Introduction to Microcontrollers

Rajat Arora

Electronics Club
IIT Kanpur

Micro-Controller

- A single chip Computer (to some extent)
- Has CPU

1. RAM
2. EEPROM
3. I/O in form of pins
4. Peripherals (Timer , Communication modes , ADC etc)

Flash Back (Takneek)

- Line Following Robots
- Wireless keyboards
- They were made using Microcontrollers

- Suppose we want to make a Line following Robot
- What do we do ?
- Use a computer with 2.4Ghz Intel core I7 with 4 Gb RAM , 500 Gb Hard disk , 1 Gb Graphics Card ??

Why not a Computer ?

- PC is a general purpose computer.
- Can run thousand of softwares
- Microsoft ppt in which you are seeing this presentation
- Games (NFS , AOE , Call of Duty)
- Highly expensive

Why MCU

- Small reflected by the word “MICRO”
- Inexpensive
- Ideal for doing repetitive tasks
- Easy to use
- Highly Efficient and fast

Selecting a MCU

- Two family of MCU extremely popular
  a) AVR
  b) PIC

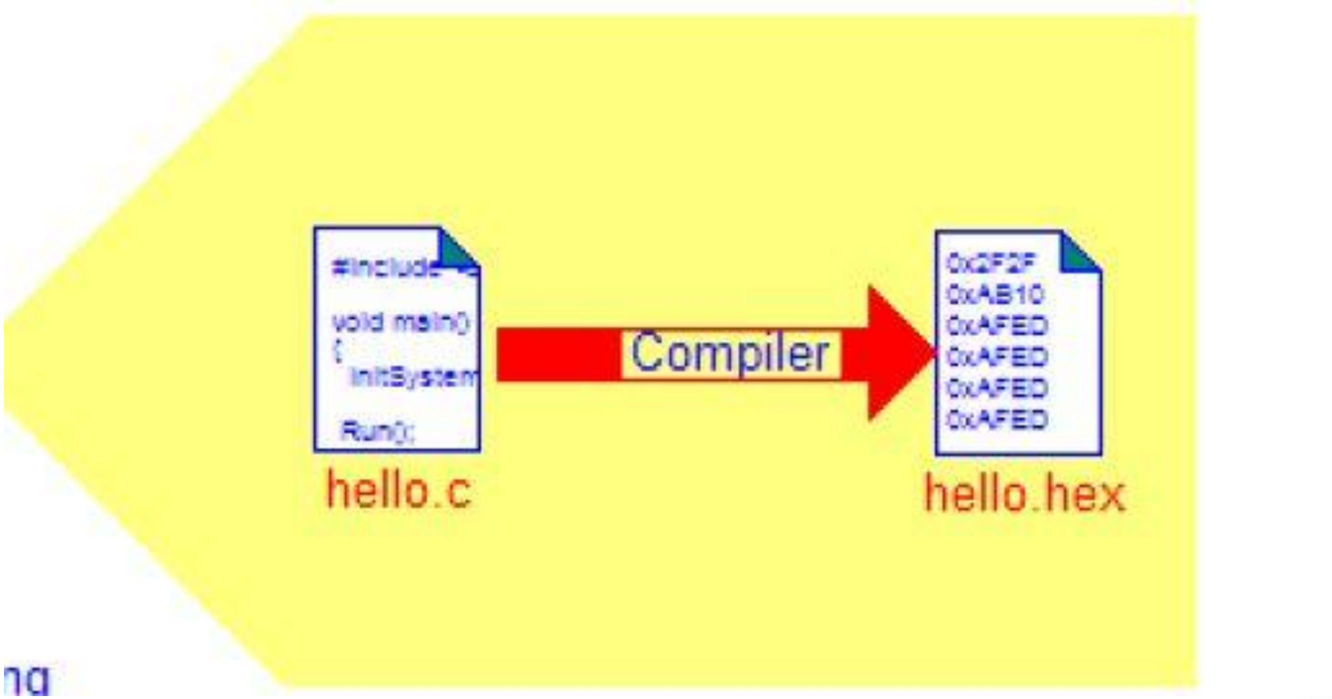
- We use AVR series of MCU from Atmel
- The instructions are fed once in the form of a Hex file

Tools Required -> CVAVR

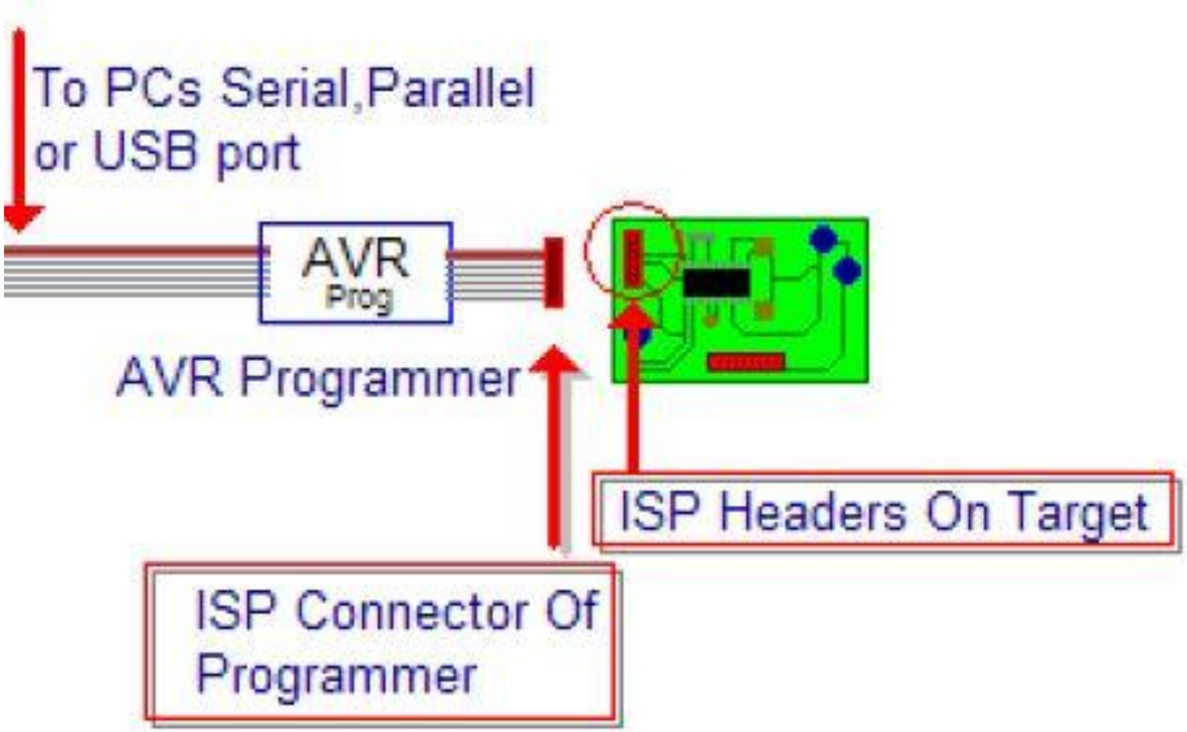
PC Running IDE for entering,editing and compiling source program.

Compiler -> CVAVR

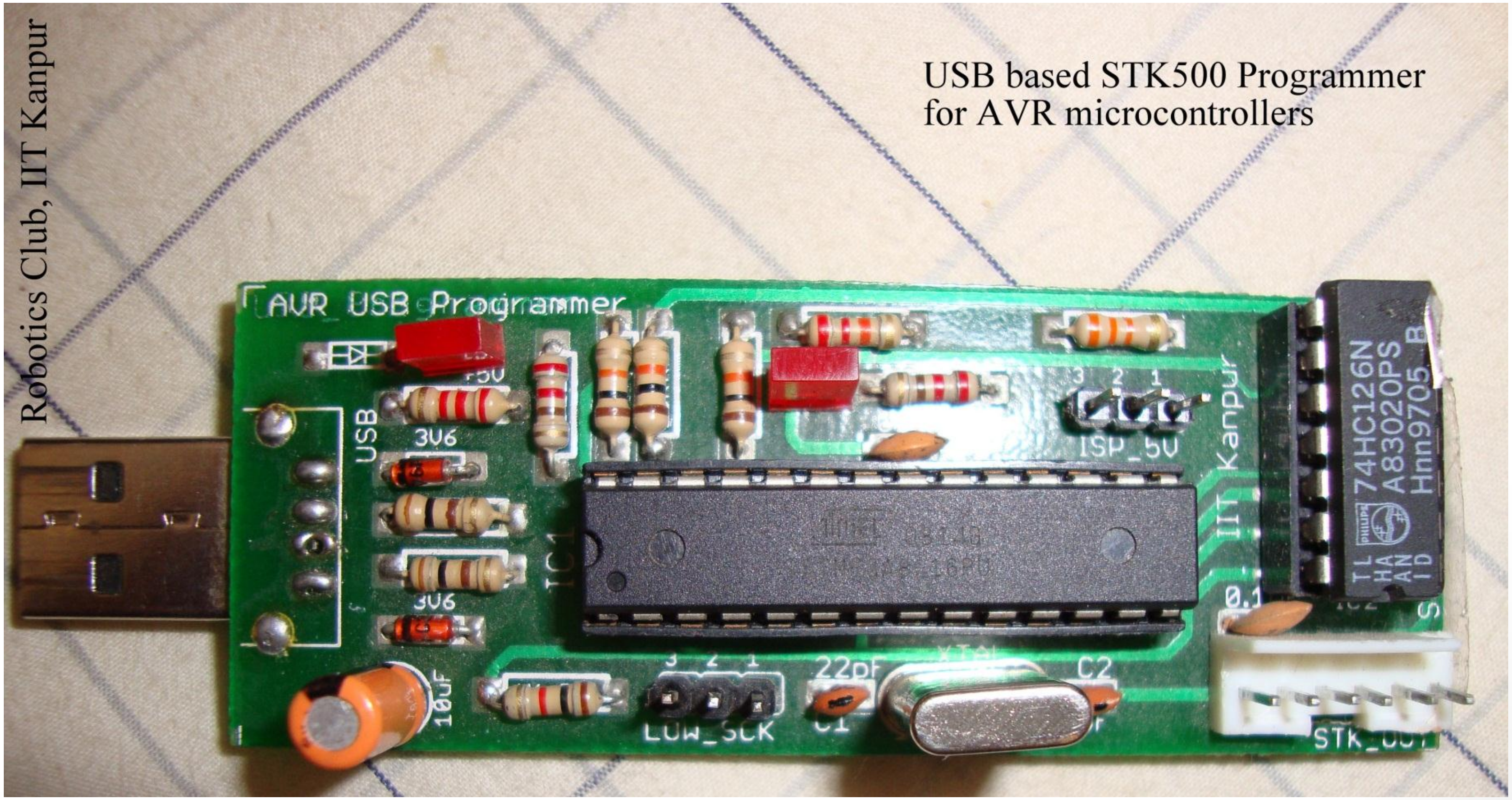
- The code is written in C language so we need to convert it into the format that Atmega understands



Transfer code to Atmega AVR Studio


To PCs Serial,Parallel or USB port
AVR Prog
AVR Programmer
ISP Headers On Target
ISP Connector Of Programmer

Avr Programmer



USB based STK500 Programmer for AVR microcontrollers

Robotics Club, IIT Kanpur

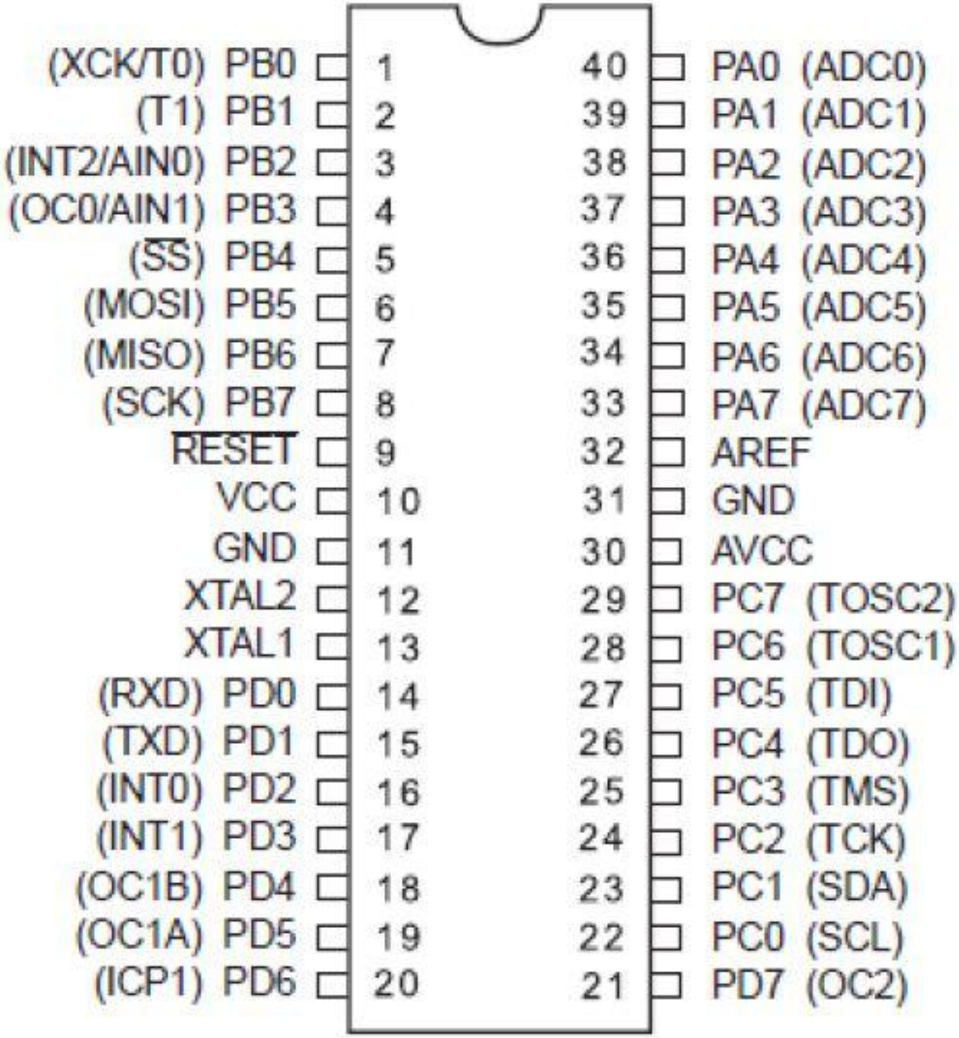
- So we need two softwares overall

  a) CVAVR –> Editor and Compiler
  b) Avr Studio –> Transfer Code to Atmega

Atmega 16

The ATmega16

- 40 pin IC.
- 32 pins for I/O.
- 8 pins reserved.
- I/O pins divided into 4 groups of 8 pins, called ports.
- Ports labeled as A, B, C and D.



Basics of C language

- If else block

-
  ```
  If(condition)
  {
   … …
  }
  else
  {
  … …
  }
  ```

While & For

- While (conditon)
  ```
  {
  … …
  }
  ```

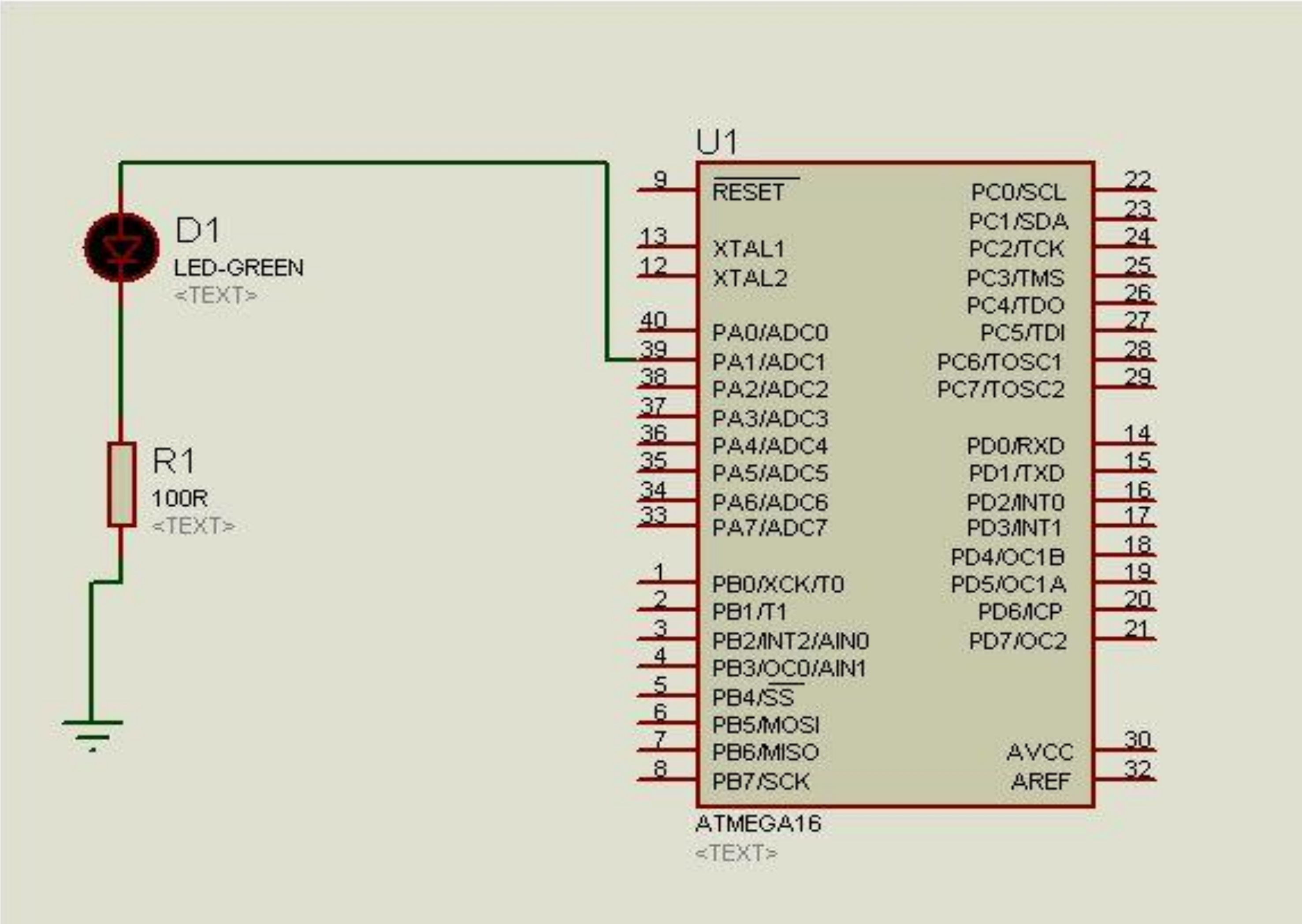
- for(initialisation; condition; increment)
  ```
  {
   … …
  }
  ```

Some C operators

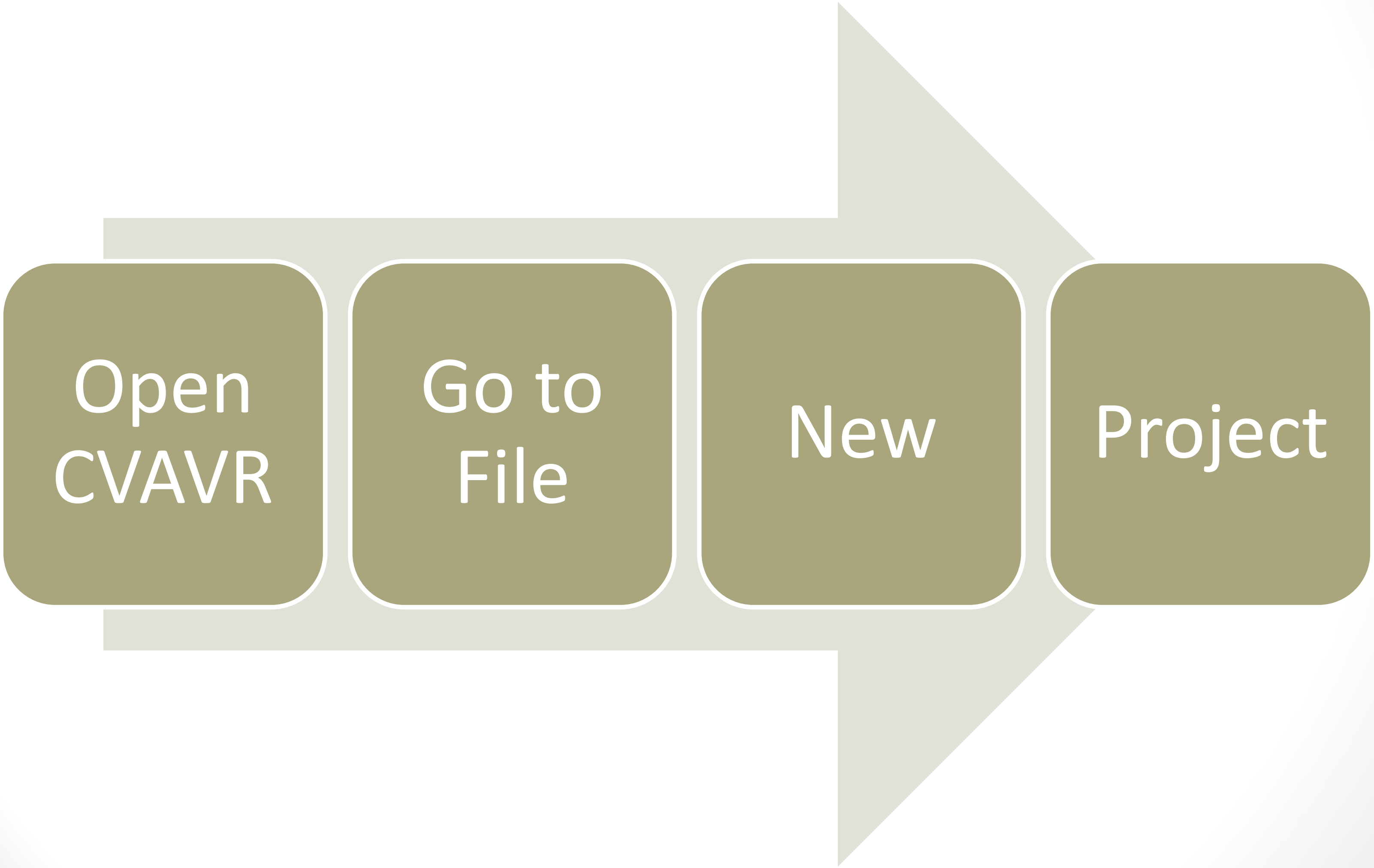
- | is bitwise OR.
  Eg. `10100111 | 11000101  = 11100111`
- & is bitwise AND.
  Eg. `10100111 & 11000101 = 10000101`
- ~ is bitwise NOT.
  Eg. `~10100110 = 01011001`
- << is shift left. >> is shift right.

- Lets Begin by blinking a simple LED

Circuit Diagram



Getting Started with CVAVR



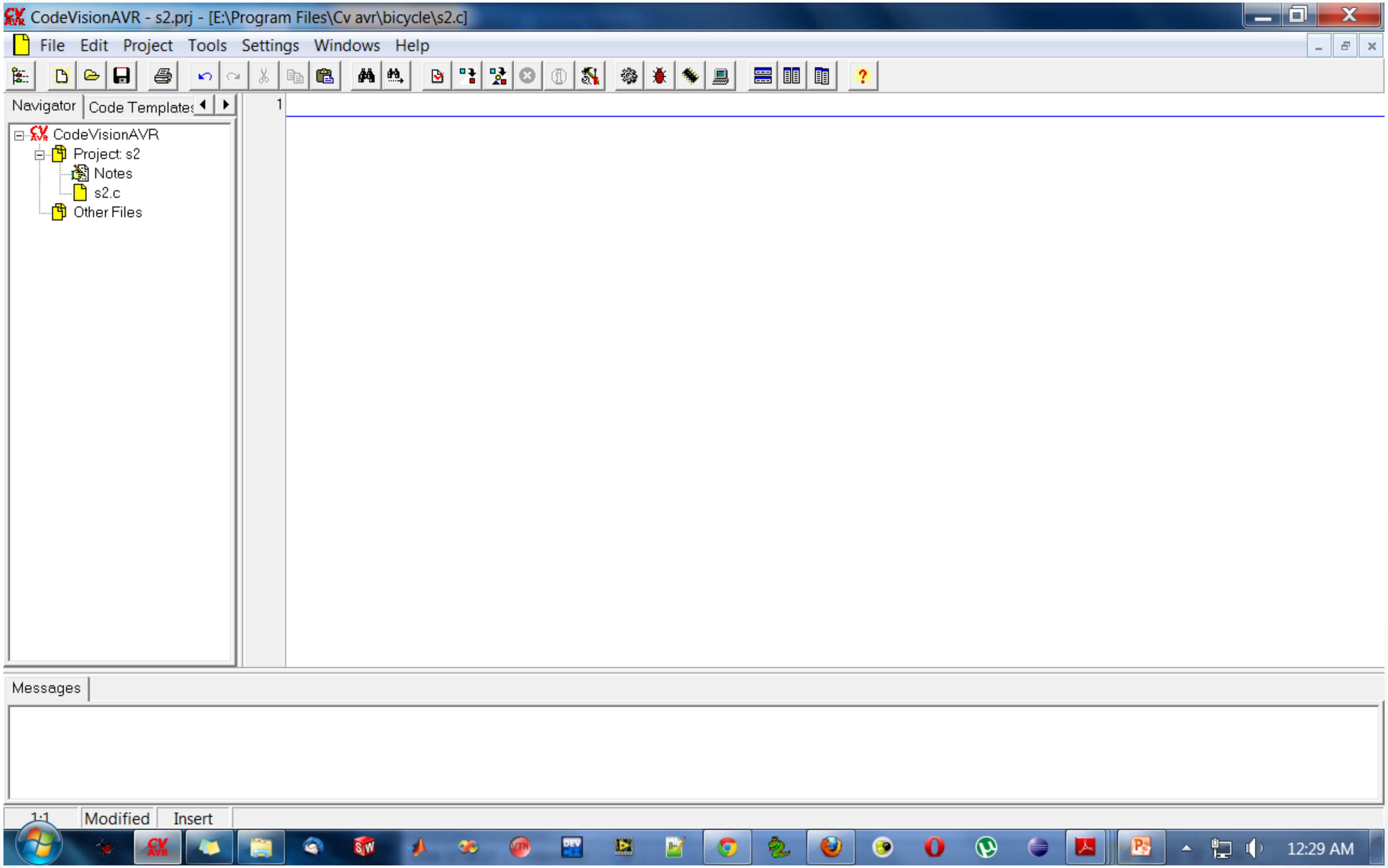
Open CVAVR

Go to File

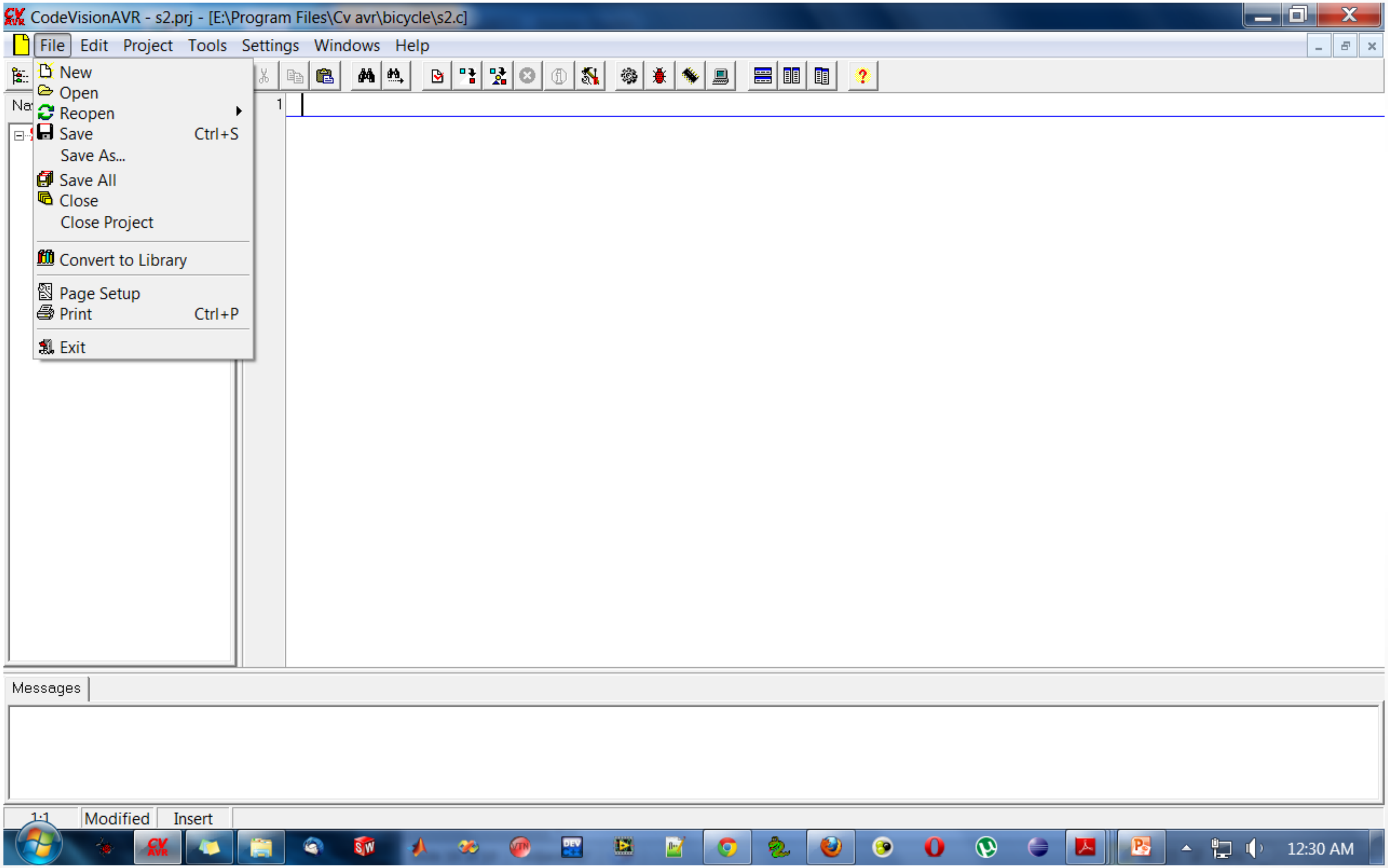
New

Project

Open CVAVR

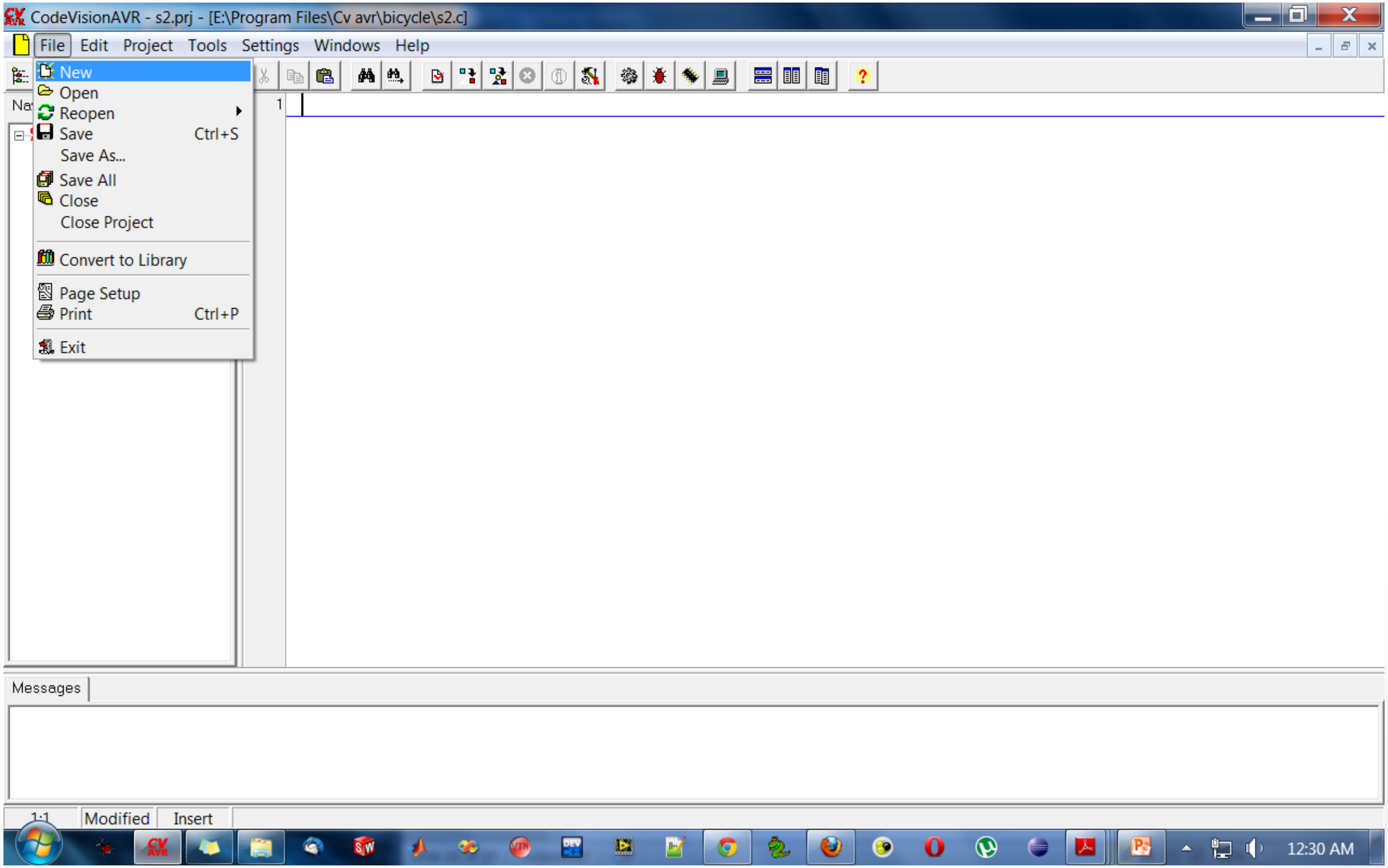

CodeVisionAVR - s2.prj - [E:\Program Files\Cv avr\bicycle\s2.c]
File Edit Project Tools Settings Windows Help
Navigator
Code Templates
CodeVisionAVR
Project: s2
Notes
s2.c
Other Files
1
Messages
1:1
Modified
Insert
12:29 AM

Go to File

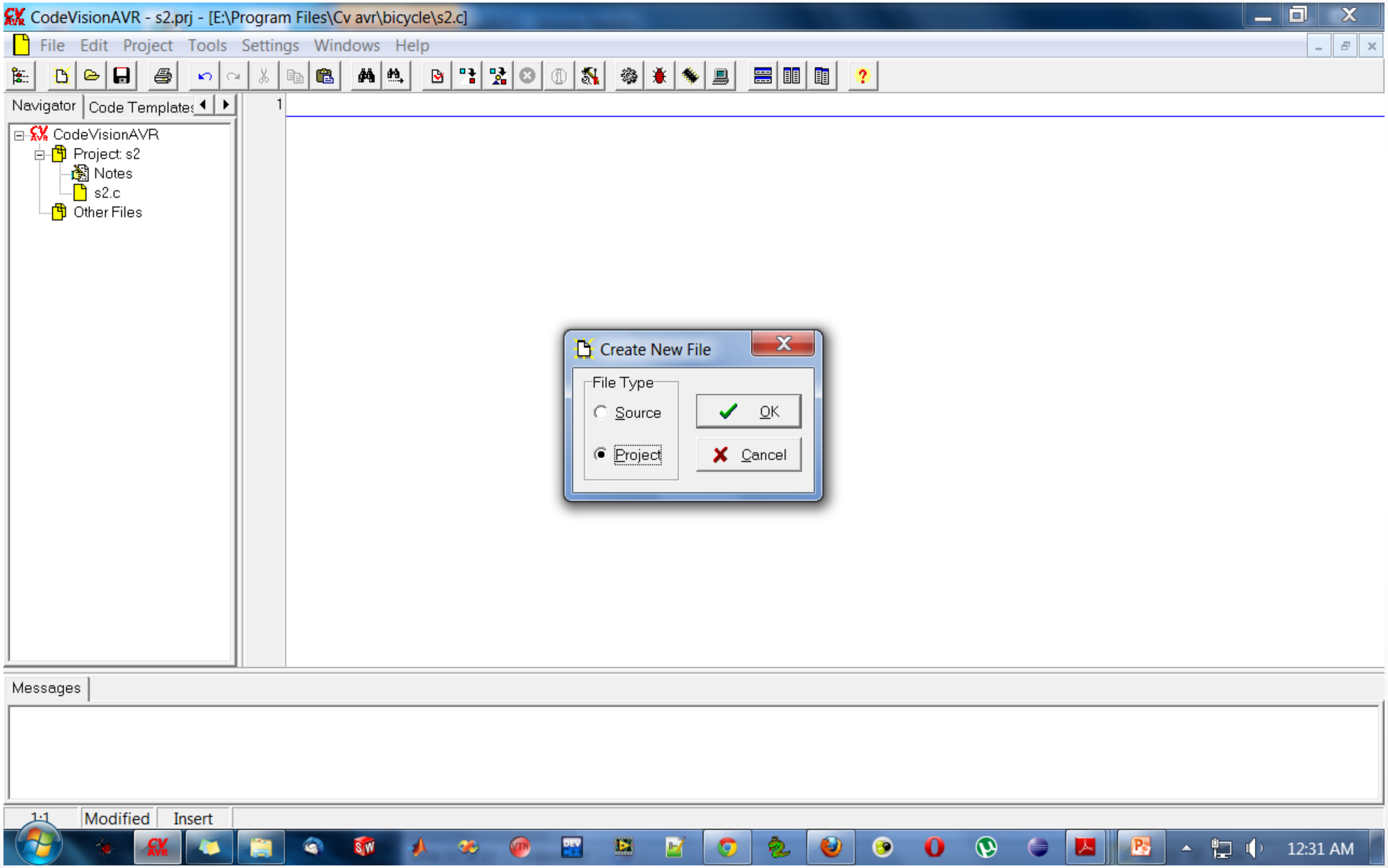


CodeVisionAVR - s2.prj - [E:\Program Files\Cv avr\bicycle\s2.c]
File
Edit
Project
Tools
Settings
Windows
Help
New
Open
Reopen
Save Ctrl+S
Save As...
Save All
Close
Close Project
Convert to Library
Page Setup
Print Ctrl+P
Exit
Messages
1:1
Modified
Insert
12:30 AM

Click on New


CodeVisionAVR - s2.prj - [E:\Program Files\Cv avr\bicycle\s2.c]
File
Edit
Project
Tools
Settings
Windows
Help
New
Open
Reopen
Save Ctrl+S
Save As...
Save All
Close
Close Project
Convert to Library
Page Setup
Print Ctrl+P
Exit
Messages
1:1
Modified
Insert
12:30 AM

Select Project- > Click OK



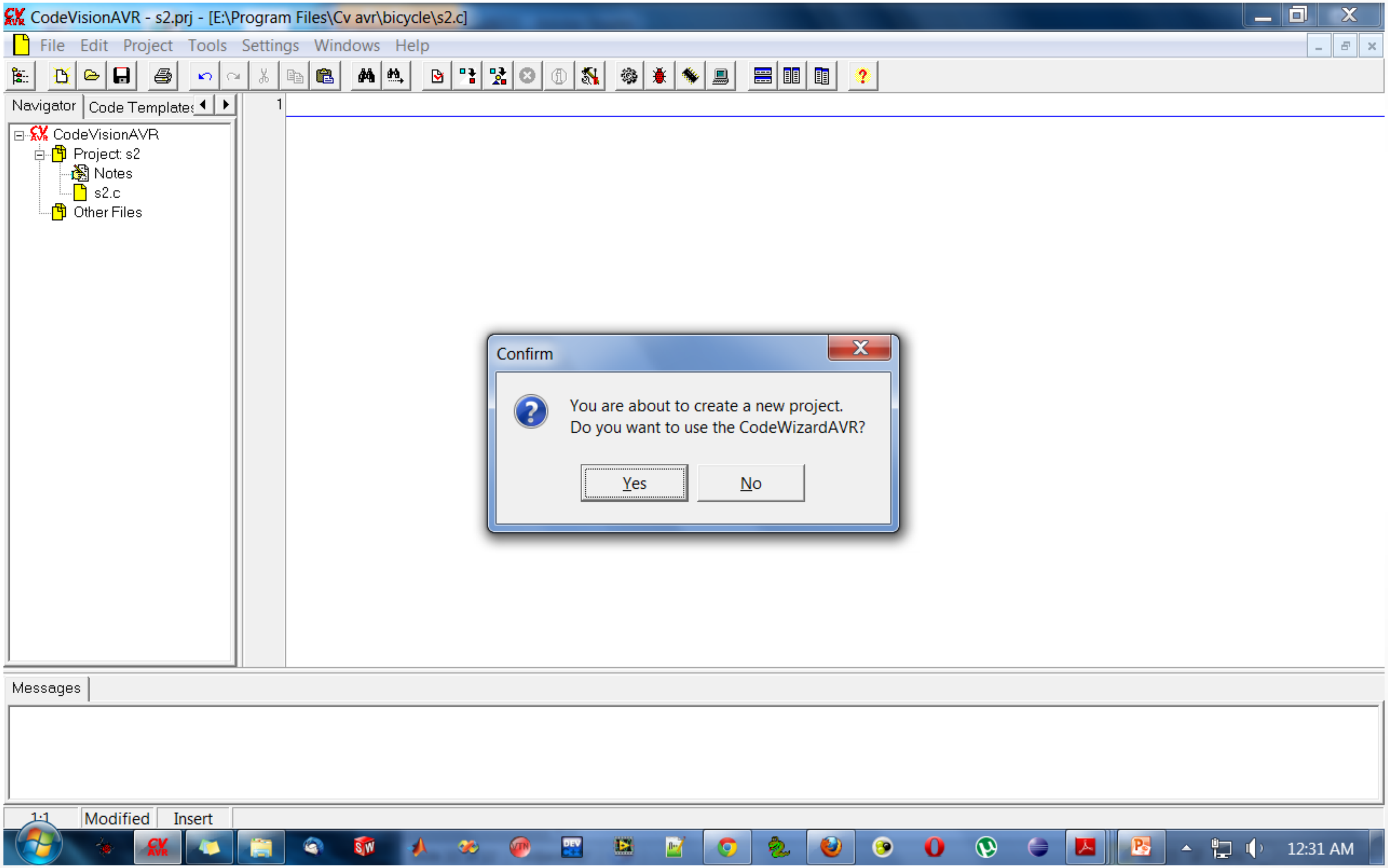
Create New File

File Type

○ Source

● Project

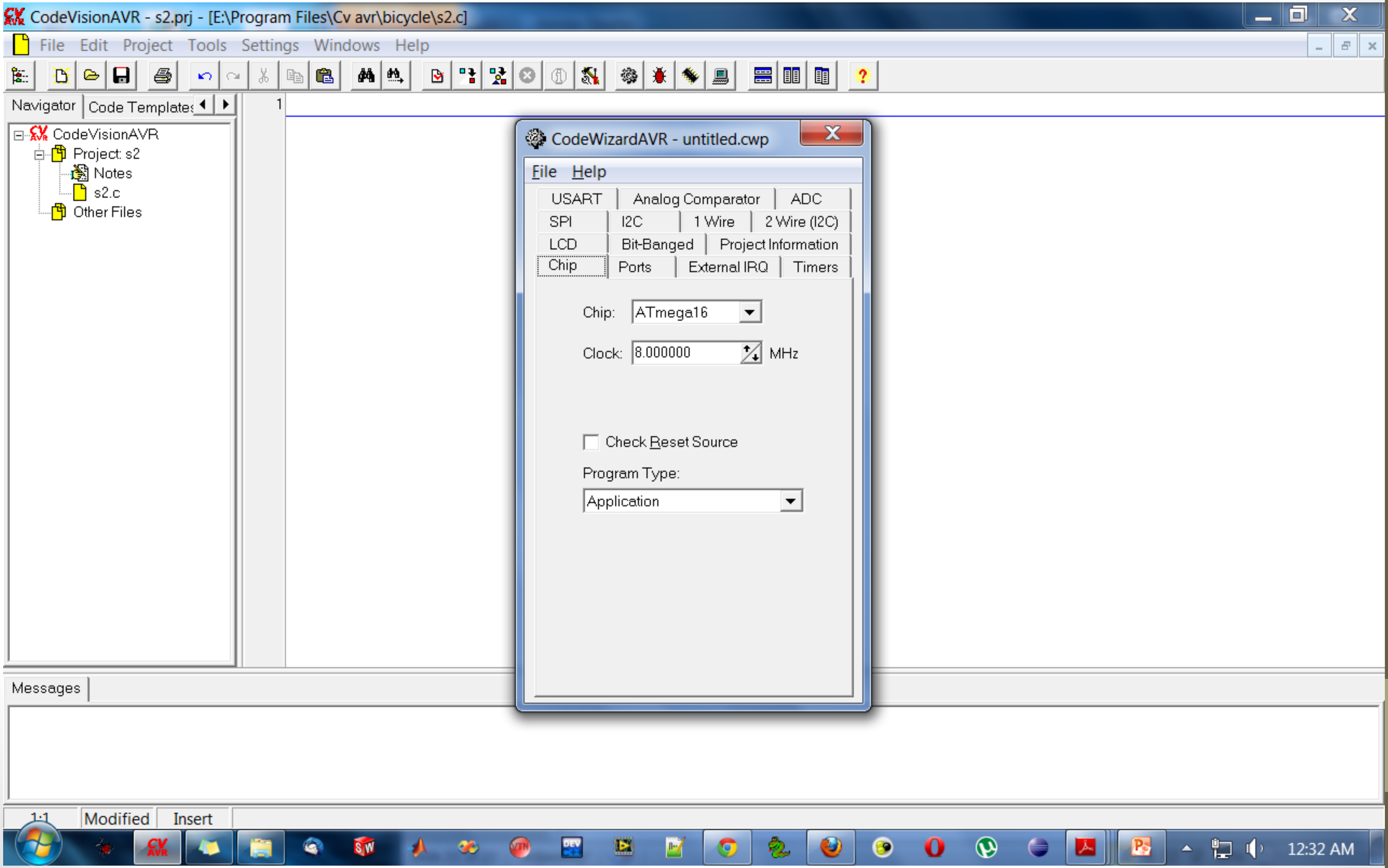
Click YES



Confirm

You are about to create a new project.
Do you want to use the CodeWizardAVR?

Select Chip



Chip: ATmega16

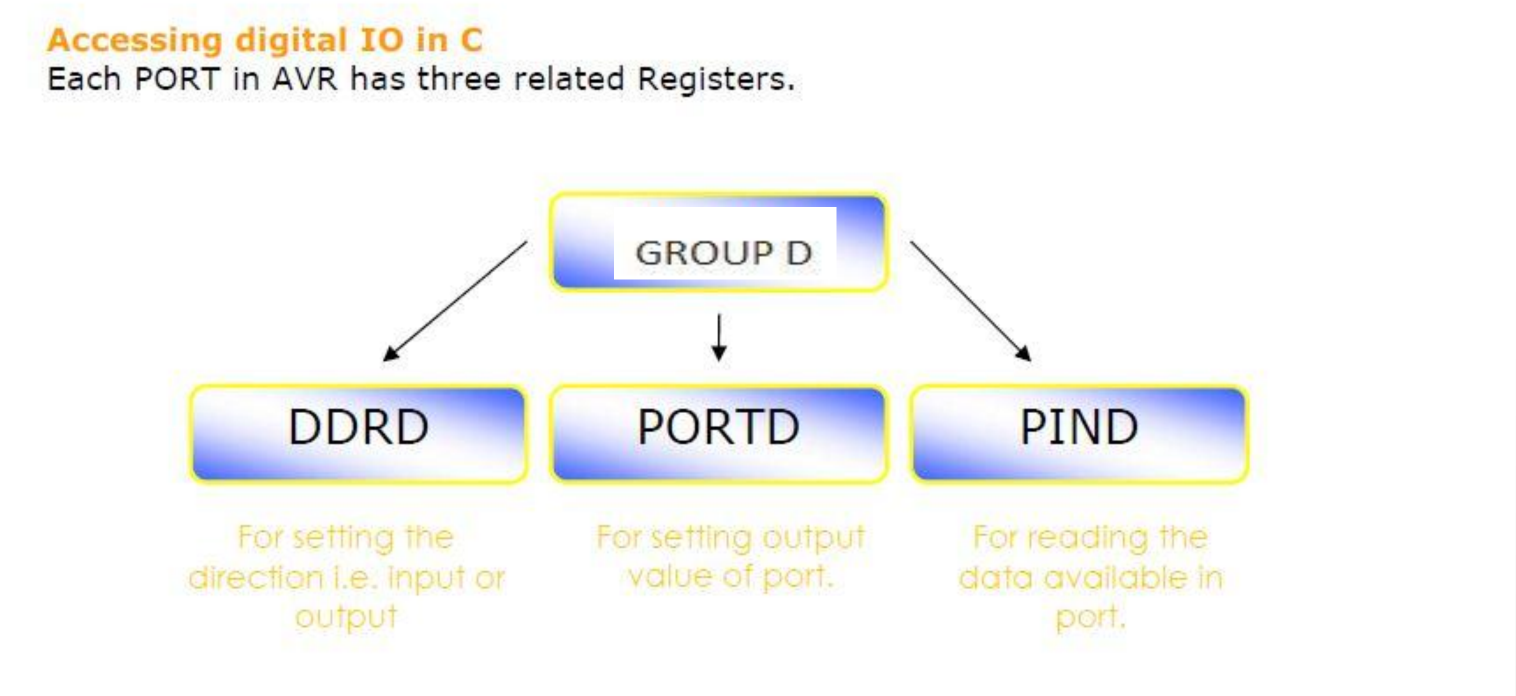
Clock: 8.000000 MHz

□ Check Reset Source

Program Type: Application

Introduction to I/O

- Atmega has total of 40 pins out of which 32 pins can be used as Input or Output
- These 32 pins are divided into 4 groups of 8 pins PORTA, PORTB , PORTC , PORTD



Accessing digital IO in C

Each PORT in AVR has three related Registers.

For setting the direction i.e. input or output

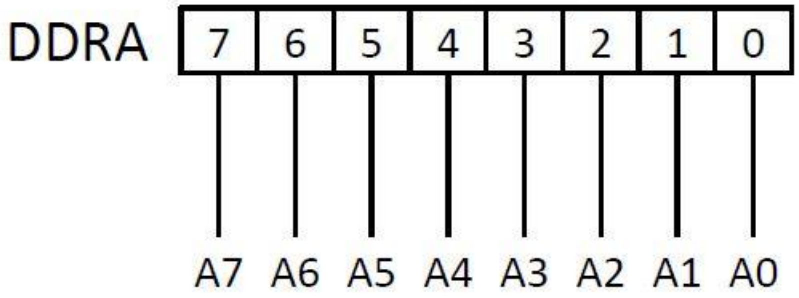
For setting output value of port.

For reading the data available in port.

Data Direction register (DDR)

- This sets direction for all pins (32)
- Direction for these pins can be Input or Output
- To blink an LED we need to set pin as “OUTPUT” but “HOW“ ?

- DDRA = 0b00000001 ;
- DDRA = 0x01 ;
- 1 Stands for Output & 0 stands for Input



DDRA	7	6	5	4	3	2	1	0
	A7	A6	A5	A4	A3	A2	A1	A0

Interpretation of DDR values

- If a bit on the **DDR** register is 0, then the corresponding pin on the associated port is set as input.
- Similarly, if the bit is 1, then the pin is set as output.
- Example: if DDRA = 0b10010110, then:

DDRA	OP	IN	IN	OP	IN	OP	OP	IN
	MSB							LSB

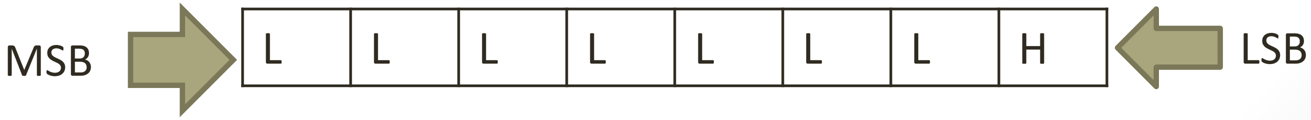
What Next ?

- We have set the Pin as Output
- What else do we need to light the LED ??
- Supply of 5 Volts !!! This is given by PORT Register

PORT Register

- Only after you have set the Pin to Output you can control them through this Register
- It is a 8 bit register . It corresponds to the pin in same manner as that of DDR Register
- Used to set output value (0 or 1) only if the corresponding Pin has been set as output by DDR Register
- PORTA= 0b 00000001;

  or
- PORTA= 0x01 ;
- 1 stands for 5V
- 0 stands for 0V



MSB	L	L	L	L	L	L	L	H	LSB

Simple Questions

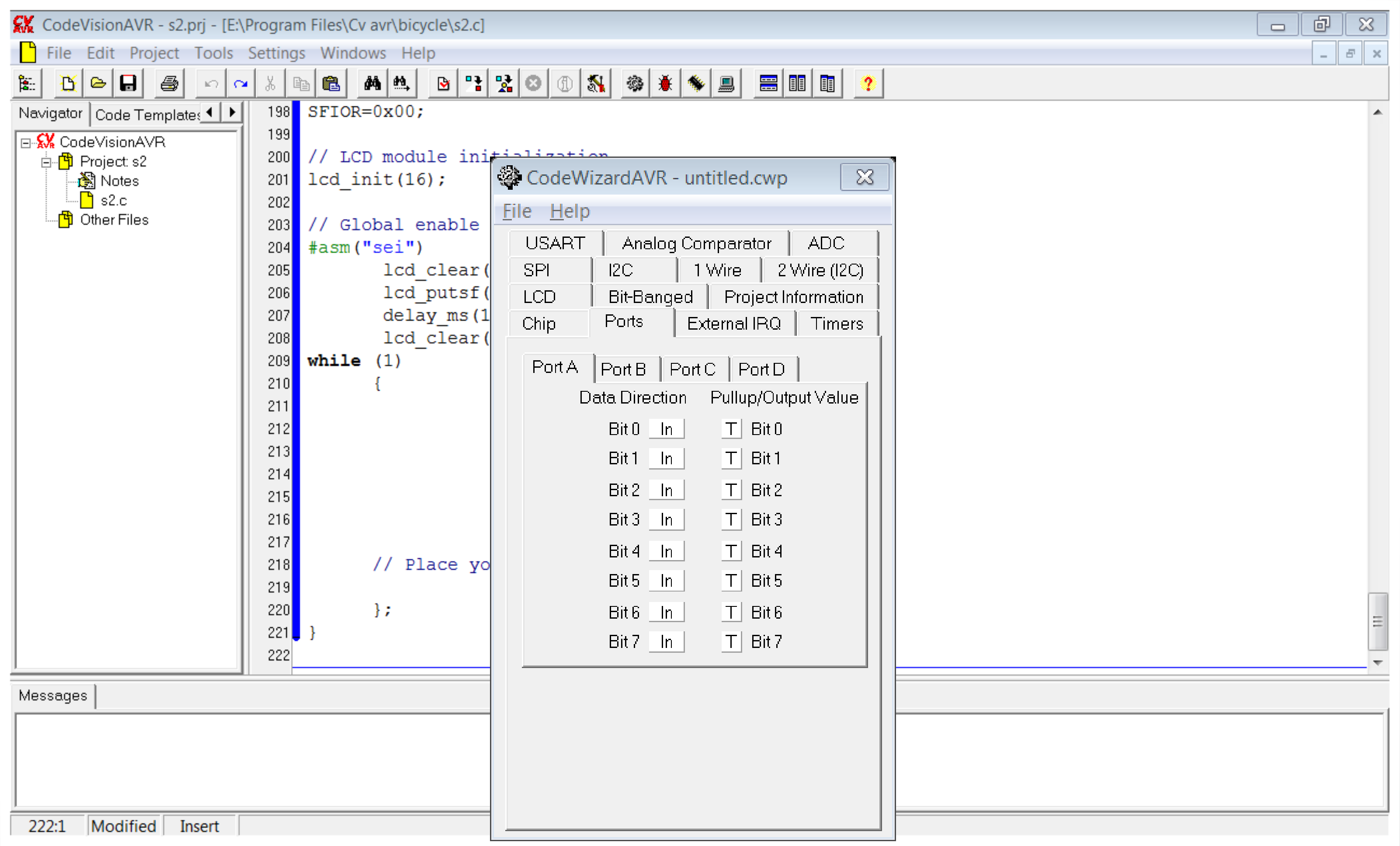
- DDRA= 0b 00101100
- DDRD = 0xf4
- DDRC = 0b 01111110
- DDRB = 0x3b

Assume all 32 pins set as output

- PORTA = 0b00001100;
- PORTD = 0b11110000;
- PORTB.4=1;
- PORTC.2=1;

Setting I/O

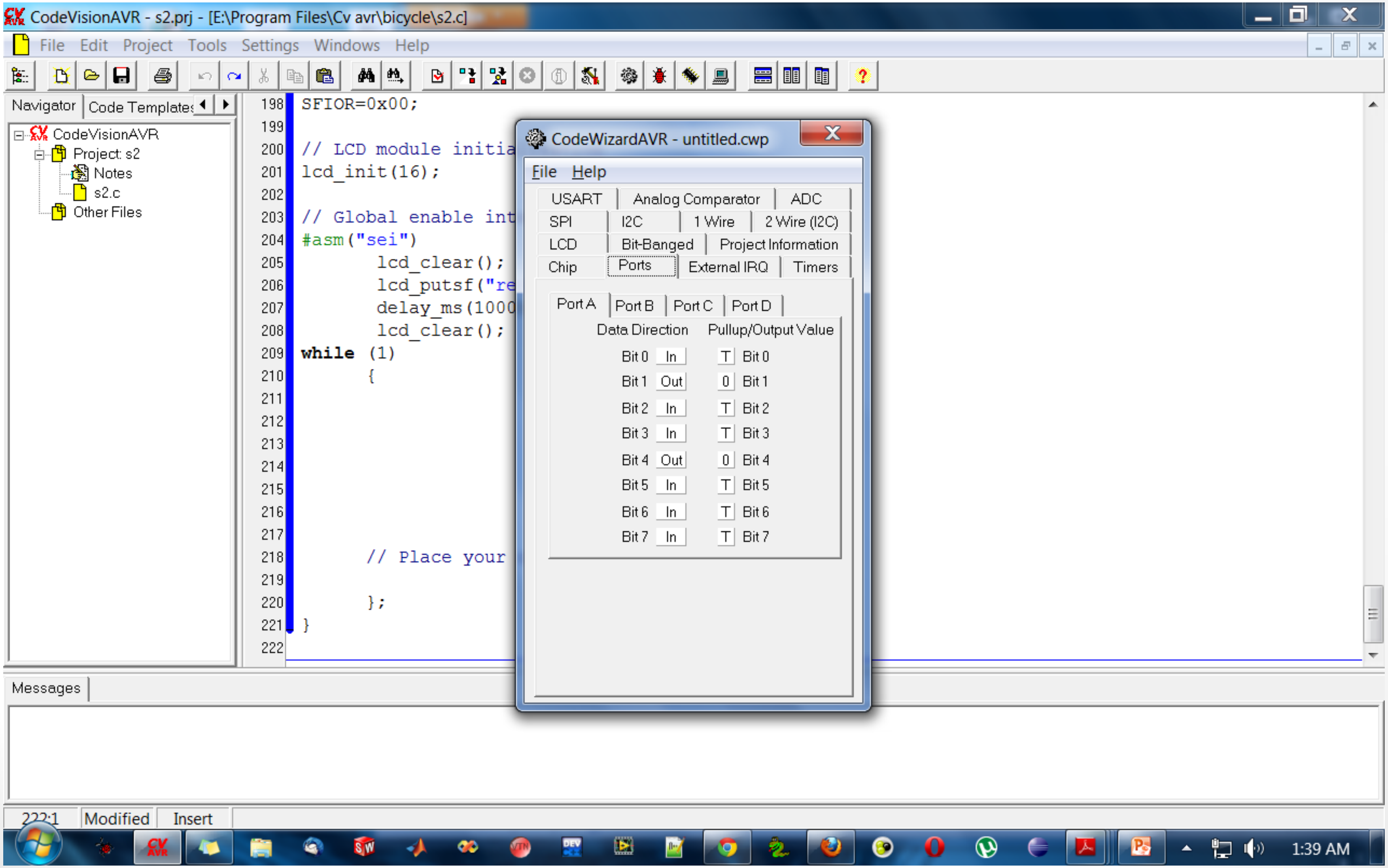
Go to Ports



```
SFIOR=0x00;

// LCD module ini
lcd_init(16);

// Global enable
#asm("sei")
       lcd_clear(
       lcd_putsf(
       delay_ms(1
       lcd_clear(
while  (1)
       {







           // Place yo

       };
}
```

	Data Direction	Pullup/Output Value	
Bit 0	In	T	Bit 0
Bit 1	In	T	Bit 1
Bit 2	In	T	Bit 2
Bit 3	In	T	Bit 3
Bit 4	In	T	Bit 4
Bit 5	In	T	Bit 5
Bit 6	In	T	Bit 6
Bit 7	In	T	Bit 7

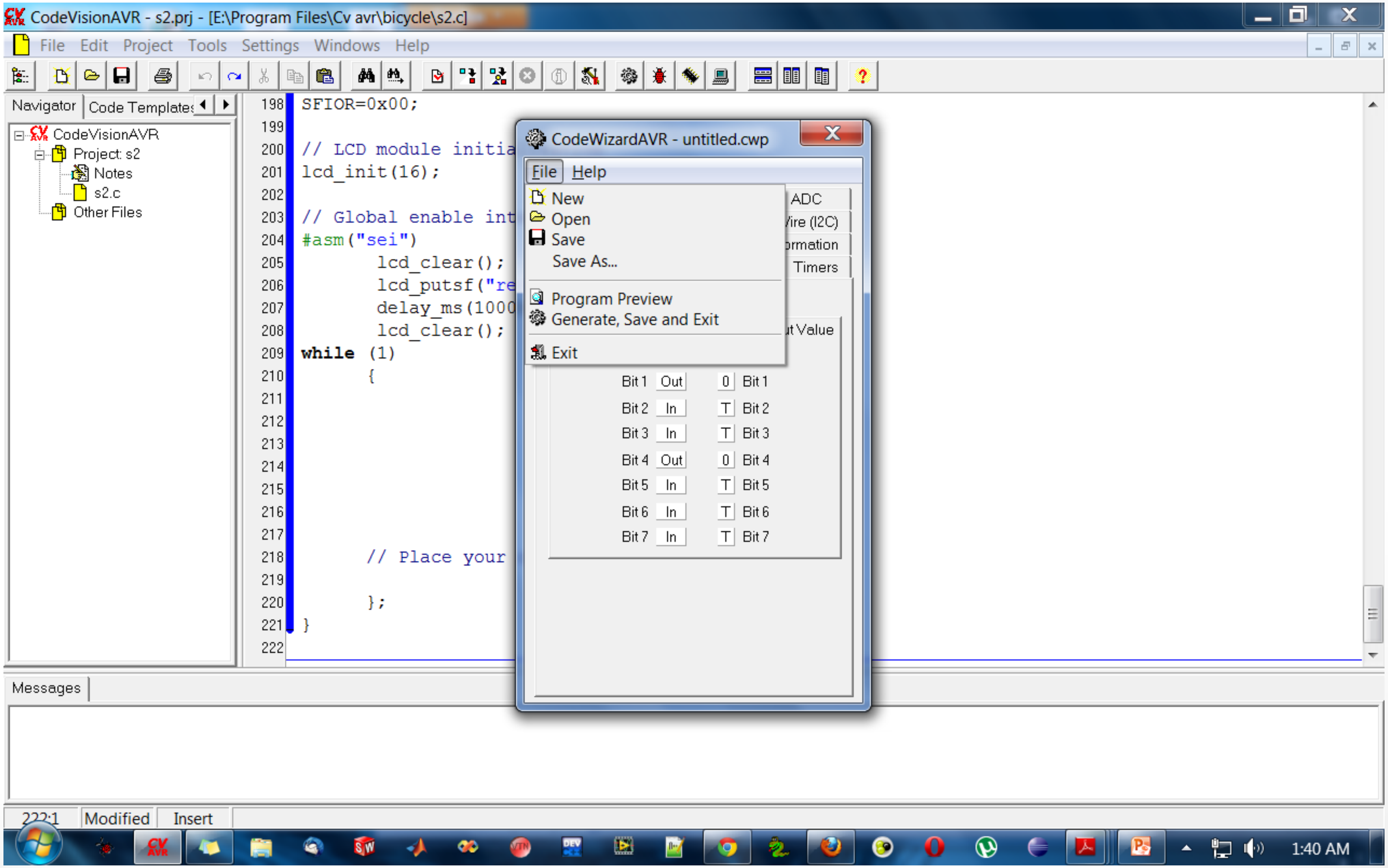
- Click on In to make that pin Output
- Can do so for all four ports



```
SFIOR=0x00;

// LCD module initia
lcd_init(16);

// Global enable int
#asm("sei")
        lcd_clear();
        lcd_putsf("re
        delay_ms(1000
        lcd_clear();
while (1)
      {




        // Place your

      };
}
```

Data Direction		Pullup/Output Value	
Bit 0	In	T	Bit 0
Bit 1	Out	0	Bit 1
Bit 2	In	T	Bit 2
Bit 3	In	T	Bit 3
Bit 4	Out	0	Bit 4
Bit 5	In	T	Bit 5
Bit 6	In	T	Bit 6
Bit 7	In	T	Bit 7

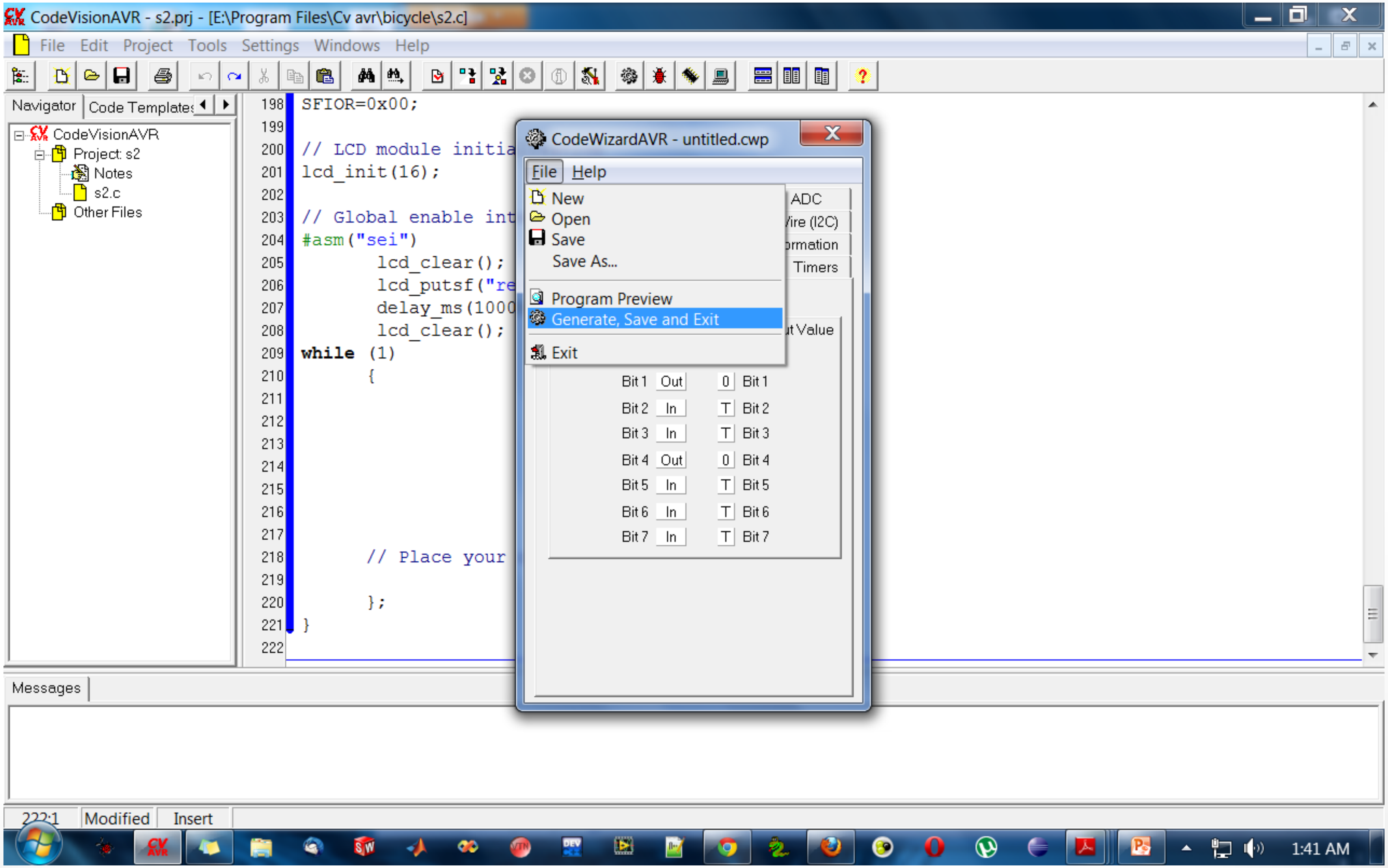
Click on File



```
SFIOR=0x00;

// LCD module initia
lcd_init(16);

// Global enable int
#asm("sei")
        lcd_clear();
        lcd_putsf("re
        delay_ms(1000
        lcd_clear();
while (1)
      {




      // Place your

      };
}
```

Bit 1	Out	0	Bit 1
Bit 2	In	T	Bit 2
Bit 3	In	T	Bit 3
Bit 4	Out	0	Bit 4
Bit 5	In	T	Bit 5
Bit 6	In	T	Bit 6
Bit 7	In	T	Bit 7

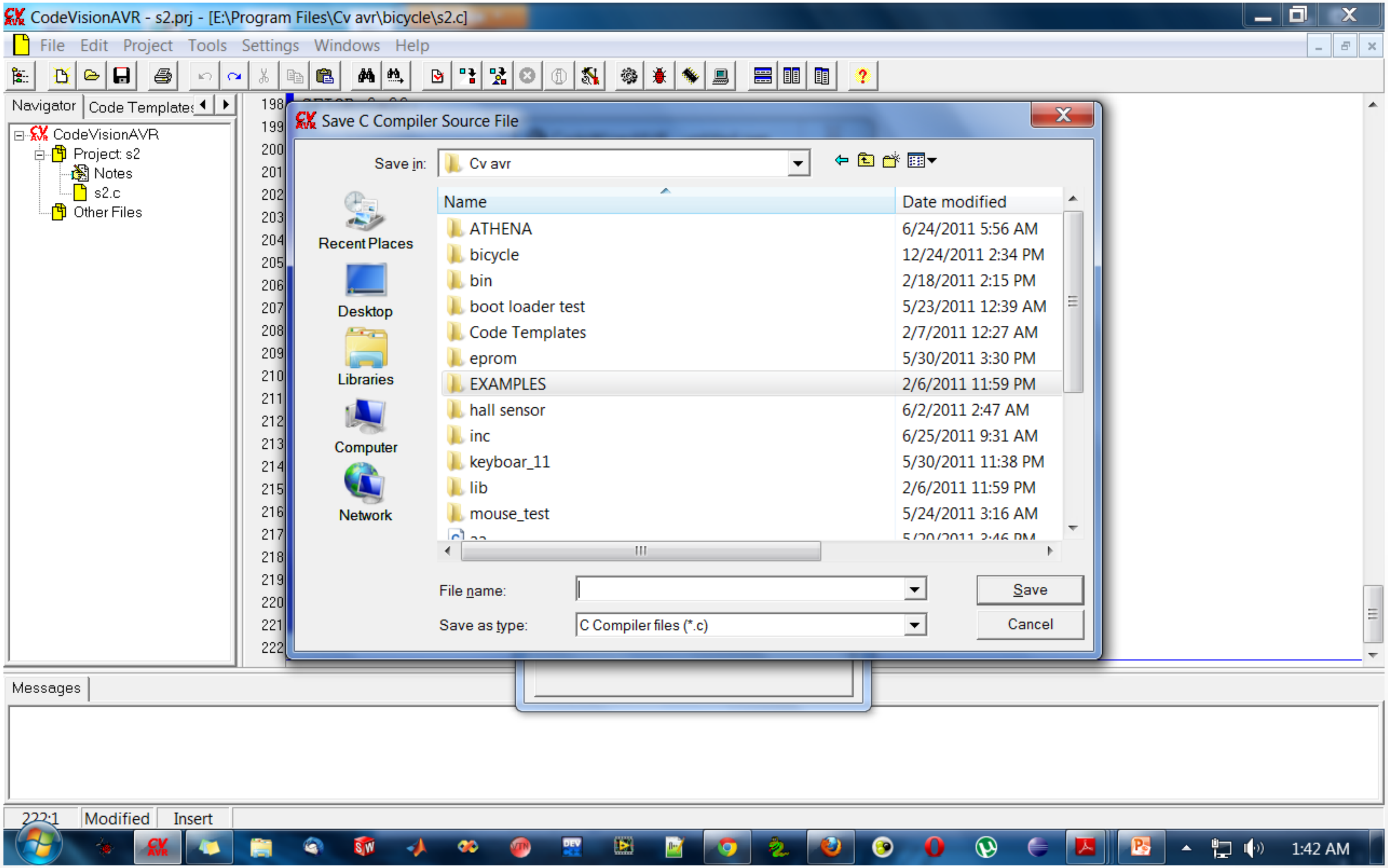
Generate Save and Exit



```
SFIOR=0x00;

// LCD module initia
lcd_init(16);

// Global enable int
#asm("sei")
        lcd_clear();
        lcd_putsf("re
        delay_ms(1000
        lcd_clear();
while (1)
      {




          // Place your

      };
}
```

Bit 1	Out	0	Bit 1
Bit 2	In	T	Bit 2
Bit 3	In	T	Bit 3
Bit 4	Out	0	Bit 4
Bit 5	In	T	Bit 5
Bit 6	In	T	Bit 6
Bit 7	In	T	Bit 7

Enter name (3 times)



Save in: Cv avr

Name	Date modified
ATHENA	6/24/2011 5:56 AM
bicycle	12/24/2011 2:34 PM
bin	2/18/2011 2:15 PM
boot loader test	5/23/2011 12:39 AM
Code Templates	2/7/2011 12:27 AM
eprom	5/30/2011 3:30 PM
EXAMPLES	2/6/2011 11:59 PM
hall sensor	6/2/2011 2:47 AM
inc	6/25/2011 9:31 AM
keyboar_11	5/30/2011 11:38 PM
lib	2/6/2011 11:59 PM
mouse_test	5/24/2011 3:16 AM
aa	5/20/2011 3:46 PM

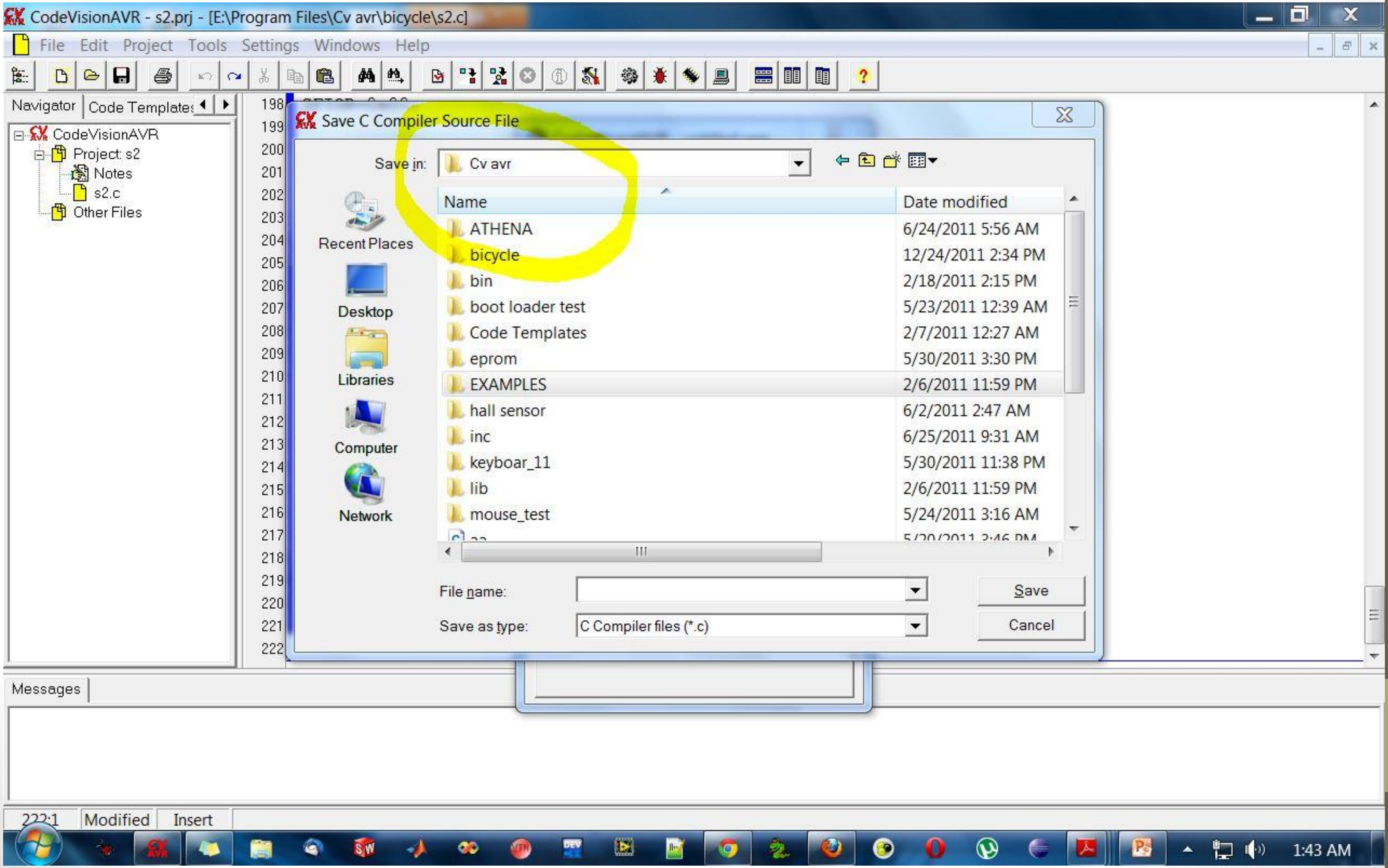
File name:

Save as type: C Compiler files (*.c)

222:1 Modified Insert

1:42 AM

Where is the code stored ?



CodeVisionAVR - s2.prj - [E:\Program Files\Cv avr\bicycle\s2.c]

Save C Compiler Source File

Save in: Cv avr

Name	Date modified
ATHENA	6/24/2011 5:56 AM
bicycle	12/24/2011 2:34 PM
bin	2/18/2011 2:15 PM
boot loader test	5/23/2011 12:39 AM
Code Templates	2/7/2011 12:27 AM
eprom	5/30/2011 3:30 PM
EXAMPLES	2/6/2011 11:59 PM
hall sensor	6/2/2011 2:47 AM
inc	6/25/2011 9:31 AM
keyboar_11	5/30/2011 11:38 PM
lib	2/6/2011 11:59 PM
mouse_test	5/24/2011 3:16 AM

File name:

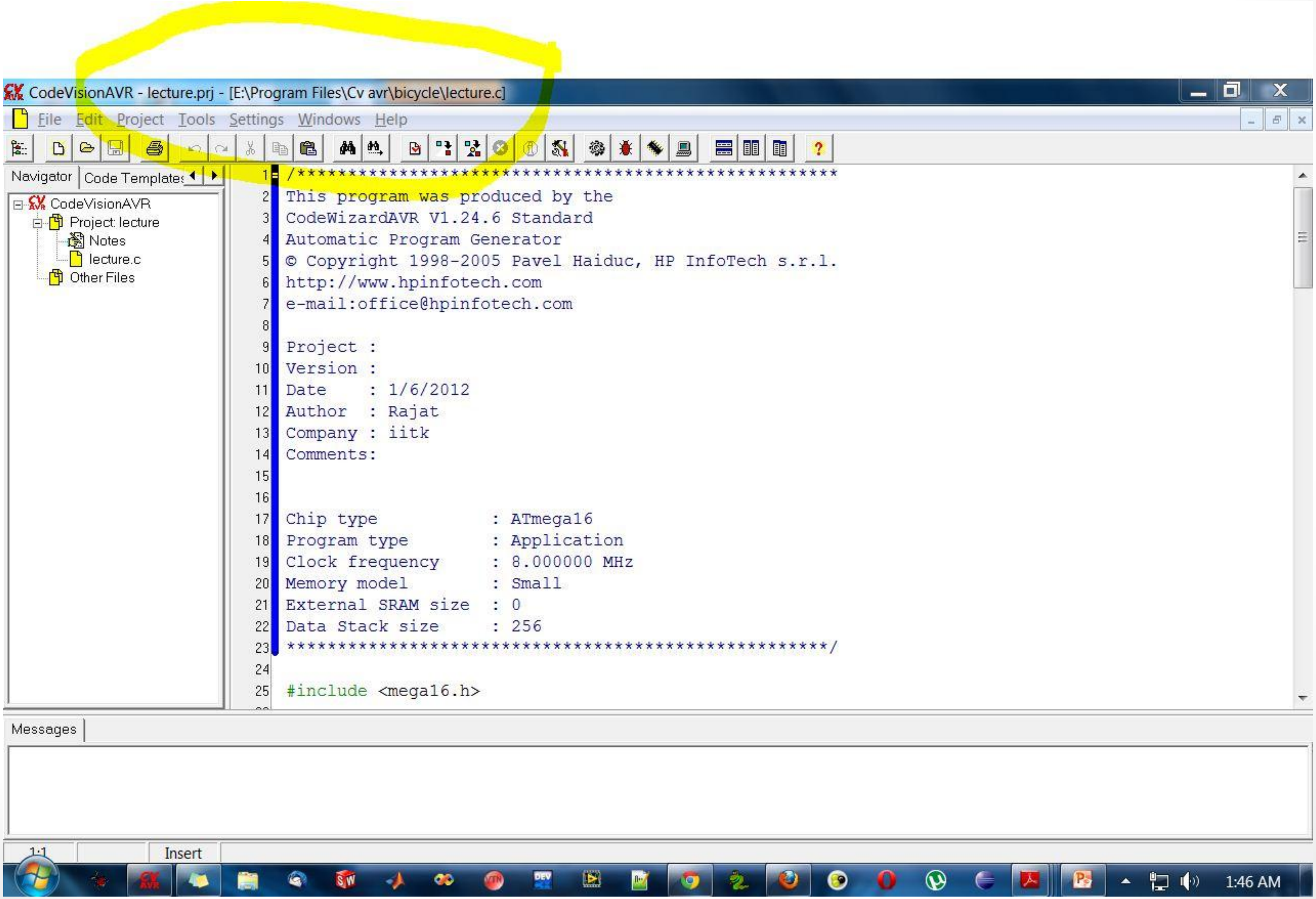
Save as type: C Compiler files (*.c)

222:1 Modified Insert

1:43 AM

Then Click Save

Name of Project & Location



CodeVisionAVR - lecture.prj - [E:\Program Files\Cv avr\bicycle\lecture.c]

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.6 Standard
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project :
Version :
Date    : 1/6/2012
Author  : Rajat
Company : iitk
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 8.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
```

Writing the Code

- NOTE : We write our code in While block
-
  ```
  While (1)
  {
  PORTA.1=1; // sets the Pin to 5 volts
  PORTA.1=0; //  sets the Pin to 0 volts
  }
  ```

- This makes the LED to blink but we cannot see blinking !!!

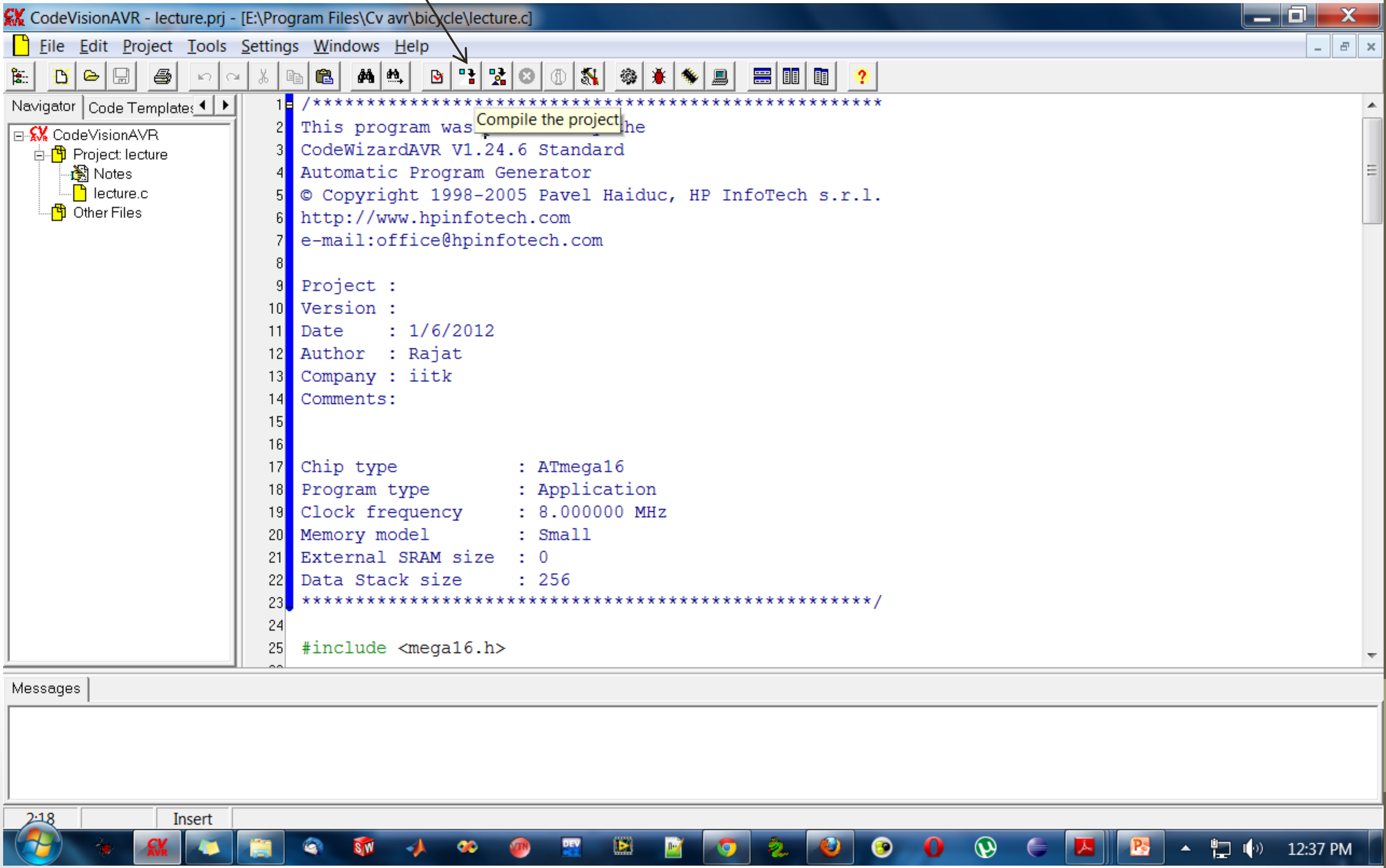
- This is because Atmega runs at a frequency of 8000000 Hz
- We need to introduce delay so as to see blinking
- Use header file delay.h
- Function to be used → delay_ms(time in millis);

```
While (1)
{
delay_ms(1000);
PORTA.1=1;
delay_ms(1000);
PORTA.1=0;
}
```

How to compile

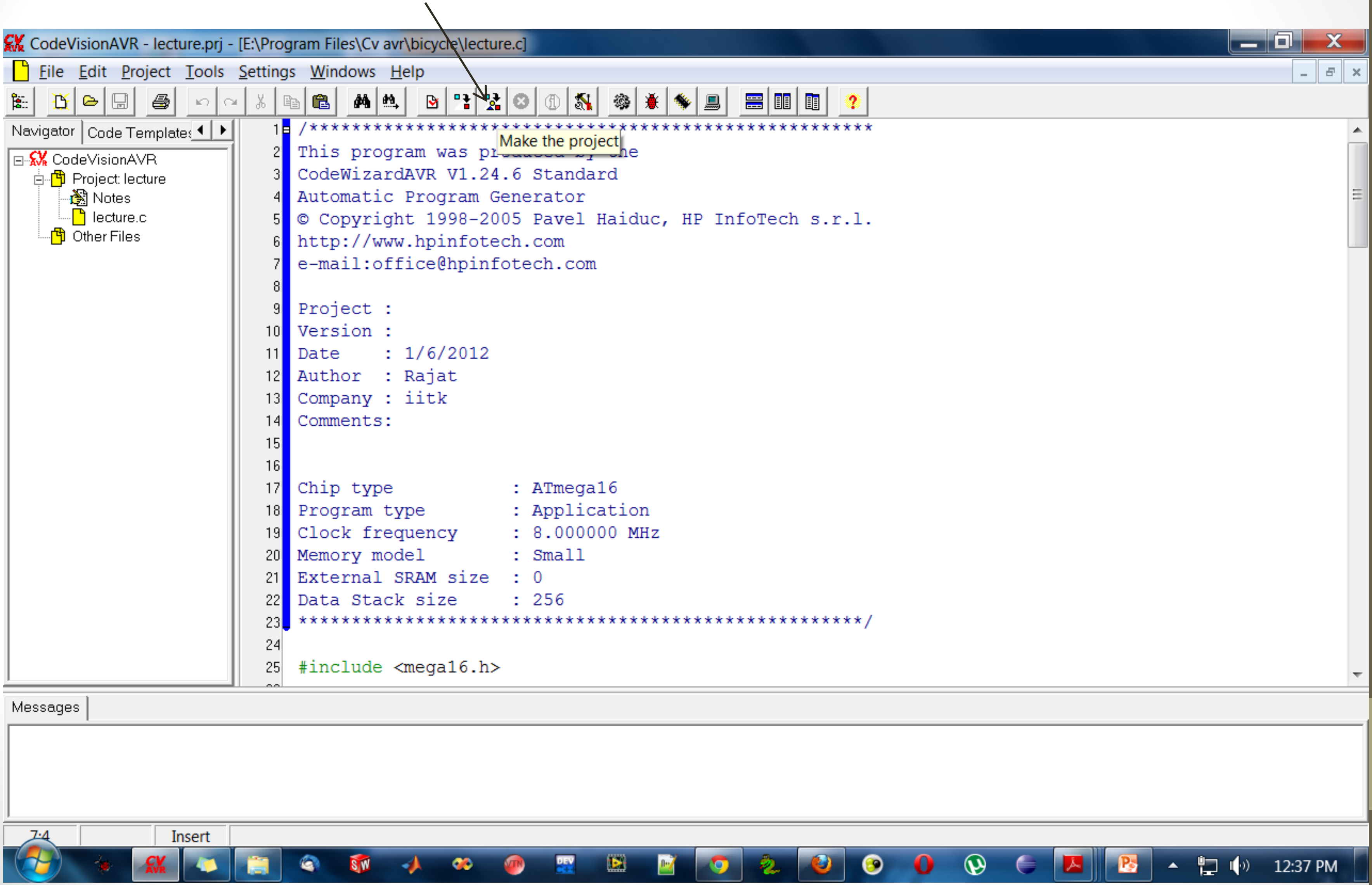
- Code is written in C language but Atmega understands Hex file
  so we need to convert the C file to Hex file

Compiling



```
/*****************************************************
This program was                    he
CodeWizardAVR V1.24.6 Standard
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project :
Version :
Date    : 1/6/2012
Author  : Rajat
Company : iitk
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 8.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
```

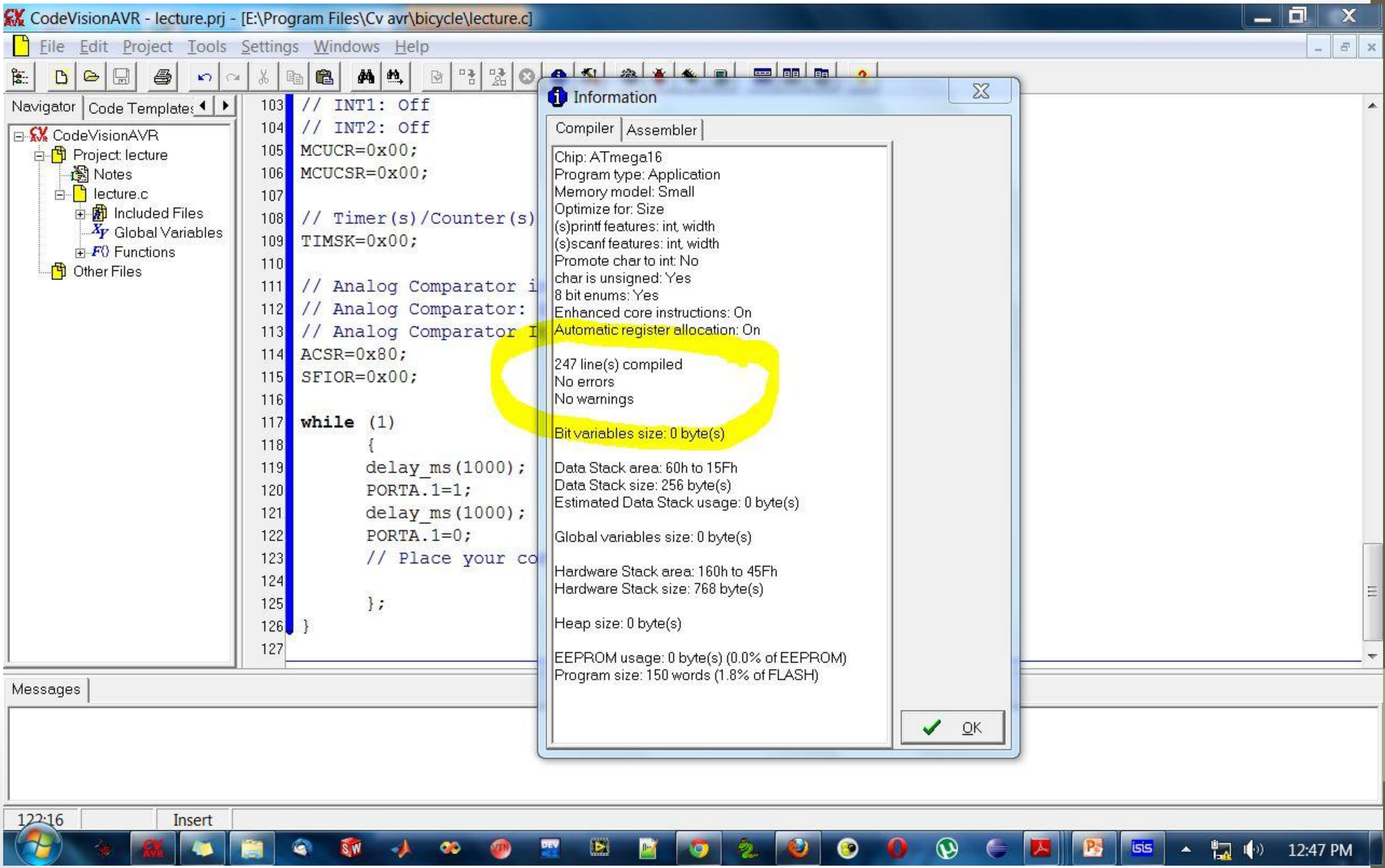
Make the Project



CodeVisionAVR - lecture.prj - [E:\Program Files\Cv avr\bicycle\lecture.c]
File Edit Project Tools Settings Windows Help
Make the project
Navigator
Code Templates
CodeVisionAVR
Project: lecture
Notes
lecture.c
Other Files
Messages
7:4
Insert
12:37 PM

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.6 Standard
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project : 
Version : 
Date    : 1/6/2012
Author  : Rajat
Company : iitk
Comments: 


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 8.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
```

Check for errors



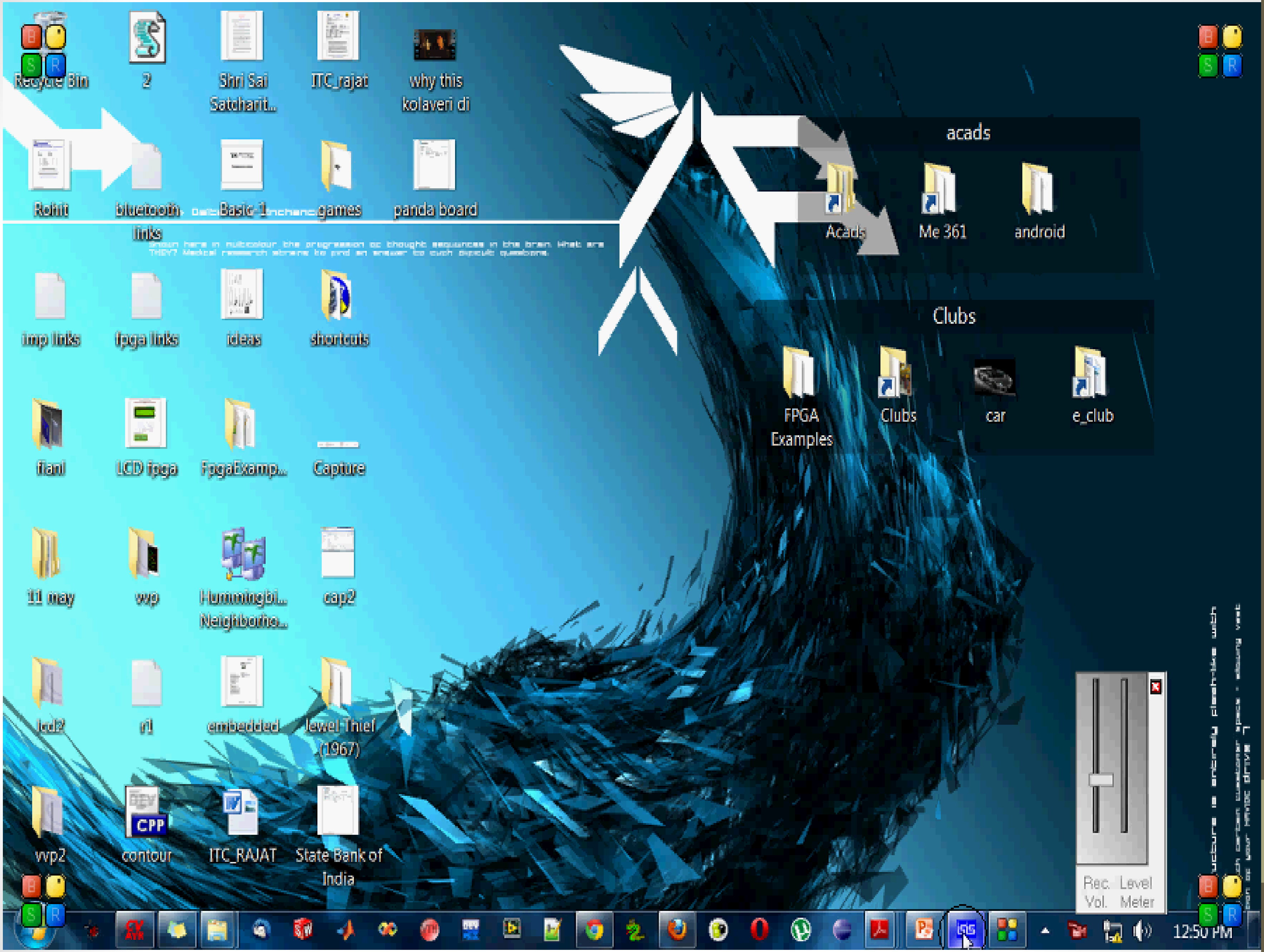
```
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s)
TIMSK=0x00;

// Analog Comparator i
// Analog Comparator:
// Analog Comparator I
ACSR=0x80;
SFIOR=0x00;

while (1)
      {
      delay_ms(1000);
      PORTA.1=1;
      delay_ms(1000);
      PORTA.1=0;
      // Place your co

      };
}
```

```
Chip: ATmega16
Program type: Application
Memory model: Small
Optimize for: Size
(s)printf features: int, width
(s)scanf features: int, width
Promote char to int: No
char is unsigned: Yes
8 bit enums: Yes
Enhanced core instructions: On
Automatic register allocation: On

247 line(s) compiled
No errors
No warnings

Bit variables size: 0 byte(s)

Data Stack area: 60h to 15Fh
Data Stack size: 256 byte(s)
Estimated Data Stack usage: 0 byte(s)

Global variables size: 0 byte(s)

Hardware Stack area: 160h to 45Fh
Hardware Stack size: 768 byte(s)

Heap size: 0 byte(s)

EEPROM usage: 0 byte(s) (0.0% of EEPROM)
Program size: 150 words (1.8% of FLASH)
```

Hex File

- You can find the Hex file in Bin folder or the EXE folder of the directory where You installed CVAVR

- So we Have our Code ready
- Feed this code to Atmega using Programmer (we will see this in workshop)
- Lets see the code in action



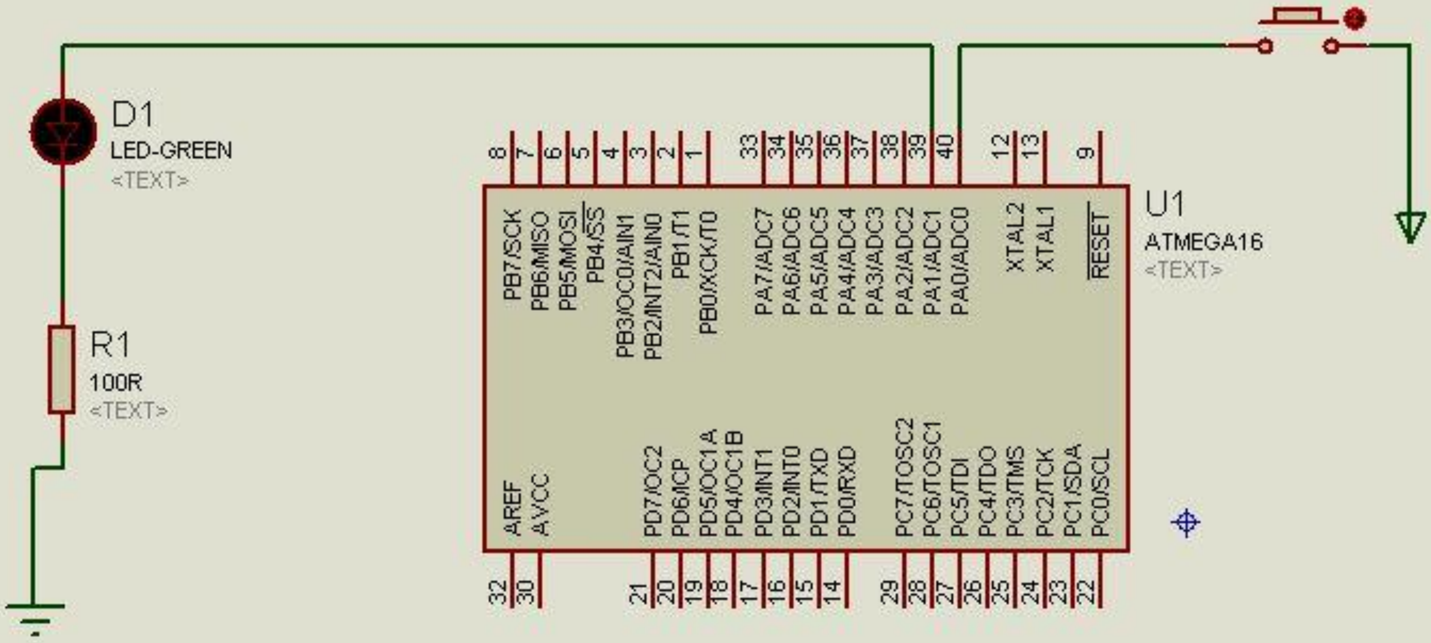
Shown here in multicolour the progression of thought sequences in the brain. What are THEY? Medical research strives to find an answer to such difficult questions.

Lets add an Input

- Most Common Input → Button

- Since we have already made A0 as Input we connect a button to that pin
- If button is pressed light the LED else turn it off
- First draw the Circuit Diagram

Circuit Diagram



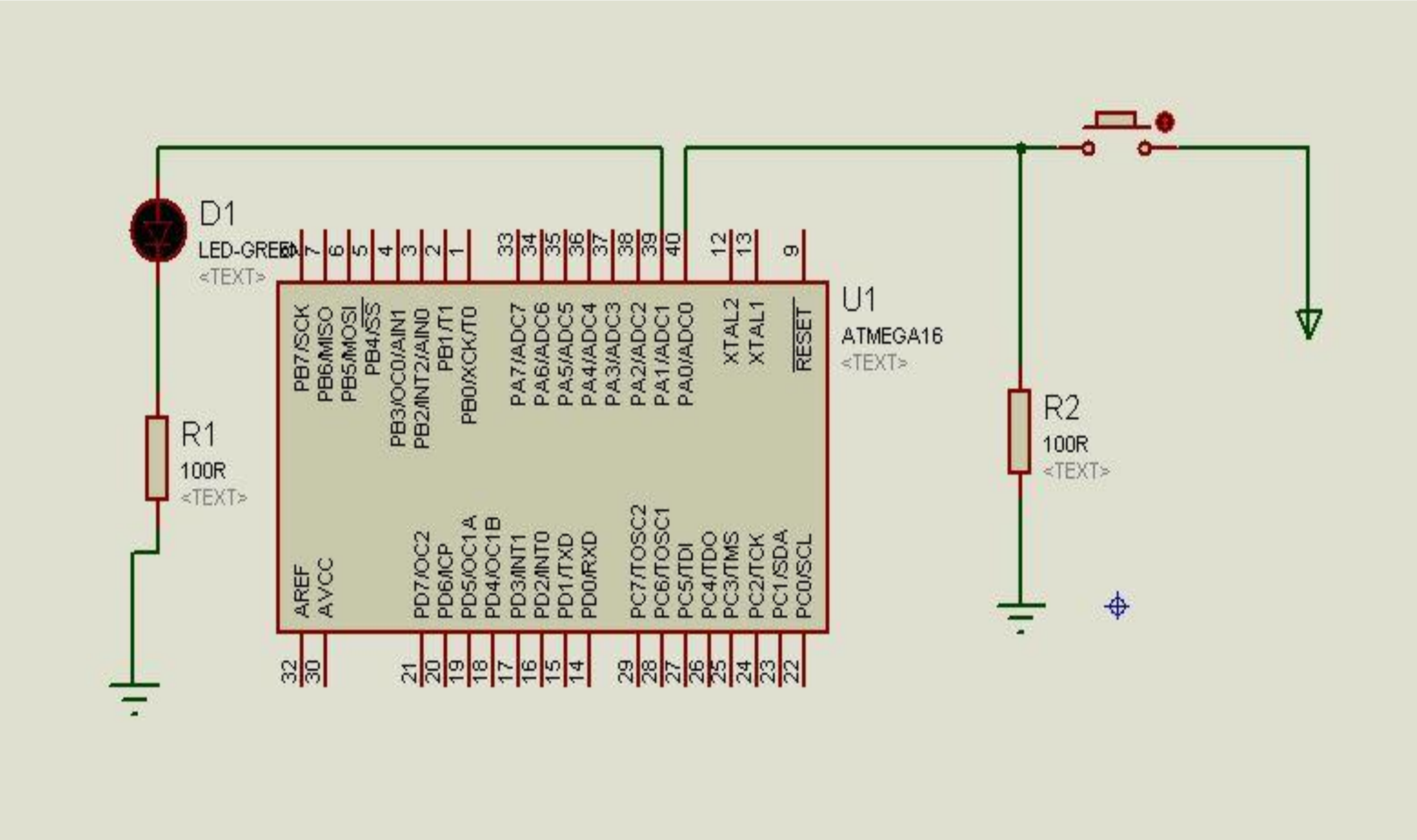
- Never leave any Input pin unconnected / floating
  at any point of time while your circuit is working
- In Last Circuit A0 is floating when button is not pressed
  so our Circuit Diagram is wrong

- What is the Voltage at the   Floating PIN ?

- Not 5 V
- Not 0V
- Its UNDEFINED
- So never leave an input pin unconnected
- Use the Concept of Pull up / Pull down

- In Layman terms
  - PULL DOWN : Gives 0V when unconnected
  - PULL UP : Gives 5V when unconnected
- Connect the PIN to Ground through a resistance for pulling down
- Connect the PIN to 5V through a resistance for Pulling up

Correct Circuit Diagram



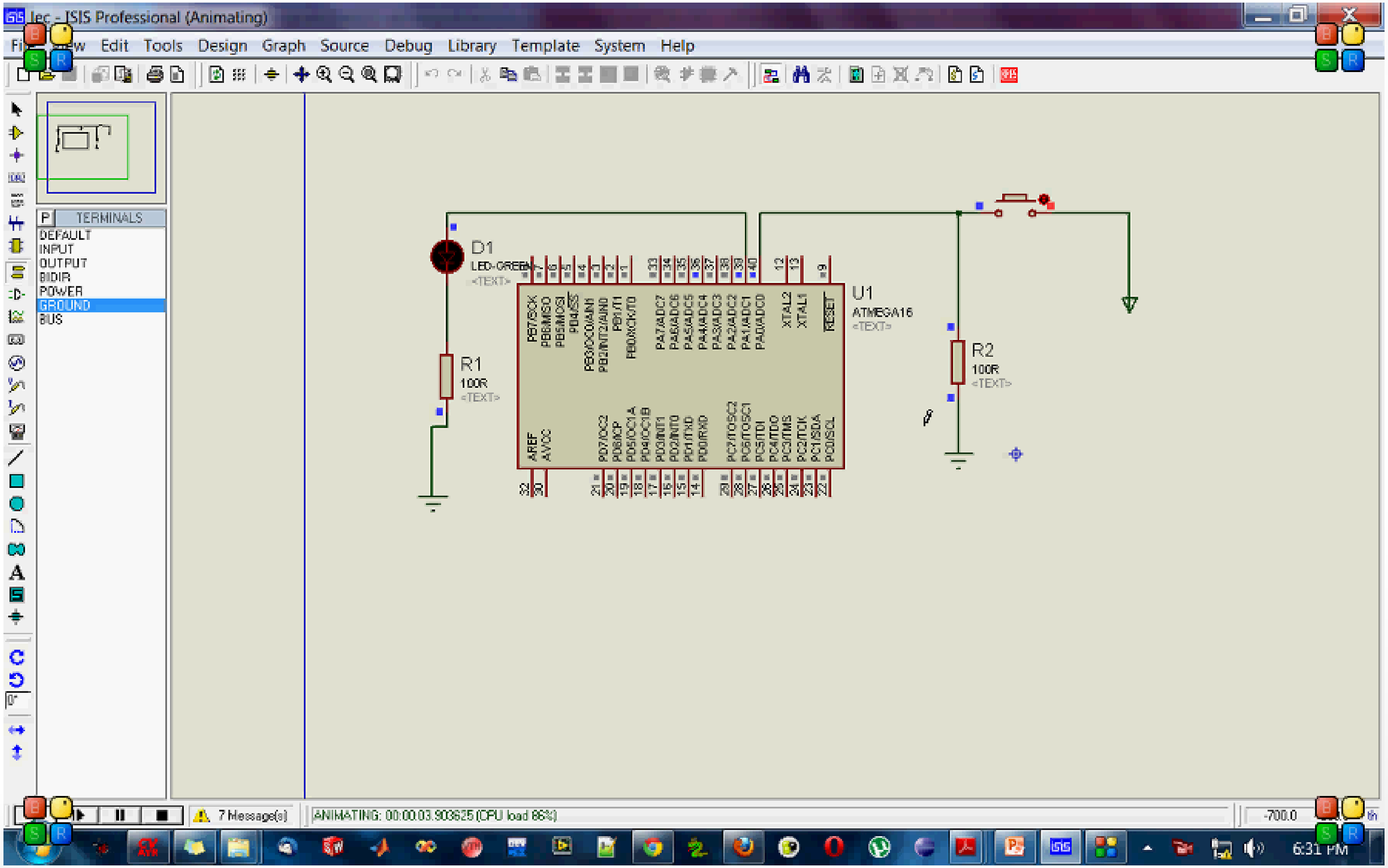
PIN Register

- It is a 8 bit register . It corresponds to the pin in same manner as that of DDR Register
- It is used to read voltage at a pin
- To be used only after the pin has been set as input by DDR register

Using Pin Register

```
int a;    // Define the variable a to store the value of voltage
a=PINA.0;  // read value at pin A.0 (make sure it is input)
If (a==1) //  if voltage is 5V
{
PORTA.1=1;  // Light the LED
}
else
{
PORTA.1=0;  // Turn off the LED
}
```

Code in Action



lec - ISIS Professional (Animating)

7 Message(s)

ANIMATING: 00:00:03.903625 (CPU load 86%)

Thank You