

How to burn Arduino code to AVR Chips

This works for only Atmega8A, Atmega32A, Atmega32 for now.

Long story short, do the following things and I think you'll be done.

Disclaimer

If you mess up the whole process and brick your chip I won't be responsible. No need to worry though, you can do it if you do as I say.

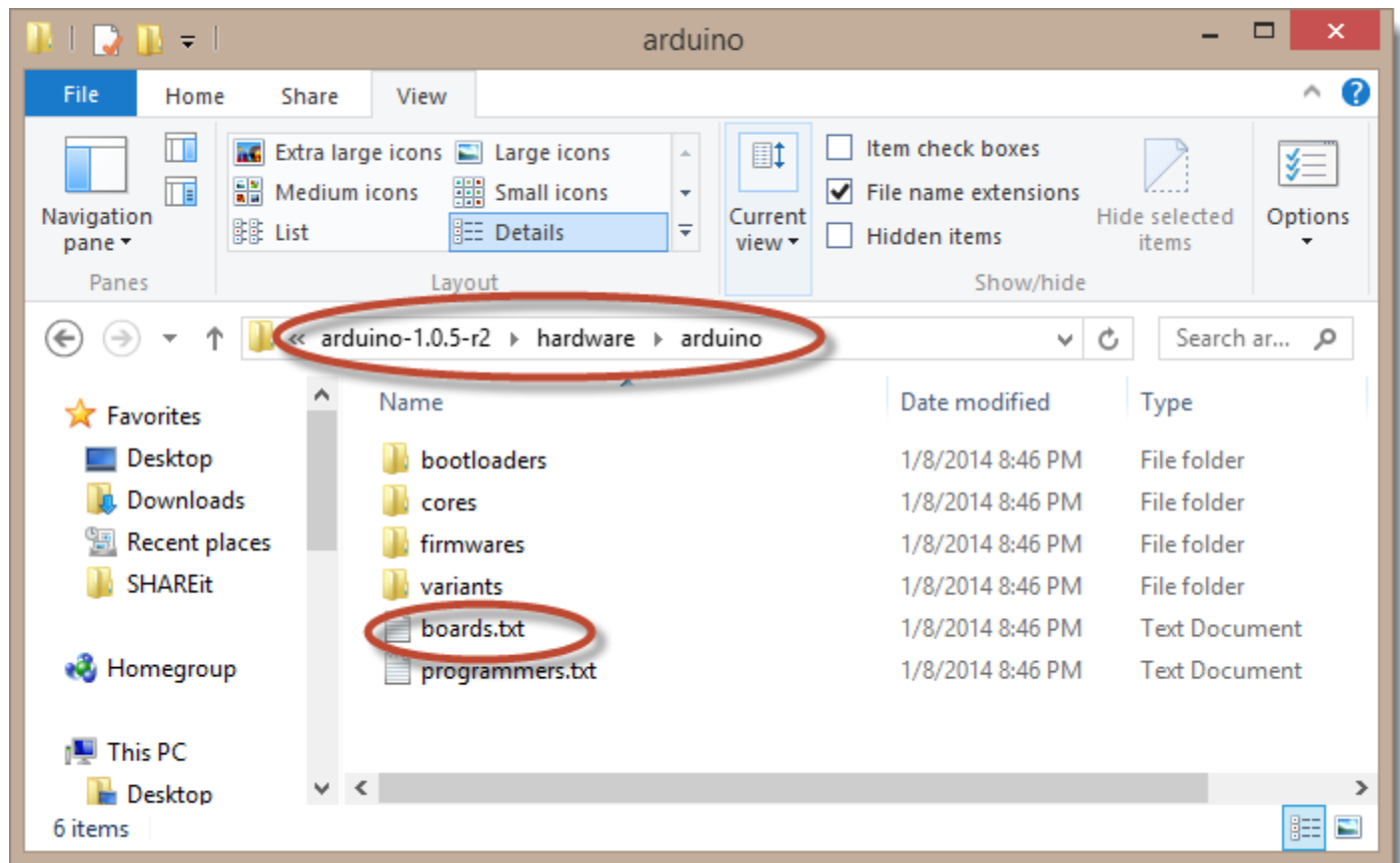
The things you should gather before you proceed:

- An ATmega chip, ATmega8/32/32A/8A should work
- An Arduino IDE (I've tested this on the version 1.0.5-r2)
- A USBasp (Works also with avr uploader as long as it uses USBasp driver it should work) with A-B Usb Cable
- Some premium jumper wires
- A led to test (with small resistance, say 330 Ohm would be perfect)
- Download the boards.txt files and header pin definition files
- **Keep a backup of your boards.txt file!!!**

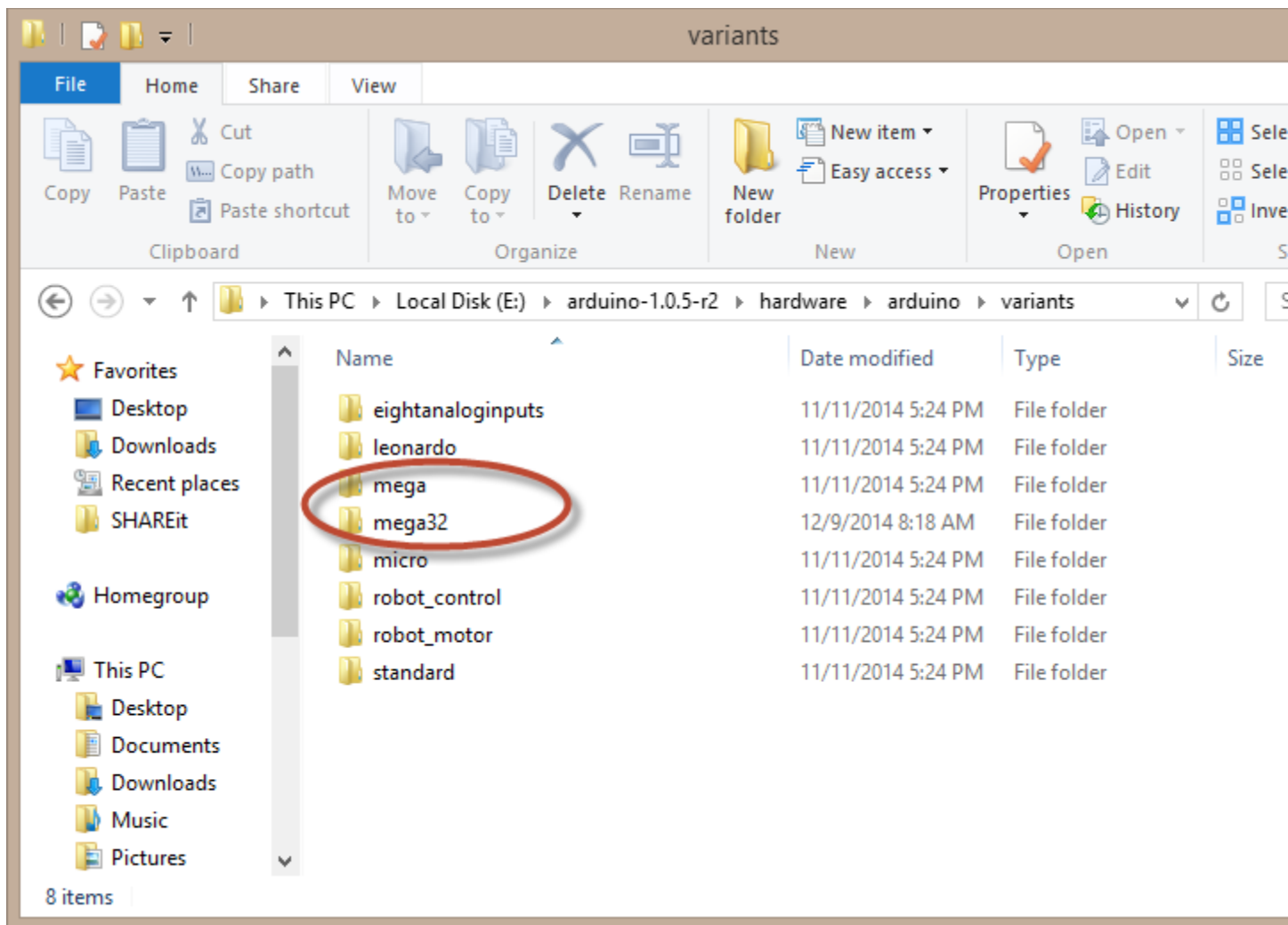
I think that's it, let's follow the procedure.

How to

- If you have your Arduino IDE open, please close it first
- copy the Downloaded boards.txt file and then paste it in your \arduino-1.0.5-r2\hardware\arduino dir, for me it was E:\arduino-1.0.5-r2\hardware\arduino

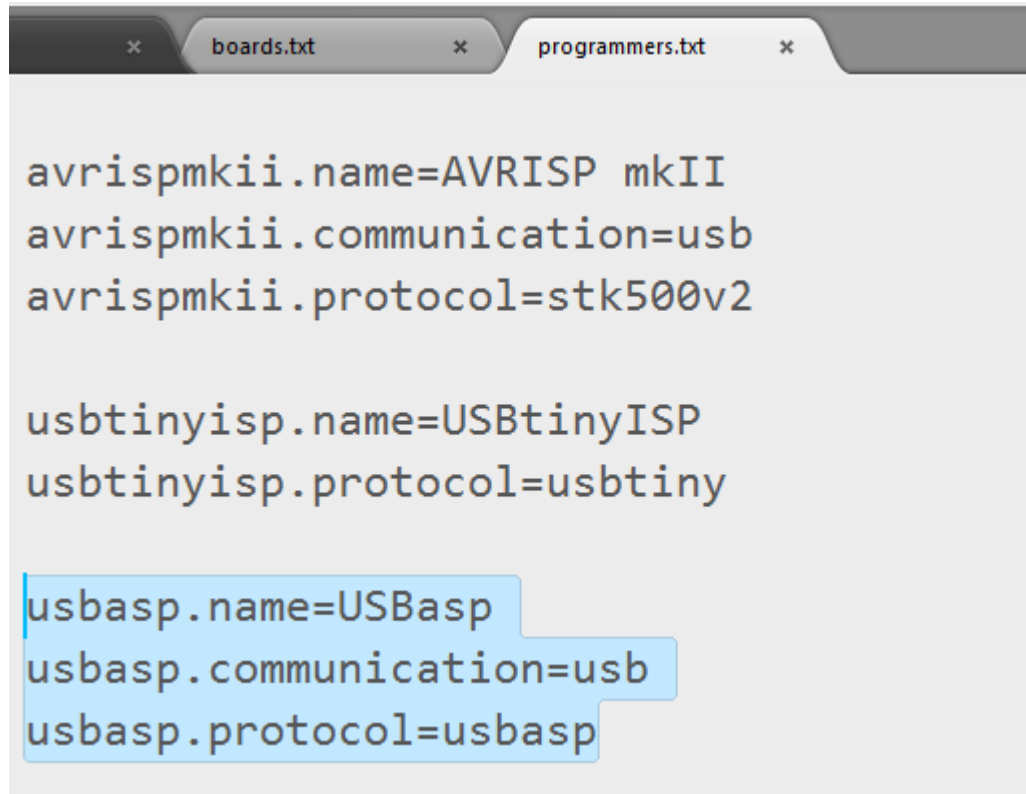


- Now copy the mega and mega32 folder to the \arduino-1.0.5-r2\hardware\arduino\variants dir, for me it was E:\arduino-1.0.5-r2\hardware\arduino\variants [A mega file maybe already there, if it is then replace it anyway]



- Add and save this text to your `programmers.txt` (`dir: \arduino-1.0.5-r2\hardware\arduino`) file if doesn't exist

```
usbasp.name=USBasp
usbasp.communication=usb
usbasp.protocol=usbasp
```

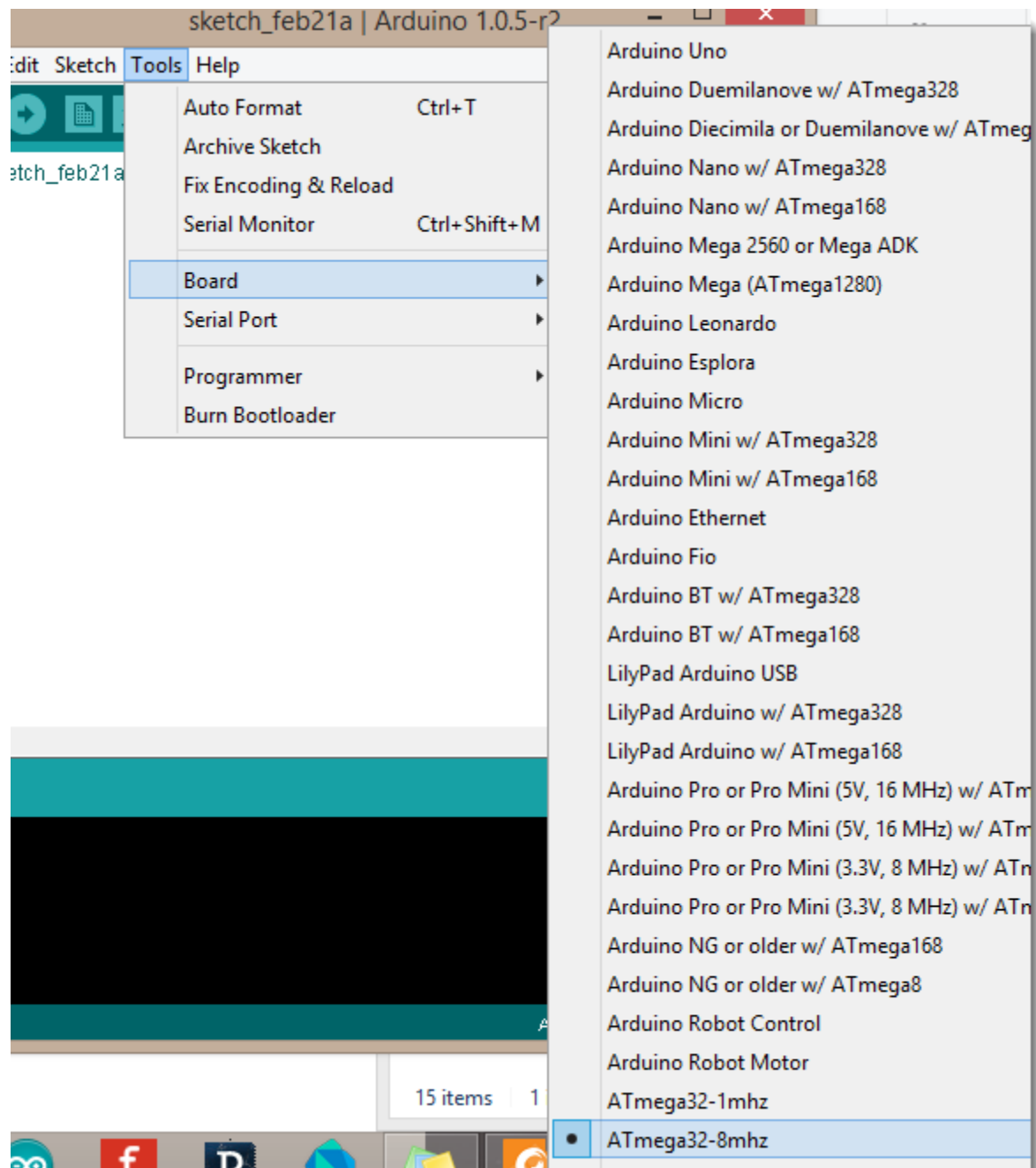
A screenshot of a code editor window with two tabs: 'boards.txt' and 'programmers.txt'. The 'boards.txt' tab is active and contains configuration lines for three boards: 'avrispmkii', 'usbtinyisp', and 'usbasp'. The 'usbasp' section is highlighted with a blue selection box.

```
avrispmkii.name=AVRISP mkII
avrispmkii.communication=usb
avrispmkii.protocol=stk500v2

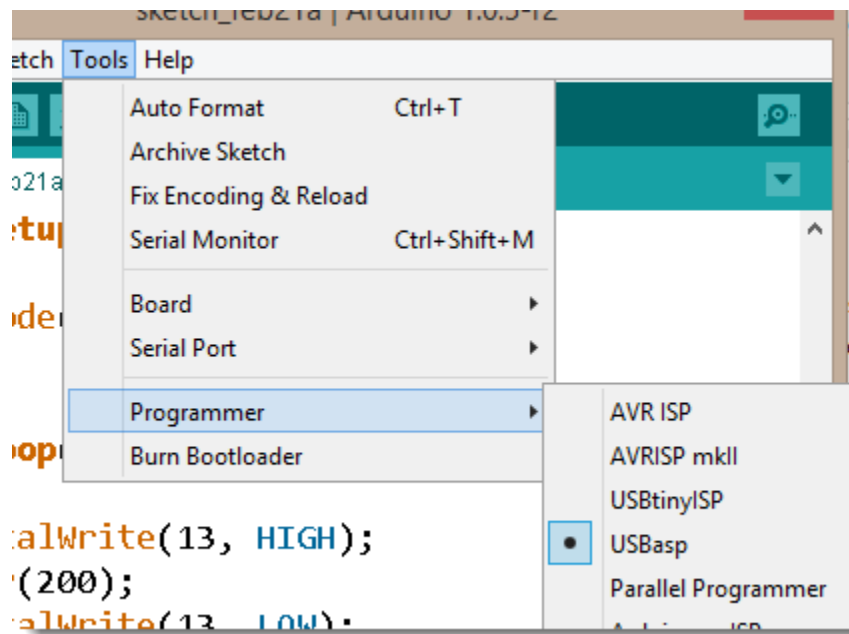
usbtinyisp.name=USBtinyISP
usbtinyisp.protocol=usbtiny

usbasp.name=USBasp
usbasp.communication=usb
usbasp.protocol=usbasp
```

- Now Open the `Arduino IDE` and select your MCU



- Now select the programmer (USBasp)

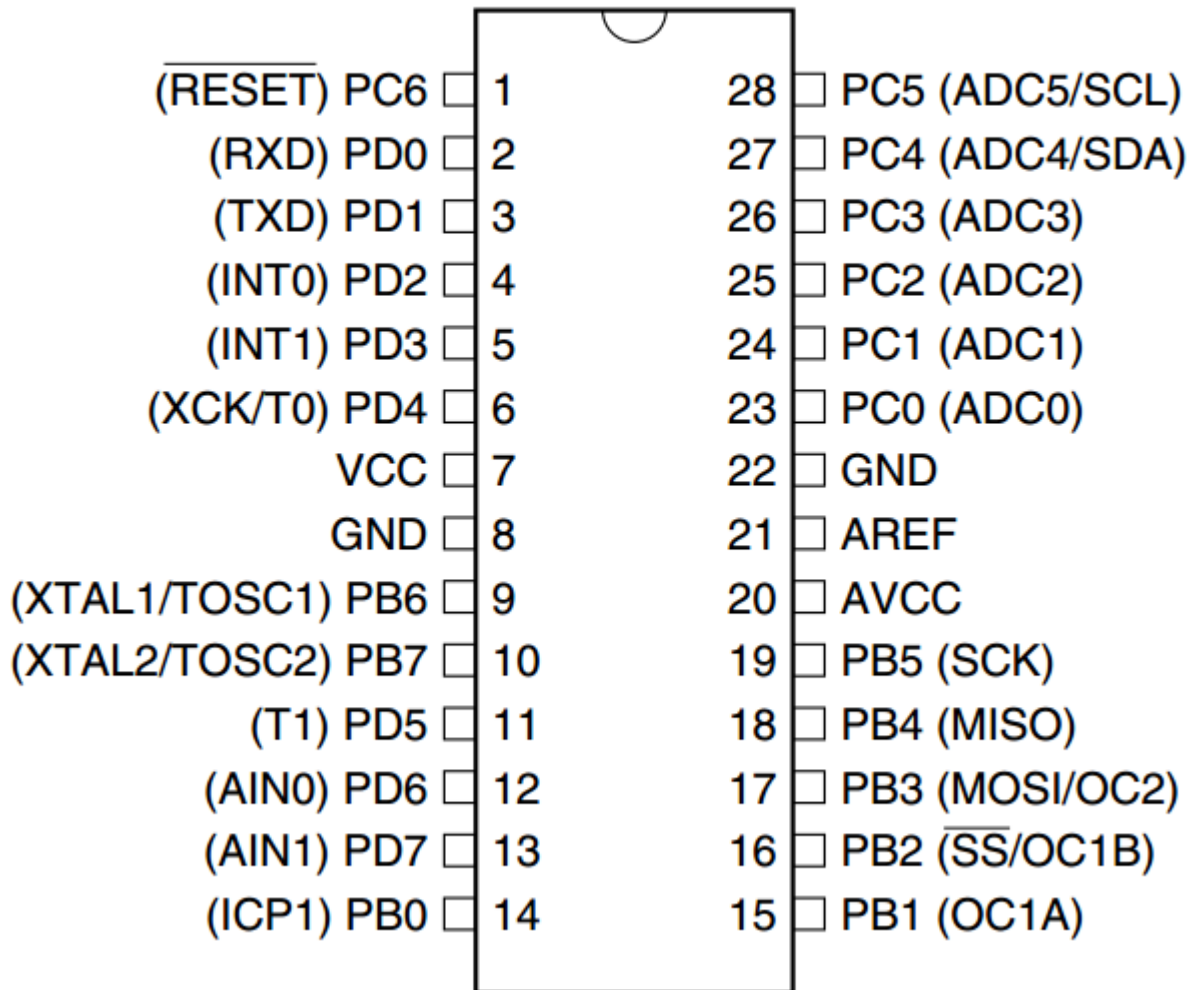


- Connect your USBasp to your pc and connect the following pins from USBasp to your chip
 - MISO
 - MOSI
 - SCK
 - RESET
 - VCC
 - **GND Make sure you short the AVCC and VCC together and put +5V and short the other GND and GND together and connect GROUND**

For ATmega32

Follow this image to find out the necessary pins

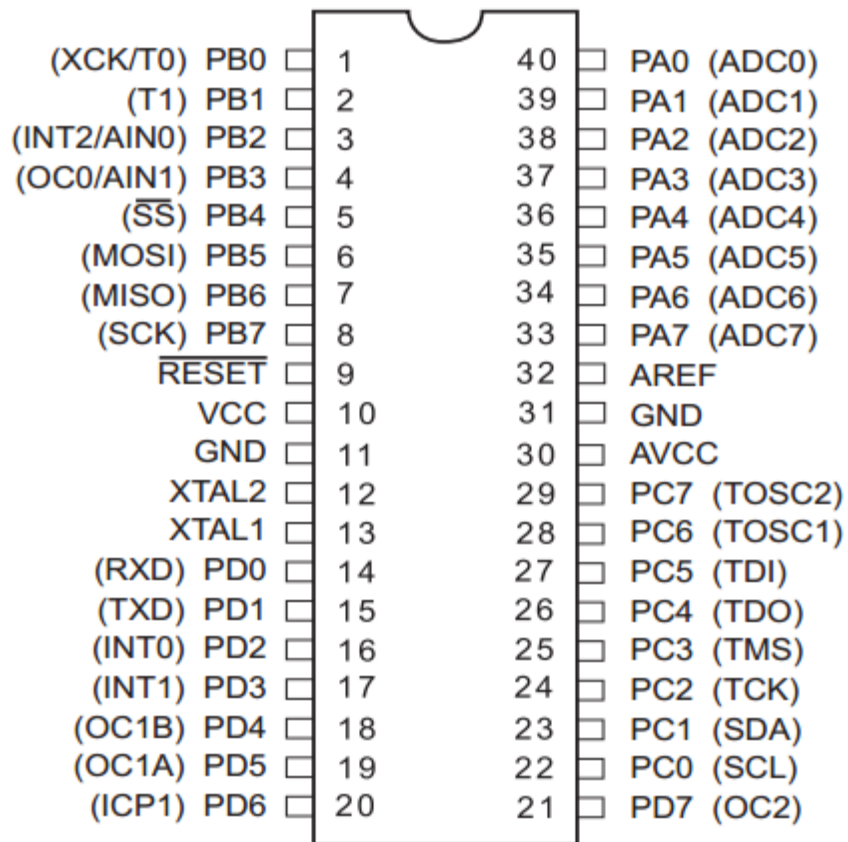
PDIP



For Atmega8

Follow this image to find out the necessary pins

PDIP



Now you've connected all the things together click upload, if it uploads without fail then you're done!

- If you want to test a led, follow the uploaded pinout diagram to find out which is the 13 pin on your MCU or other pins (you get the idea, don't you?) then connect it and test it

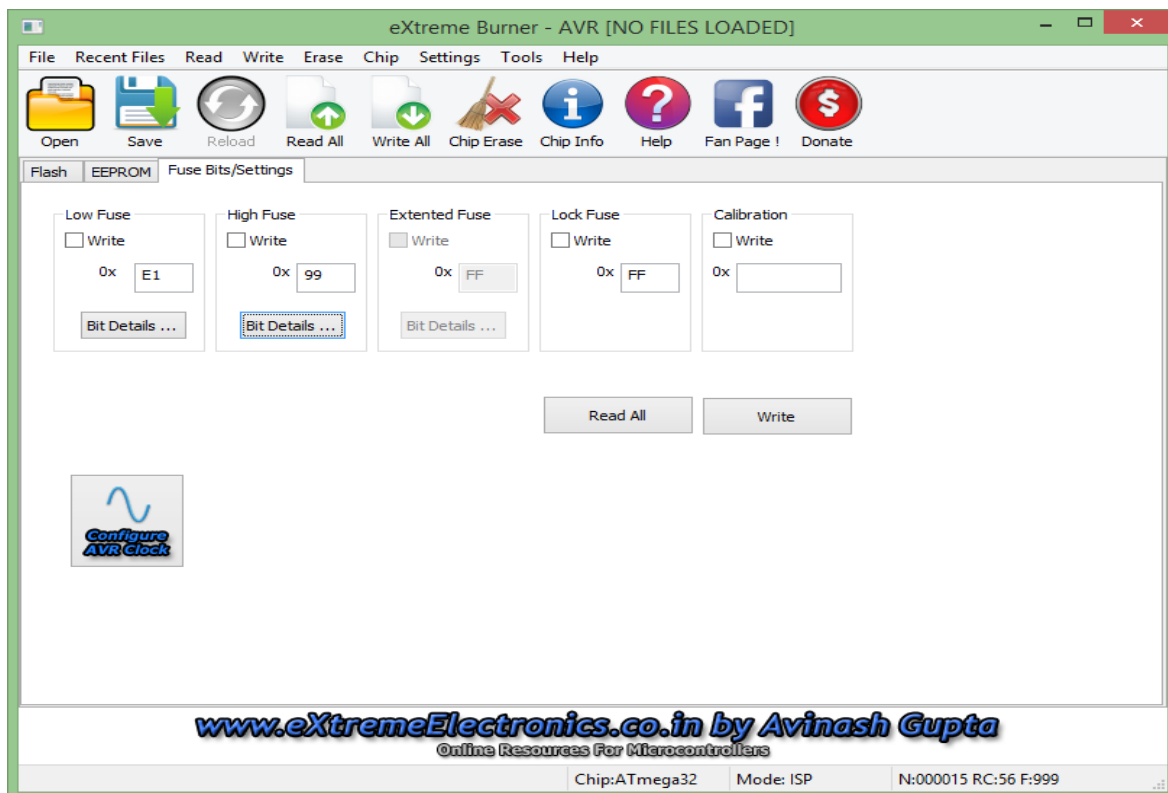
Setting the CLOCK

While burning the Arduino code in AVR chip then it's a problem in clock for timing and I found a solution through this way Just changing the fuse bit of AVR microcontroller....

Changing fuse bit we need these....

1. eXtreme Burner
2. 16MHz crystal clock
3. 22pF capacitor

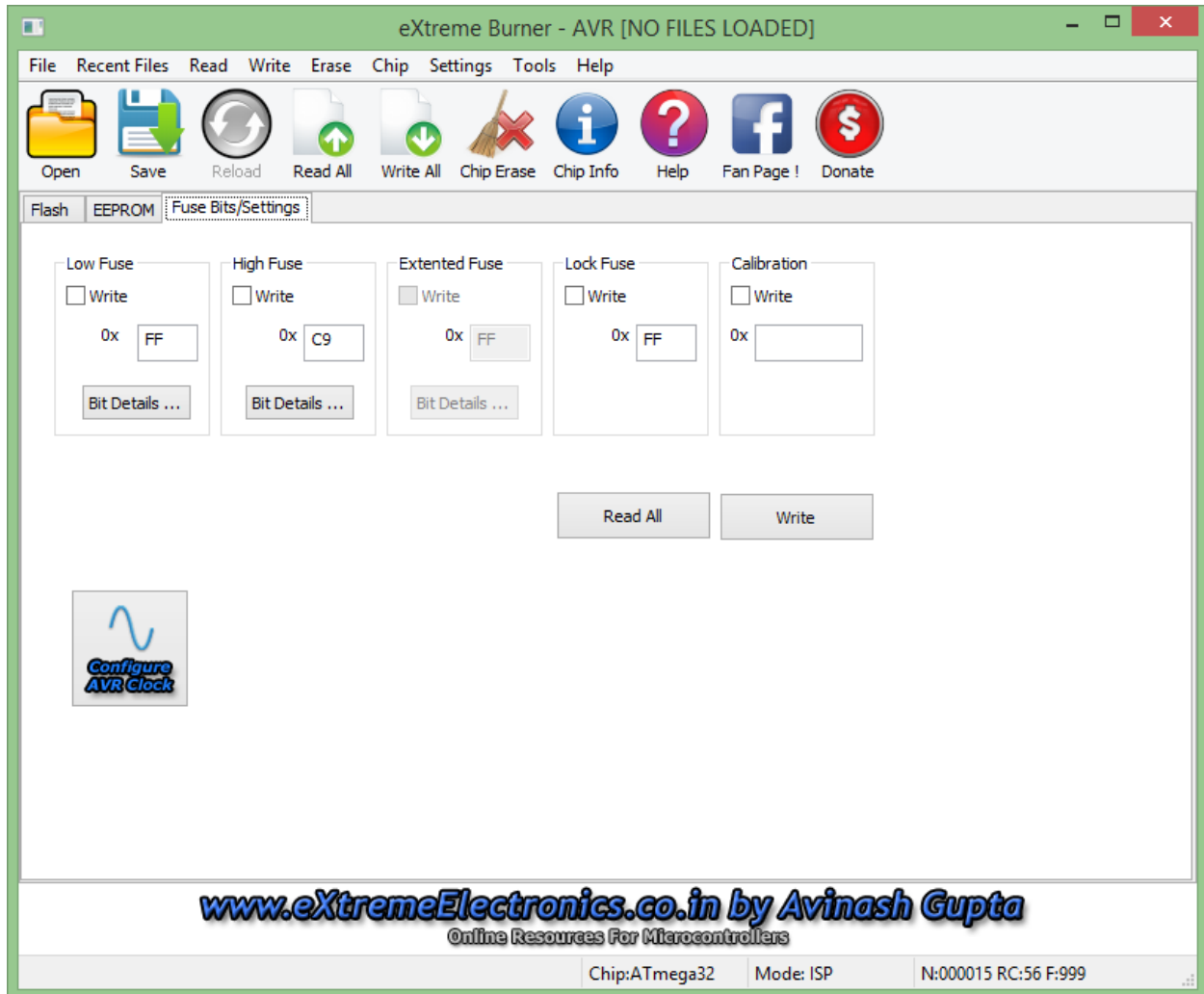
Here I change the Fuse bit of ATmega32 Atfirst connect the MCU with usbasp then by default Fuse bits are like this..



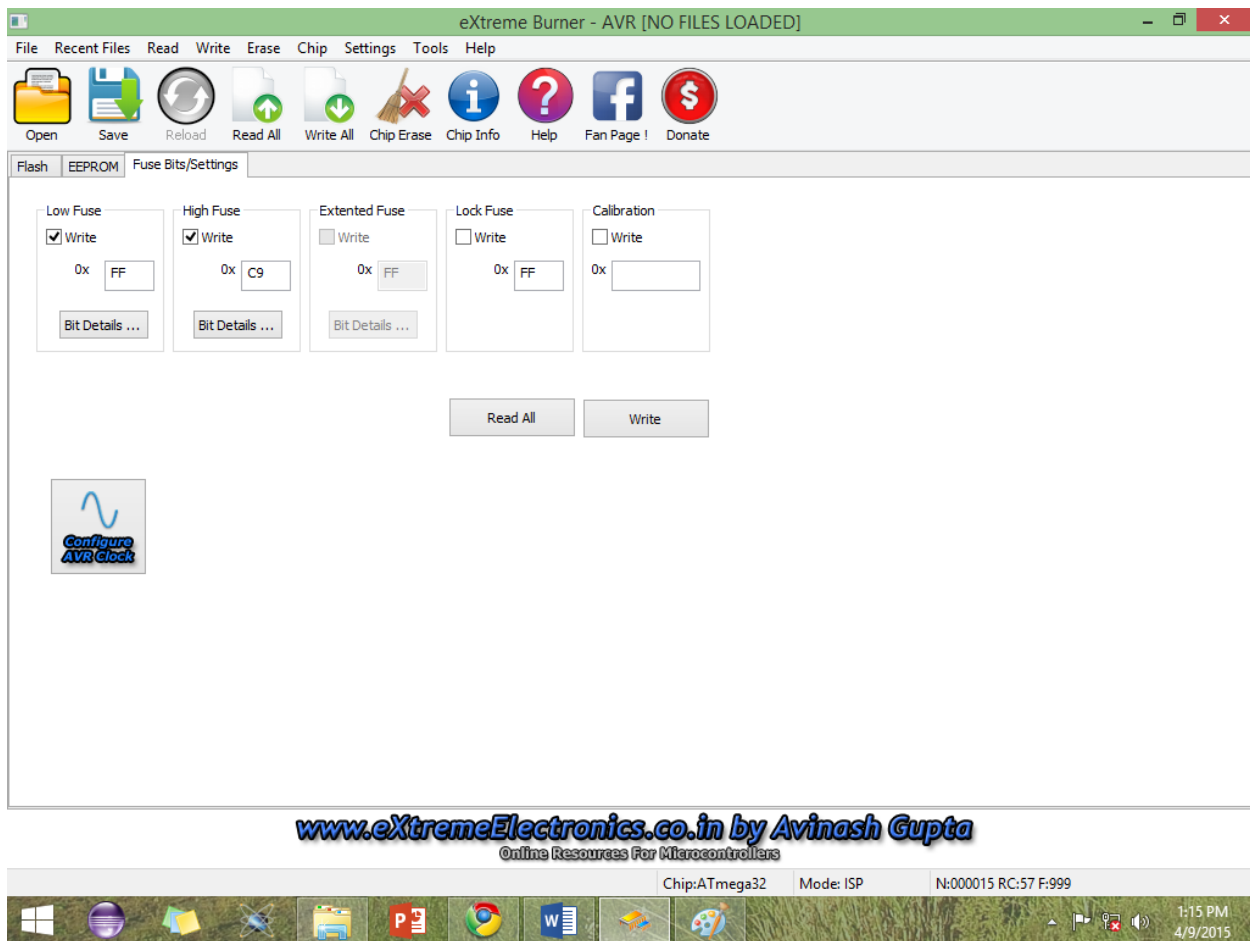
Now changing the HIGH and LOW Fuse bits like this

LOW = 0xFF

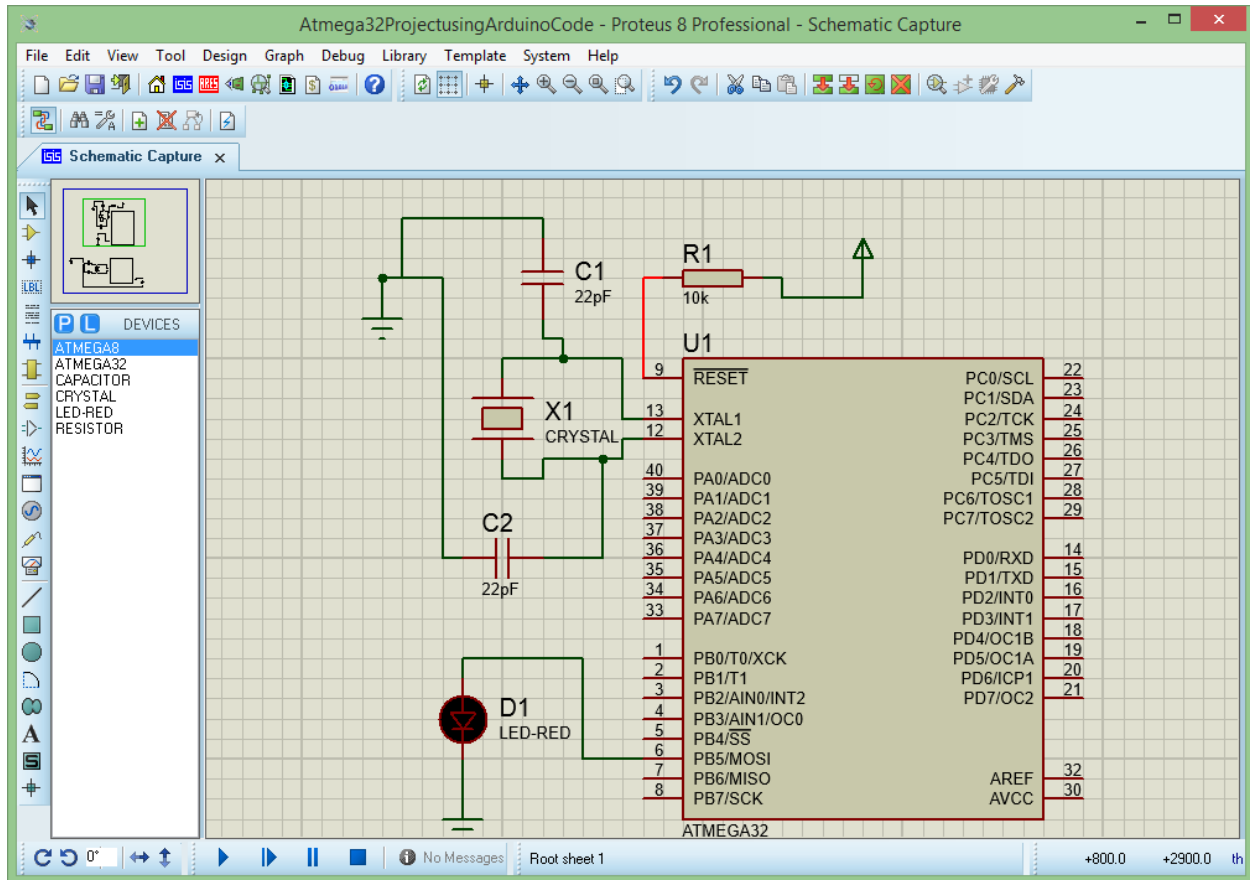
HIGH = 0xC9 according to the following figure



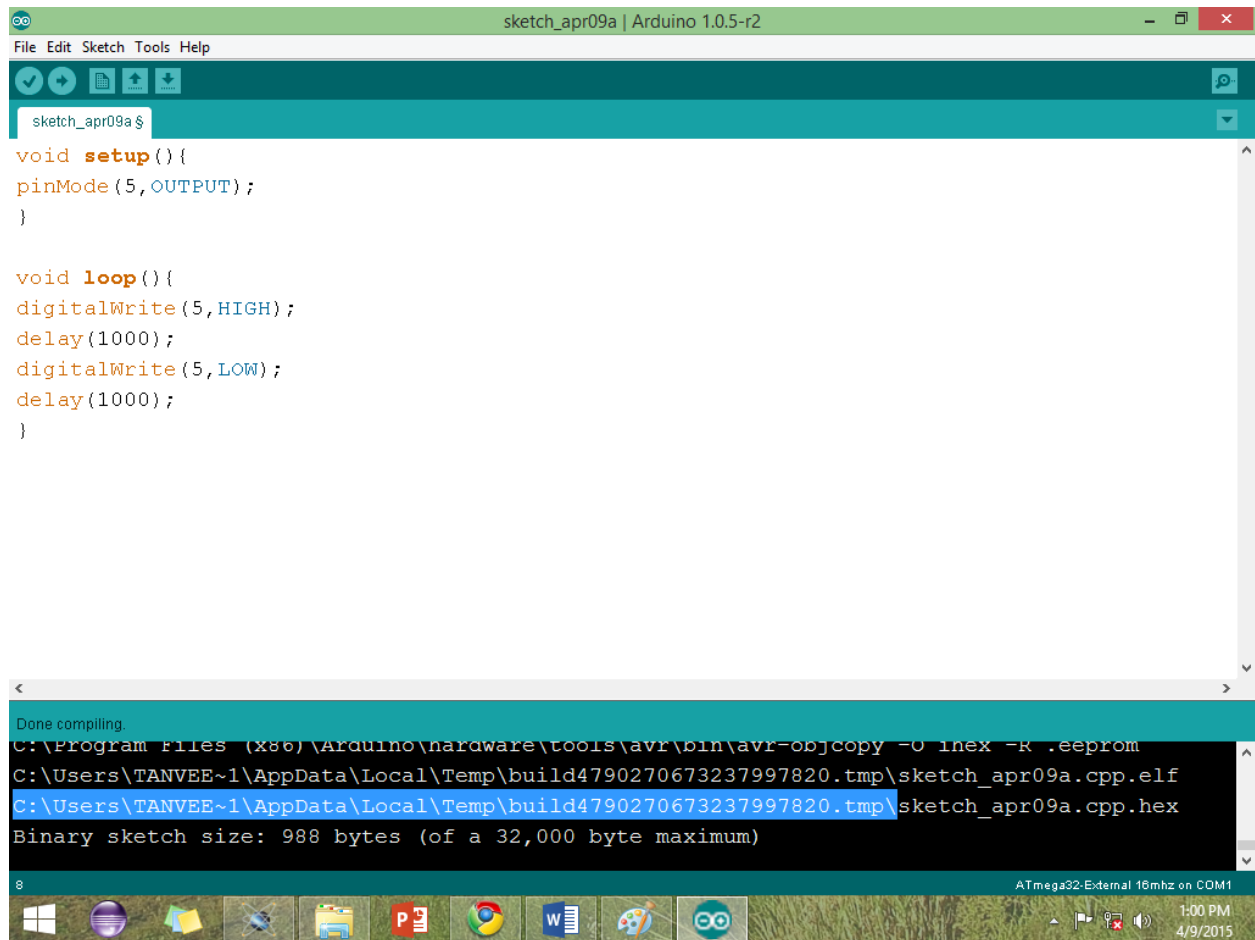
Now writing the fuse bits to the MCU just mark both Write check boxes of Low Fuse and High Fuse part....



Now we are done ..Its the time to implement the hardware setup like the following proteus simulation..



The hex file is generated in this directory..



The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for opening, saving, and running. The main text area contains the following C++ code:

```
void setup() {  
  pinMode(5, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(5, HIGH);  
  delay(1000);  
  digitalWrite(5, LOW);  
  delay(1000);  
}
```

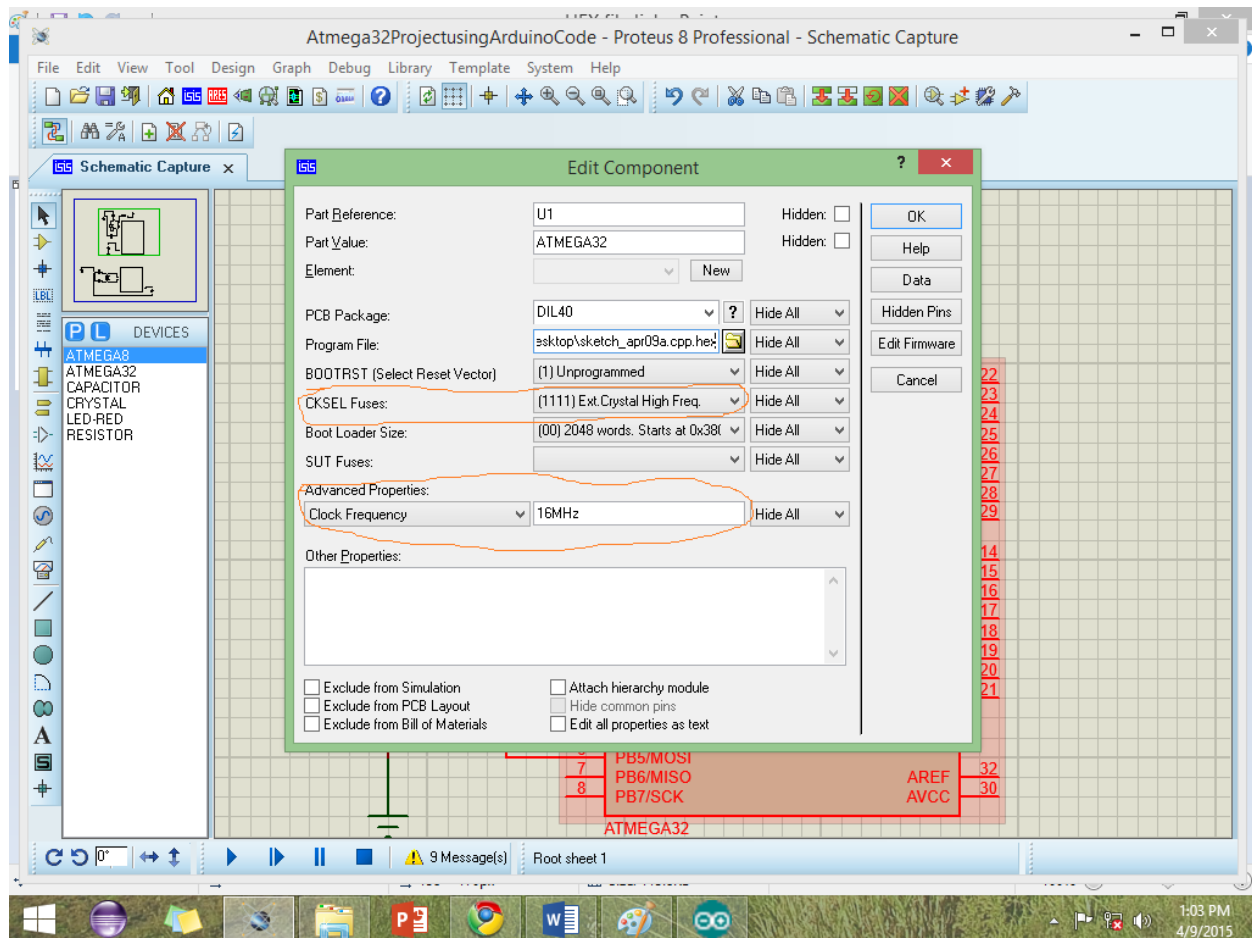
The bottom status bar shows 'Done compiling.' and the following command line output:

```
C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avr-objcopy -O hex -R .eeprom  
C:\Users\TANVEE~1\AppData\Local\Temp\build4790270673237997820.tmp\sketch_apr09a.cpp.elf  
C:\Users\TANVEE~1\AppData\Local\Temp\build4790270673237997820.tmp\sketch_apr09a.cpp.hex  
Binary sketch size: 988 bytes (of a 32,000 byte maximum)
```

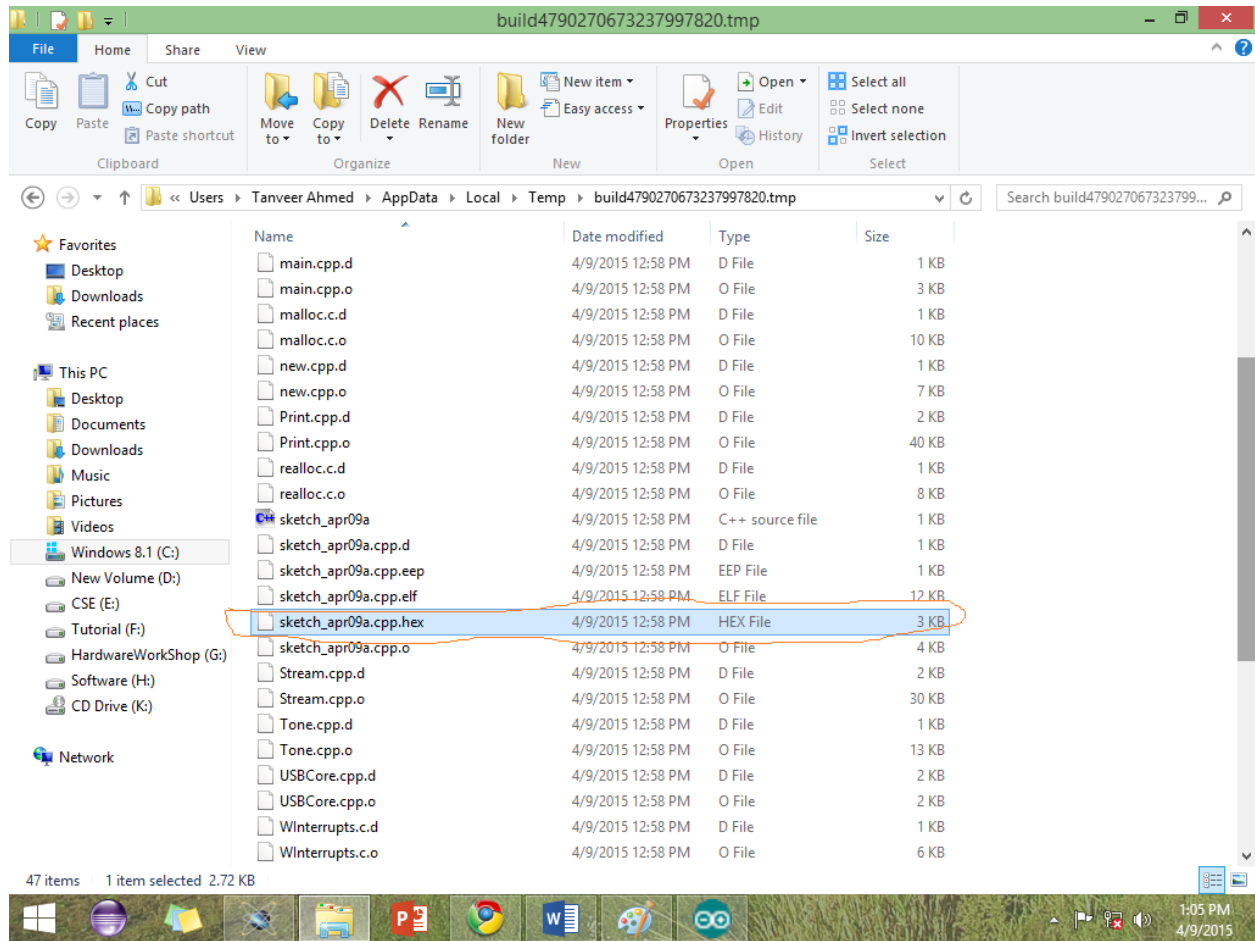
The bottom status bar also displays '8' and 'ATmega32-External 16mhz on COM1'. The Windows taskbar at the bottom shows various application icons and the system clock indicating 1:00 PM on 4/9/2015.

CHANGING FUSE BITS IN PROTEUS

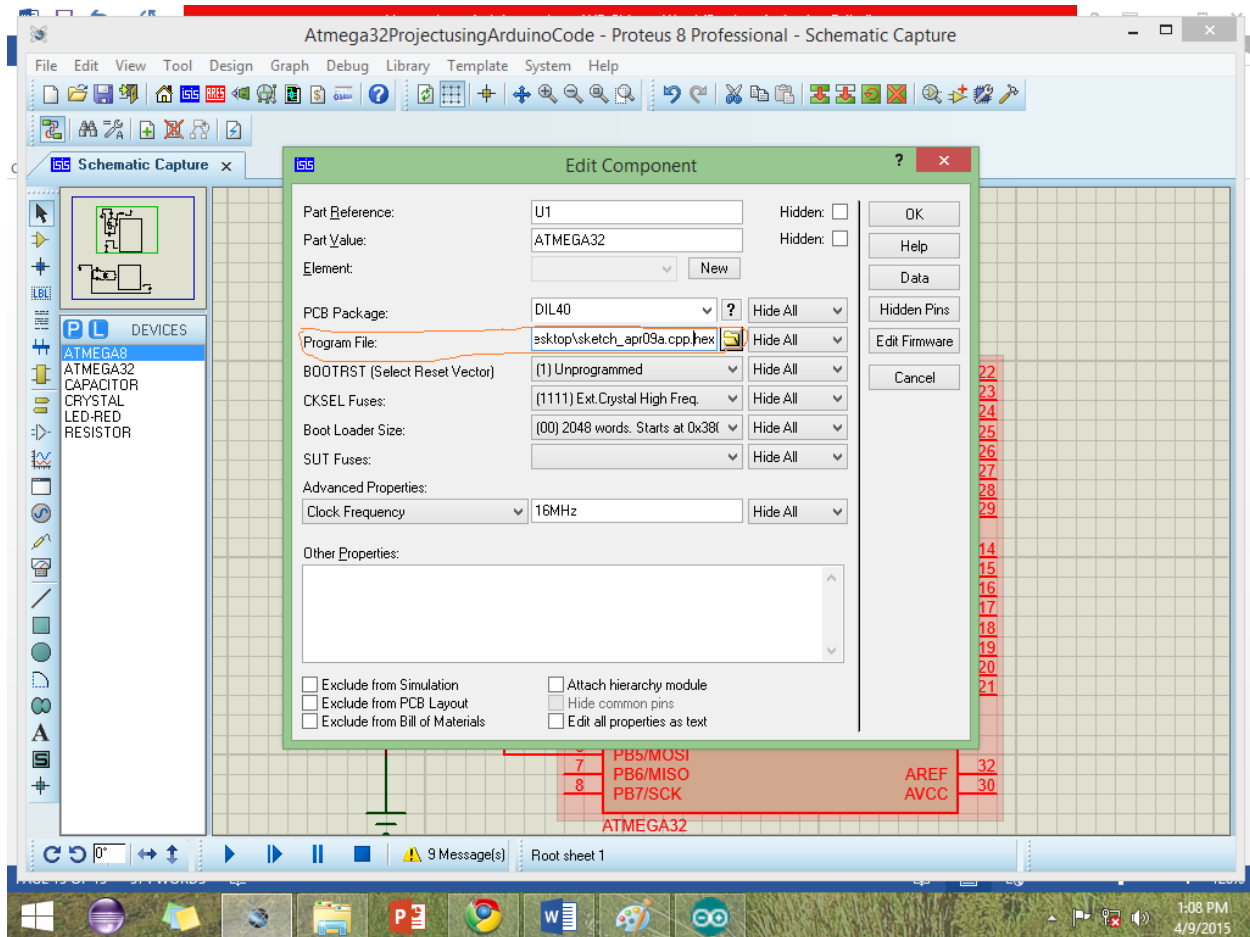
Now going to that directory we get the hex file and upload it to the MCU.. In proteus simulation we can change the fuse bits like the following...



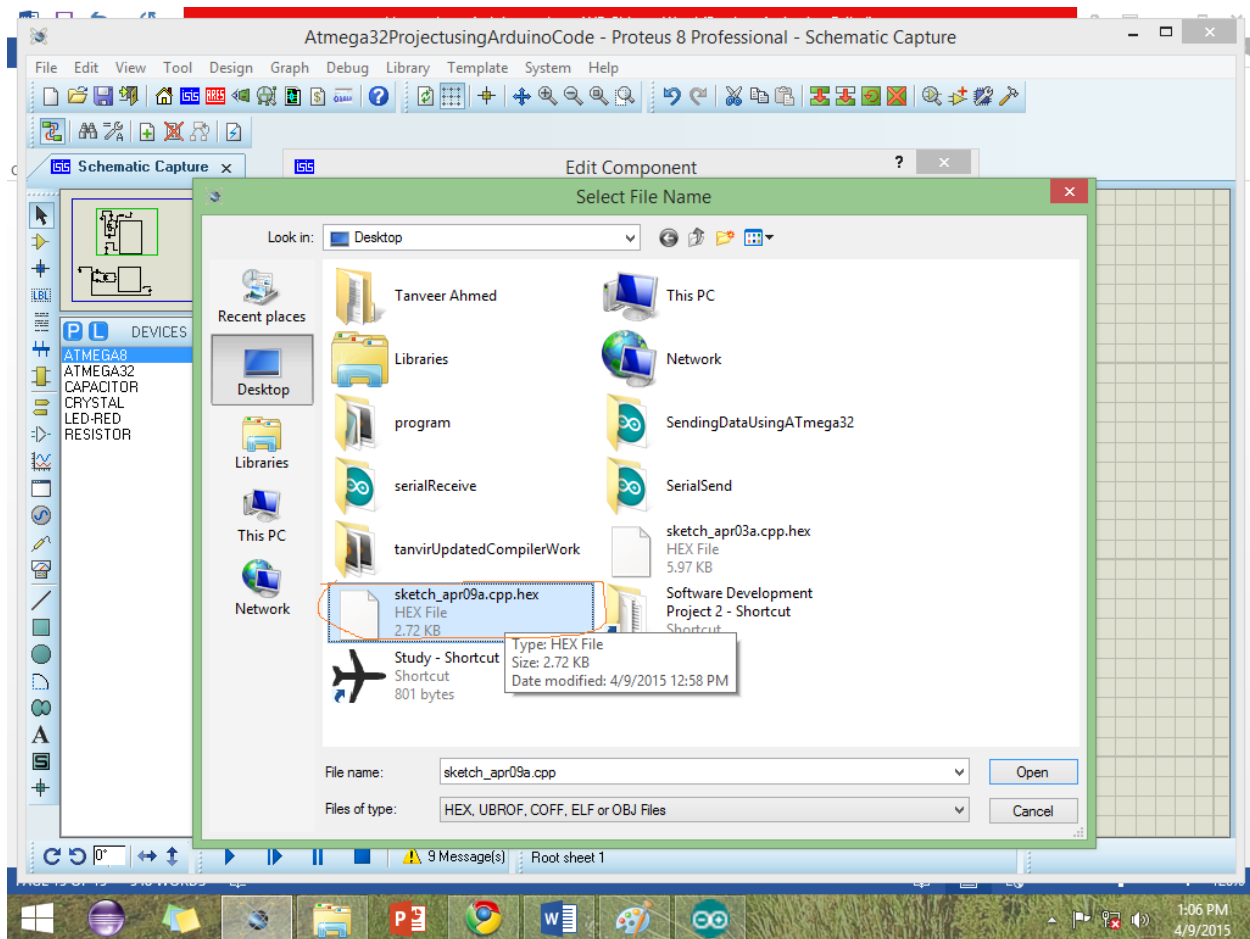
Now uploading the hex file to the MCU...



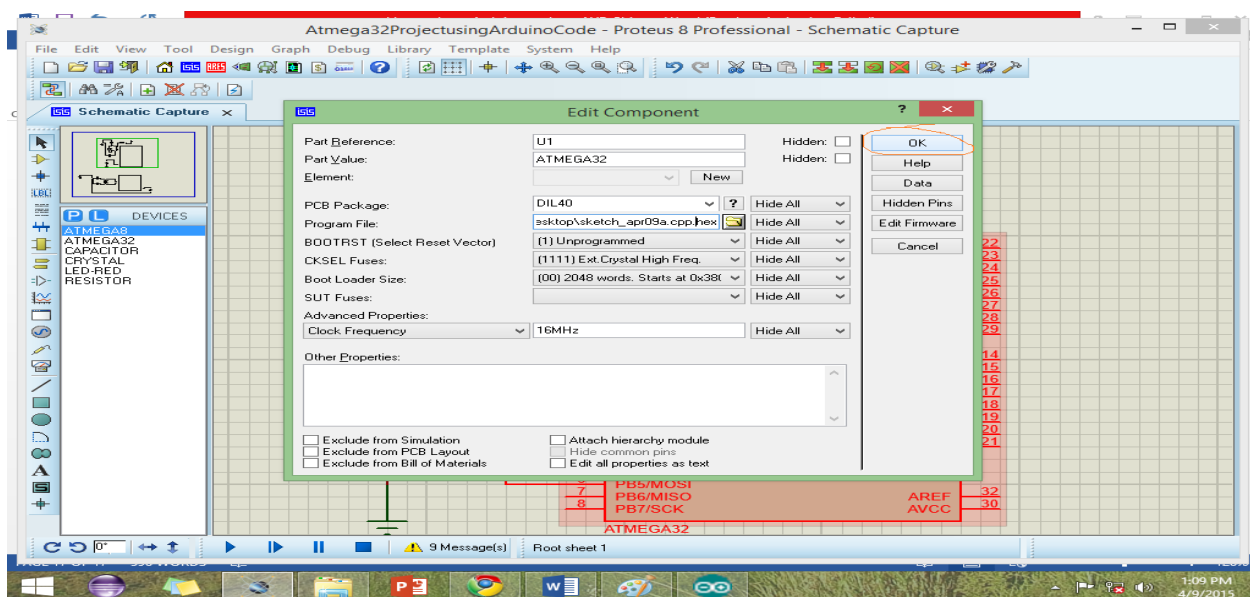
At first I copy the hex file in desktop so the I can get the hex file as soon as possible to get that ... Now I insert the hex file



Now I go to my desktop to get the hex file....



Now upload the hex file in the MCU.....



Now it's the time to see what happened to our MCU.....

