CS10 – Clock Selection

Modes in Timer 1

Mode	WGM13	WGM12	WGM11	WGM10	Timer/Counter Mode of Operation	Top	Update of OCR1x	TOV1 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	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

What is Input Capture?

- In input capture function is widely used for many applications. Among them, a few are
  - Recording the arrival time of an event,
  - Pulse width measurement and
  - Period measurement.
- In ATmega32, Timer1 can be used as the Input Capture to detect and measure the events happening outside the chip.
- Upon detection of an event, the TCNT1 register value is loaded into ICR1 register.
- And the ICF1 flag is set

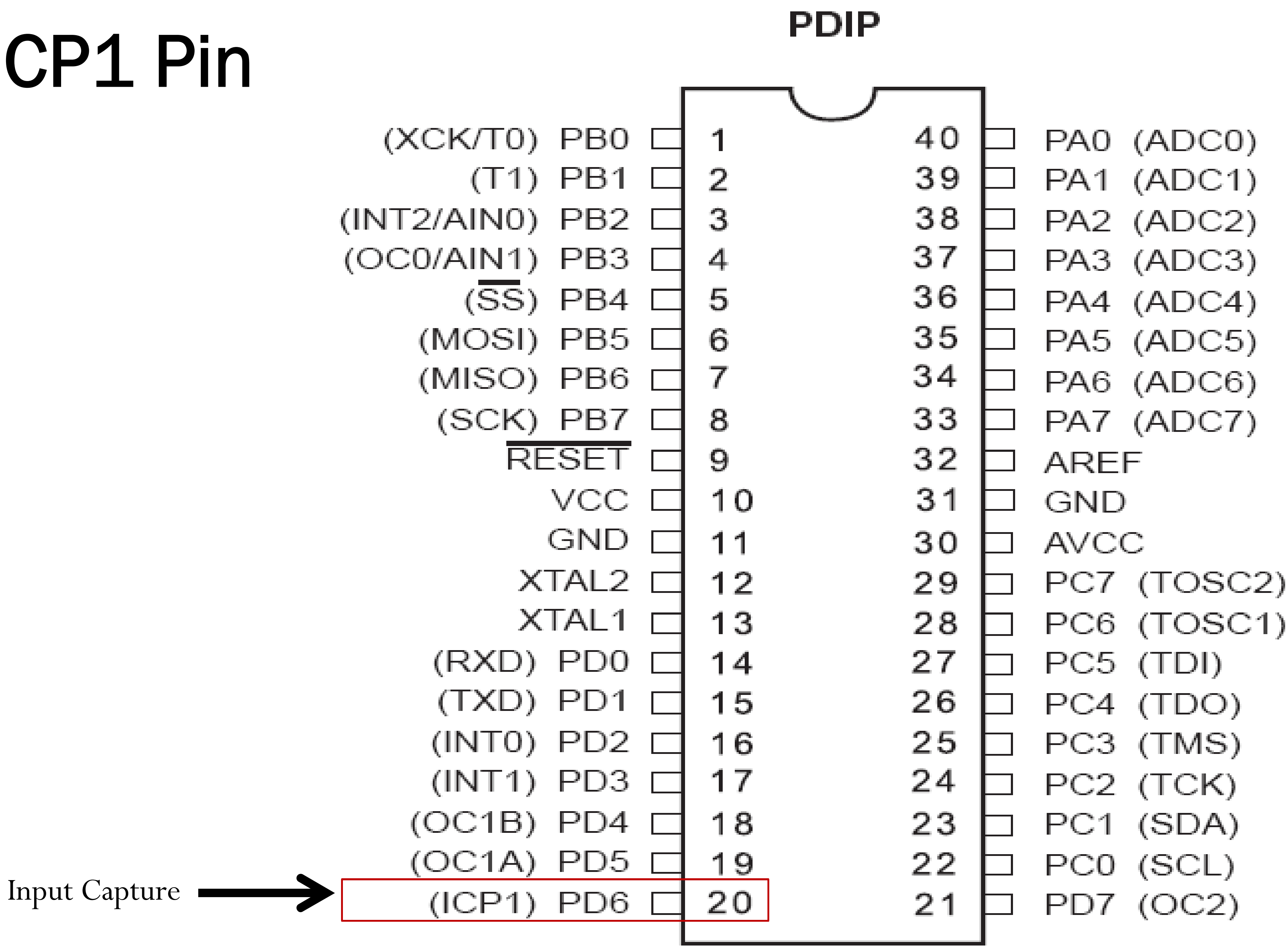
TIMSK Register

OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

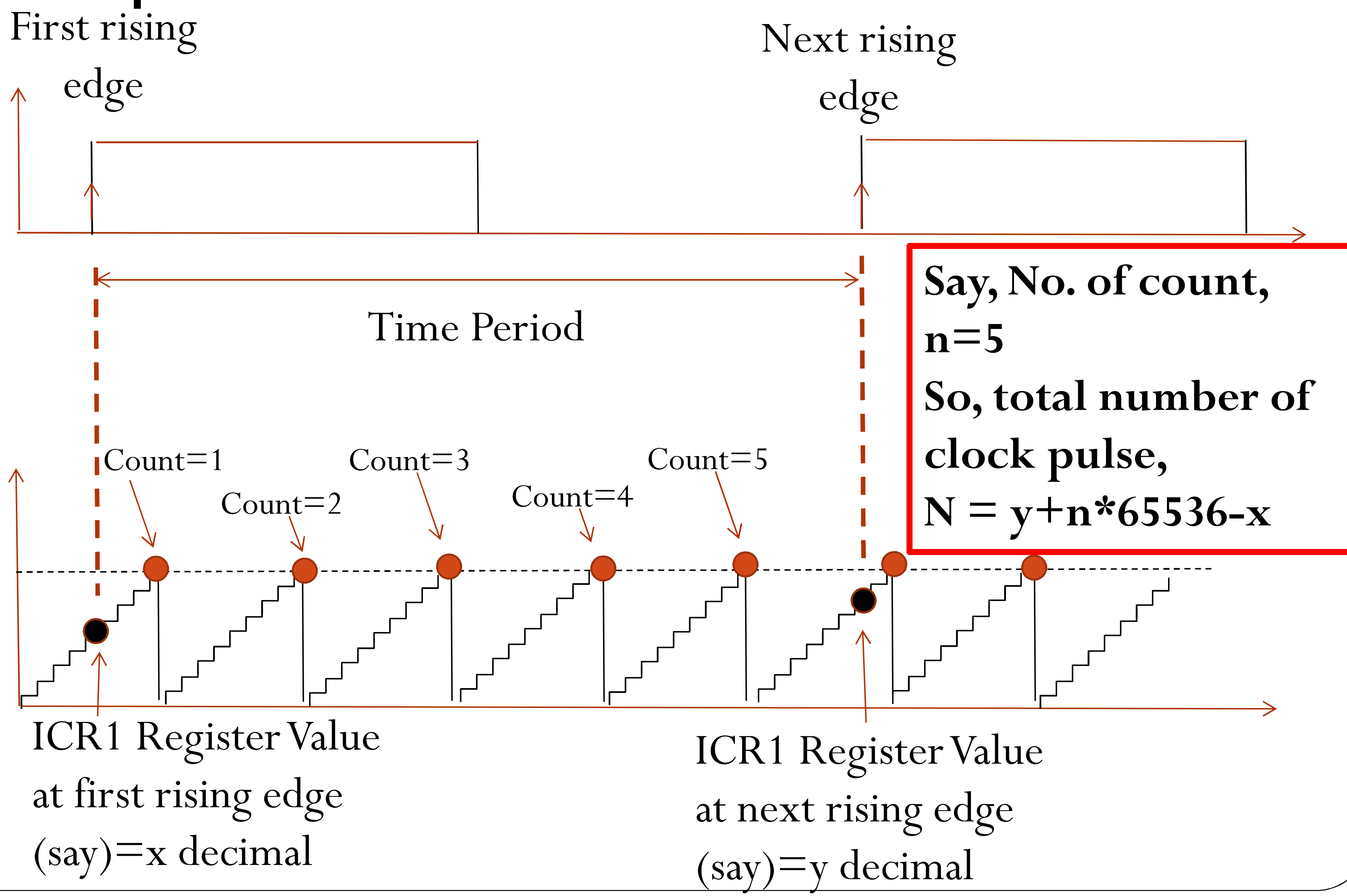
TIFR Register

OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

ICP1 Pin


PDIP
(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP1) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)
Input Capture

Principle of Calculating Time Period of a Square Wave



Bits in TCCR1x Registers

TCCR1A Register

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

TCCR1B Register

ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

ICNC1 – Input Capture Noise Canceller , normally not set

ICES1 – Input Select Edge Select, 0-falling and 1- rising edge

Steps for the Calculation of the Period of a Square Wave

- **Step#1**: Enable both TOV1 and TICIE1 bit of TIMSK register.
- **Step#2**: In the ISR for Timer overflow increment a count, indicating number of overflow, n between two successive input edges.
- **Step#3**: In the ISR for input capture, store the number in ICR1 register as a present value, y.
- **Step#4**: Subtract this present value, y from the previous value, x stored in the previous input capture and add with number of ticks in the complete cycle of overflow. The number, N=y+n*256-x.
- **Step#5**: Calculate the time for the number of total ticks (N). This time is the time period.
- **Step#6**: Replace the previous value with the present value and reset the value of the count.

The Code (1^{st} Part)

```
#include <mega32.h>
#include <stdlib.h>
#include <alcd.h>

long int new_value=0;
long int prv_value=0;
char disp[16];
int count=0;
float Tp=0;
float Ttick=1/(2000*1e3);
float Tov=65535*Ttick;
```

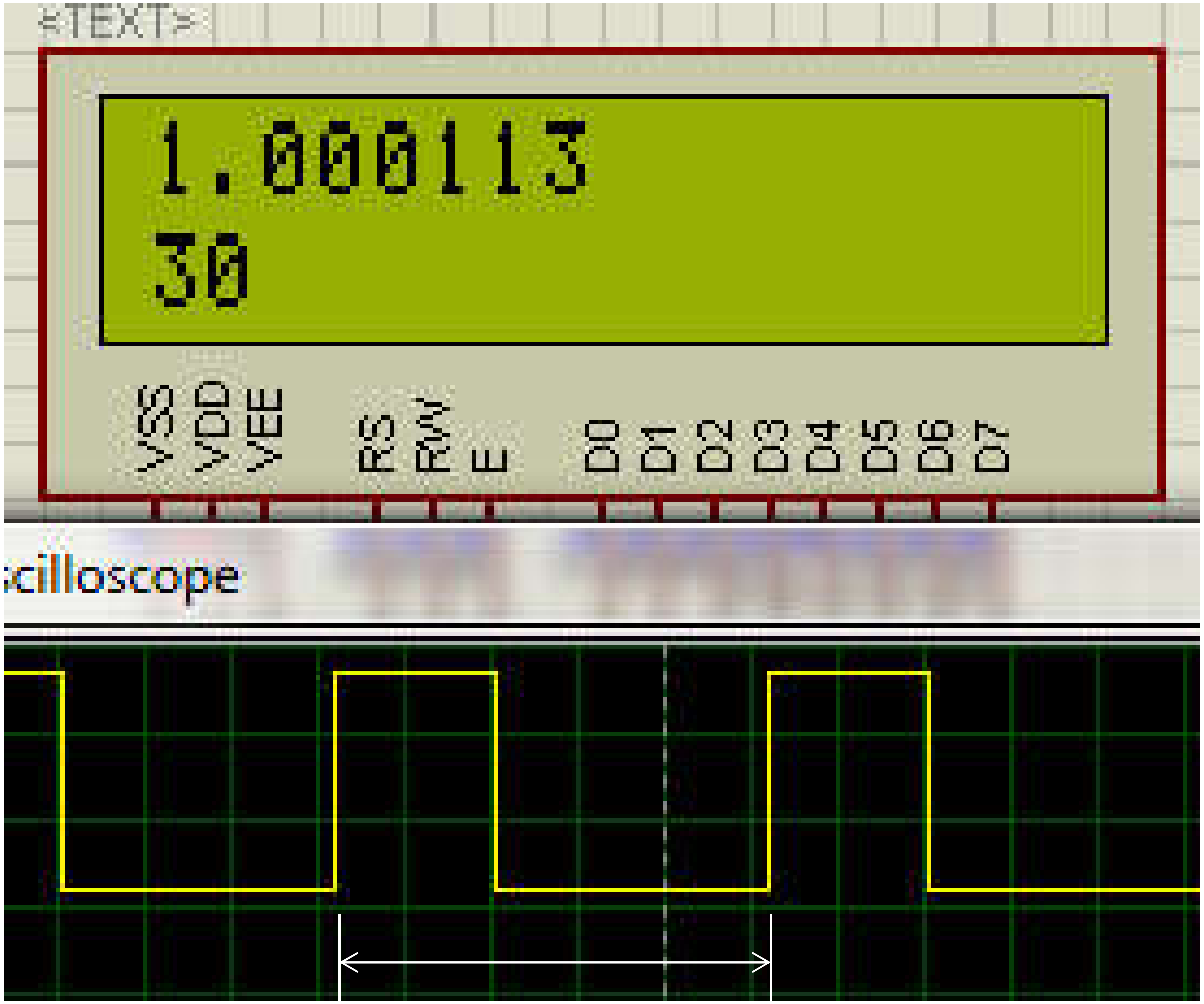
The Code (2nd Part)

```
//Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
 count++;
}
//Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
new_value=ICR1H*256+ICR1L;
Tp=count*Tov+(new_value-prv_value)*Tpulse;
prv_value=new_value;
ftoa(Tp,6,disp);
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts(disp);
itoa(count,disp);
lcd_gotoxy(0,1);
lcd_puts(disp);
count=0;
}
```

The Code (3rd Part)

```
void main(void)
{
TCCR1A=(0<<COM1A1) |
(0<<COM1A0) | (0<<COM1B1) |
(0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) |
(1<<ICES1) | (0<<WGM13) |
(0<<WGM12) | (0<<CS12) |
(1<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer(s)/Counter(s) Interrupt(s)
initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2)
| (1<<TICIE1) | (0<<OCIE1A) |
(0<<OCIE1B) | (1<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1)
      {
      }
}
```

The Input in Virtual Oscilloscope and Output in LCD Display

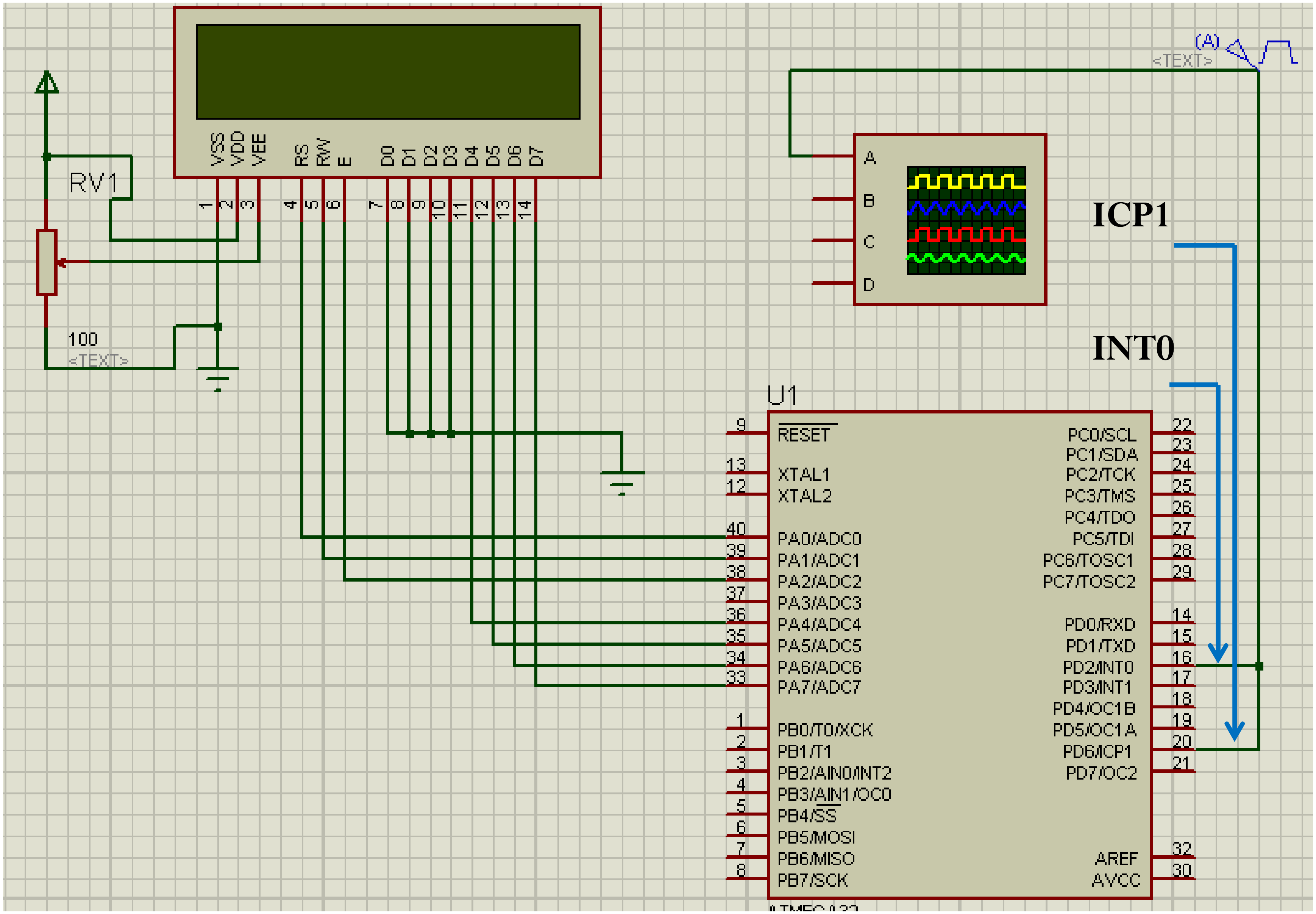


5 div, 1 div = 0.2 s, Tp = 1s

How Duty Cycle is Measured Using Input Capturing ?

- Period is measured by input capturing as we explained earlier.
- The pulse width is measured by using interrupt with sensing at "triggering at both edge"
- The time measurement technique is done by using the values stored in Input Capture Register and the value of the count.
- The duty cycle is then obtained from T_{ON}/T_{period}.

The Simulation



The Code (First Part)

```
#include <mega32.h>
#include <stdlib.h>
#include <alcd.h>
#include <string.h>

long int new_value_TP=0;
long int prv_value_TP=0;
long int new_value_DC=0;
long int prv_value_DC=0;
char disp1[16];
char disp2[16];
int count_TP=0;
int count_DC=0;
float TP=0;
float TDC=0;
float freq=0;
float duty=0;
float Ttick=1/(16*1e6);
float Tov=65535/(16*1e6);
int edge=1;
```

The Code (2^{nd} Part)

```
//Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    count_TP++;
    count_DC++;
}
```

```
//Timer1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
new_value_TP=ICR1H*256+ICR1L;
TP=count_TP*Tov+(new_value_TP-prv_value_TP)*Ttick;
freq=1/TP;
prv_value_TP=new_value_TP;
count_TP=0;
}
```

The Code (3rd Part)

```
interrupt [EXT_INT0] void
ext_int0_isr(void)
{
    if (edge==1)
    {
prv_value_DC=TCNT1H*
256+TCNT1L;
    edge=2;
    count_DC=0;
    }
    else
    {
new_value_DC=TCNT1H*2
56+TCNT1L;

TDC=count_DC*Tov+(new_
value_DC-
prv_value_DC)*Ttick;
    edge=1;
    count_DC=0;
    }
}
```

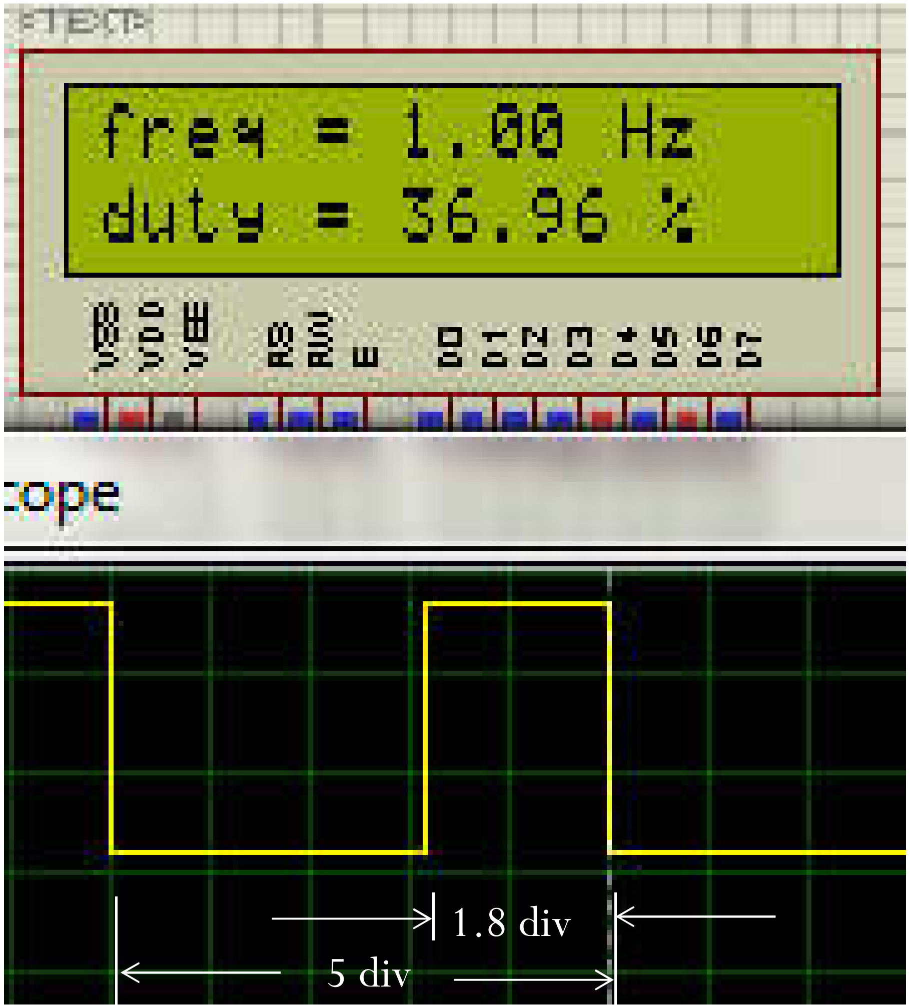
The Code (4^{th} Part)

```
void main(void)
{
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (1<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (1<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (1<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
GICR|=(0<<INT1) | (1<<INT0) | (0<<INT2);
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (1<<ISC00);
MCUCSR=(0<<ISC2);
GIFR=(0<<INTF1) | (1<<INTF0) | (0<<INTF2);
```

The Code (5th Part)

```
lcd_init(16);
// Global enable interrupts
#asm("sei")
while (1)
    {
     if (count_TP==0)
     {
       if (TDC!=0)
           duty=100-TDC*100/TP;
       ftoa(freq,2,disp1);
       strcpy(disp2,"freq(Hz) = ");
       strcat(disp2,disp1);
lcd_clear();
       lcd_gotoxy(0,0);
       lcd_puts(disp2);
       ftoa(duty,2,disp1);
       strcpy(disp2,"duty (%) = ");
       strcat(disp2,disp1);
       lcd_gotoxy(0,1);
       lcd_puts(disp2);
     }
    }
}
```

Input in Virtual Oscilloscope and Output in LCD Display



1 div=0.2s

So, T = 5div =1s.

f=1Hz

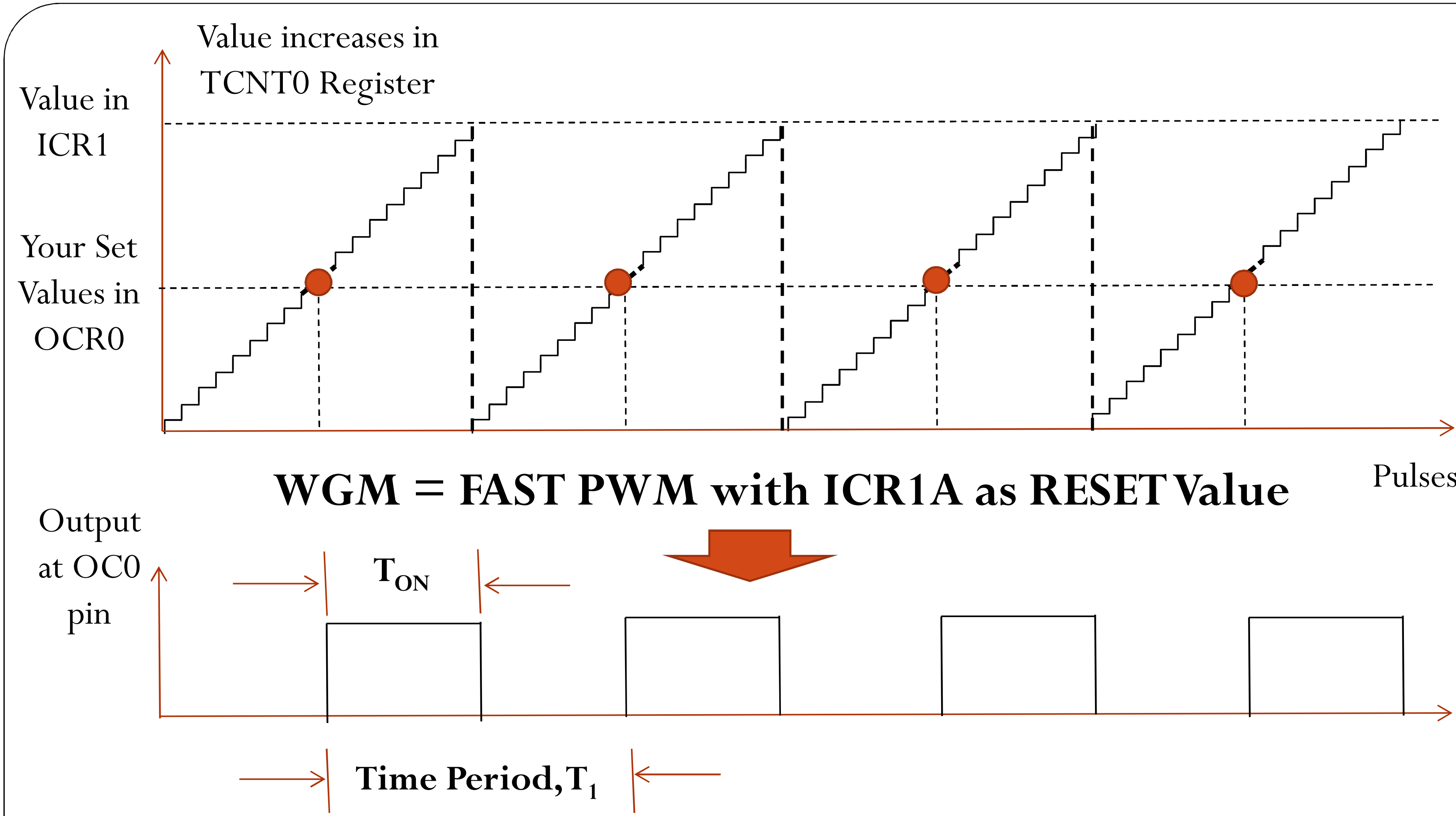
Duty cycle

=1.8*100/5

=36%

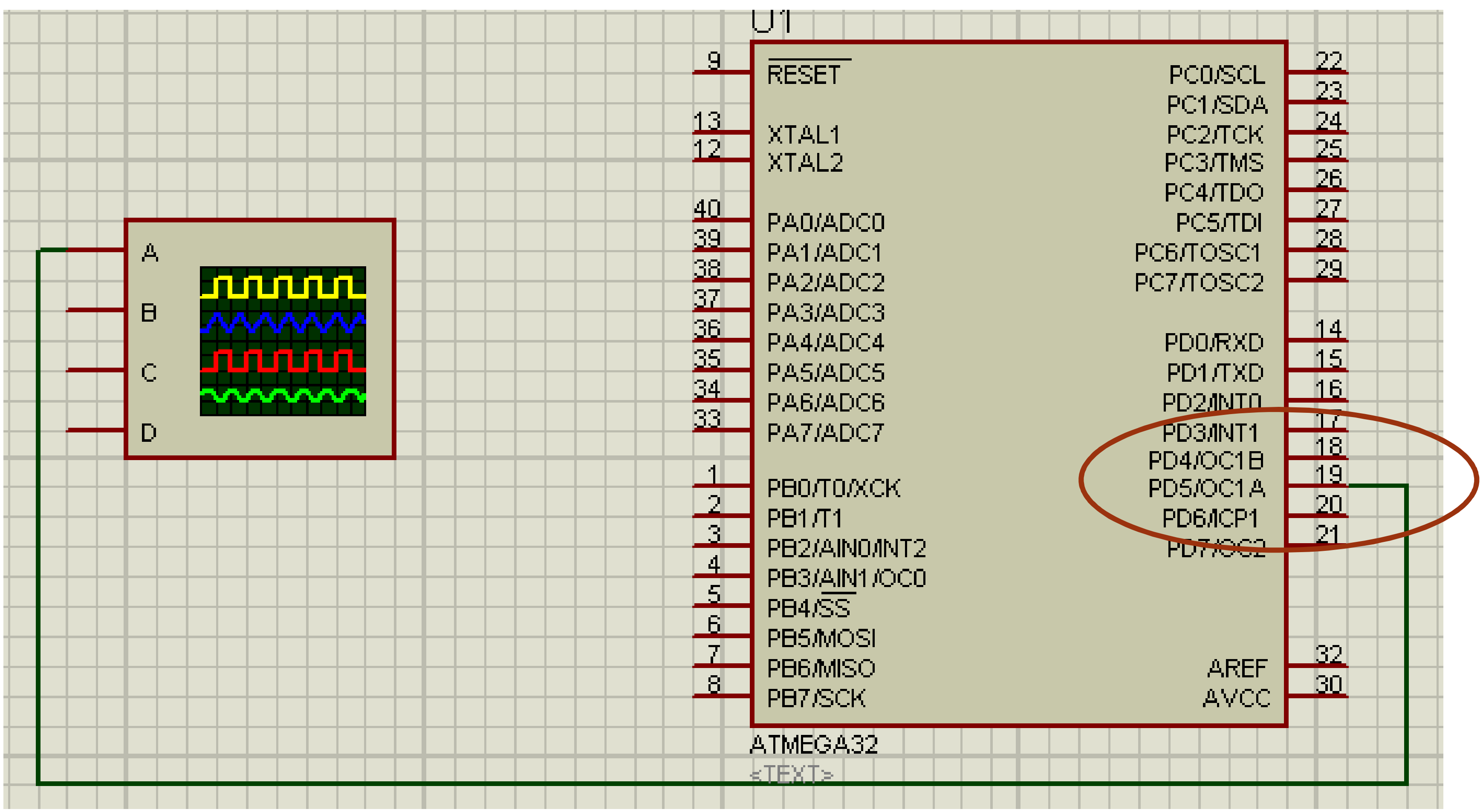
Generation of PWM having Variable Frequency

- Previously we discussed about fixed frequency PWM
- It was fixed frequency because the Timer register Resets at TOP value
- In order to obtain variable frequency PWM we need to have a choice of WGM where Timer Register Resents at ICR1 register value.
- Let us follow the diagram of the next slide.



12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Connection at OC1A (PD.5) pin



Calculation

- Fosc=16MHz
- Ttick=(1/16) uS
- OCR1=1FFF h = 8191 d
- ICR1 = AFFF h = 45055 d
- Tp = ICR1 * Ttick

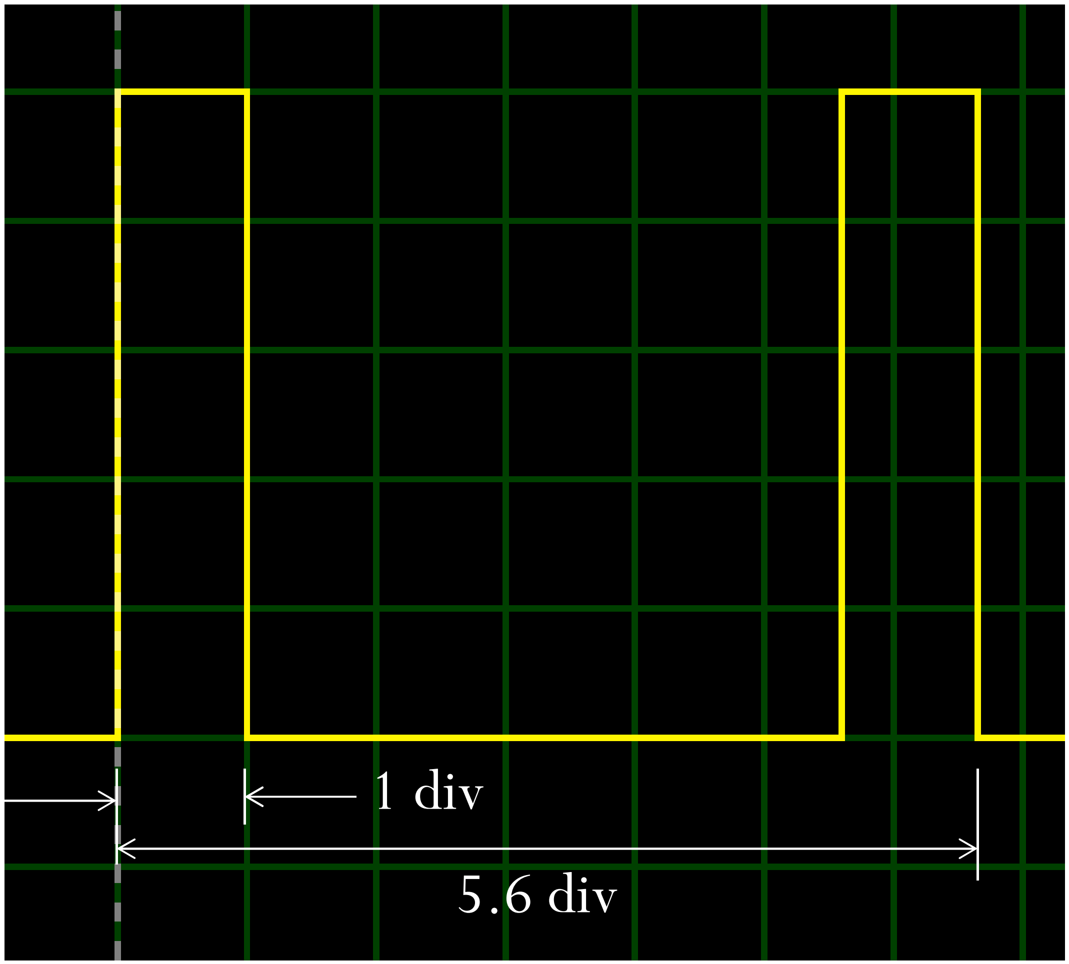
  =45055 * (1/16) * (1e-6)=0.00282s=2.82mS
- T_{ON}=OCR1A * Ttick

  =8191* (1/16) * (1e-6)=0.000511s=0.511mS

The Code

```
#include <mega32.h>
void main(void)
{
DDRD.5=1;
TCCR1A=(1<<COM1A1) |
(0<<COM1A0) |
(0<<COM1B1) |
(0<<COM1B0) |
(1<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) |
(0<<ICES1) |
(1<<WGM13) |
(1<<WGM12) | (0<<CS12)
| (0<<CS11) | (1<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0xAF;
ICR1L=0xFF;
OCR1AH=0x1F;
OCR1AL=0xFF;
OCR1BH=0x00;
OCR1BL=0x00;
while (1)
    {
    }
}
```

Output in Virtual Oscilloscope



1 div =0.5 s

So,

TON=0.5 s

and

Tp=2.8 s

Thanks