Experiment-3

Aim: Hands-on experimentation of ATMega32 Timer Programming in C.

Objectives: After successfully completion of this experiment students will be able to,

- Use C language for ATMega32 microcontroller programming on AVRStudio.
- Experiment with Timer of ATMega32 on ATMega32 AVR Development Board.

Equipment required:

- Windows7 or later based host computer
- ATMega32 Development board
- USBasp Programmer
- Jumper Wires
- LED

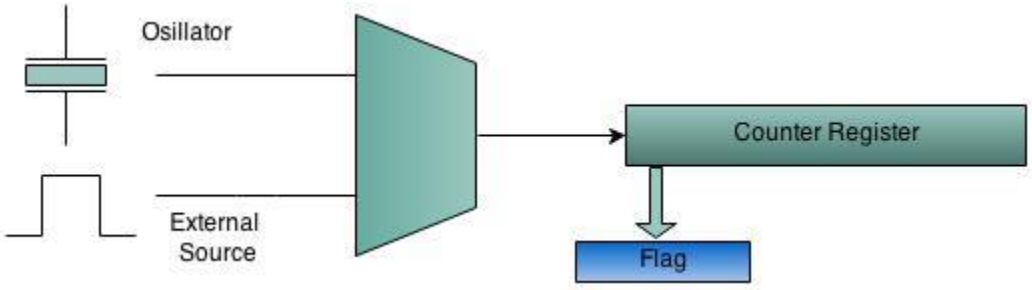
Software required:

- AVR Studio7 installation setup
- USBasp driver installation setup

Theory:

AVR Timer programming Basics

Timers come in handy when you want to set some time interval like your alarm. This can be very precise to a few microseconds.



Timers/Counters are an essential part of any modern MCU. Remember it is the same hardware unit inside the MCU that is used either as Timers or Counter. Timers/counters are an independent unit inside a microcontroller. They basically run independently of what task the CPU is performing. Hence they come in very handy, and are primarily used for the following:

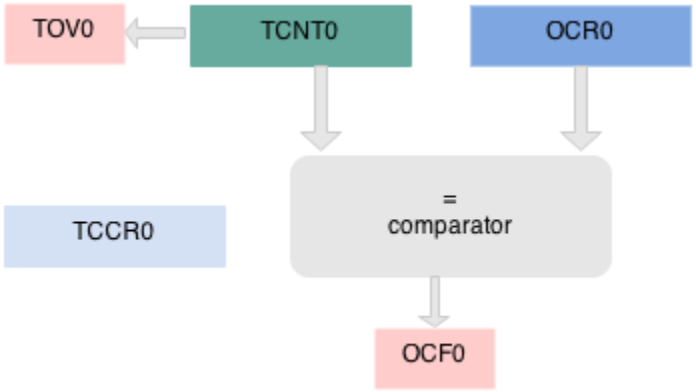
- **Internal Timer**: As an internal timer the unit ticks on the oscillator frequency. The oscillator frequency can be directly fed to the timer or it can be pre-scaled.

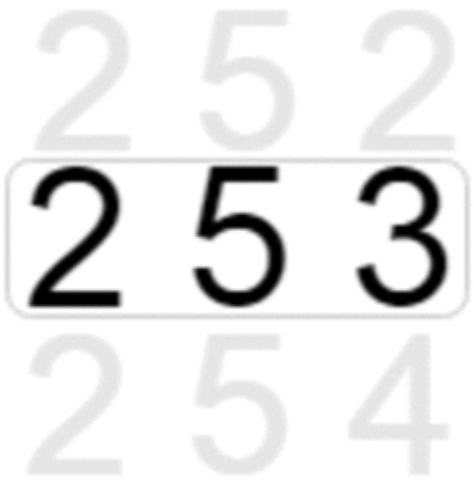
In this mode it used to generate precise delays. Or as a precise time counting machine.

- **External Counter**: In this mode the unit is used to count events on a specific external pin on a MCU.
- **Pulse width Modulation(PWM) Generator**: PWM is used in speed control of motors and various other applications.

Atmega32 has 3 timer units, timer 0, timer 1 and timer 2 respectively.

Timer 0 Basics





Timer 0 is a 8 bit timer. It basically means it can count from 0 to 2^8 255. The operation of timer 0 is straight forward. The TCNT0 register hold the timer Count and it is incremented on every timer **"tick"**. If the timer is turned on it ticks from 0 to 255 and overflows. If it does so, a Timer OverFlow Flag(TOV) is set.

You can as well load a count value in TCNT0 and start the timer from a specific count. Another interesting feature is that a value can be set in the Output Compare Register (OCR0), and whenever TCNT0 reaches that value, the Output Compare Flag (OCF0) flag is Set.

Timer/Counter 0

TCNT0							
D7	D6	D5	D4	D3	D2	D1	D0

The configuration of the Timer can be set using the TCCR0 register shown below. With this you can basically select two things:

1. The Frequency of the Clock Source with CS02, CS01, CS00 bits.
2. The mode of the timer. For the first example we will use it in normal mode where it ticks from zero to the highest value(255)

Timer Counter Control Register 0

TCCR0							
D7	D6	D5	D4	D3	D2	D1	D0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

D2	D1	D0	Clock Source
CS02	CS01	CS00	Freq
0	0	0	No Clock (Stopped)
0	0	1	Clk
0	1	0	Clk/8
0	1	1	Clk/64
1	0	0	Clk/256
1	0	1	Clk/1024
1	1	0	Clk/T0-Falling edge
1	1	1	Clk/T0-Rising Edge

D6	D3	PWM
WGM00	WGM01	Mode
0	0	Normal
0	1	CTC (Clear timer on compare match)
1	0	PWM (Phase correct)
1	1	Fast PWM

The Timer/counter Interrupt Flag Register(TIFR) holds the two basic flags we need the TOV and OVF.

Timer/Counter 0 Flag Register

TIFR							
D7	D6	D5	D4	D3	D2	D1	D0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

Timer 0 Example

Toggle LED connected to PD4 every 1000msec using Timer Zero with 1024 pre-scalar in normal mode.

What is the Max delay Timer 0 overflow generates? Okay, lets calculate. The Explore Ultra AVR dev board comes with a 1MHz on board crystal and the fuse bits are set appropriately. If we use the highest pre-scalar of 1024, calculation shows it can generate a delay of 16milliseconds every time timer zero overflows.

$$Ftimer = CPU Frequency/Prescalar $$ $$Ftimer = 1MHz/1024 = 0.976KHz $$ $$Ttick = 1/ 0.976K = 1.024 ms seconds$$ $$Ttotal = 1.024m s X 255 = 261.270ms$$

Of-course 261.270ms is not enough, so the next obvious question is:

How many times should the timer overflow to generate a delay of approximately 1000msec?

$$ OverFlowCount = 1000ms/261.270ms = 3.82 ≈ 4 $$

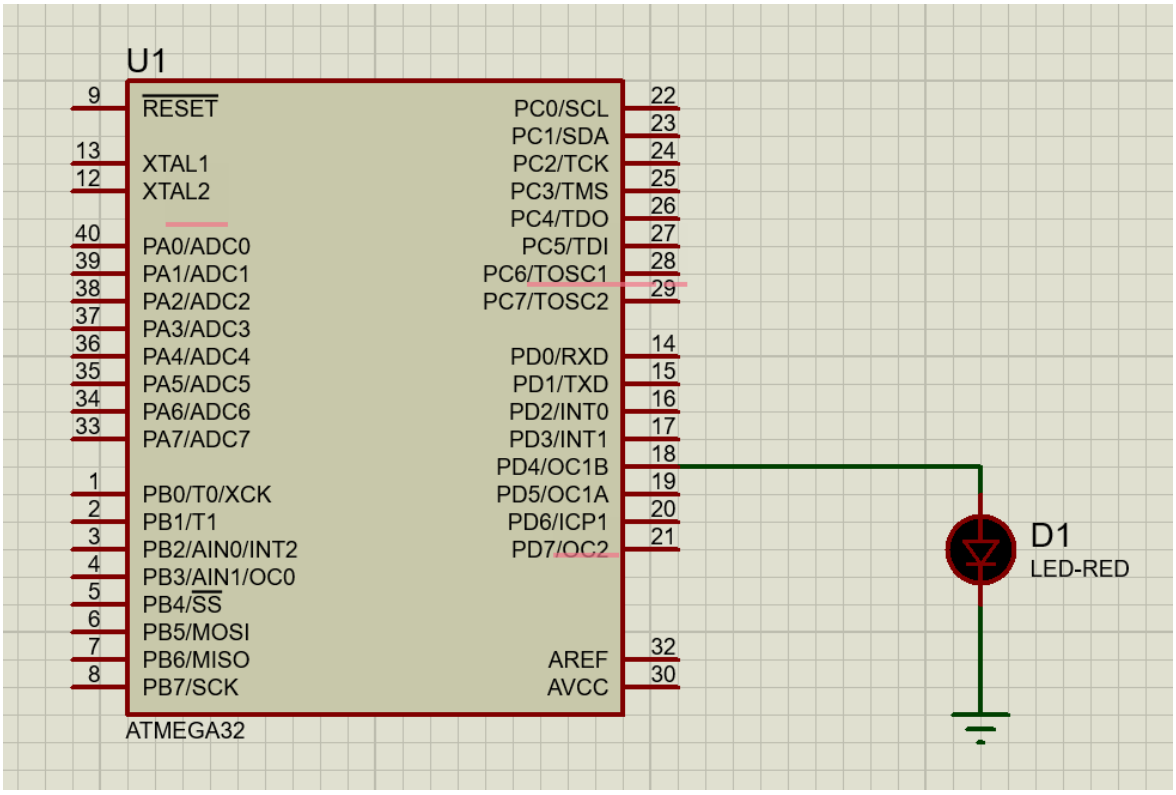
Now let's write a simple program which will toggle a port pin (PD4) after the timer 0 overflows 6 times.

1. Load TCNT0 with 0x00
2. Set CS00 and CS02 bits in TCCR0 register. This will start the timeWe will calculate the tick time in just a moment.r at Clk/1024 speed.
3. Monitor the TOV0 flag in the TIFR0 register to check if the timer has over-flowed, keep a timerOverFlowCount.
4. If timerOverFlowCount >= 6, toggle the led on PD4 and reset the count

Code:

```
#include<avr/io.h>
#define LED PD4

int main()
{
	uint8_t timerOverflowCount=0;
	DDRD=0xff;
	TCNT0=0x00;

	TCCR0 = (1<<CS00) | (1<<CS02);

	while(1)
	{

		while ((TIFR & 0x01) == 0);
		TCNT0 = 0x00;
		TIFR=0x01; //clear timer1 overflow flag
		timerOverflowCount++;
		if (timerOverflowCount>=4)
		{
			PORTD ^= (0x01 << LED);
			timerOverflowCount=0;
		}
	}
}
```

Circuit :-



CONCLUSION:

After Performing these experiments I came to know the various aspects and applications of timers

Experiment-3 **Post Lab Exercise**

Student Name: Aryan Langhanoja
Enrollment No: 92200133030

Answer the following questions:

1) What is the difference between Timer 0 and Timer 1 in terms of bits?
   - Timer0 is 8 bit timer and Timer1 is 16 bit timer
2) If the crystal frequency of Atmega32 is 8 MHz and the prescalar chosen is 256 then what is the time period and total time period for the count from 0 to 255?
   - The time period will be 32us and the total time period will be 8.192ms.
3) What is the job of the TCCR0 register?
   - TCCR0 is responsible for controlling the mode and the pre scalar which is going

to be used in the timer applications.

4) How many Timers are there in Atmega32?
   - There are 3 timers in ATMega32 Timer0 which is 8 bit ,Timer1 which is 16 bit and Timer2 which is 8 bit.
5) Which register holds the TOV0/1 bit?
   - TIFR holds TOV0