Getting Started with microcontroller

By- Swapnil Jariwala

I met with this wonderful device microcontroller when we decided to make micromouse. It took us two months to decide which controller to use (we chose wrong controller first so we have two controllers now) & two months to understand it & program it. We got very little time to actually work on controller & make micromouse.

So I am writing this article to help those who are new to microcontroller. This will save your time in getting basic info. about how to program controller, which hardware & software to use for compiling & burning.

Choosing A Microcontroller

There are many manufacturers of microcontrollers like Intel, Motorola, Microchip, Atmel , Texas Instruments & many more. Depending upon your application need you have to choose microcontroller.

I will discuss here about microcontrollers from **ATMEL** only. You will know why ATMEL only as you read more.

Most important parameters to select uc are

1) Number of I/O lines required.

2) How much Flash memory you need ?

3) How much RAM memory you need ?

After this there are many parameters like

1) EEPROM size

2) Number of timer/counter & resolution (8bit/16bit)

3) ADC, number of ADC channels & resolution (usually 8 channels & 10bit in Atmels)

4) number of USART channels

5) Number of external interrupts

& the list goes on

Don't get afraid with this list. We normally don't need many of them. Now next step is to see datasheets & choose controller with parameters matching to your application.

Some ICs are

ATmega8535

Flash : 8k RAM : 256 bytes EEPROM:256 bytes

Number of ADC channels:8 ADC resolution:10bit

ATmega16

Flash : 16k RAM : 1k EEPROM:512 bytes

Number of ADC channels:8 ADC resolution:10bit

Timer/counters: two 8 bit, one 16 bit with PWM

External interrupts : 2

ATmega162

Flash : 16k RAM : 1k EEPROM:512 bytes

Timer/counters: two 8 bit, two 16 bit with PWM

External interrupts : don't know exact number but it has more than 8

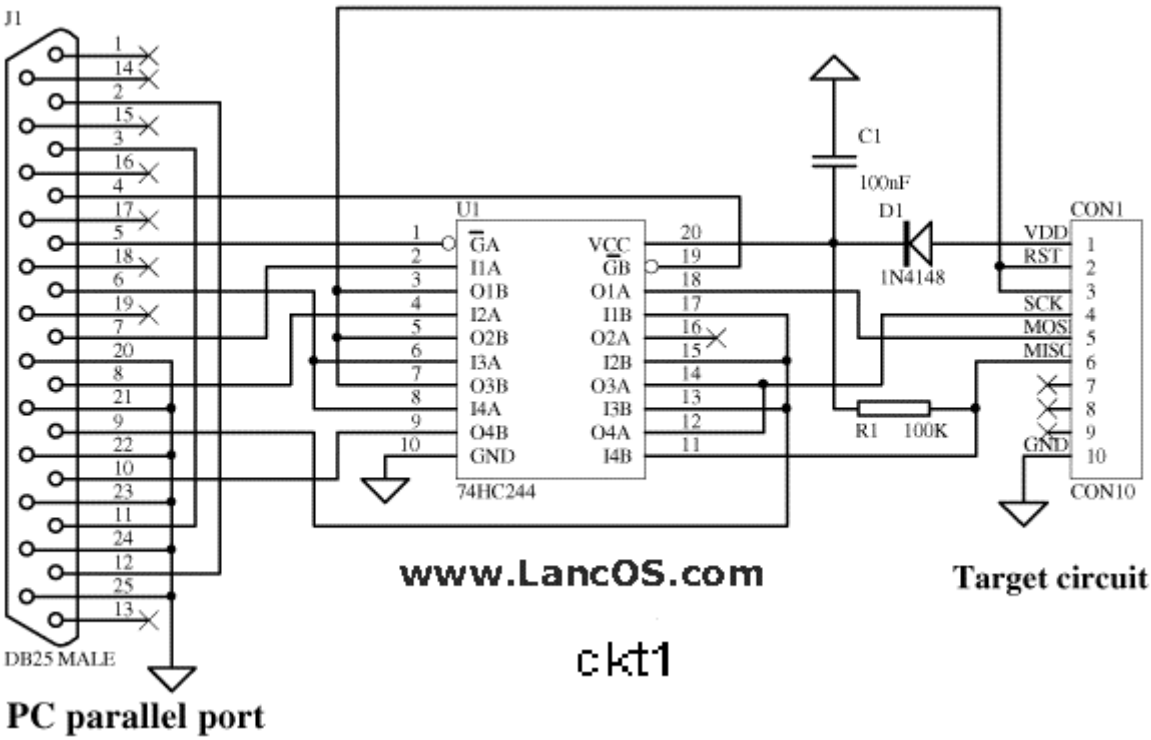
Why ATMEL?

The most important feature that attracted me to Atmel is its ease in programming the device. development tools are freely and very easily available with many options.

All Atmel uc have **In System programming** capability via SPI using only 5 pins to program. That means you don't need to pull out uc chip every time you want to program device. You can program device in the circuit you are working that's why it is called in system.

OK now you have purchased your controller & now you can't wait to program it.

Now next thing is to design a programmer ckt to program your uc as i said ATMELs use SPI (serial peripheral interface) for programming & pins used for programming are MISO, MOSI, SCK, RESET & GND.

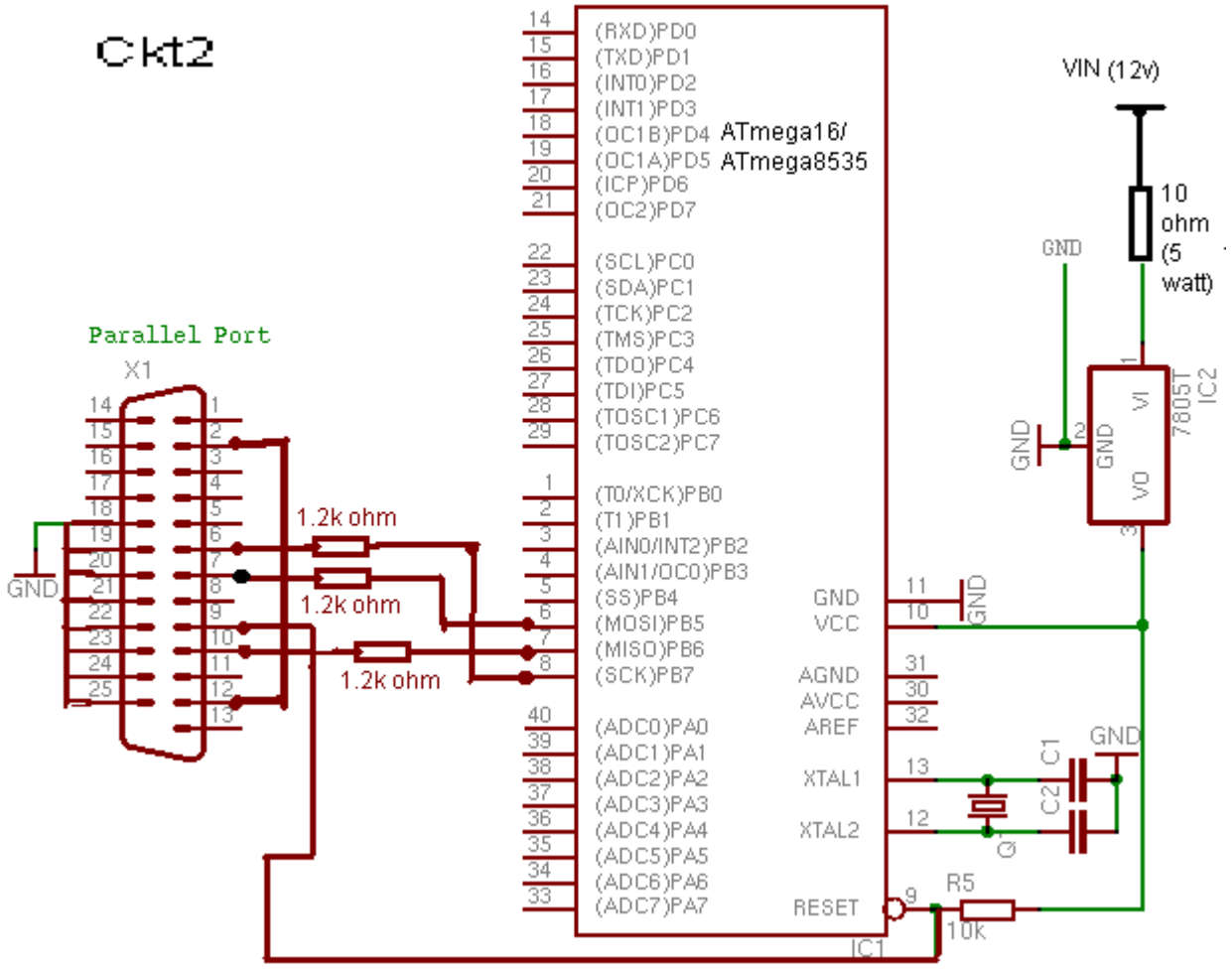


or from this pdf file provided by electronics for you isp.pdf

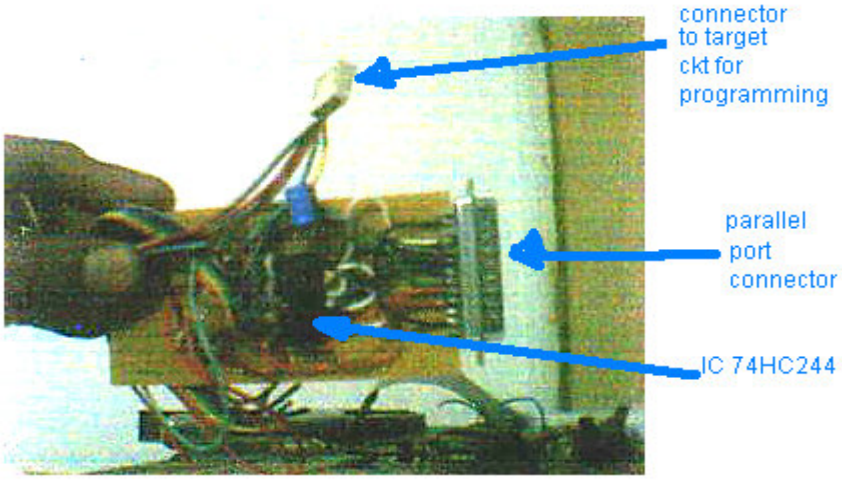
On target board (your main ckt with uc on which you are working) connect +vcc to 5v regulated power supply. you can use any regulator IC eg. 7805 +5v regulator. connect all the GND pins to ground (ATmega has two ground pins where as ATmega162 has one). In case of ATmega16 or ATmega8535 connect AVCC & AVref to +vcc.

Connect crystal between XTAL1 & XTAL2 depending upon frequency connect two capacitors across it for 8MHz C=22pF. You can leave these pins open if you want to use internal oscillator of 8MHz all ics I mentioned have internal oscillator of 8MHz so you can skip crystal.

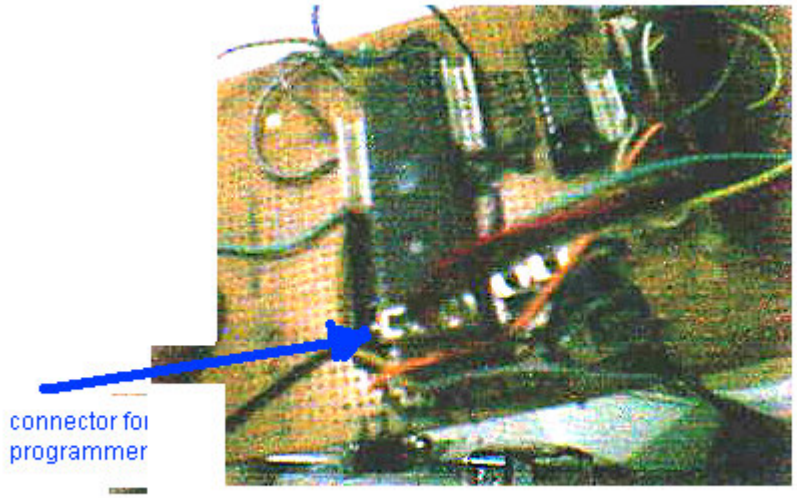
i derived ckt below from bsd programmer it is much simpler than above ckt. this ckt works for ATmega16, ATmega8535. In case of ATmega162 AREF & AVCC pins will not be present & pin numbers for +vcc & gnd will change.



Use this ckt1 if you are worried too much for your parallel port otherwise ckt2 is simple. But Take precaution take that no resistance is skipped



Programmer ckt



Target ckt board

these are low quality pics taken from my webcam of our micromouse. above programmer ckt connects to parallel port. & from programmer ckt a six pin connector connects to micromouse ckt.

Now your ckt ready for use & Now you can program it. Now lets see how to use AVR Studio & ponyprog.

Now you need to install following software packages.

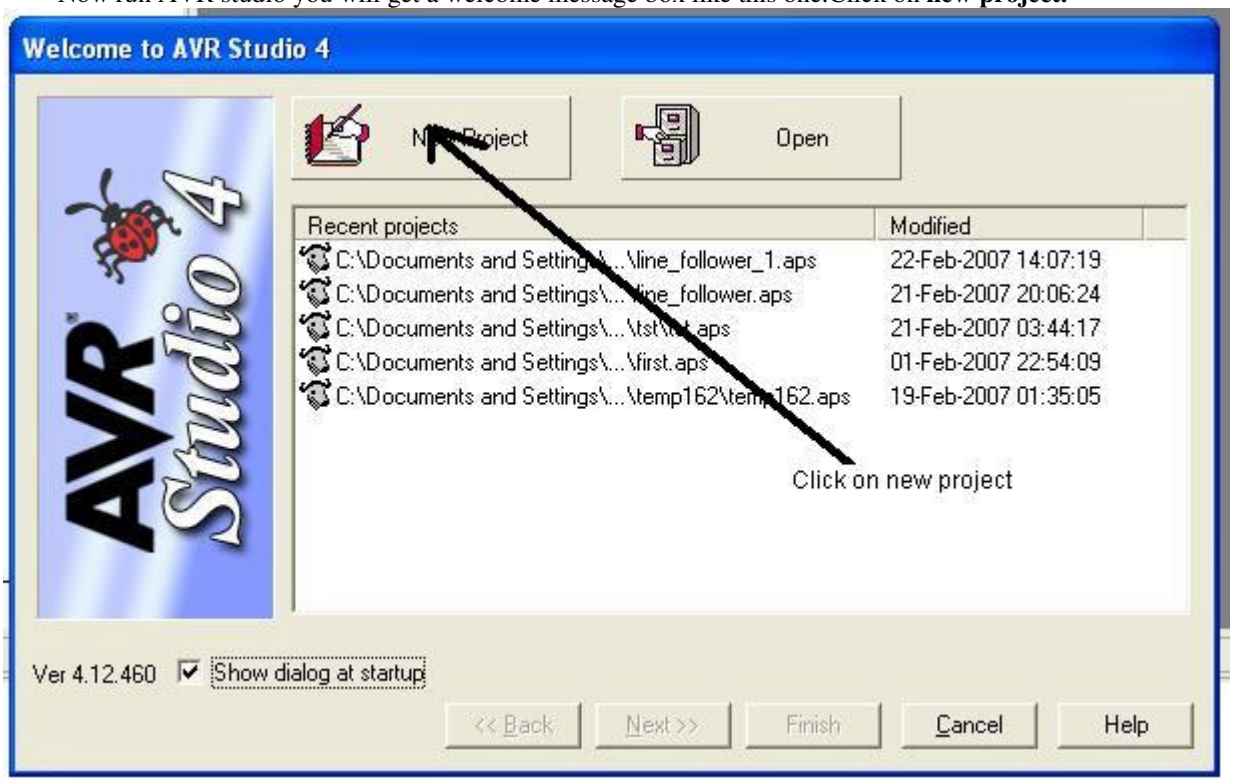
1)**WINAVR** (Search in google:winavr sourceforge.net): Winavr is c & assembly compiler for Atmel's controller it is result of free software movement so it is freeware.

2) **AVR studio** (available on www.atmel.com): AVR studio is assembler provided by atmel & after installing winavr you can comple C programs from AVR studio.

3) **ponyprog** (available on page www.lancos.com/prog.html) : For downloading the hex file created by compiler on uc.

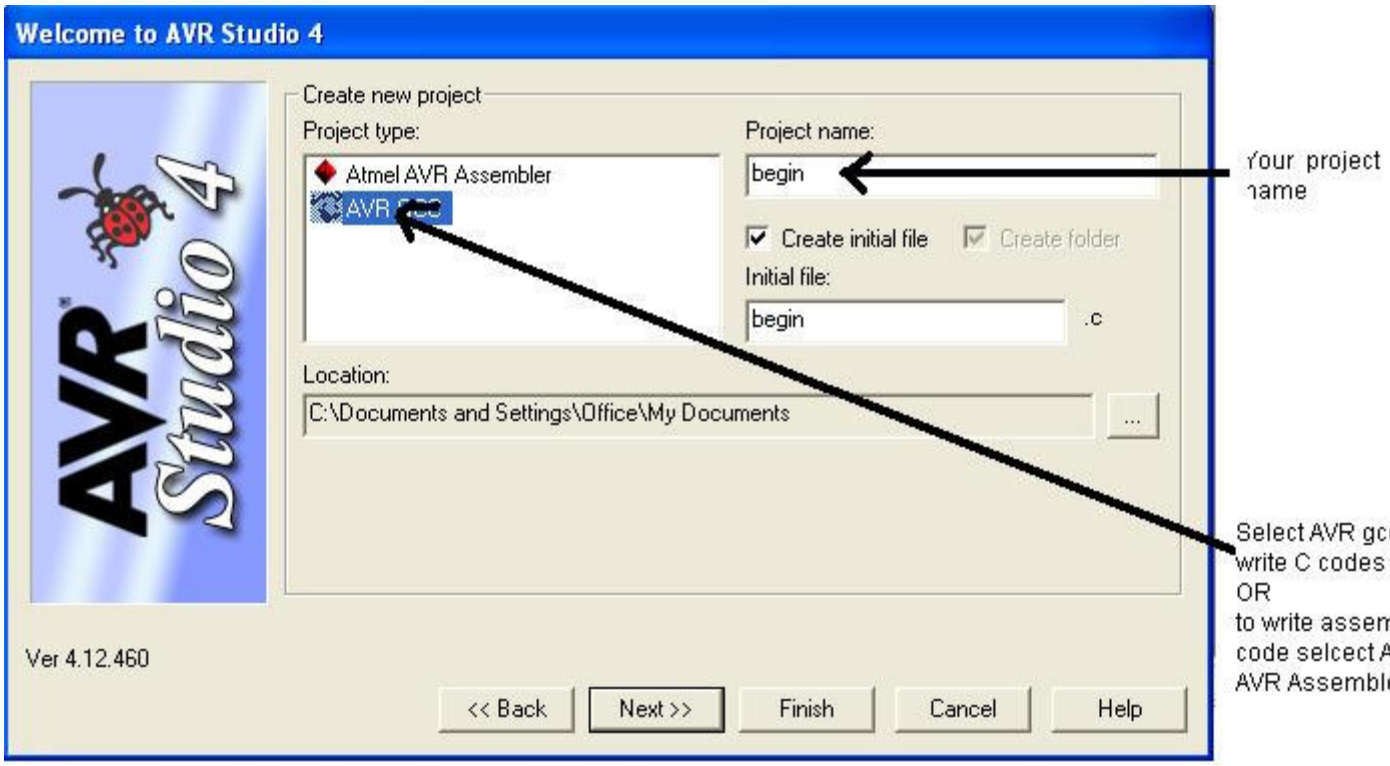
If you want these softwares you can contact me or sumedh.

Now run AVR studio you will get a welcome message box like this one.Click on **new project.**



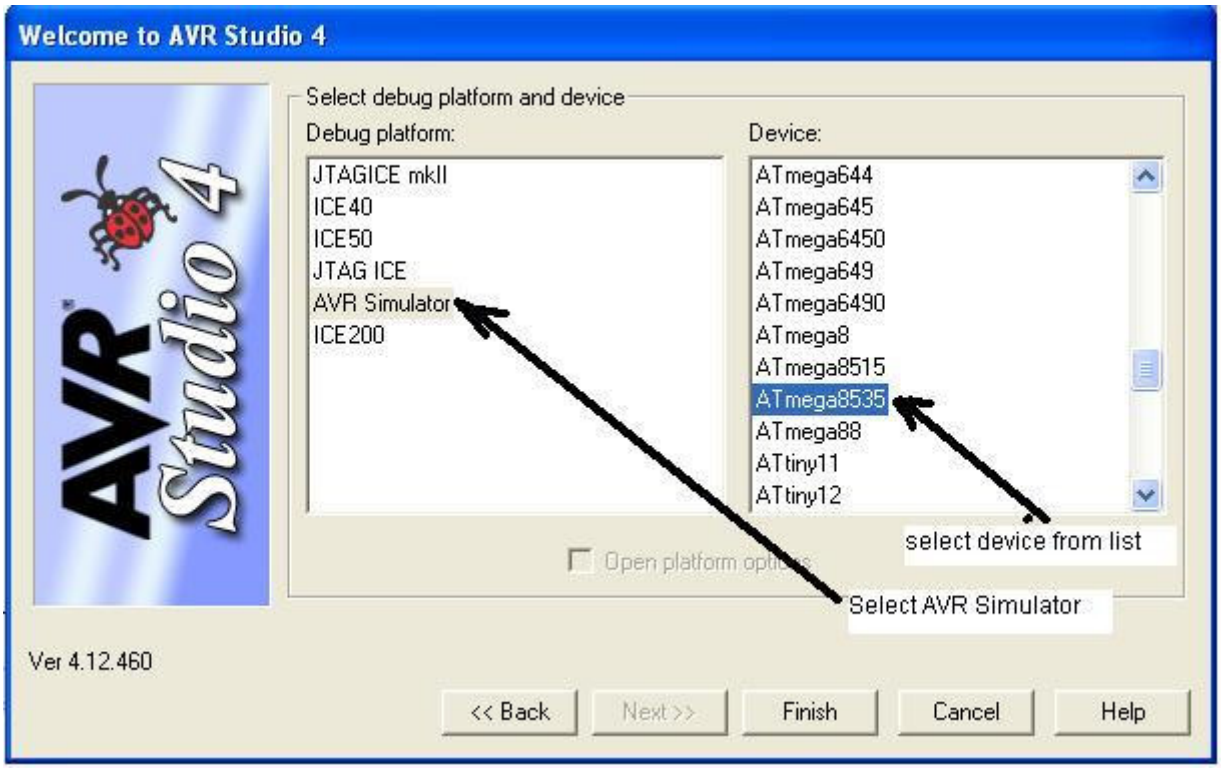
Now you have two options either you can write code in C or assembly language. you can refer datasheets for assembly language. I will be using C everywhere.

In new project window insert your project name & Click **NEXT**.



NOTE : FOR AVR GCC project you need to install WINAVR first.

Select Debug platform : AVR Simulator Device : ATmega16 or ATmega8535 in our case & Click **Finish**



Now you are ready to write a code. Type Following code as it is I will explain it in next article with all details.

```
#include<avr/io.h>

void main()

{

   DDRD=0xff;    // Defining all PORTB pins as OUTPUT pin

   PORTD=170;  // (10101010) in binary

   while(1);       // infinite loop

}
```

To assemble this code go to **Build->Build** or simply press F7

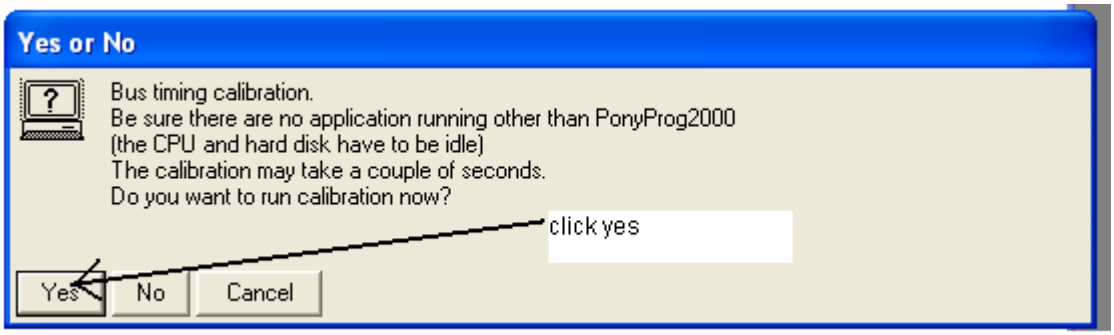
Once you have build the project avrstudio makes a .hex file in folder named **default** inside your project directory for eg. if your project name is hello. then destination of hex file is

My Documents-> hello -> default -> hello.hex

Using PONYPROG2000

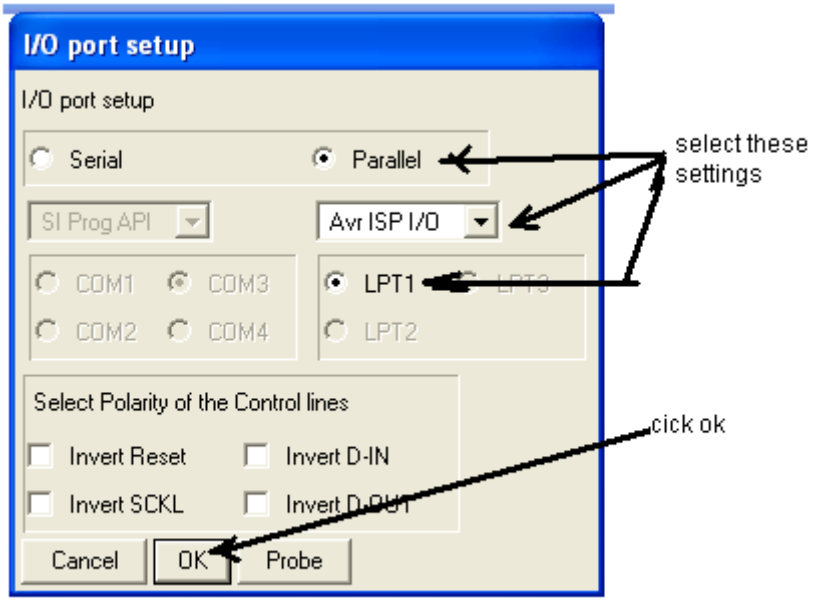
Open ponyprog2000. For the first time you need to calibrate your bus. Click on **Calibration** inside **Setup** menu. click yes.

Setup->calibration



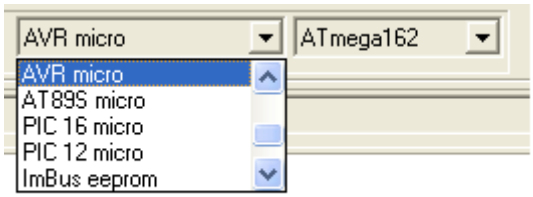
Now go to

Setup -> Interface setup.. & choose following options.



Note : On some computers Ponyprog cannot access parallel port in that case you will need to use onother software **avrdude** available with winavr (\winavr\bin\avrdude-gui.exe)

Now select device family to **AVR micro** & device number to your uc no eg. **ATmega162**.



Now open .hex file to be loaded in the uc

File->open Device file.. select your hex file

Hex file looks something like this

Now program the uc

Command -> write program (Flash)

If your hardware is right then it will display message **write successful** else **write failed .**

THATS IT YOU HAVE DONE IT.