Embended systems
Microcontrollers
Fall 2016
Laboratory Work 5

Vasile Schidu

January 22, 2017

Date Performed:	October 02, 2016
Partners:	Vasile Schidu
Instructor:	Bragrarenco Andrei

Introduction

Topic: Introduction to Micro Controller Unit programming and implementation of timer.

Objectives:

1. Studying of timer
2. Timer registers
3. Create a basic task scheduler using timer
4. Understanding of avr timers usage

Tasks: Write a C program and schematics for Micro Controller Unit (MCU) using Universal asynchronous receiver/transmitter. For writing program, use ANSI-C Programming Language with AVR Compiler and for schematics use Proteus, which allow us to explain and to demonstrate the usage of timers in avr programming.

AVR Timers

DefinitionTimers are standard features of almost every microcontroller. So it is very important to learn their use. Since an AVR microcontroller has very powerful and multifunctional timers, the topic of timer is somewhat "vast". Moreover there are many different timers on chip. So this section on timers will be multipart. I will be giving basic introduction first.

What is a timer ? A timer in simplest term is a register. Timers generally have a resolution of 8 or 16 Bits. So a 8 bit timer is 8Bits wide so capable of holding value withing 0-255. But this register has a magical property ! Its value increases/decreases automatically at a predefined rate (supplied by user). This is the timer clock. And this operation does not need CPU's intervention.

Basic operation of a timer Since Timer works independently of CPU it can be used to measure time accurately. Timer upon certain conditions take some action automatically or inform CPU. One of the basic condition is the situation when timer OVERFLOWS i.e. its counted upto its maximum value (255 for 8 BIT timers) and rolled back to 0. In this situation timer can issue an interrupt and you must write an Interrupt Service Routine (ISR) to handle the event.

Using the 8 bit timer (TIMER0)

The ATmega16 and ATmega32 has three different timers of which the simplest is TIMER0. Its resolution is 8 BIT i.e. it can count from 0 to 255. Note: Please read the "Internal Peripherals of AVRs" to have the basic knowledge of techniques used for using the OnChip peripherals(Like timer !) The Prescaler The Prescaler is a mechanism for generating clock for timer by the CPU clock. As you know that CPU has a clock source such as a external crystal of internal oscillator. Normally these have the frequency like 1 MHz,8 MHz, 12 MHz or 16MHz(MAX). The Prescaler is used to divide this clock frequency and produce a clock for TIMER. The Prescaler can be used to get the following clock for timer. No Clock (Timer Stop). No Prescaling (Clock = FCPU) FCPU/8 FCPU/64 FCPU/256 FCPU/1024 Timer can also be externally clocked but I am leaving it for now for simplicity

TIMER0 Registers As you may be knowing from the article "Internal Peripherals of AVRs" every peripheral is connected with CPU from a set of registers used to communicate with it. The registers of TIMERs are given below. TCCR0 – Timer Counter Control Register. This will be used to configure the timer.

Timer counter control register TCCR0 As you can see there are 8 Bits in this register each used for certain purpose. For this tutorial I will only focus on the last three bits CS02 CS01 CS00 They are the CLOCK SELECT bits. They are used to set up the Prescaler for timer.

Timer interrupt mask register (TIMSK) This register is used to activate/deactivate interrupts related with timers. This register controls the interrupts of all the three timers. The last two bits (BIT 1 and BIT 0) Controls the interrupts of TIMER0. TIMER0 has two interrupts but in this article I will tell you only about one(second one for next tutorial). TOIE0 : This bit when set to "1" enables the OVERFLOW interrupt. Now time for some practical codes !!! We will set up timer to at a Prescaler of 1024 and our FCPU is 16MHz. We will increment a variable "count" at every interrupt(OVERFLOW) if count reaches 61 we will toggle PORTC0 which is connected to LED and reset "count= 0". Clock input of TIMER0 = 16MHz/1024 = 15625 Hz Frequency of Overflow = 15625 /256 = 61.0352 Hz if we increment a variable "count" every Overflow when "count reach 61" approx one second has elapse.

Resourses

Short Theory: Proteus developed by Labcenter Electronics, is a software with which you can easily generate schematic captures, develop PCB and simulate microprocessor. It has such a simple yet effective interface that it simplifies the task required to be performed. This one aspect has attracted many users to select this tool amongst many others offering the same services. Atmel® Studio 6 is the integrated development platform (IDP) for developing and debugging Atmel ARM® Cortex®-M and Atmel AVR® microcontroller (MCU) based applications. The Atmel Studio 6 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. A microcontroller (sometimes abbreviated μC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications. In computing, a device driver (commonly referred to as a driver) is a computer program that operates or controls a particular type of device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.

About stdio library: Input and Output operations can be performed using theCStandardInput andOutput Library (cstdio, known asstdio.hin the C language). This library uses what are calledstreamsto operate with physical devices such as keyboards, printers, terminals or with any other type of files supported by the system. Streams are an abstraction to interact with these in an uniform way. All streams have similar properties independently of the individual characteristics of the
physical media they are associated with. Streams are handled in thecstdiolibrary as pointers toFILEobjects. A pointer to aFILEobject uniquely identifies a stream, and is used as a parameter in the operations involving that stream.

Atmel® microcontrollers: Atmel® microcontrollers (MCUs) deliver a rich blend of efficient integrated designs, proven technology, and groundbreaking innovation that is ideal for today's smart, connected products. In this era of the Internet of Things (IoT), microcontrollers comprise a key technology that fuels machine-to-machine (M2M) communications. Building on decades of experience and industry leadership, Atmel offers proven architectures that are optimized for low power, high-speed connectivity, optimal data bandwidth, and rich interface support. By using our wide variety of configuration options, developers can devise complete system solutions for all kinds of applications. Atmel microcontrollers can also support seamless integration of capacitive touch technology to implement buttons, sliders, and wheels (BSW). In addition, Atmel MCUs deliver wireless and security support. No matter what your market or device, Atmel offers a compelling solution that is tailored to your needs—today and tomorrow. Atmel is a global industry leader in the design and manufacture of microcontrollers and related system solutions, including capacitive touch solutions, advanced logic, mixed-signal, nonvolatile memory, and radio frequency (RF) components. Leveraging

one of the industry's broadest intellectual property technology portfolios and backed by a comprehensive ecosystem, Atmel MCU products enable designers to develop complete solutions for industrial, consumer, security, communications, computing, and automotive markets. Developers have the option of combining Atmel microcontrollers with industry-leading Atmel touch technology. Atmel technology for touchscreens and fixed-function buttons, sliders and wheels provides a rich user experience with unparalleled performance, while minimizing power consumption.

ATmega328

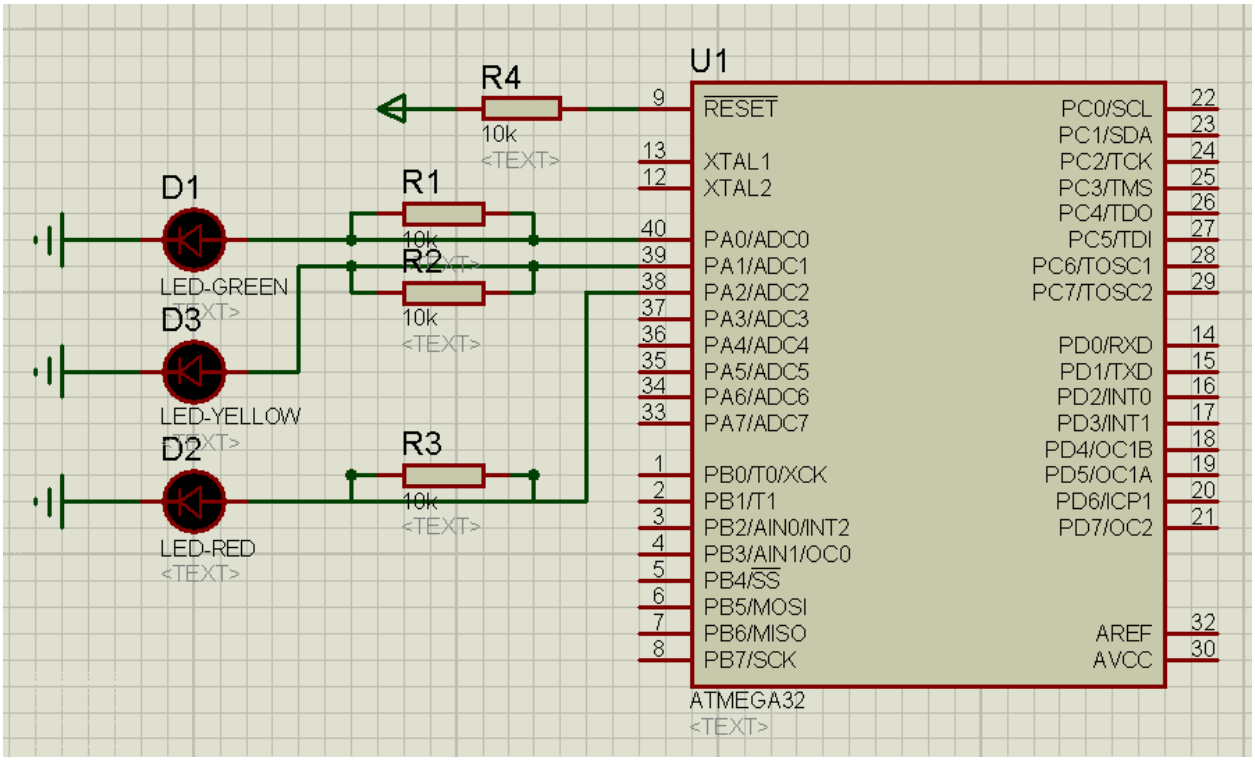
The high-performance Atmel 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1KB EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts,serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

Atmel Studio: Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits. Additionally, Atmel Studio includes Atmel Gallery, an online apps store that allows you to extend your development environment with plug-ins developed by Atmel as well as by third-party tool and embedded software vendors. Atmel Studio 7 can also able seamlessly import your Arduino sketches as C++ projects, providing a simple transition path from Makerspace to Marketplace.

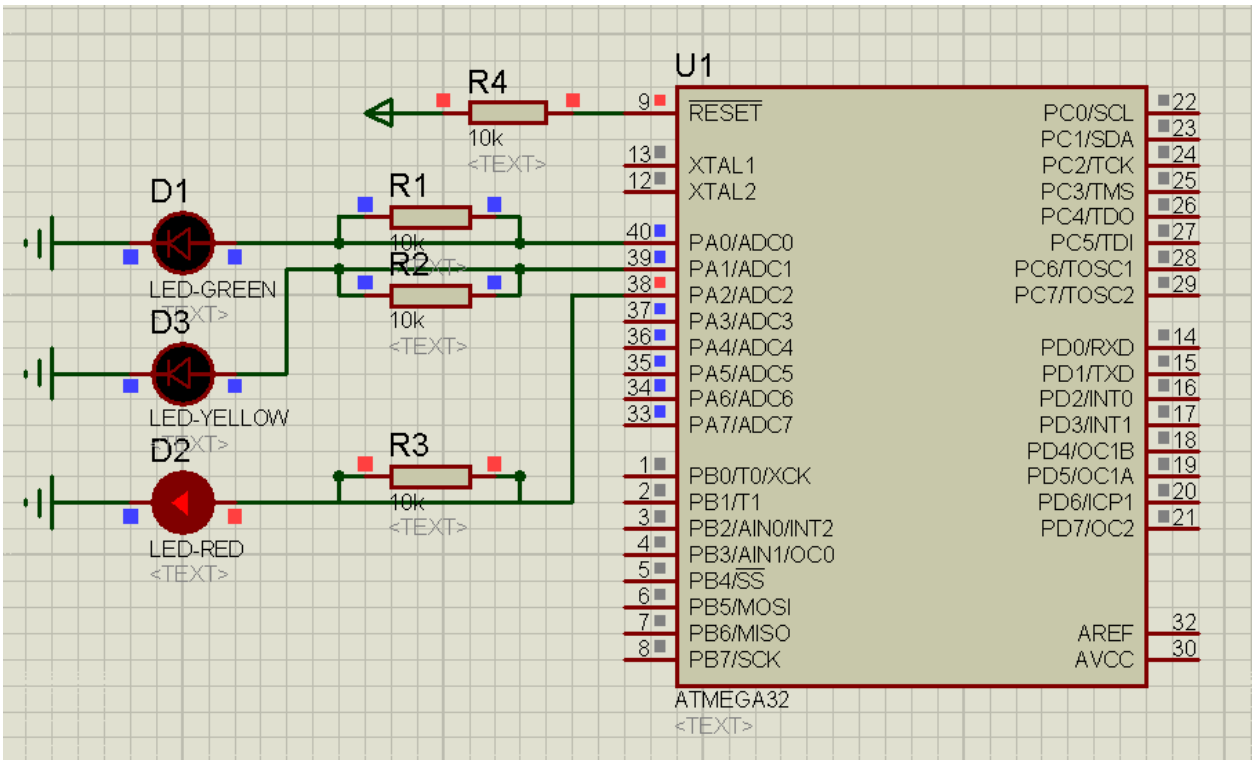
Solution

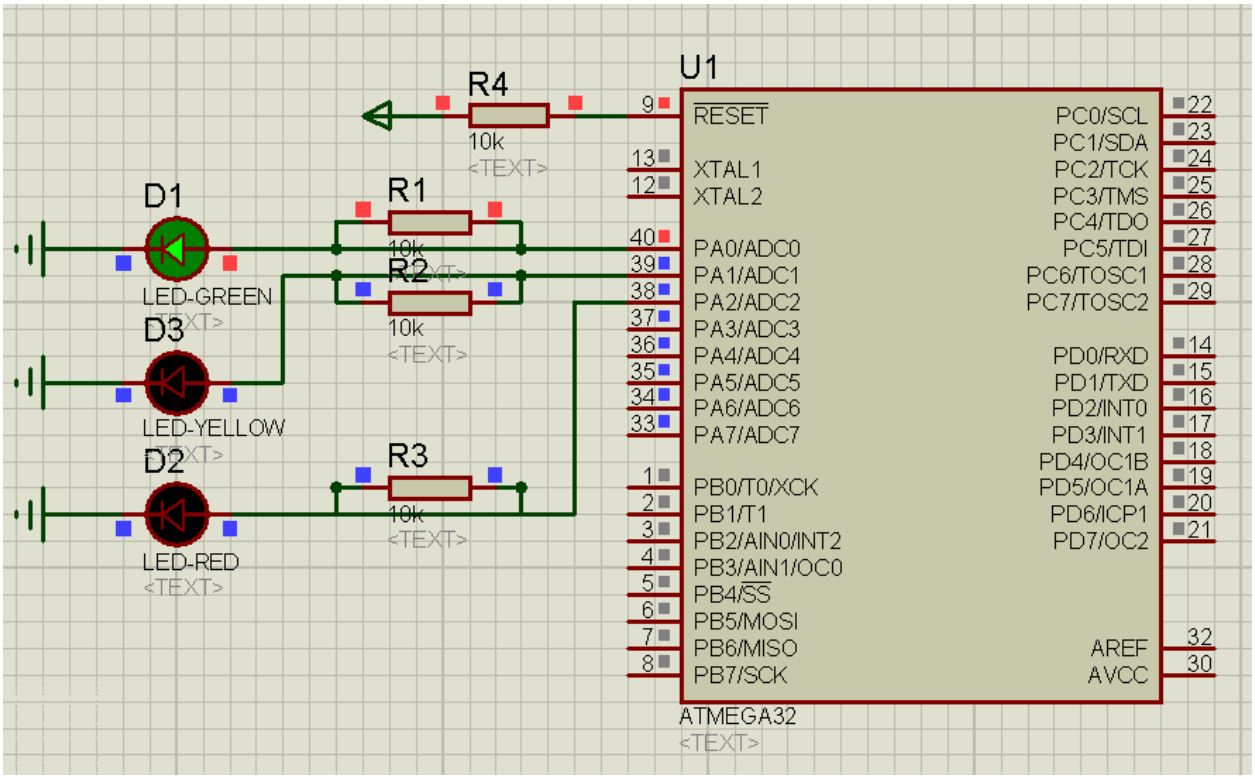
Proteus Scheme Let's take a look on this scheme. So we have Microcontroler **ATMEGA32** This microcontroler is composed from 4 ports each of thie have 8 pins.

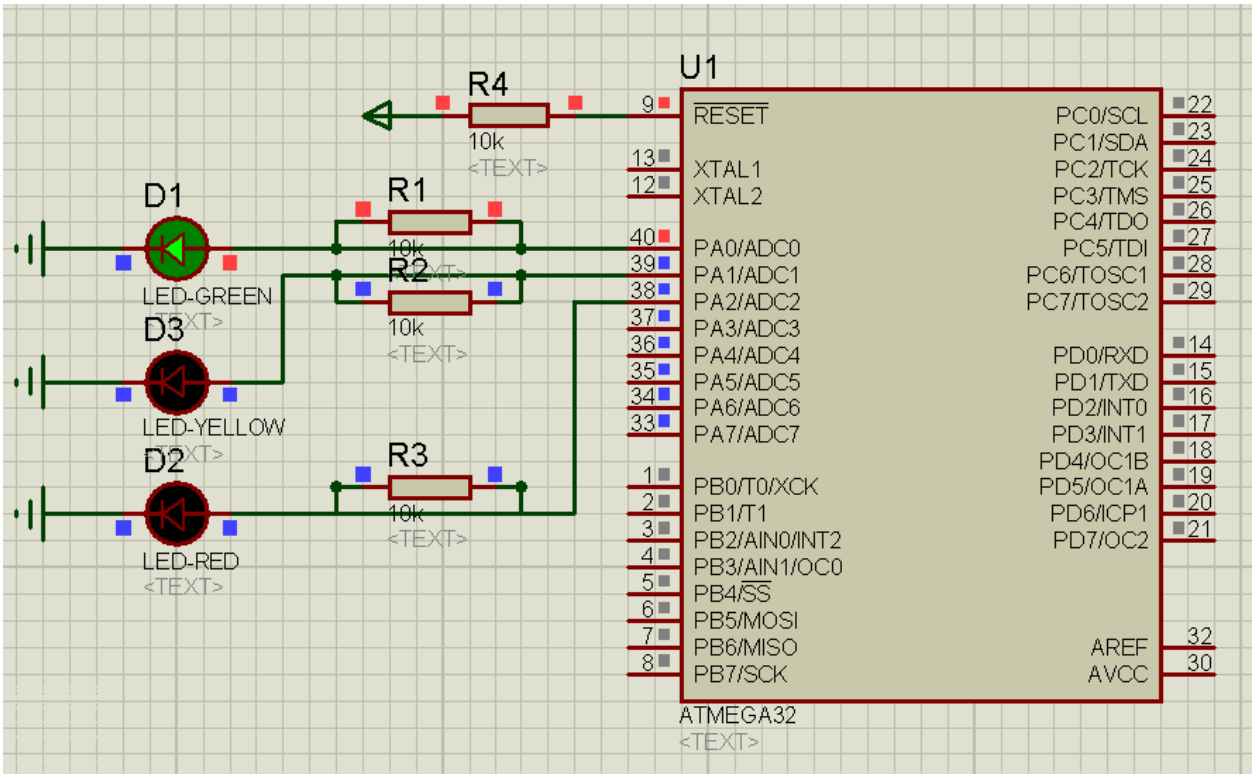
So here is our project in Proteus.



In the picture we cant see, but after an equal amount of time the led turn on and turn off one by one.




U1
R4
10k
<TEXT>
D1
R1
10k
R2
10k
LED-GREEN
D3
LED-YELLOW
D2
R3
10k
LED-RED
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32


U1
R4
10k
D1
R1
10k
R2
10k
LED-GREEN
D3
LED-YELLOW
D2
R3
10k
LED-RED
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32

Conclusion

In this laboratory work I've learned basic concepts of MCU programming in C language and building a simple printed circuit using Proteus using timers. I've understood that timers is a very important component in avr programming.