MICROCONTROLLER TUTORIALS 1

WHAT IS AN EMBEDDED SYSTEM?

An embedded computer is frequently a computer that is implemented for a particular purpose. In contrast, an average PC computer usually serves a number of purposes: checking email, surfing the internet, listening to music, word processing, etc... However, embedded systems usually only have a single task, or a very small number of related tasks that they are programmed to perform.

Every home has several examples of embedded computers. Any appliance that has a digital clock, for instance, has a small embedded microcontroller that performs no other task than to display the clock. Modern cars have embedded computers onboard that control such things as ignition timing and anti-lock brakes using input from a number of different sensors.

In general, an Embedded System:

- Is a system built to perform its duty, completely or partially independent of human intervention.
- Is specially designed to perform a few tasks in the most efficient way.
- Interacts with physical elements in our environment, viz. controlling and driving a motor, sensing temperature, etc.

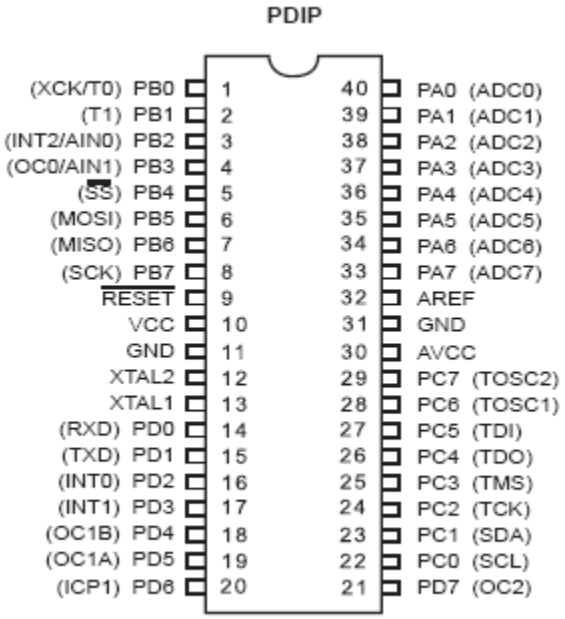
An embedded system can be defined as a control system or computer system designed to perform a specific task. Examples:

- Pen drives (for controlling the communication between P.C. and Flash Chip and also the small LED!)
- Hard disks(again for the same purpose)
- Mouse(Reads and Interprets the Sensors and send final result to P.C.),Keyboards
- Printers: Ever opened a printer for installing ink cartridge? Then you must have seen the printed head. There are motors to control the print head and the paper movement. Your P.C. is not directly connected to them but there is built in MCU of printer to control all these. Your P.C. just sends the data (pixels) through the communication line (USB or parallel).But the MCU used here is fairly fast and has lots of RAM.
- Automobiles
- Calculators, Electronic wending machines, Electronic weighing scales, Phones(digital with LCD and phonebook)
- Cell phones

WHAT IS A MICROCONTROLLER?

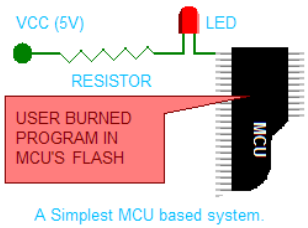
A microcontroller is an integrated chip that is often part of an embedded system. The microcontroller includes a CPU, RAM, ROM, I/O ports, and timers like a standard computer, but because they are designed to execute only a single specific task to control a single system, they are much smaller and simplified so that they can include all the functions required on a single chip.

In a microcontroller all that you have to do is to make proper connections of the pins and then feed a computer program into it. After that your microcontroller responds in accordance with the program that has been fed into it. In a microcontroller program you receive the inputs from a set of input pins that you specify and then process the input and produce your output on a set of output pins in form digital signal. However in order to connect the pins, you need to know the pin diagram of the MCU you are using. The pin diagram of Atmega 16/32 has been given below:



A SIMPLE MCU BASED SYSTEM.

A simplest MCU system may look like below



A Simplest MCU based system.

The program it is executing is like this (In C language).The MCU contains a flash memory where it stores its program (it is like a hard disk to it). The flash memory can be easily erased and a new program can burned. This makes them very flexible. MCUs can be programmed few thousand times before they die.

A SMALL NOTE ABOUT "DELAY"

C has inbuilt libraries which contain many pre-built functions. One such function is "Delay", which introduces a time delay at a particular step. To invoke it in your program, you need to add the following line at the beginning of your code:

```
#include <delay.h>;
```

Thereafter, it can be used in the program by adding the following line:

```
delay_ms(X);
```

Where X is the time delay you wish to introduce at that particular step in milliseconds.

A SAMPLE PROGRAM

```
#include <delay.h>;

void main()
{
    SetPortDirection();
    while(1)
    {
        PORTA=0b00000001;
        delay_ms(500);
        PORTA=0b00000000;
        delay_ms(500);
    }
}
```

PORTS

A MCU has some ports. Ports are PINs on the MCU that can be turned on and off by program. On means 5V and off means 0V or GND .This behavior is for OUTPUT mode. They can also be put in INPUT mode. In INPUT mode they can read what is the signal level on them (only on and off).If voltage is more than a threshold voltage (usually half the supply) it is reported as ON(1) otherwise OFF(0).This is how MCU control everything .Majority of the PINs of a MCU are PORT so you can hookup lots of gizmos to it !!!They are named PORTA ,PORTB ,PORTC ,PORTD etc .They are of one byte which means 8 Bitts all bits of them are connected to external pins and are available outside the chip. In smaller chips only some of the eight bits are available. Setting PORTB=0b00000001 will set PORTB's zeroth bit high that is 5V while remaining PINs will be low (GND). [NOTE: To write a binary number in c prefix it with 0b ex 0b00001000. It is decimal 8 not 1000!!!]

What the above program does:

STEP 1 SetPortDirection(); This Function Makes the PORTB as OUTPUT. Its implementation detail is not shown.

STEP 2 PORTB=0b00000001; makes the 0th bit high, switching off the L.E.D. because other end of LED is connected to VCC (i.e. Supply voltage 5V). Note that the 0 in 0b is a “zero”, not an “oh”.

STEP 3 delay_ms(500); Halts the MCU for 0.5 Sec

STEP 4 PORTB=0b00000000; Switches on the LED

STEP 5 delay_ms(500); STEP 2 to 5 are within an infinite while loop so it runs till MCU is powered. The program is compiled, and burned into the chip and power is turned on and woillaaaa that LED is blinking once every second. Although the given program doesn't do something very important and can be done without a MCU and in a cheap manner, but it introduce you to microcontroller programming .Doing this with MCU has some advantages also. You can change the way LED blinks by simply writing a new program without touching the hardware part. You can also connect 8 LEDs to all 8 PINs of the PORTB, write a nice program them to light them in various pattern and you have a deluxe decorating lights!!! Which you can't make easily without MCU. So you have seen that major functioning of a MCU project is made in software and hardware part is simple and small. So you have learned the basics of MCUs and their use.

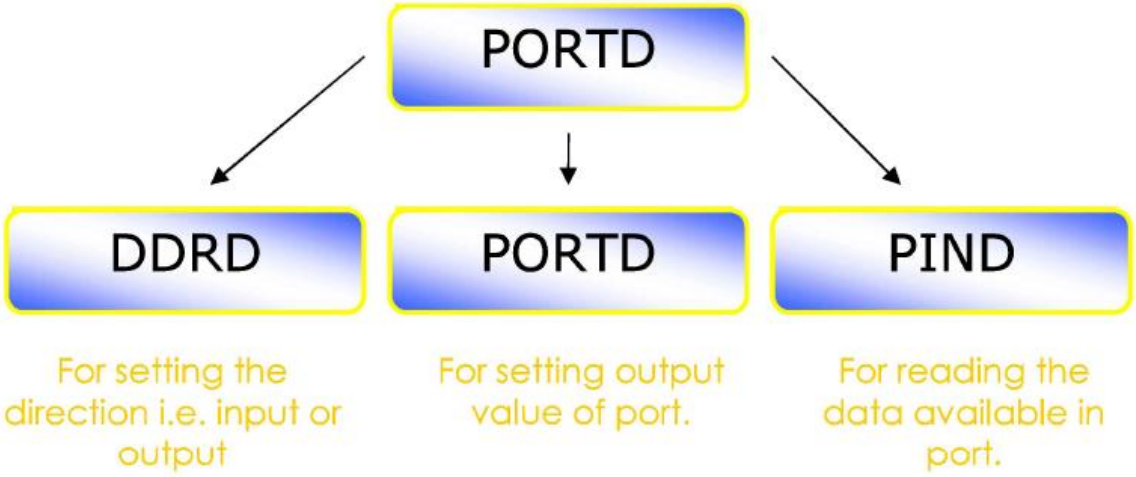
BASIC DIGITAL I/O

Digital IO is the most fundamental of connecting a MCU to the external world. The interfacing is done through PORT. A PORT is a point where data internal to the MCU chip comes out. They are present in form of PINS of the IC. Most of the Pins (32 out of 40) are dedicated to this function and other pins are used for Power supply, clock sources etc. Once introduced to the concept of Ports the next task is to learn how to use these Ports to get Input from a port and to Output to a Port. IO is usually done by controlling the values of certain registers associated with these PORTs. A register is

a variable (usually 8 bit) whose value can be changed and read from within your program just like you do for any other variable. Therefore IO is done in a very simple fashion simply by altering or reading the values of certain variables. There are 32 pins available for IO (8 pins per Port) and each pin can be set to either take input (voltage high or low) or to output a digital value (0 or 1). In order to control which pin should do input and which should do output, there is a register called DDR (Data Direction Register), whose value tells the microprocessor which is to be set to input and which to output. For example , to set Pin no 0, 2, 3, 7 of Port A to input and other pins of PORT A to output, the command would be:

DDRA=0b01110010;

Here DDRA means the DDR register of PORT A. 0b means that we are entering the number in binary. The sequence of 0s and 1s indicate which pin is to be input and which is to output. A 0 means input and 1 means output. We could equivalently write: DDRA=114; (decimal equivalent of 01001110). Next comes how to read/write data from/to these pins. For this task there are two registers, PORT and PIN. **PIN** (**P**ort **In**put) is the register that reads the input from the pin. For example in order to read the value of pin number 3 of port A into a variable x, the command would be: x=PINA.3; or equivalently x=PINA&0b00001000;.where the '&' is binary AND. x would be 0 if PINA.3 is set to low otherwise it would be a non zero value. You can only read from a PIN register, you cannot write into it. PORT is the register that is used to output values. For example to output 1 on pin no 5 of Port D, you would say PORTD.5=1; or PORTD=PORTD|0b00100000 after setting pin 5 of port D to output. Here Boolean algebra is used and the reader is supposed to be familiar with such concepts of ANDing and ORing bits. To summarize, IO is done through PORTS and each PORT is associated with 3 registers for IO.



A SAMPLE PROGRAM

Suppose that you have a LED and a switch. Now you want that when you press the switch the LED is switched OFF, otherwise it is ON. Also suppose that you have made proper connection of the mcu, that is provided power connections (GND on the GND terminals-11 & 31, and Vcc on 10 and 30 and also on $\overline{\text{RESET}}$. Then your next task is to connect the output of your switch to appropriate pin, say

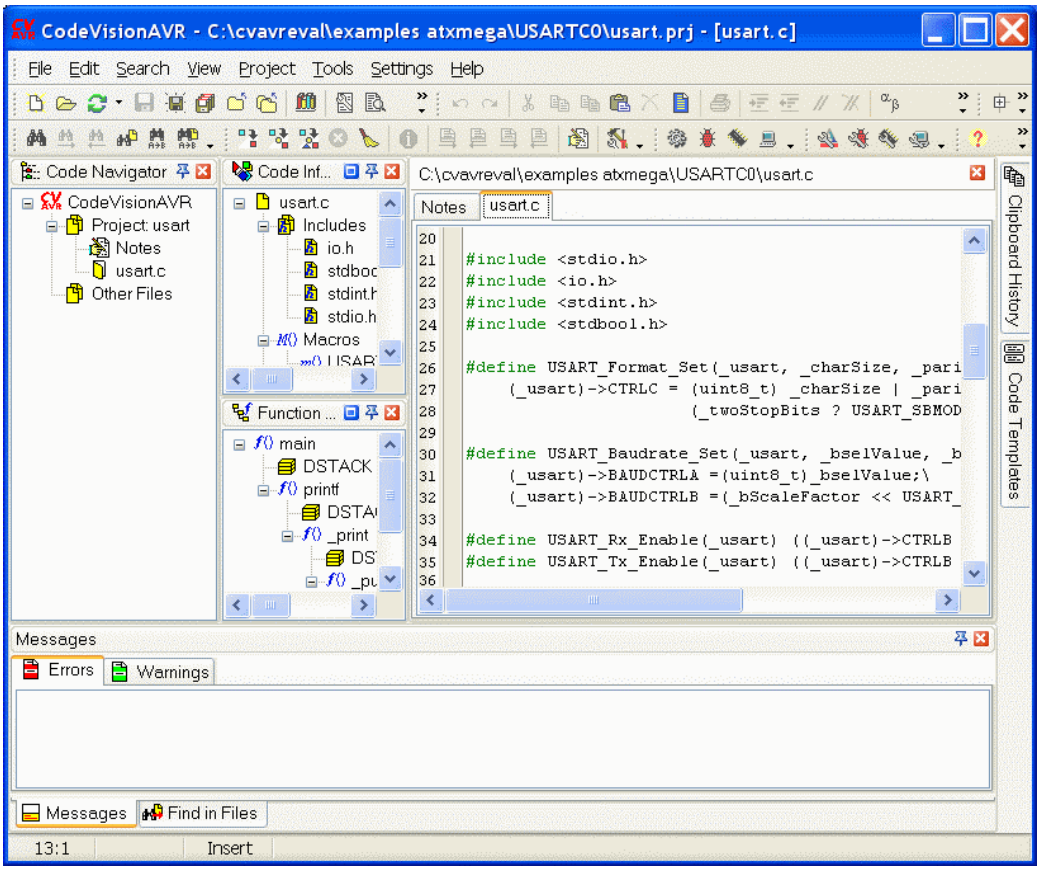
connect it to Pin 0 of Port C, and connect LED to, say Pin 2 of Port C. Then the Programme to do the above mentioned task will be:

```
#include <delay.h>;

DDRC=0b00000100   //it is a good practice to set unused pins to input.
While (1)
{
      If(PINC.0==1)
      {
            PORTC.2=0;
            delay_ms(100);
      }
      else
      {
            PORTC.2=1;
            delay_ms(100);
      }
}
```

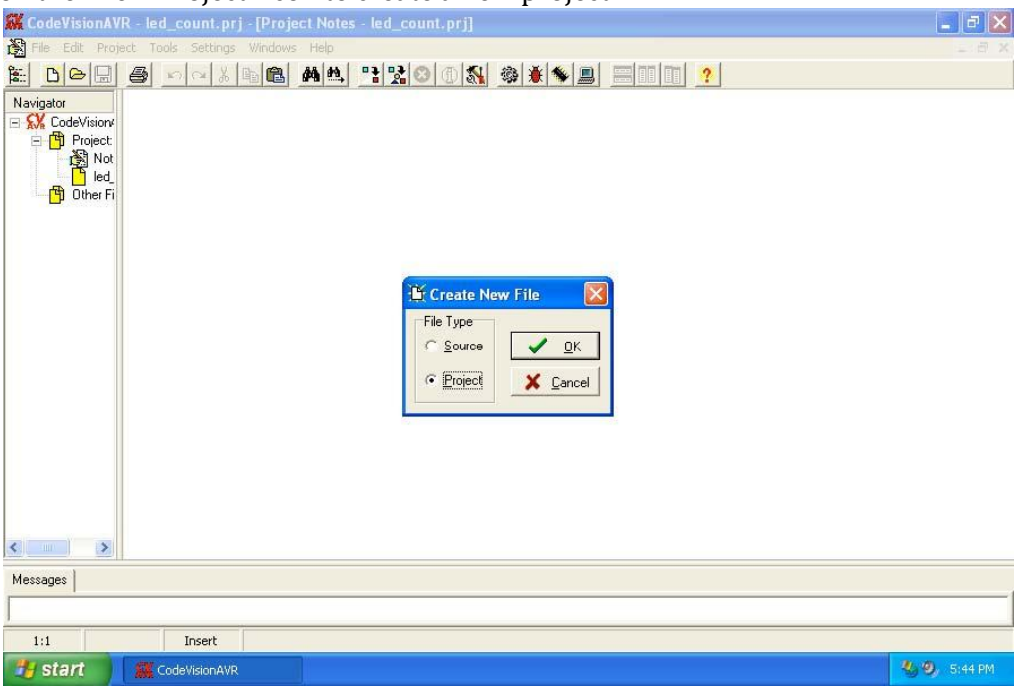
GETTING STARTED WITH CODEVISIONAVR

CodeVisionAVR (CVAVR) is the C-program language compiler that shall be used to program the MCU. CVAVR is a highly versatile software which offers “High Performance ANSI C Compiler, Integrated Development Environment, Automatic Program Generator and In-System Programmer for the Atmel AVR family of microcontrollers.” After installing and setting up CVAVR, a typical screen with a program open looks like this:



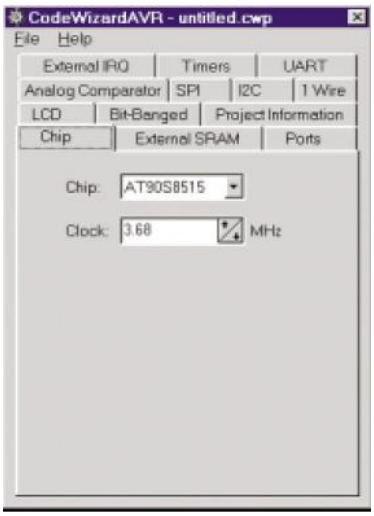
CREATING A NEW PROJECT:

- Open up CodeVisionAVR on your PC.
- Click on the “New Project” icon to create a new project.

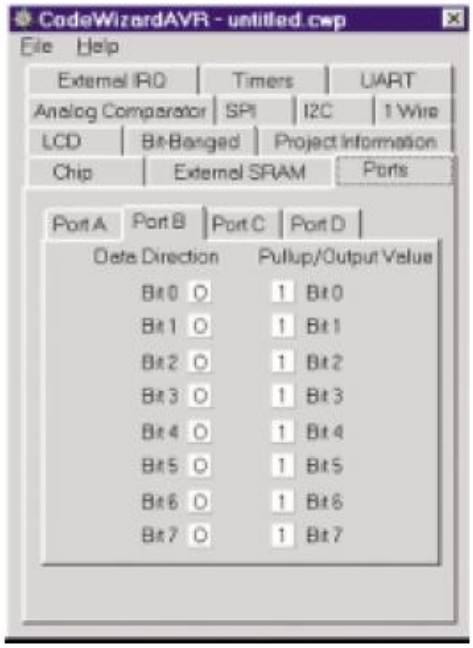


- When the “Create New File” dialog box pops up, click “Project” then “OK.”

- A dialog box titled “Confirm” will pop up asking if you would like to use “CodeWizardAVR.” This is a helpful tool which will help you automatically generate the proper code depending on your MCU. Select “Yes”. The following window will open:



- Select the appropriate Microcontroller and its appropriate frequency.
- Now, click on the Ports tab to determine how the I/O ports are to be initialized for the target system:



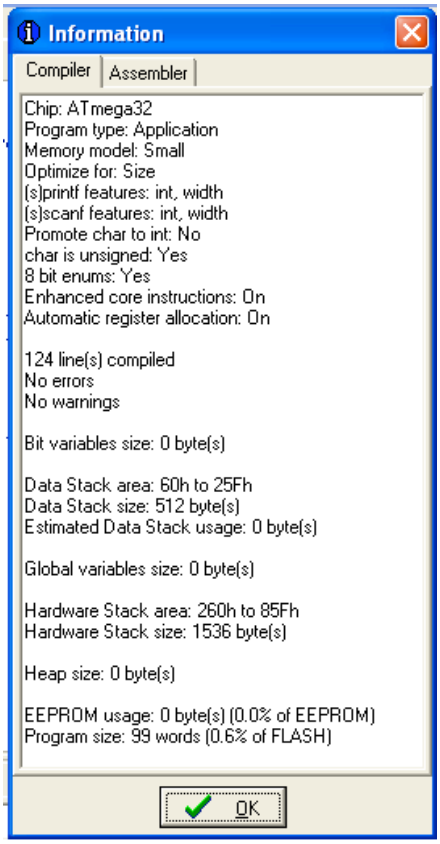
The default setting is to have the ports for all the target systems to be inputs.

You can also change other settings in this window such as Timers, etc. These topics shall be covered in the following tutorials.

- By selecting the File -> Generate, Save and Exit option, the CodeWizard will generate a skeleton C program with the appropriate Port initializations. Many Save File prompts shall open – these are the project files generated by the wizard. Save them with appropriate names.

- Now, type your code in the source code window.
- Once you're done with the creation of the source code, you can "make" the project by clicking on the "Assemble" button.

- A dialog box appears. Make sure that the message says "No errors," otherwise, go back to your code and fix the errors.

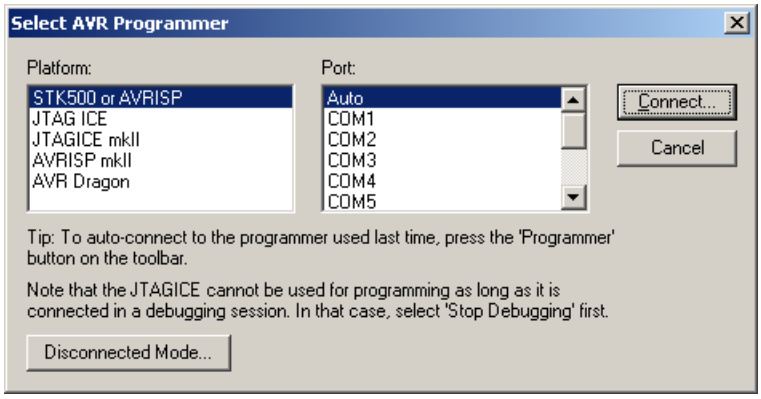


- If there are no errors in the compilation, it is time to program the chip.

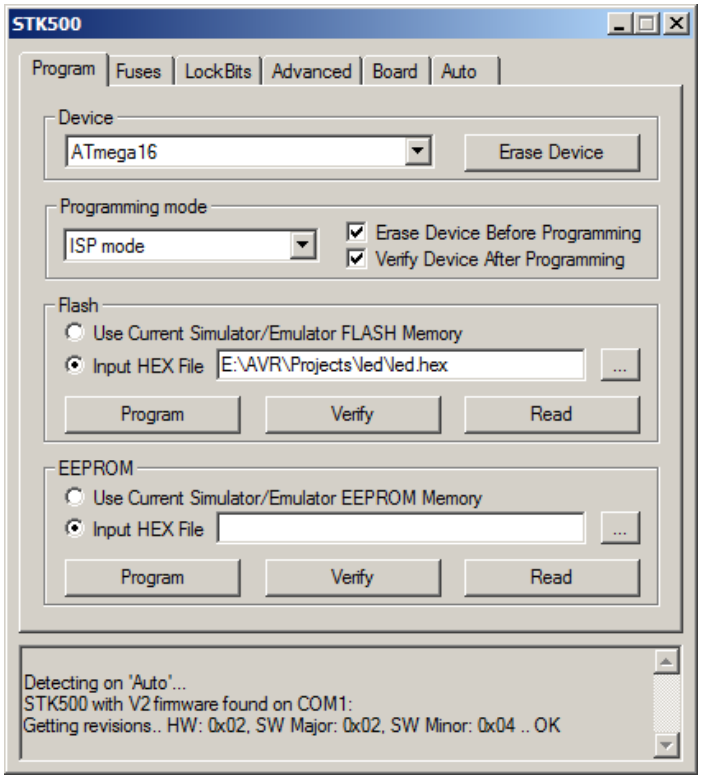
PROGRAMMING THE MCU USING AVR STUDIO

Now that the program is ready, it is time to put it on the chip. This is accomplished by a software known as AVR Studio:

- Start AVR Studio. It will immediately ask you to start a new project. Click on Cancel.
- In AVR Studio, select menu Tools | Program AVR | Connect.
- In the 'Select AVR Programmer' dialog box, choose 'STK500 or AVRISP' as the platform and 'Auto' as Port. Then click button **Connect**.



- Depending on the version of your AVR Studio, a message about firmware *may* appear. For now, this message can be discarded by clicking button **Cancel**.
- In the 'STK500' dialog box that appears, select the generated hex file as 'Input Hex File'. Then, click the button **Program** to download the HEX file to the AVR chip.



- The program will now run on the microcontroller.