Introduction to Microcontrollers

Rajat Arora
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 -&gt; CVAVR

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

USB based STK500 Programmer for AVR microcontrollers
• 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 (condition)
  
  {
  
    ... ... 
  
  }

• 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.
• Let's Begin by blinking a simple LED
Getting Started with CVAVR
Open CVAVR
Go to File
Click on New
Select Project - > Click OK
Click YES
Select Chip

Chip: ATmega16
Clock: 8.000000 MHz
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.

- **DDRD**  
  For setting the direction i.e. Input or Output

- **PORTD**  
  For setting output value of port.

- **PIND**  
  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 Diagram]
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
• 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
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
• Click on In to make that pin Output
• Can do so for all four ports
Click on File
Generate Save and Exit
Enter name (3 times)
Where is the code stored?
Then Click Save
This program was produced by the CodeWizardAVR V1.24.6 Standard Automatic Program Generator © Copyright 1998-2005 Pavel Maiduc, HP InfoTech s.r.l. http://www.hppinfo.com e-mail:office@hppinfo.com

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

#include <mega16.h>
Make the Project

This program was produced by
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
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
Let's 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
• 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
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
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
Thank You