Embedded System on AVR Microcontroller (ATMEGA32)

Exp1: Introduction of Microchip Studio, SimulIDE and Hardware Programming Tool(Khazama AVR Programmer) with LED Blink

Submitted by
Ronit Dutta, MS in IOT and Signal Processing
Department of Electrical Engineering, IIT Kharagpur

Under the guidance of
Aurobinda Routray, Professor, Electrical Engineering

Department of Electrical Engineering
Indian Institute of Technology Kharagpur
January, 2024

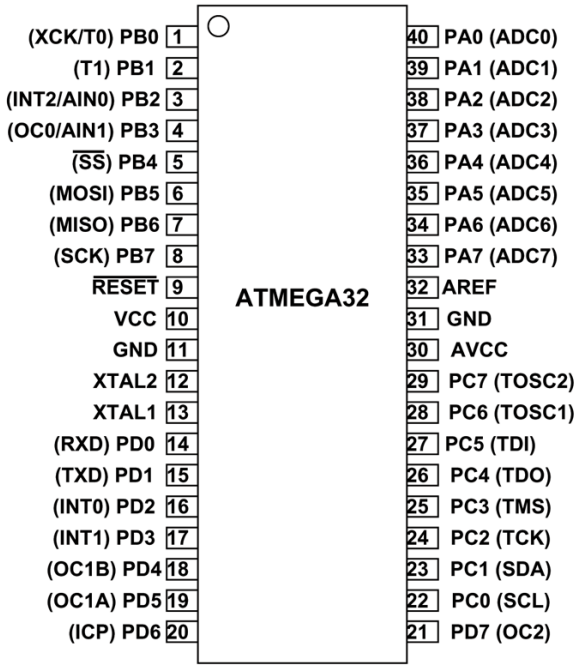
• Embedded System Lab Experiments

Experiment No.	Title of the Experiment	Week
Exp1	Introduction of Microchip Studio, SimulIDE and Hardware Programming Tool(Khazama AVR Programmer) with LED Blink	1
Exp2	Changing of Duty Cycle of LED Blink; Digital Output through 7 segment LED display to increment and decrement the number with SRAM memory mapping	2
Exp3	Accepting Digital Inputs through PUSH Button Pull Up Network (Internal Pull Up/ External Pull Up) Pull Down Network (External Pull Up)	3
Exp4	Interrupts in AVR (ATMEGA32): INT0, INT1	4
Exp5	Write library for 16x2 LCD & display text on it	5
Exp6	Waveform generation through Digital to Analog Converter DAC0808 and LM358 (i) Sawtooth (+ve slope, -ve slope) (ii) Sine, Miscellaneous	6
Exp7	Timer and Counter: PWM	7
Exp8	Analog to Digital Conversion (Internal 10bit ADC) of sinusoidal: Aliasing Effect	8 & 9
Exp9	Communication Protocols: UART, SPI & I2C	10
Exp10	Filter Design (Hardware and Software)	11 & 12

• Essential Requirements of the course

- ***Windows platform is needed***
- **3 Software are required**
  - I. **Microchip Studio: This is required to write the code for the ATMEGA32 microcontroller**
  - II. **SimulIDE: An open source simulation platform of ATMEGA32**
  - III. **Khazama AVR Programmer: This is required to flash the code in the Hardware Platform of ATMEGA32**

- **Few main features of ATMEGA32 for the Embedded System course**
  - **High-performance, Low-power Atmel®AVR® 8-bit Microcontroller**
  - **131 Powerful Instructions – Most Single-clock Cycle Execution**
  - **32 × 8 General Purpose Working Registers**
  - **Up to 16 MIPS Throughput at 16MHz**
  - **32Kbytes of In-System Self-programmable Flash program memory**
  - **1024Bytes EEPROM**
  - **2Kbytes Internal SRAM**
  - **Write/Erase Cycles: 10,000 Flash/100,000 EEPROM**
  - **Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes**
  - **One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode**
  - **Four PWM Channels**
  - **8-channel, 10-bit ADC- 8 Single-ended Channels**
  - **Programmable Serial USART**
  - **Master/Slave SPI Serial Interface**
  - **Byte-oriented Two-wire Serial Interface**
  - **External and Internal Interrupt Sources**
  - **4.5V - 5.5V for ATmega32**
  - **0 - 16MHz for ATmega32**



- # Setup for Coding Environment on Microchip Studio

Step 1:

Install Microchip Studio from the below link to setup coding environment.

https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#Downloads

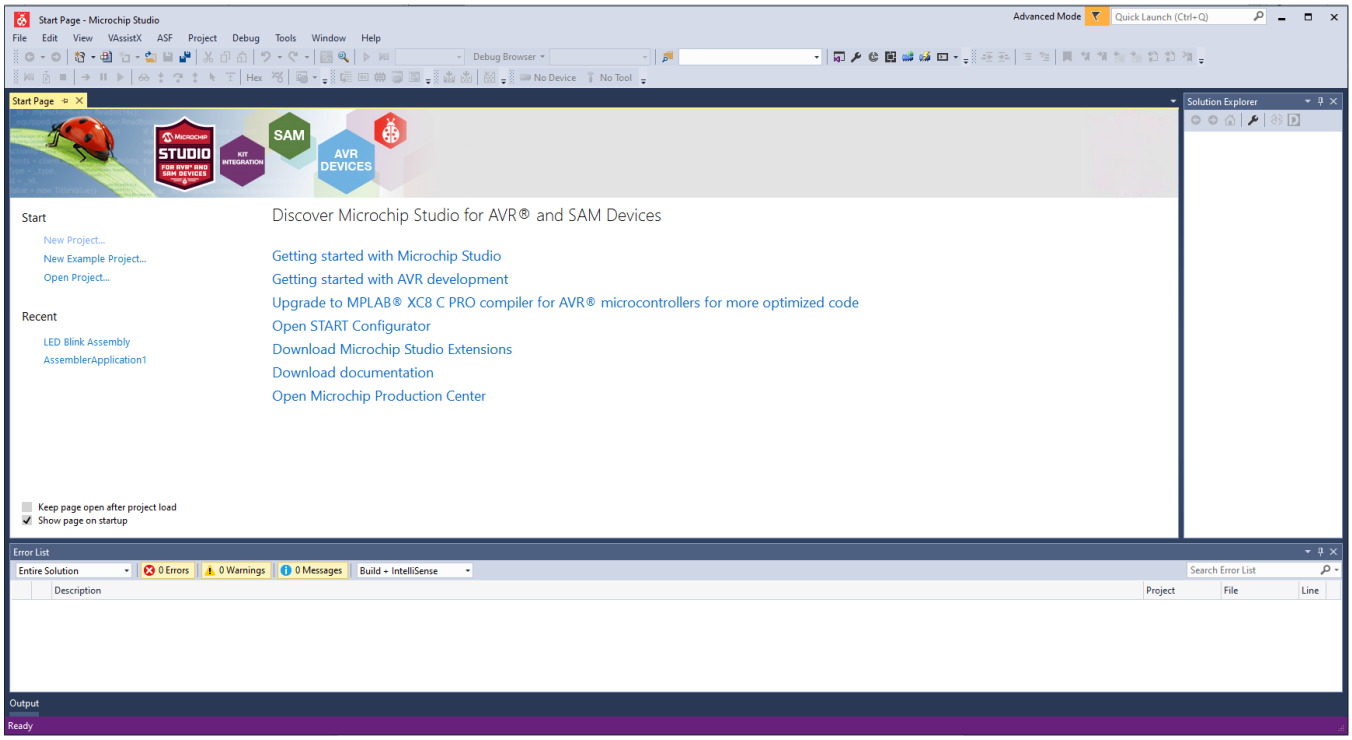
Download Microchip Studio

Title	Version Number	Date	Download
Microchip Studio for AVR and SAM Devices- Offline Installer	7.0.2594	20 Jun 2022	Download
Microchip Studio for AVR and SAM Devices- Web Installer	7.0.2594	20 Jun 2022	Download

Install with web installer or offline installer as your choice.

Step 2:

Click on the Microchip Studio icon to run the IDE(Integrated Development Environment)



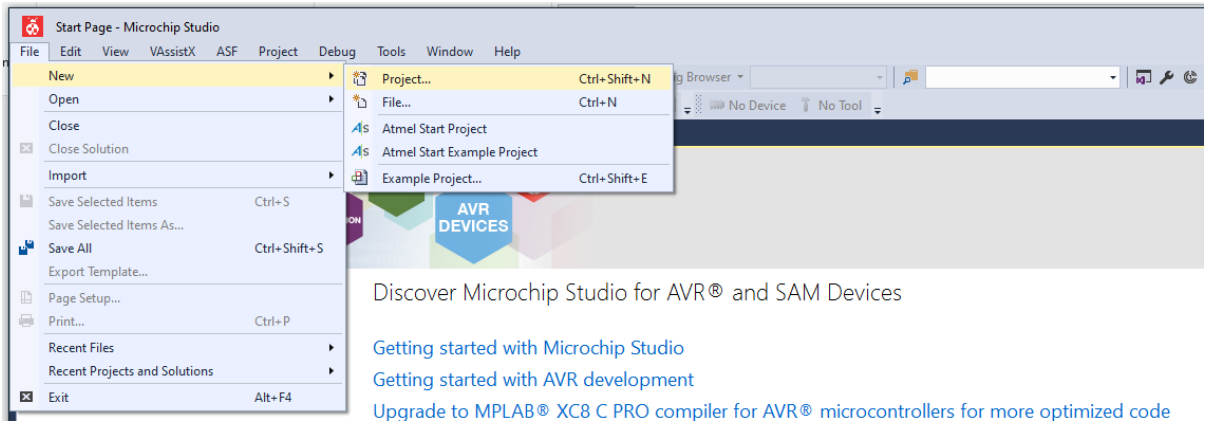
Now we can go for Assembly Coding or C/C++ Coding.

- # Experiment: LED Blinking

Assembly coding for LED Blinking on ATMEGA32

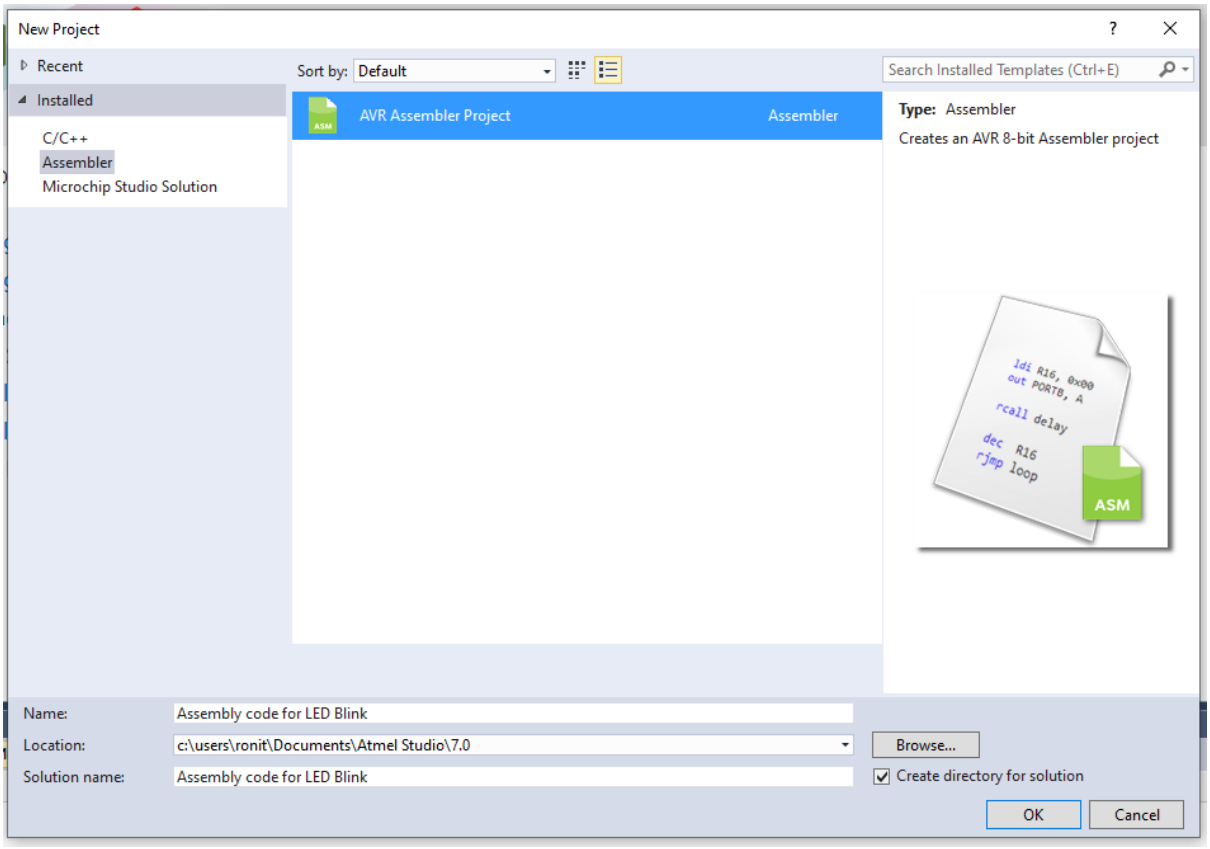
Step 1:

Click on File then New then Project



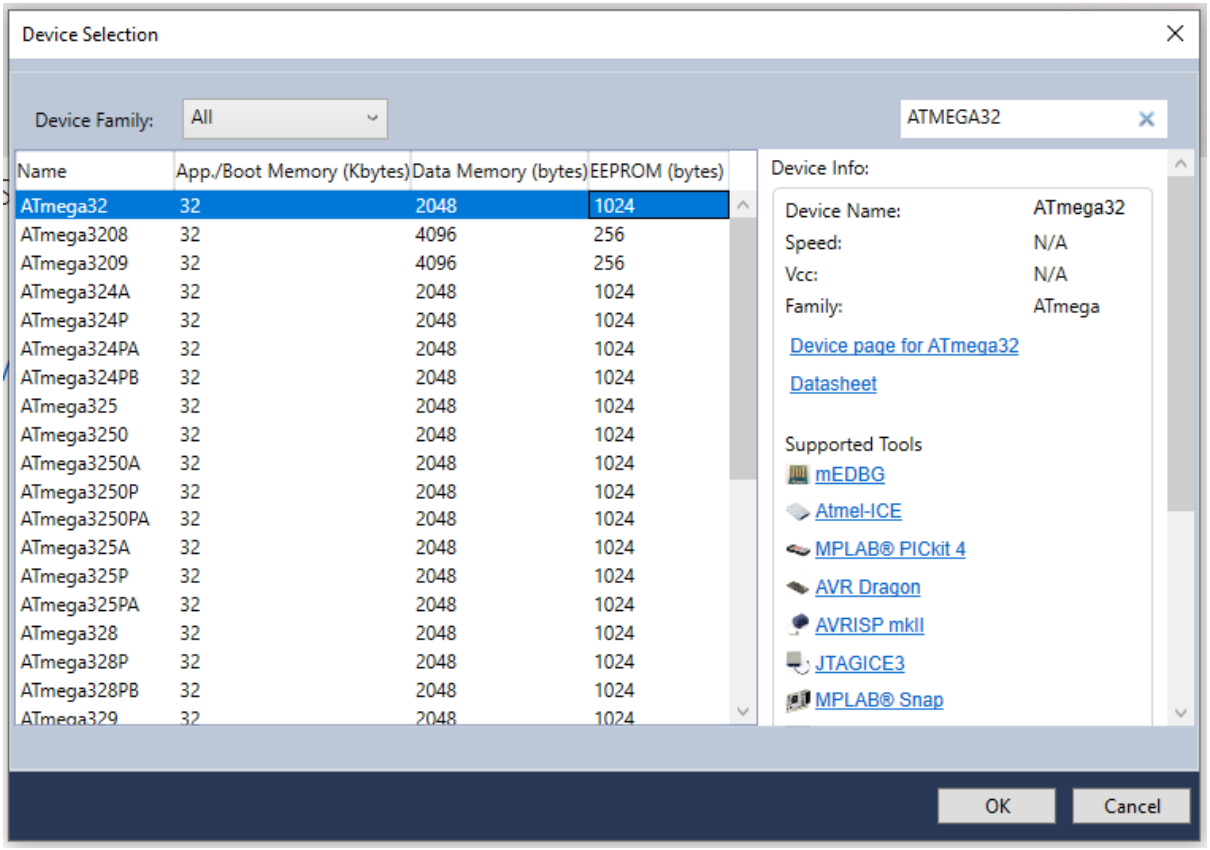
Step 2:

Then click on Assembler and then AVR Assembler Project and then rename it then click Ok.



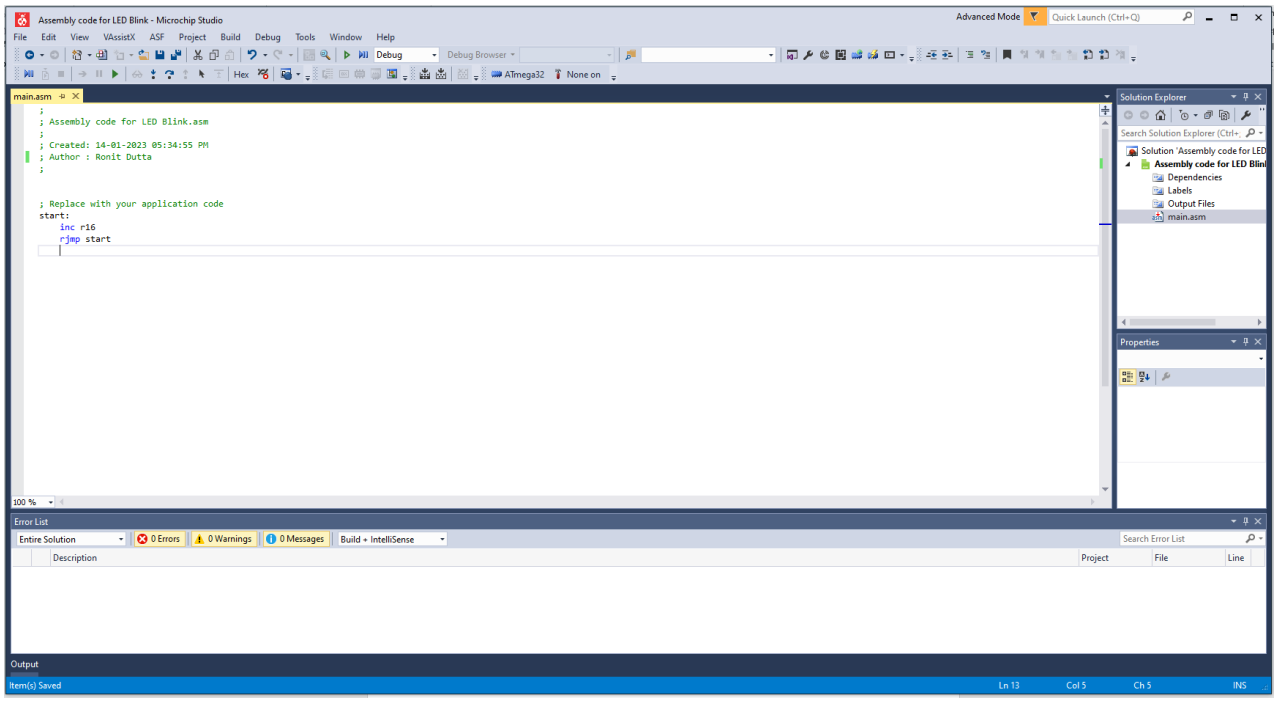
Step 3:

Select the device ATMEGA32 then click OK.



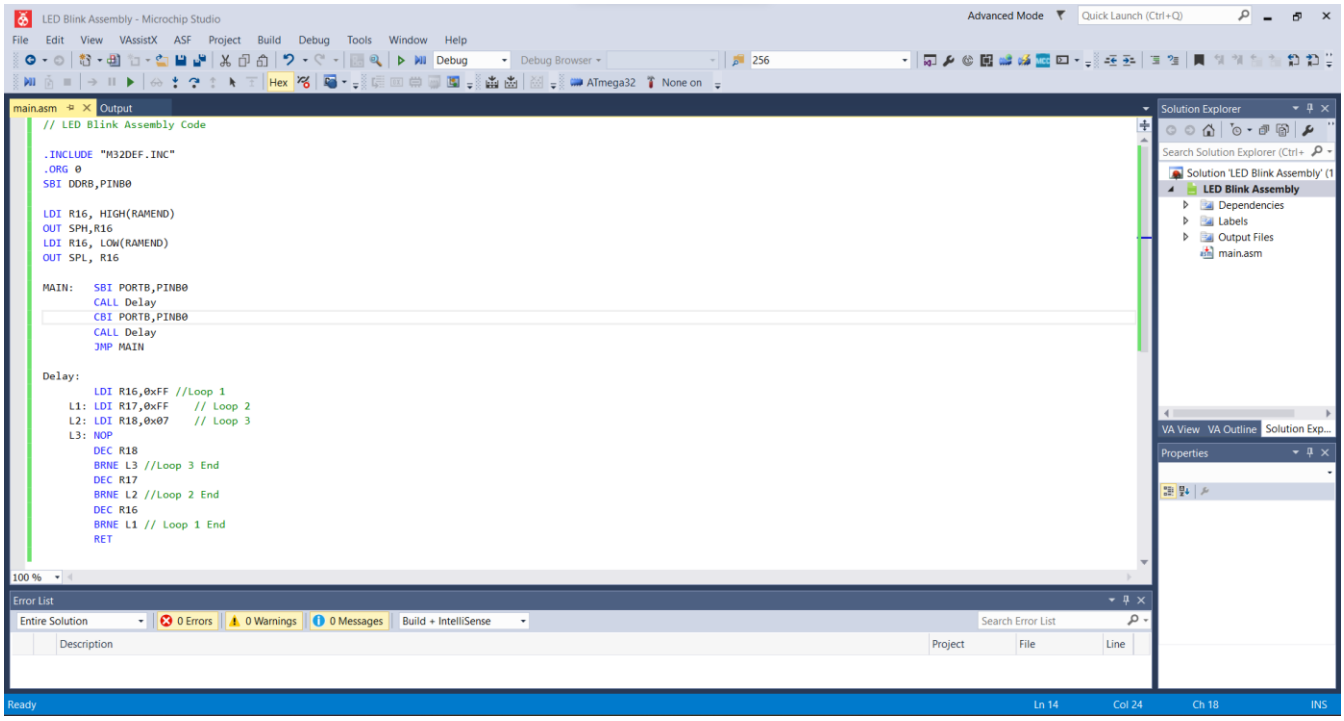
Step 4:

Write the assembly code in the main.asm



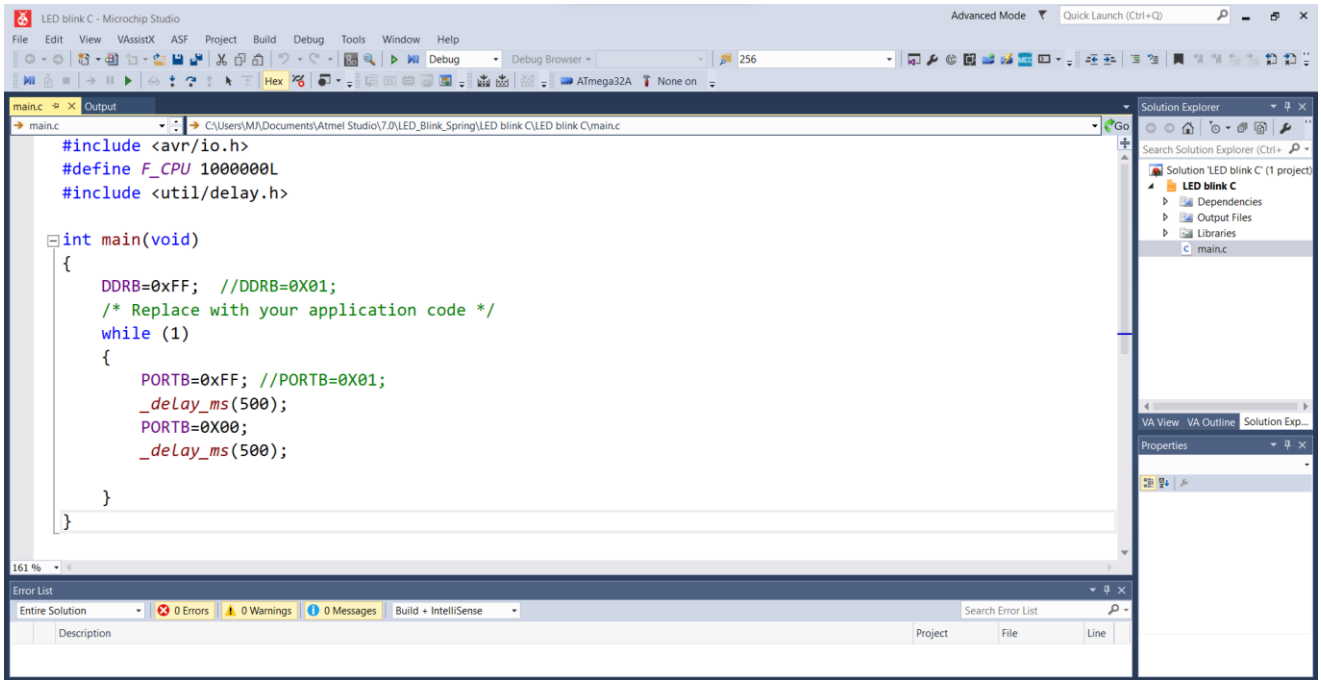
Step 5:

After writing the assembly code go to the build menu then build solution to generate the HEX file.




```
// LED Blink Assembly Code
.INCLUDE "M32DEF.INC"
.ORG 0
SBI DDRB,PINB0

LDI R16, HIGH(RAMEND)
OUT SPH,R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

MAIN:  SBI PORTB,PINB0
               CALL Delay
               CBI PORTB,PINB0
               CALL Delay
               JMP MAIN

Delay:
               LDI R16,0xFF //Loop 1
       L1:     LDI R17,0xFF  // Loop 2
       L2:     LDI R18,0x07  // Loop 3
       L3:     NOP
               DEC R18
               BRNE L3       //Loop 3 End
               DEC R17
               BRNE L2       //Loop 2 End
               DEC R16
               BRNE L1       // Loop 1 End
               RET
```

C coding for LED Blinking on ATMEGA32

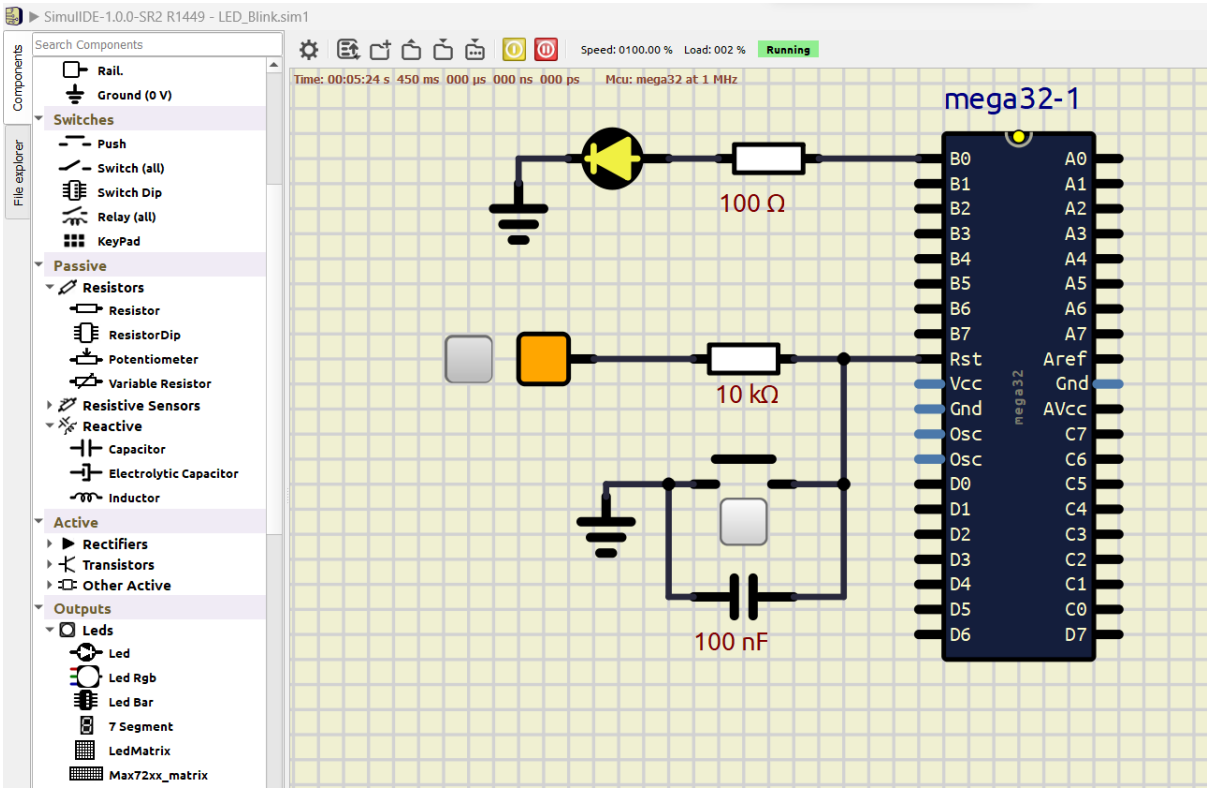



```c
//C code for LED Blink

#include <avr/io.h>
#define F_CPU 1000000L
#include <util/delay.h>

int main(void)
{
    DDRB=0xFF;   //DDRB=0X01;
    /* Replace with your application code */
    while (1)
    {
        PORTB=0xFF;  //PORTB=0X01;
        _delay_ms(500);
        PORTB=0X00;
        _delay_ms(500);

    }
}
```

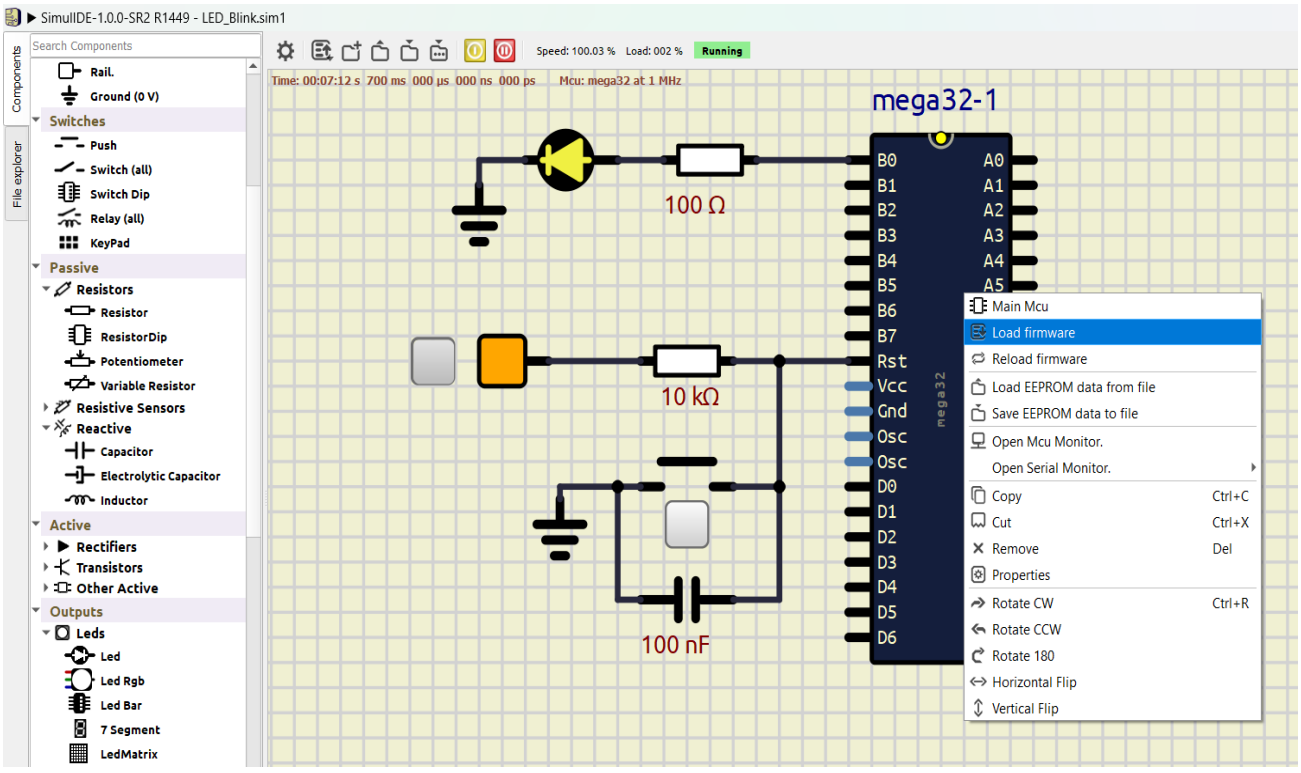
- # Simulation on SimulIDE

SimulIDE Download Link: https://simulide.com/p/downloads/

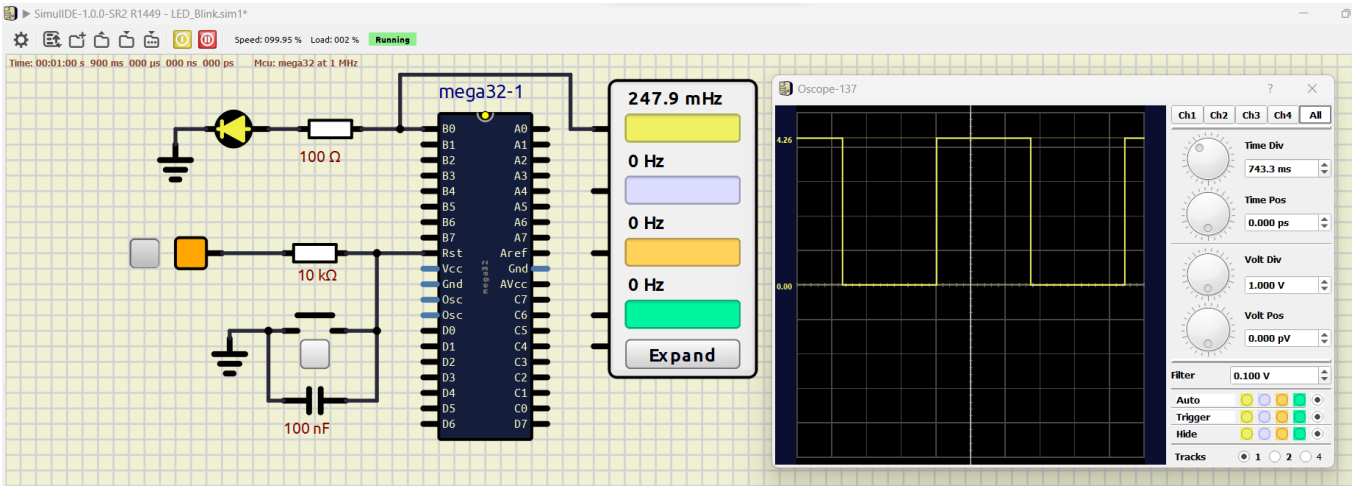
Make the below circuit on SimulIDE



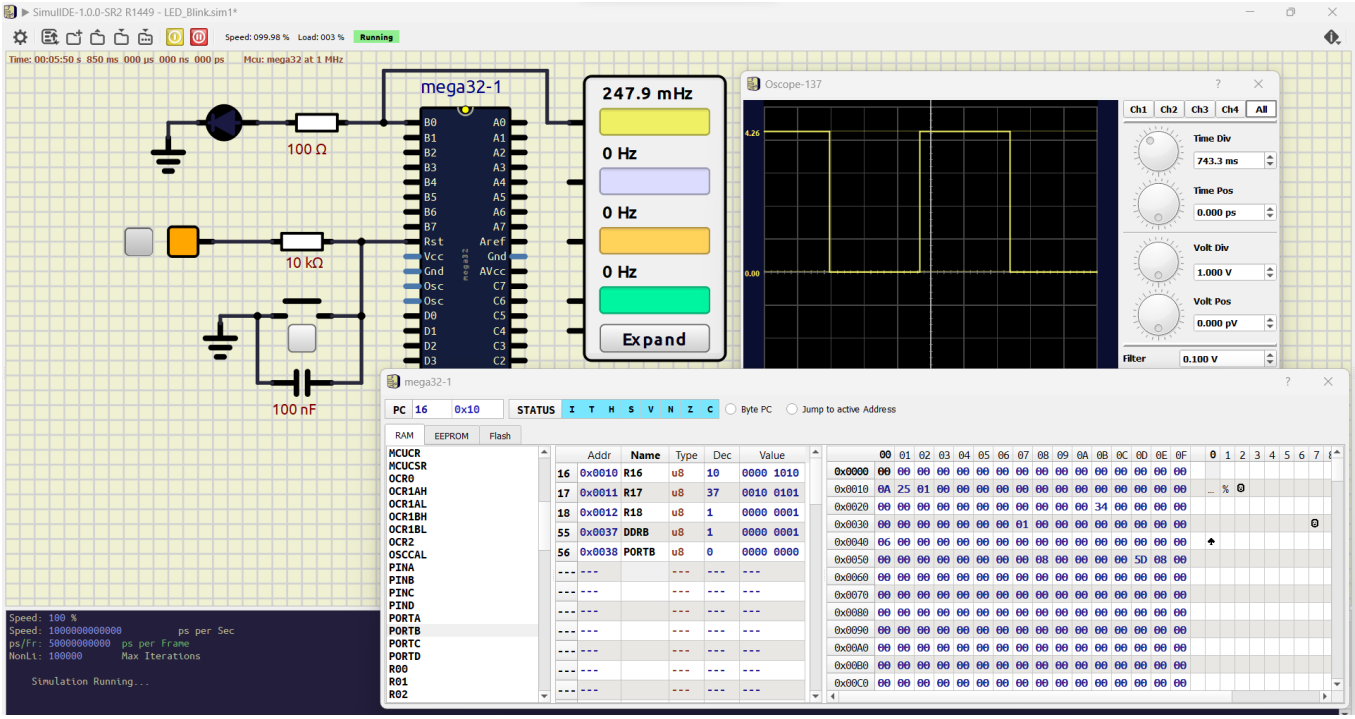
Then right click on the microcontroller and upload the HEX file on it. Set the proper clock frequency then simulate it.



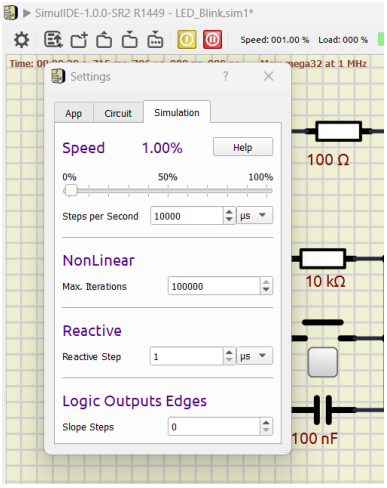
Attach an oscilloscope to observe the generated square wave signal (LED Blink Signal).



Right click on the microcontroller and click on “Open Mcu Monitor”. Check the RAM status.



To check the simulation less than Real-Time speed then click on settings and decrease the simulation speed.



- # Setup for Program Flashing Environment

For the flashing of ATMEGA32, we have required two software

I. Zadig tool
II. Khazama AVR Programmer

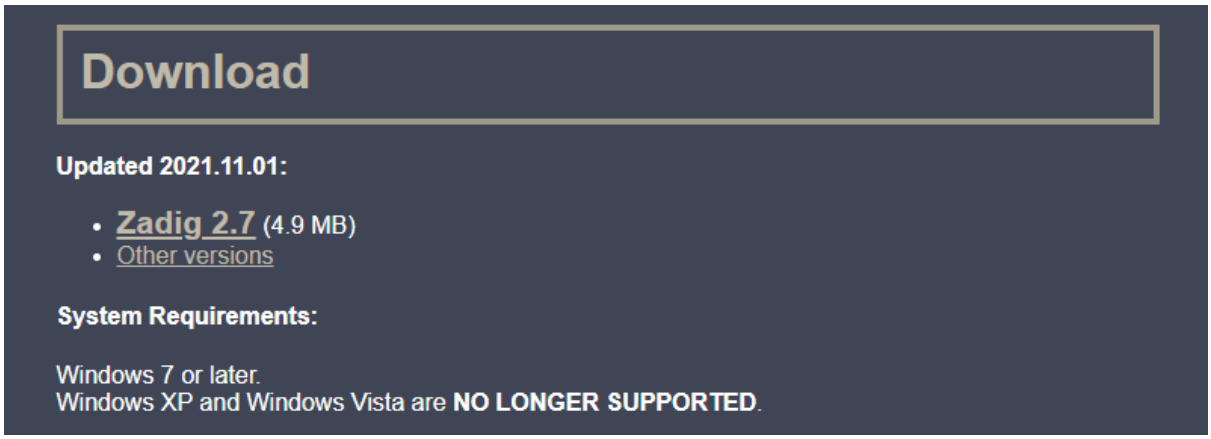
Zadig tool:

Zadig tool is required for the installation of the USBASP AVR Programmer driver.

Step 1:

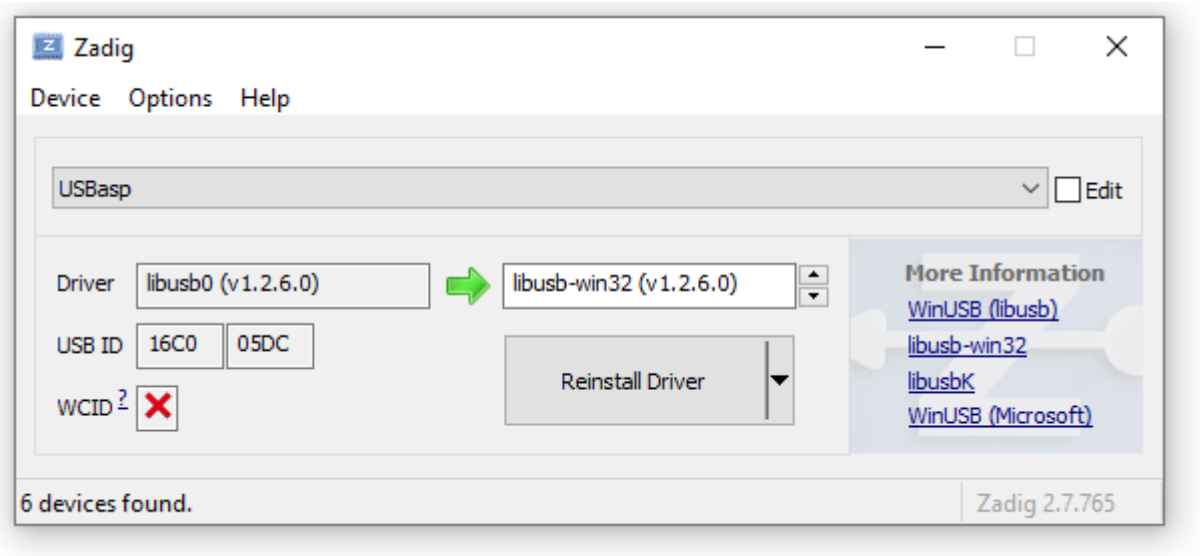
https://zadig.akeo.ie/

Click on the above link and download the software package.



Step 2:

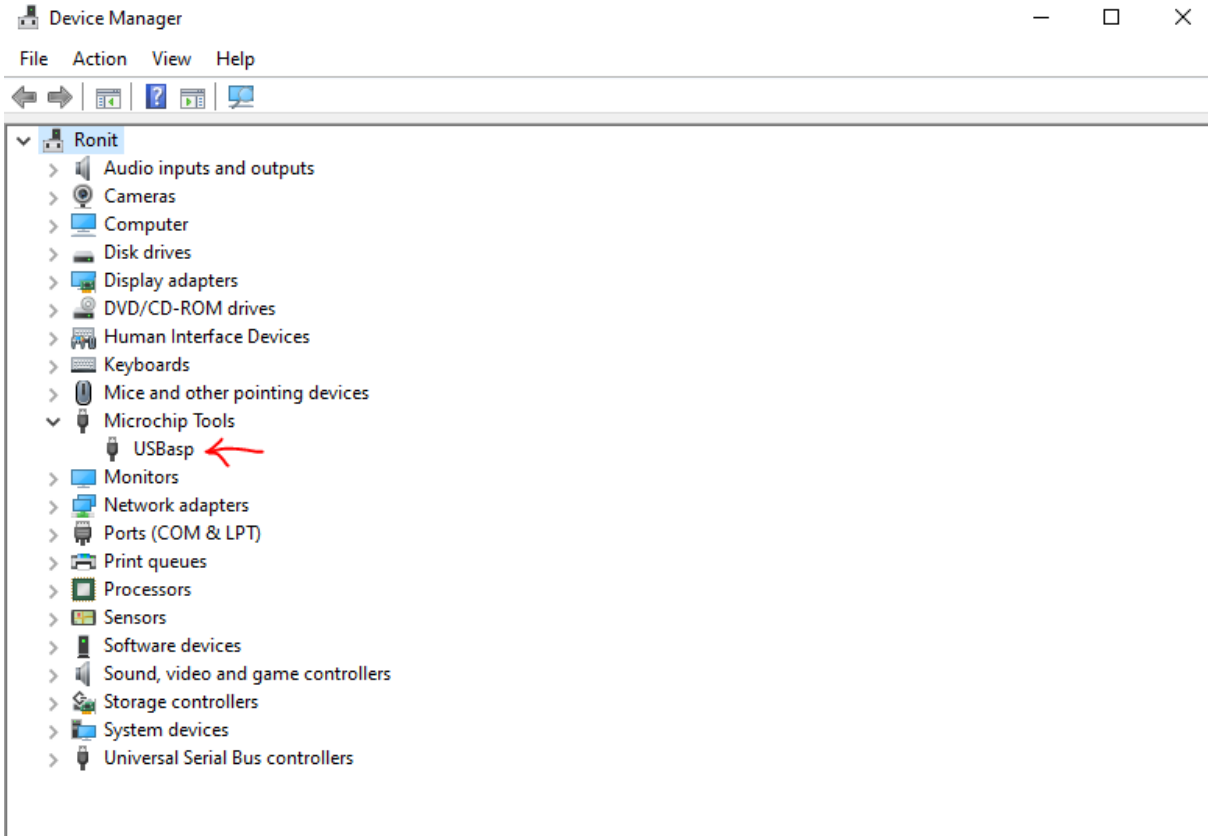
Double Click on the the zadig file and select USBasp from the dropdown menu



Select libusb-win32 driver for the USBasp programmer then click on install driver to install successfully.

Step 3:

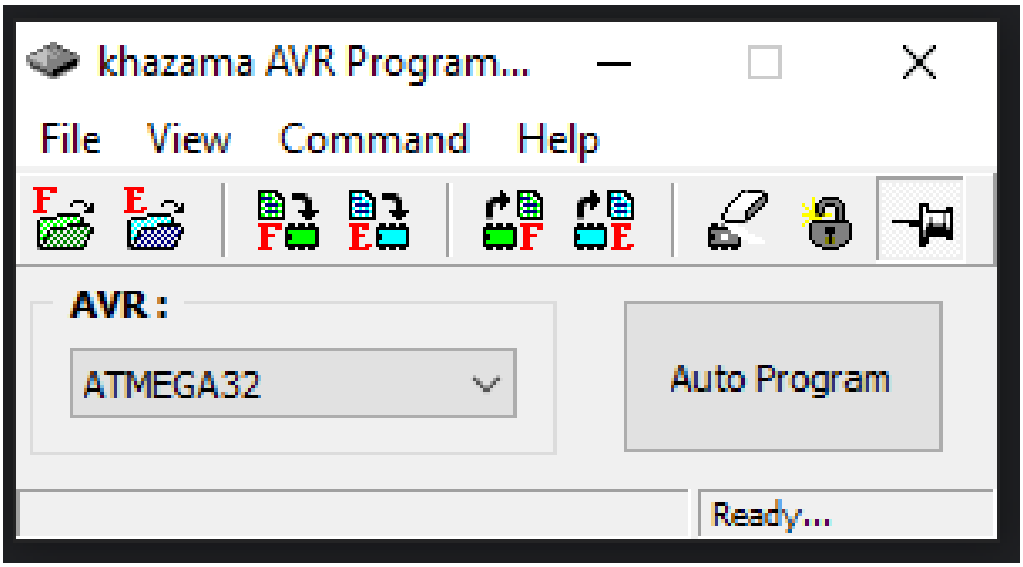
Go to device manager and check the USBasp under the Microchip/libusb-win32 Tools dropdown menu to verify.



Now, we are going to install Khazama AVR Programmer.

Khazama AVR Programmer

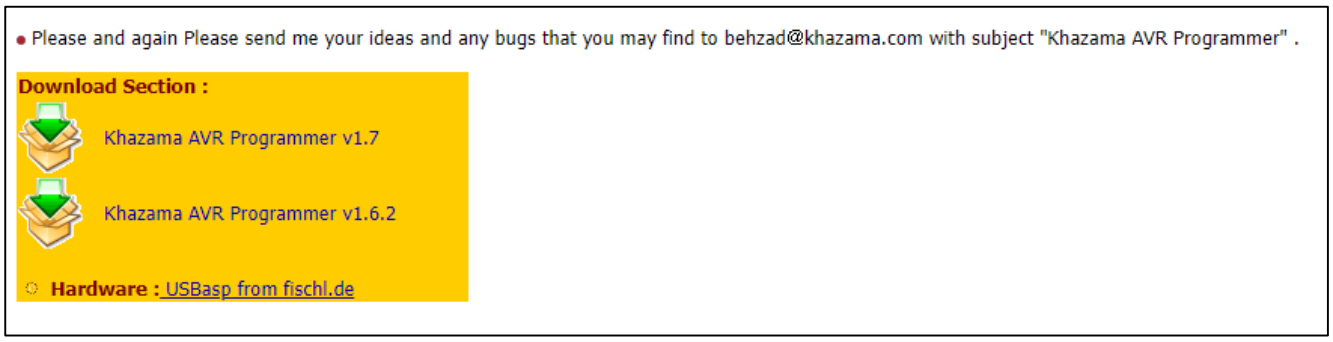
Khazama AVR Programmer is required to flash any AVR Group microcontroller. Here we are going to flash ATMEGA32 as our target microcontroller.



Step 1:

http://khazama.com/project/programmer/

Click on the above link and download the latest software package.



A .rar file will be downloaded.

Step 2:

Extract the .rar file and click on the .exe file to install the Khazama AVR Programmer.

Step 3:

We have to setup Fuses bits and Lock bits.

Fuse bits and Lock bits are master registers whose values directly affects functioning of microcontroller.

ATmega32 microcontroller has two fuse bytes namely high fuse and low fuse. Both of them are 8 bits.

In fuse bits 0 means programmed and 1 means not programmed.

The default value of ATmega32 fuse bit is 0x99E1 i.e. high fuse =0x99 and low fuse =0xE1.

Bit No.	7	6	5	4	3	2	1	0
High Fuse	OCDEN	JTAGEN	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST
Default Value	1	0	0	1	1	0	0	1
Bit No.	7	6	5	4	3	2	1	0
Low Fuse	BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
Default Value	1	1	1	0	0	0	0	1

OCDEN : This pin is used to enable or disable on chip debugging. On chip debugging provides a real time emulation of microcontroller when running in target system. By default it is disabled as 1 means not programmed.

JTAGEN : There is a built in JTAG interfaces for debugging. It is enabled in new microcontroller. This is the reason why some newbies say “PORTC of ATmega32 not working!!” Disable it if you are not using JTAG by making JTAGEN bit 1(high).

SPIEN : 0 value (programmed) means Serial programing of ATmega32 is enabled. Don't change this unless you have a parallel programmer! Because once disabled ATmega32 can't be programmed using serial programmer.

EESAVE : If programmed (0) it will save EEPROM from erasing during chip erase else EEPROM would also be erased with flash.

BOOTSZ0 and BOOTSZ1 : These are used to set the boot loader size.

CKSEL [3-0] : These bits are used to select different clock options available.

Device clocking options	CKSEL 3..0
External crystal/ceramic resonator	1111-1010
External low frequency crystal	1001
External RC oscillator	1000-0101
Calibrated Internal RC oscillator	0100-0001
External clock	0000

The default value of CKSEL3..0 is 0001 i.e. internal RC oscillator running at 1 MHz. If you want to add external crystal you need to change these values according to the table above. Some common fuse bits values are given in the end of the article.

CKOPT : The CKOPT Fuse Selects between two different oscillator amplifier modes. When CKOPT is programmed, the oscillator output will oscillate with a full rail-to-rail swing on the output. When not programmed, the oscillator has a smaller output swing. If you are using external crystal oscillator it is better to program CKOPT i.e. CKOPT=0.

BODEN : ATmega32 has an On-chip Brown-out Detection (BOD) circuit for monitoring the VCC level during operation by comparing it to a fixed trigger level. When the BOD is enabled (BODEN programmed), and VCC decreases to a value below the trigger level, the Brown-out Reset is immediately activated. When VCC increases above the trigger level, it starts the microcontroller again.

BODLEVEL : The trigger level for the BOD can be selected by this fuse bit. When programmed (0) the trigger level is 4V and when not programmed (1) the trigger level is 2.7V.

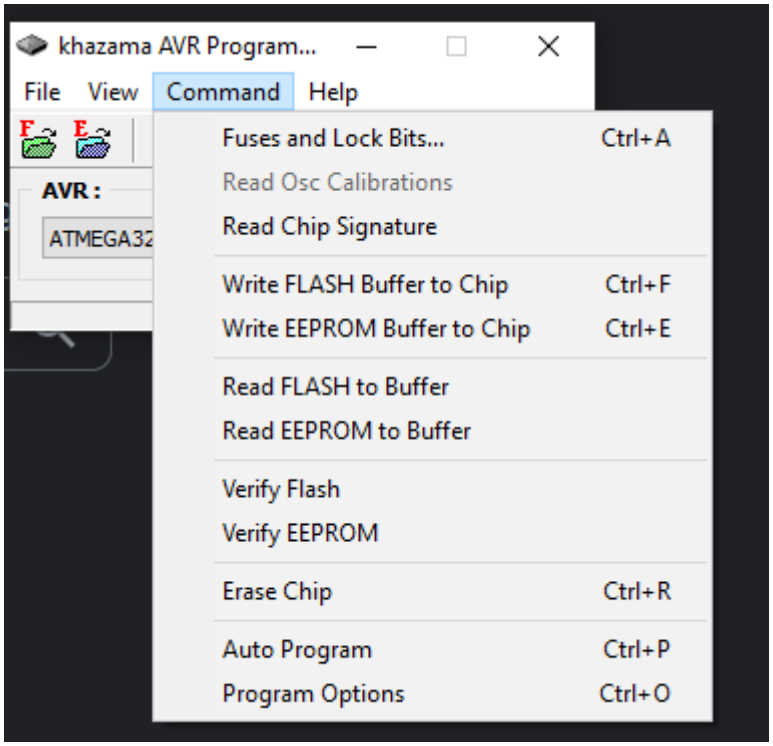
BOOTRST : If BOOTRST bit is programmed (0), the device will jump on first address boot-loader block.

Some fuse bits values are

Low Fuse	High Fuse	Comments
0xE1	0x99	Default, Internal 1MHz
0xE2	0x99	Internal 2MHz, rest all default
0xE3	0x99	Internal 4MHz, rest all default
0xE4	0x99	Internal 8MHz, rest all default
0xEE	0xC9	External 12/16MHz, JTAG disabled
0xE1	0xD9	Internal 1MHz, JTAG disabled
0xE2	0xD9	Internal 2MHz, JTAG disabled
0xE3	0xD9	Internal 4MHz, JTAG disabled
0xE4	0xD9	Internal 8MHz, JTAG disabled

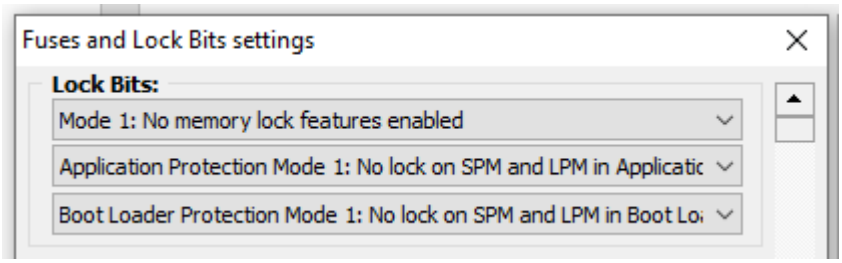
Lock Bits: These bits are very important to lock the microcontroller for the further programming. We have to choose these very carefully.

Run Khazama AVR Programmer and go to command & click on Fuses and Lock Bits for setting the bits.



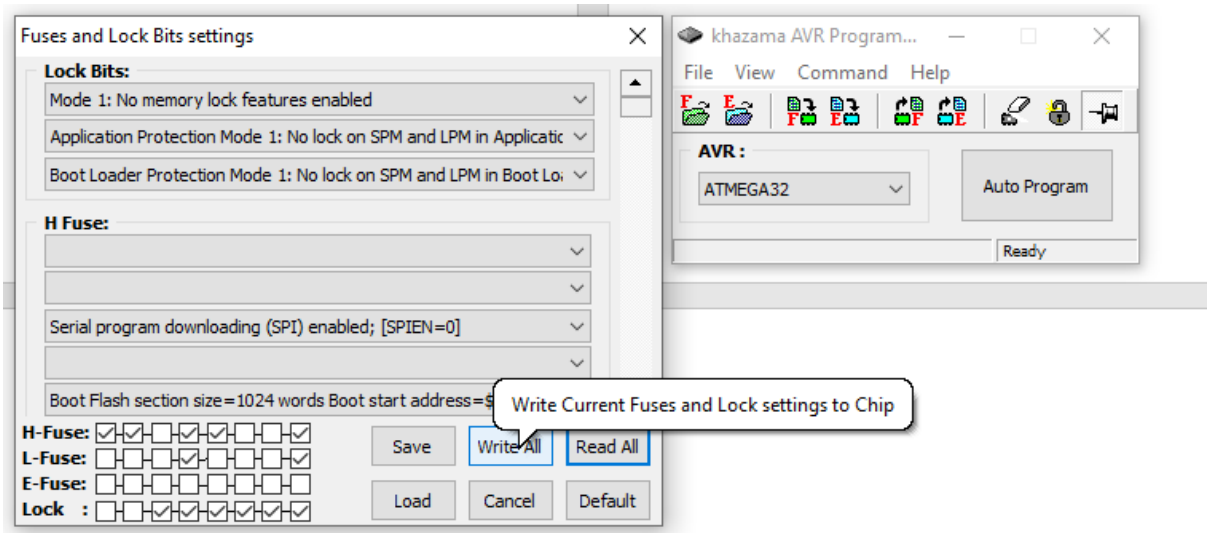
Step 4:

Select the Lock Bits as the following fashion



Step 5:

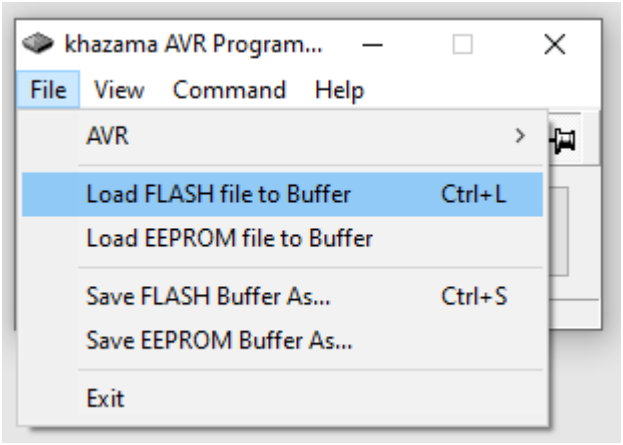
Set H-Fuse and L Fuse as our desired manner. Then click on Write All tab.



Note: To check the previously set Fuse bits and Lock bits just click on Read All tab

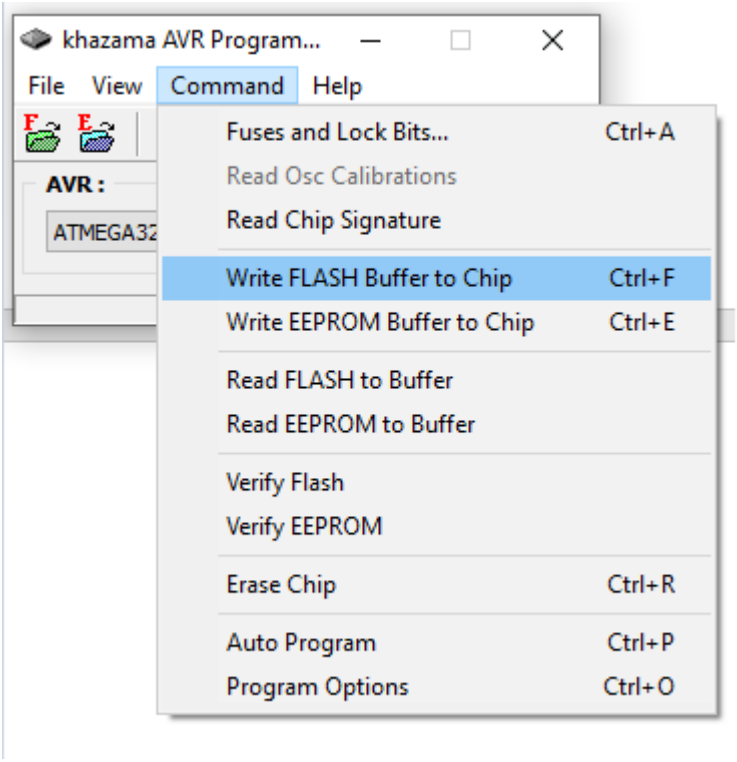
Step 6:

Then to load .hex file into Khazama AVR Programmer click on file & then click on Load FLASH file to Buffer and then select your .hex file.

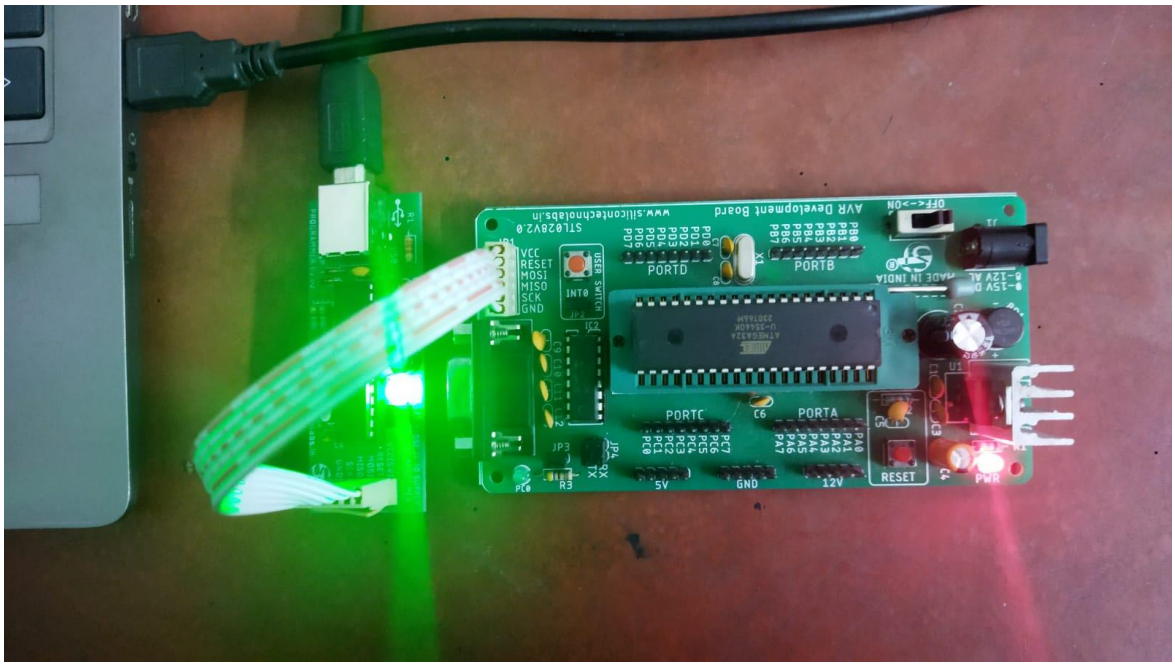


Step 7:

To flash the .hex file into the targeted microcontroller just click on command & then Write FLASH Buffer to Chip



After flashing the code Just press on the Reset button at your Hardware setup to run the Embedded System.



Assignment: Change the duty cycle of the LED Blink with 25% duty cycle and 75% duty cycle.