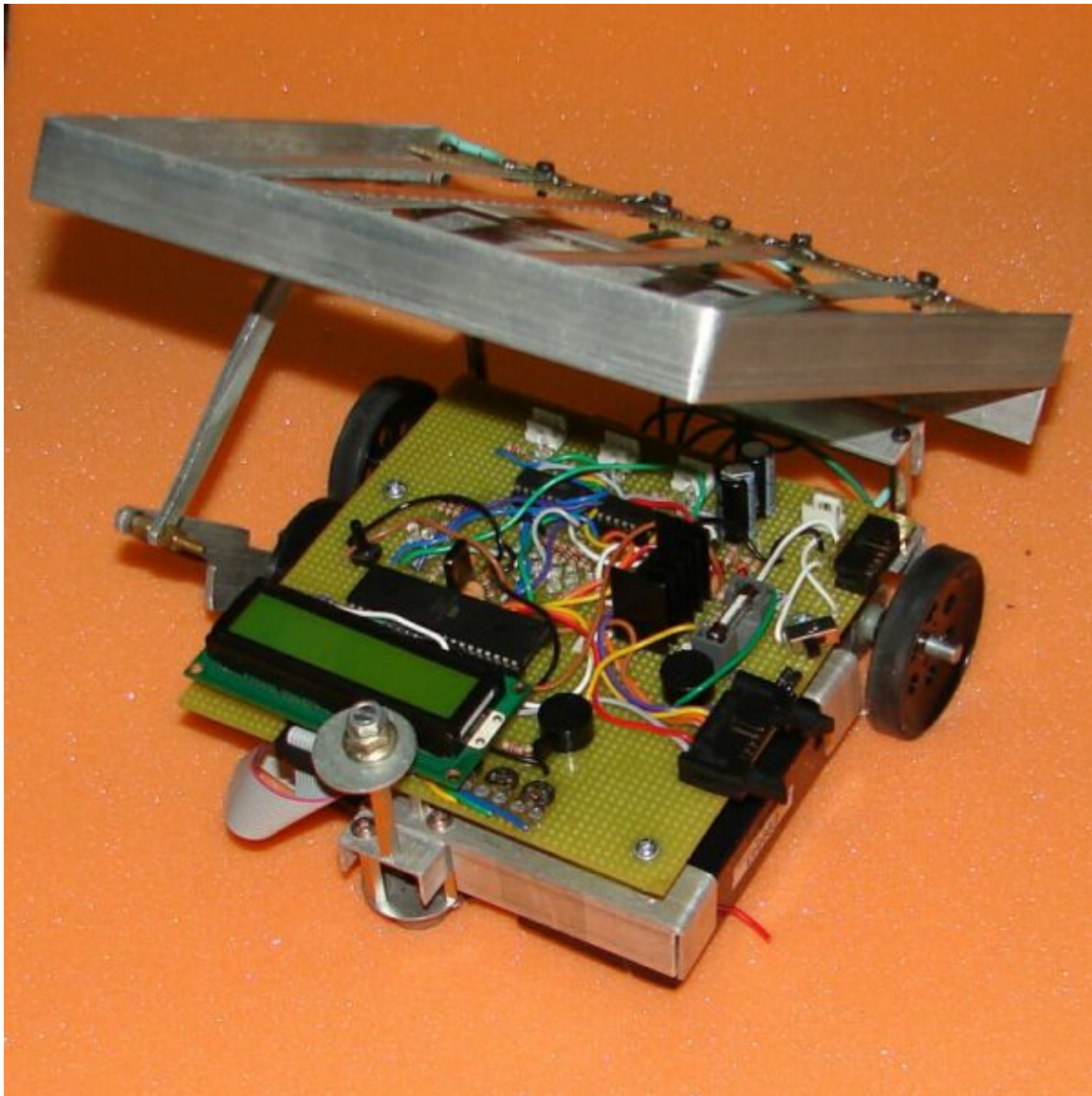


Building Low Cost Autonomous Robots

Chapter 3: Robot Testing



Sachitanand Malewar
Director, NEX Robotics

<http://www.nex-robotics.com/sachitanand.php>

CHAPTER 3

ROBOT TESTING

In this chapter we are going to test our assembled robot. For this we will need two additional things.

1. Power supply for board testing and battery charging
2. Programmer for the robot
3. Battery charger circuit.

Power Supply Construction:

First we are going to build power supply for the robot. We are going to test robot using external power supply. If we use lead acid battery in initial testing then in the case of short circuit large amount of current can flow which can burn all the connecting cables.

Determining transformer rating:

Our battery has charging current limit is of 0.32 Amp.

Our robot requires maximum current of 1 Amp.

If we consider transformer or AC adaptor from market with 1A. rating probably it will be of 0.5Amp capacity. So we have to get at least 2 Amp rating transformer to get 1A. current capacity. Another option is to use old adaptors of some expensive gadget like old HP610 printer. Adapters which come with some good quality gadgets (which are not made in China☺) will have true rating written on them as this current rating is not there selling point.

Unfortunately if you have to buy Chinese adaptor get adaptor with only 2Amp rating. Open the adaptor you will find that it has bridge rectifier consisting only 1N4007 diode. This has 1 Amp peak rating. It will have only 1000uF capacitor with 16V rating. We have to open this adaptor and throw away its rectifier and place your own rectifier. In this rectifier we will use two 1N4007 rectifier bridges in parallel so that it can sink 2Amp peak current. Use 2200uF capacitor with voltage rating of at least 25V. Avoid buying capacitors on which Samsung or any other brand. I like to use orange colored KELTRON (Karla Electronics) capacitors. This is Indian company which never compromises in quality.



Figure 3.1: Building your own power supply

Never ever use small SMPS as in many SMPS designs to reduce cost resistive feedback is taken from 230V AC section. They have very small leakage current. It can damage your microcontroller very easily. You can check this by standing on wet floor and touching your wet finger on the negative terminal of the DC output. If you feel tingle at your finger tips then SMPS might be leaking 230V AC in to DC supply line.

Building programmer for ATMEGA16 microcontroller:

We are using ATMEGA16 microcontroller for our robot. This microcontroller supports In System Programming (ISP). It means that you don't need to remove microcontroller for programming, it can be programmed by connecting external programmer. We are going to build a simple programmer for this microcontroller based on parallel port of the PC.

All the AVR series microcontrollers support In System Programming (ISP). SPI bus of the microcontroller along with RESET pin is used for the programming.

Pins involved in the ISP process are as followed:

1. Vcc: It is supply provided from our microcontroller board to ISP programmer
2. Ground: It is common ground between our board and ISP
3. RESET: It is the RESET pin of the ATMEGA16 microcontroller
4. MOSI: Master Out Slave In of the microcontroller
5. MISO: Master in Slave Out of the microcontroller
6. SCK: SCK pin of the microcontroller.

SPI bus is a serial bus which is used for serial communication between multiple microcontrollers. Same bus also contains necessary logic for ISP. When RESET pin is pulled up to Vcc all the pins of the microcontroller becomes tristated i.e. isolated except the pins of SPI bus. When RESET is pulled up this bus is used for loading “.HEX” file in to microcontroller's flash.

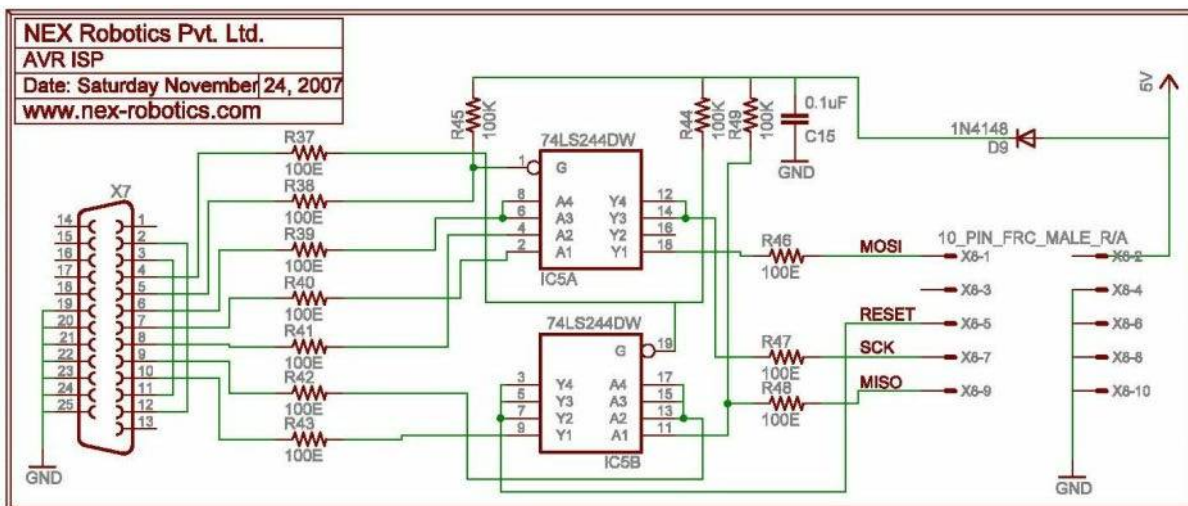


Figure 3.2: Circuit diagram of the Parallel port AVR ISP AVR

In the above circuit connector X7 is the DB25 male printer port connector. Resistors R37-R43 are used to protect printer port. IC 74 244 is used as buffer. R46-R48 are used to prevent excess of current flowing through ISP ports. Capacitor C15 is used for noise suppression. D9 is used to protect circuit from reverse polarity. Connector X6 is 10 pin FRC right angle connector. You have to make sure that length of the programming cable should no exceed 25cm else programmer becomes unstable. This is because of line capacitance ringing starts occurring at SCK line.



Figure 3.3: In System Programmer for AVR microcontrollers from NEX Robotics

Battery Charger Circuit:

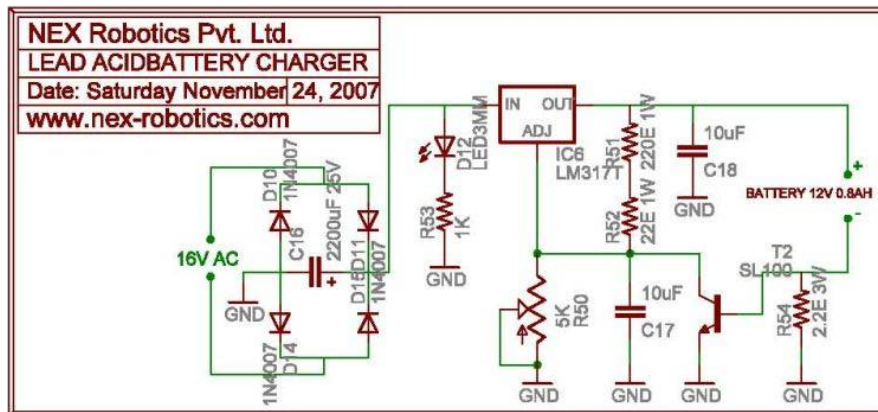


Figure 3.4: Circuit diagram of Lead Acid Battery charger

Lead acid battery with 12V rating when fully charged gives about 13.8V and almost discharged gives about 11.2V. Our battery has current rating of 0.8Ah and its maximum charging current is 0.32Ah. To charge our battery we have to make a charger in such a way that it gives 13.8V regulated output voltage and its maximum current is limited at 0.32Amp. In the above charger LM317 variable voltage regulator is used. It is tuned at 13.8V using potentiometer R50. To limit its maximum charging current below 0.32A transistor T2 and R54 are used. R54 is 2.2 ohm 3 watt resistor. When current reaches above 0.318A voltage across the resistor R54 cross 0.7V which turns on transistor T2. It shorts R50 and reduces voltage in such a way that current is kept below 0.318A. When a fully discharged battery is connected to this charger initially current limiter kicks in to action. Battery voltage gradually increases while current is limited below 0.32A.

Once voltage is reached at 13.8V current limit circuit is turned off and voltage is maintained at 13.8V.

Programming the robot:

We can use Open source IDEs (Integrated Development Environment) like WinAVR, AVR GCC, AVR Studio or proprietary IDEs like ICC AVR, IAR, and Code Vision AVR etc.

We are going to use proprietary IDE ICC AVR which you can download from <http://www.imagecraft.com>. It's a 45 days evaluation version which will work as full version for 45 days. Advantage of this IDE is that it has code generator with it so we don't have to know microcontroller to program it. This statement will get clearer as we proceed further. Once we get familiar with the microcontroller we can switch over to free IDEs

Introduction to ICC AVR:

Setting Up a New Project

Step 1: Setting default project options

In this step we are going to tell the IDE that from now on we are going to use ATMEGA16 microcontroller.

Run ICCAVR and then click on the Project Tab and select "Options"

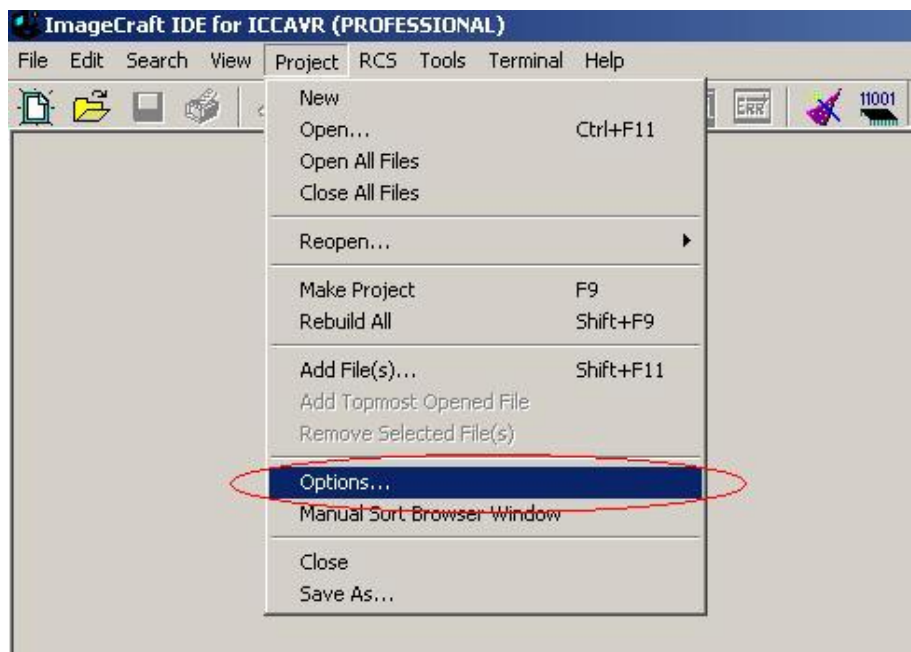


Figure 3.5

It will open Compiler options. Select ATMEGA16 microcontroller in device configuration. Then select set as default and close the window.

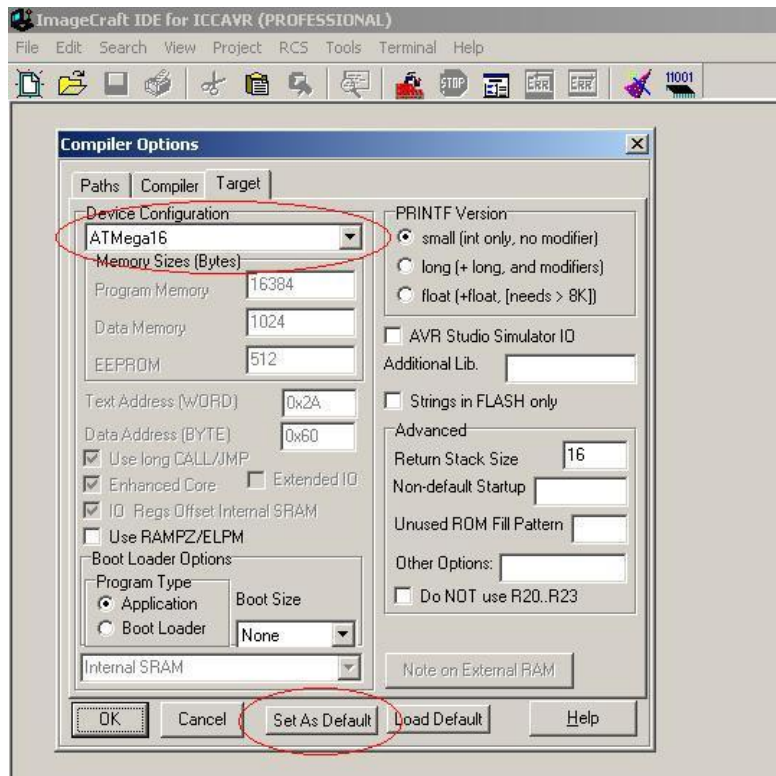


Figure 3.6

Step 2:

Create a Folder where all files related to this New Project will be stored

Step 3:

Run ICCAVR and then click on the Project Tab and select the option “New” from the drop-down list.

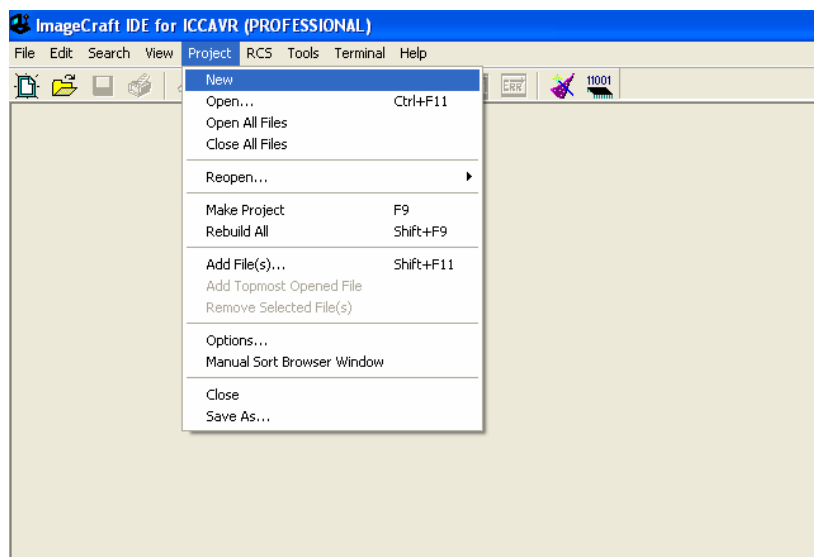


Figure 3.7

Step 4:

Now you will be prompted to assign a name to your project. So browse to the Folder that you have created in Step 2 and once there, assign a name and click on “Save”

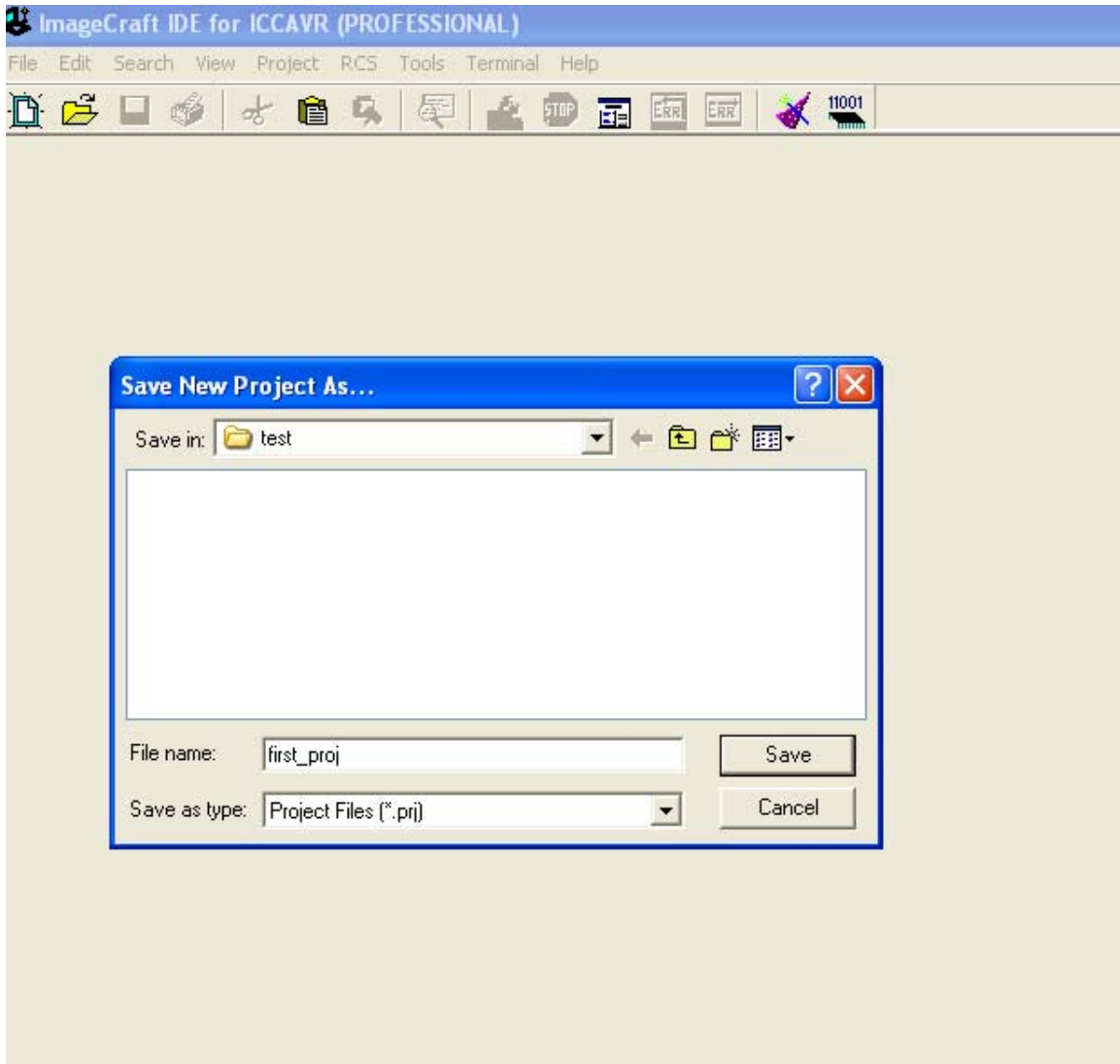



Figure 3.8

Step 5:

Next comes the task of writing the source code and this is where the magic of ICCAVR lies. It makes the task simple for you by automatically generating code to initialize all hardware resources that you may need and in the mode that you need them. It is called

the Application Builder and it is started by clicking on this icon  in the toolbar. As you can see in the figure below, all you need to do is to graphically specify your requirements and it takes care of the rest. This way you are free to just concentrate on the

application that you are building. We prefer to teach you how to use this feature by taking up actual examples and therefore this will be dealt with in detail later.

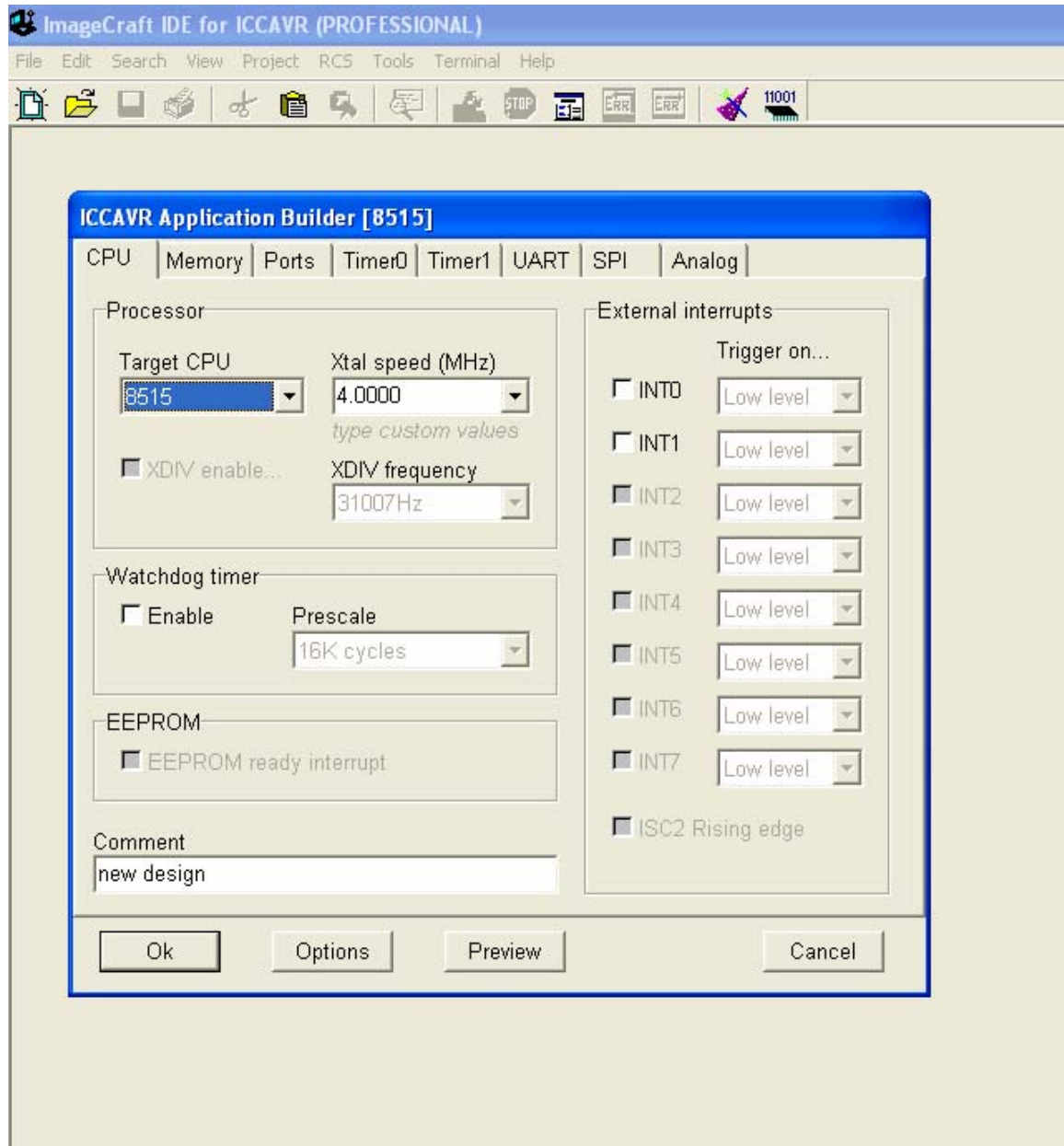


Figure 3.9


Step 6:

Write the code required for your application.

Step 7:

To include this file as a source file of your project, right click on File in the Project Window and select Add Files and browse to the file you saved in Step 4 and then click on Project->Rebuild All (Shift+F9). Correct any errors if present.

Step 8:

Click on the *Tools* tab and select the *In System Programmer* option or click on the  icon on the toolbar.

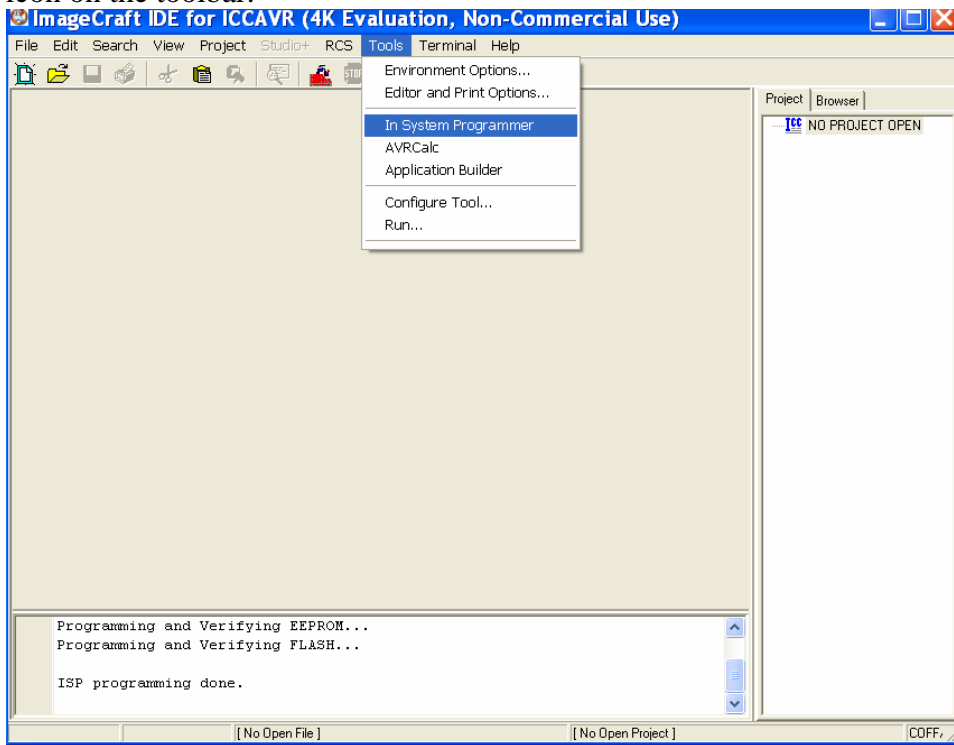


Figure 3.10

Step 9:

Load the hex file

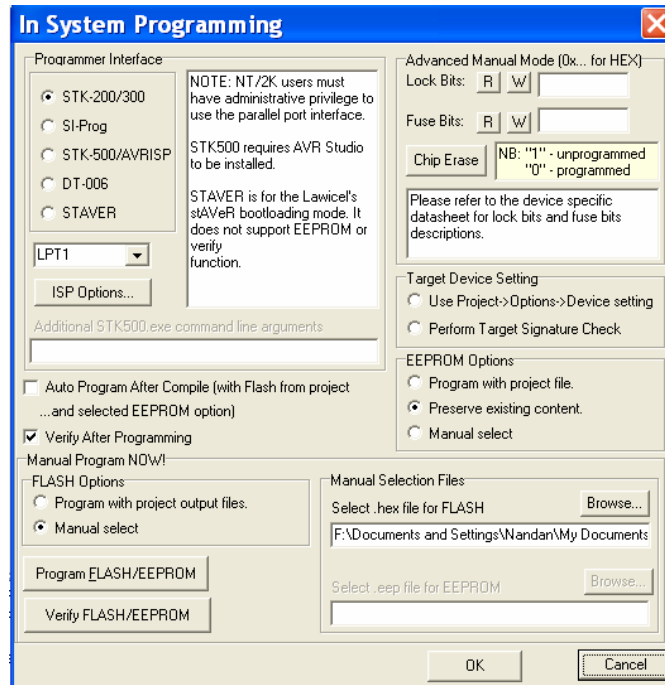


Figure 3.11

In the *Programmer Interface* section select *STK-200* programmer and select *LPT1* (parallel port) option.

In *Target Device Setting* section, select *Perform Target Signature Check* option.

Deselect the *Auto Program after Compile* option and select *Verify after Programming* option.

In *EEPROM Options*, select *Preserve existing content* option.

In *Manual Program NOW* section, specify the path of the hex file to be loaded.

Now load the hex file into the microcontroller by clicking on the *Program FLASH/EEPROM* button