Ministerul Educatiei al Republicii Moldova
Universitatea Tehnica a Moldovei
Filiera Anglofona

Report

Laboratory Nr.6

Embeded Systems

Performed By: Daniel Ciobanu

Verified By: Andrei Bragarenco

Chisinau 2017

Topic : PID Controller

Theory:

A **microcontroller** is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, peripherals, automobiles and household appliances for computer systems. Due to that, another name for a microcontroller is "embedded controller." Some embedded systems are more sophisticated, while others have minimal requirements for memory and programming length and a low software complexity. Input and output devices include solenoids, LCD displays, relays, switches and sensors for data like humidity, temperature or light level, amongst others.

AVR Timers – TIMER0

In AVR, there are three types of timers – TIMER0, TIMER1 and TIMER2. Of these, TIMER1 is a 16-bit timer whereas others are 8-bit timers.

TCNT0 Register

The **Timer/Counter Register** – TCNT0 is as follows:

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

TCNT0 Register

This is where the uint 8-bit counter of the timer resides. The value of the counter is stored here and increases/decreases automatically. Data can be both read/written from this register.

Activate the timer:

TCCR0 Register

The **Timer/Counter Control Register** – TCCR0 is as follows:

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

Choose prescaler:

Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Interrupts:

In most microcontrollers, there is something called interrupt. This interrupt can be fired whenever certain conditions are met. Now whenever an interrupt is fired, the AVR stops and saves its execution of the main routine, attends to the interrupt call (by executing a special routine, called the Interrupt Service Routine, ISR) and once it is done with it, returns to the main routine and continues executing it.

TIMSK Register

The **Timer/Counter Interrupt Mask** – TIMSK Register is as follows. It is a common register for all the three timers. For TIMER0, bits 1 and 0 are allotted. Right now, we are interested in the 0th bit **TOIE0**. Setting this bit to '1' enables the TIMER0 overflow interrupt.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

TIFR Register

The **Timer/Counter Interrupt Flag Register**– TIFR is as follows. Even though we are not using it in our code, you should be aware of it.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

Setting Up TIMER0 in Fast PWM mode

Setting up the TIMER0 in fast pwm mode is very easy and just require one line of code. You only need to deal with one register named TCCR0 (**T**imer **C**ounter **C**ontrol **R**egister For Timer **0**). You just need to set up various bits in it to get the required setting. The various bits of TCCR0 is given below.

TCCR0

This register is used for configuring the TIMER0. See Timer Tutorial for more info. The explanation of various bits of this register is as follows.

Bit No	7	6	5	4	3	2	1	0
Name	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Initial Val	0	0	1	0	0	0	0	0

WGM – Wave Form Generation Mode

Mode	WGM00	WGM01	Mode Of Operation
0	0	0	Normal
1	0	1	PWM Phase Correct
2	1	0	CTC
3	1	1	Fast PWM

From the table it is clear that for Fast PWM we need mode 3. To get it we must set WGM00=1 and WGM01=1

COM – Compare Output Mode

These bits are used to set the Output mode in various Wave form generation mode. For Fast PWM mode these can be used to achieve following output modes.

COM01	COM00	Output Mode
0	0	Normal Port Operation (OC0 disconnected)
1	0	RESERVED

0	1	Non Inverted PWM
1	1	Inverted PWM

We need the "Non Inverted PWM output mode" so we set COM01=0 and COM00=1

CS – Clock Select

These are used to set an Input Clock for TIMER. We set them as follows to get Ftimer=F_CPU

CS02 = 0

CS01 = 0

CS00 = 1

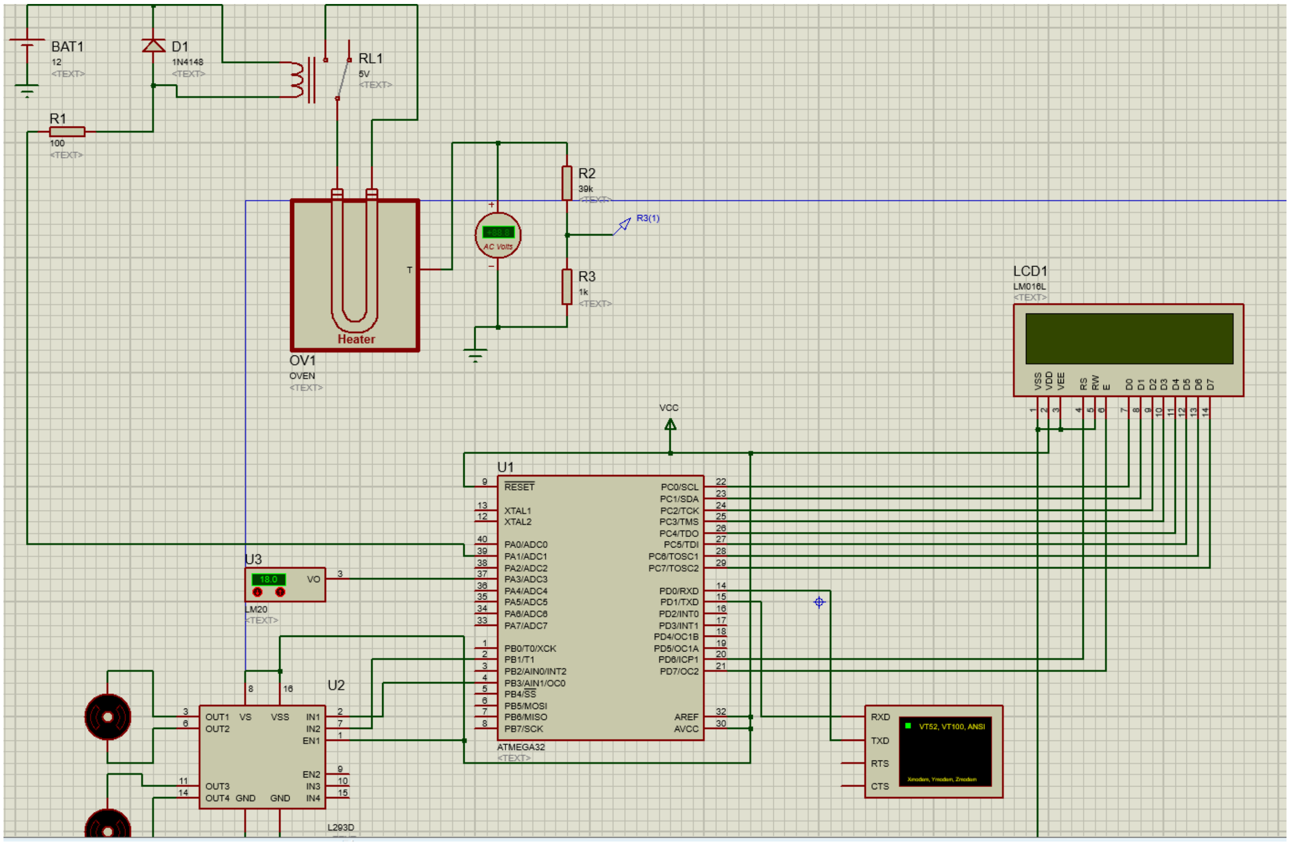
Now the TIMER is in Fast PWM mode to vary its output duty cycle we just need to set the OCR0 (Output Compare Register for Timer 0). For example setting it to 0 will generate PWM with duty cycle 0% (Totally off) while setting it to 128 will generate 50% duty cycle and 255 will generate 100% duty cycle signals.

Note: The output waveform is available in the associated Output Compare Pin of the microcontroller. For example for Timer 0 the associated OC pin is OC0. You can find its location from Pin diagram in datasheet. In ATmega16 and ATmega32 it is on PORTB bit 3, i.e. PB3. ***This pin must be set to output to get the PWM signals.***

Implementation:

In this laboratory work i did kind of a PID control. This means that the sensor gets the data from the outside and sends it to the drives. They having the data, analyze it and take actions given the circumstances.

The scheme in Proteus:



Conclusion :

Doing this laboratory work i learned how to work with a full system constructed by different deices and how to manage them. I also learned how to use pwm for atmega32.