AVR Trainer Kit

User Manual

TECHSHOP
Bangladesh

Contents

Document Title	AVR Trainer Kit User Manual
Version	1.3
Date	2014-11-14
Status	Design Changed
Document Control ID	AVR Trainer Kit_Manual_V1.3

General Notes

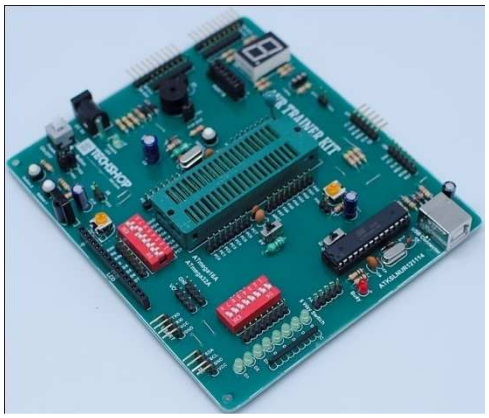
TechShop offers this information as a service to its customers, to support application and engineering efforts that use the products designed by TechShop. The information provided is based upon requirements specifically provided to TechShop by the customers. TechShop has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by TechShop within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

Version History

Date	Version	Description of change	Designer
2010-01-27	1.0	Origin	FahadMirza
2013-05-30	1.1	Design Changed	FahadMirza
2014-08-01	1.2	PS2 Removed	Nur Mohammad
2014-11-14	1.3	Single Layer PCB	Nur Mohammad

1. Introduction



Many of us made our first steps in embedded world with AVR Trainer Kit™. Today it has thousands of users: students, hobbyists, enthusiasts and professionals.

We asked ourselves what we can do to make such a great board even greater. And we made some brilliant changes. We focused all of our creativity and knowledge into making a revolutionary new design, unlike any previous version of the board. We now present you with the new version, it's portable, pluggable. We hope that you will be thrilled with your new board, just as we are.

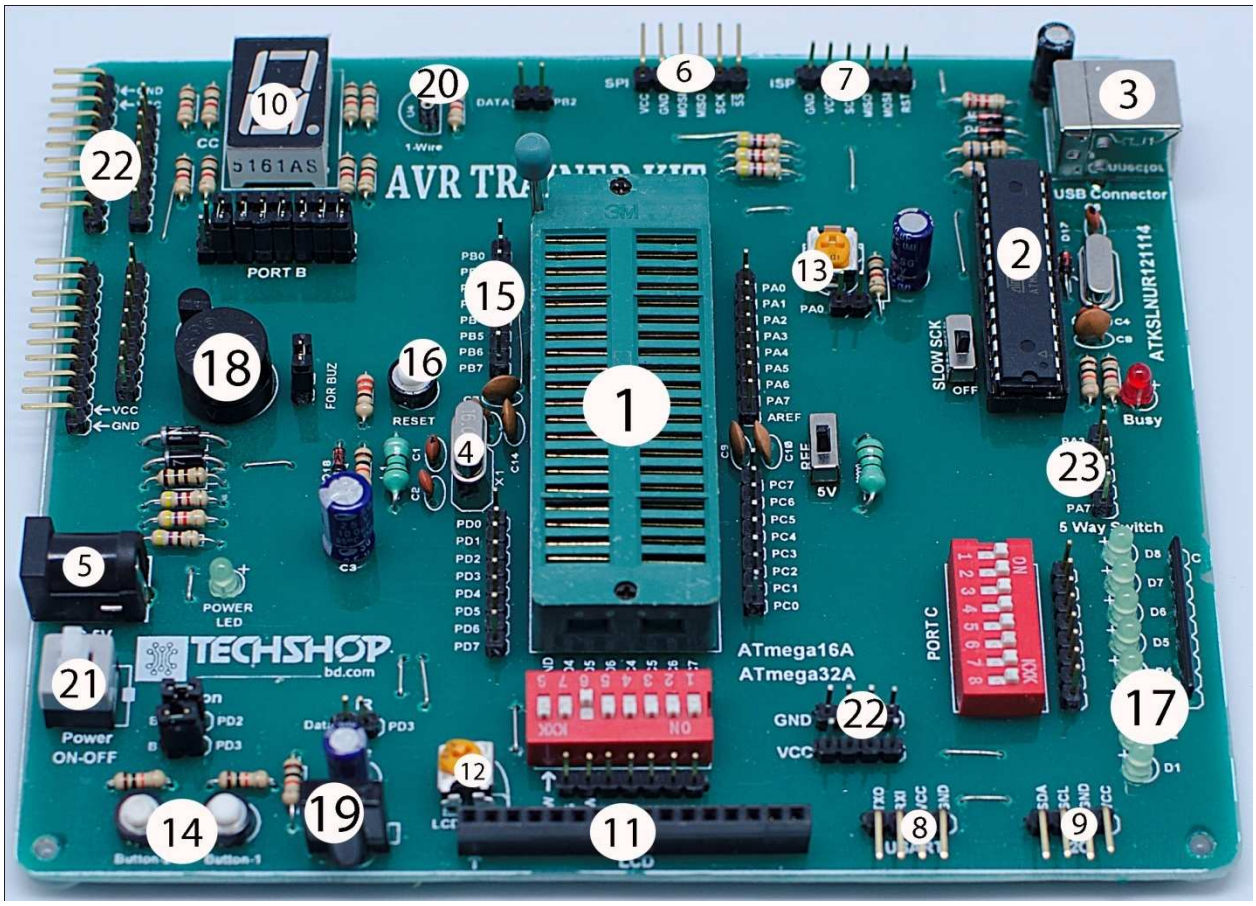
AVR Trainer Kit is anexcellent developmentkit for novices for programming and experimenting with AVR microcontrollers from Atmel. It includes an On-board programmer which provides an interface between PC and microcontroller. All you have to do is to write the program in any AVR compiler, generate the hex file, connect the board to your computer's USB port, place the microcontroller on the ZIF socket and load the program to the microcontroller. A number of buttons, jumpers and connectors are there to help the microcontroller to be interfaced with the peripherals.

1.1 Kit contents

This kit contents:

- AVR Trainer Kit Board.
- Standard USB cable (A to B).
- Some jumpers and Connectors.
- User Guide DVD

1.2 What's On Board



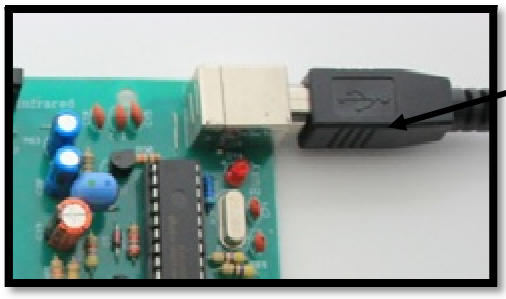
1)ZIF Socket for ATmega16A/ATmega32A, with accessible I/O, grouped by PORT.
2)On board Programmer.
3)USB connector, connection with PC.
4)Crystal Oscillator (Default 16MHz). It's replaceable.
5)External Supply Socket (5V).
6)SPI Pinout.
7)ISP Pinout.
8)USART Pinout.
9)I2C Pinout.
10)Seven Segment Display Interfacing.
11)LCD Interfacing Connector.
12)LCD Contrast.
13)ADC Interfacing.
14)Push Button Interfacing.
15)I/O Expander.
16) RESET Circuitry.
17) 8 LEDs Interfacing.
18) Buzzer Interfacing.
19) Infrared Interfacing.

20)1-WireCommunication Interfacing.
21) Power Switch.
22) Vcc-GND power pin.
23) Joystick Pinout

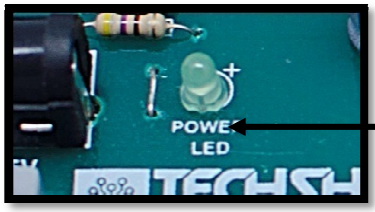
2. How do I Start?

Step 1:

Use the USB cable to connect the AVR Trainer Kit development board to your PC. One end of the USB cable provided with a connector of the USB B type should be connected to the development board as shown in figure below; whereas, the other end of the cable (A type) should be connected to the PC. As soon as the board is connected to the PC, the green LED marked as 'power' will glow.



Connecting AVR Trainer Kit to a computer's USB Port



Power Indicator LED

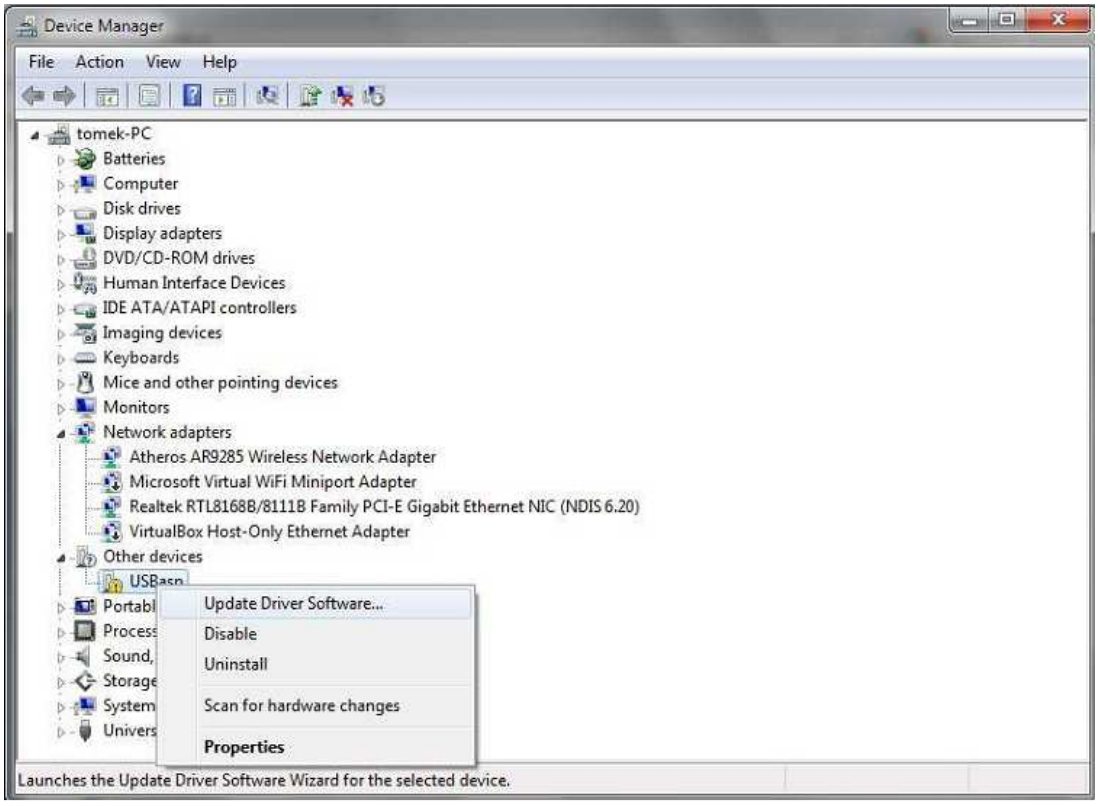
Step 2:

Now you'll need to install the driver of the programmer. On Linux and MacOS X no kernel driver is needed. Windows requires a driver. In order to complete the installation, you need to follow several steps. This procedure will only focus on Window 7.

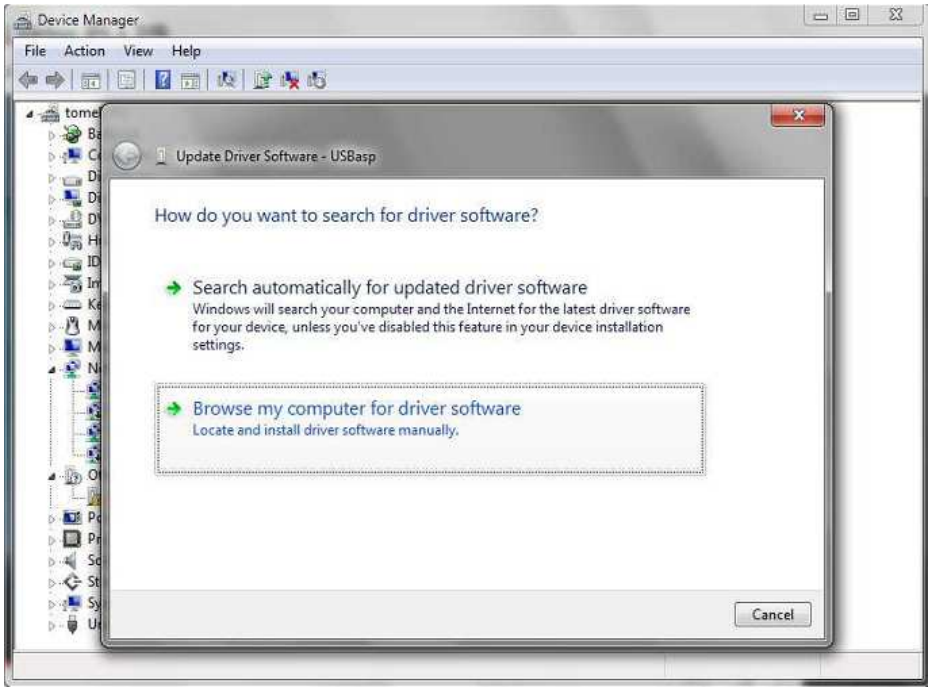
2.1 Driver Installation

1. Unzip the 'Driver' file (included in the DVD and in our site) and connect AVR Trainer Kit to the USB port of your PC.

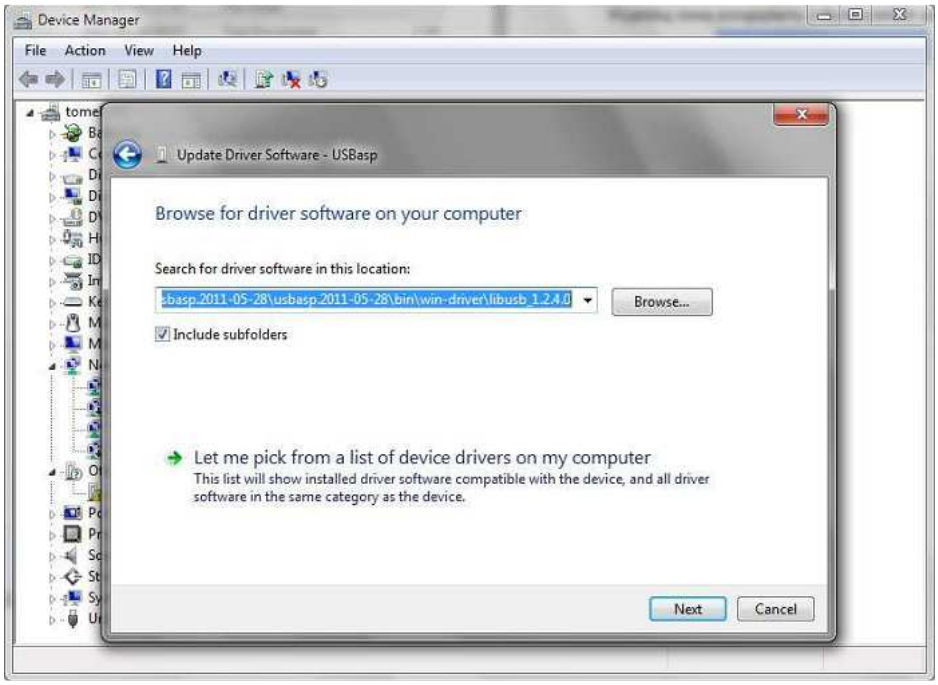
2. Open Device Manager, find the entry for the USBasp and it should be displayed with a yellow alert icon on it. Then right click on the device and select “Update Driver Software”.



3. After you left click the “Update Driver Software”, it will come out with “How do you want to search for driver software?” Then choose the second one which is “Browse my computer for driver software” and click into it.

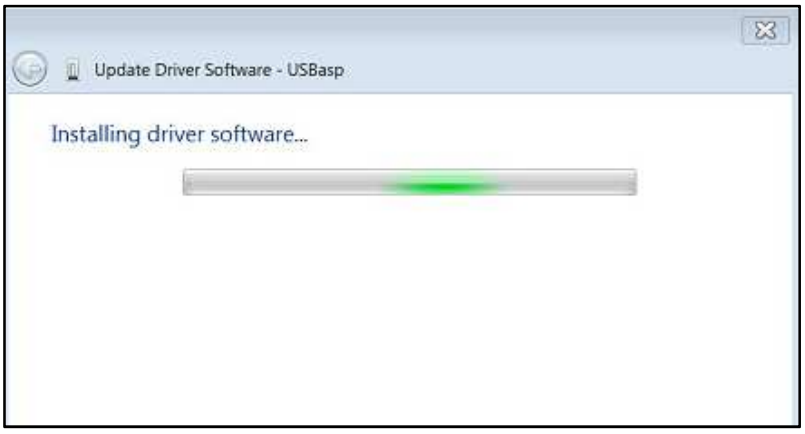


4. After that, you will see the screen which will prompt out “Browse for driver software on your computer”. Select the driver folder (…\Driver\libusb_1.2.4.0),then click “Next”.

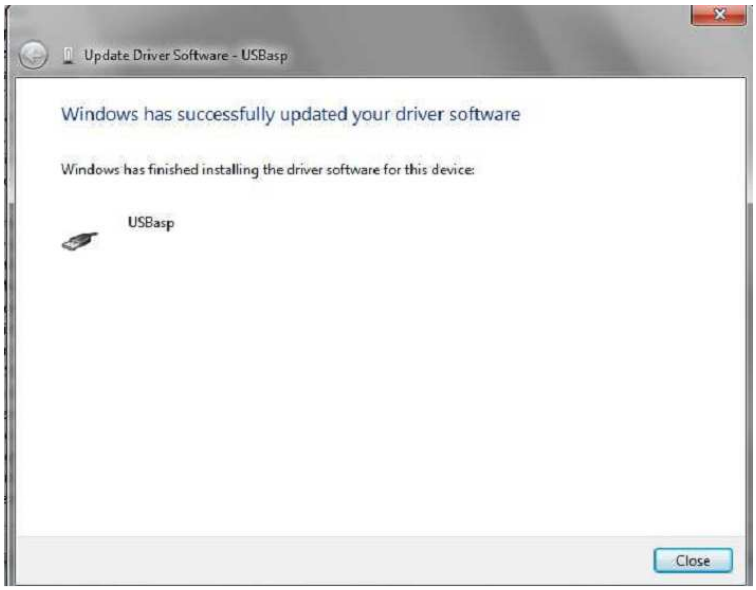


5. Next, the windows will prompt out a “Windows Security” with a red warning dialog. Do not worry about it, and just click “Install this driver software anyway” and the driver will install.

6. After click it, the next step is to wait a few seconds to let your computer to process the installation of driver software.

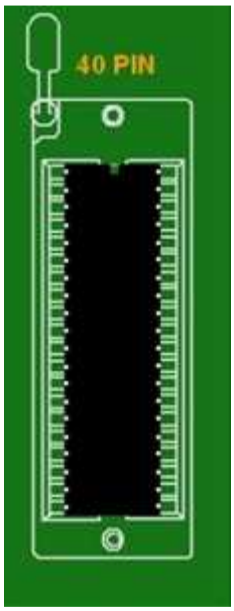


7. Now, you can use the programmer to do the programming for the microcontroller.



Step 3:

AVR Trainer Kit™ supports two most popular mcu, ATmega16A and ATmega32A. Now place your desired mcu into theZIF socket, as shown below. You are good to go now.



NOTE: You can load other microcontrollers through ISP pinout.

2.2 Programming Software

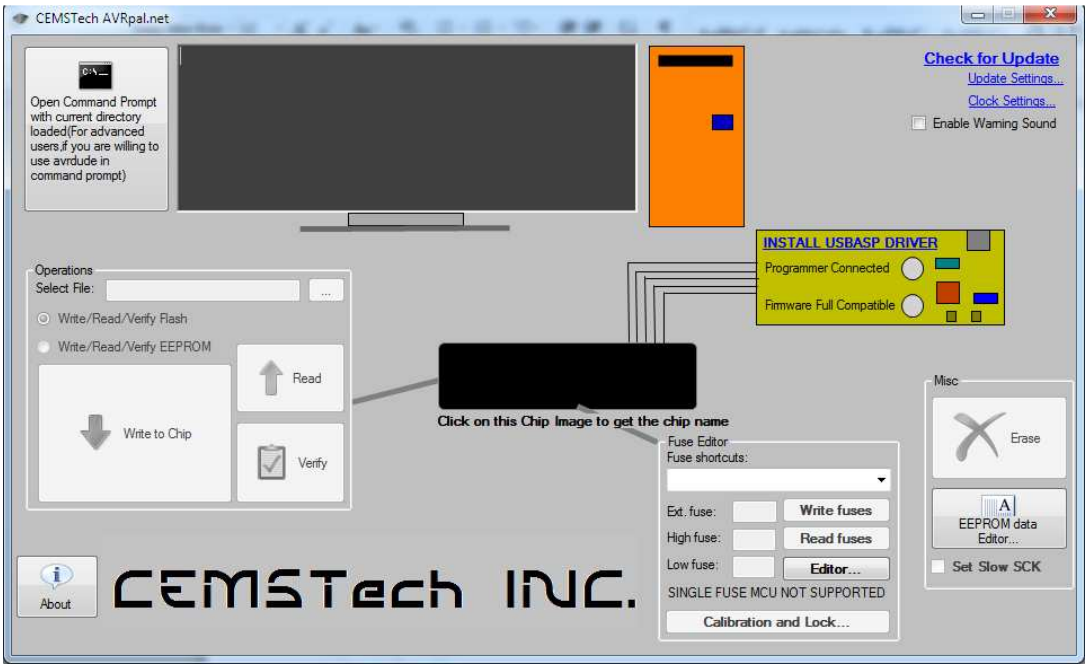
There are varieties of software which can work for the programmer. These are:

- AVRpal – Version 3.1 or later. A great GUI of avrdude. We normally use this software.
- **eXtreme Burner** – An easy to use GUI application. Our second favorite software.
- **Khazama AVR Programmer** – An AVRdude GUI for MS Windows.
- **BASCOM-AVR** – Version 1.11.9.6 or later.
- **ProgISP 1.72** – Supports various hardware, including USBasp, STK200 etc.

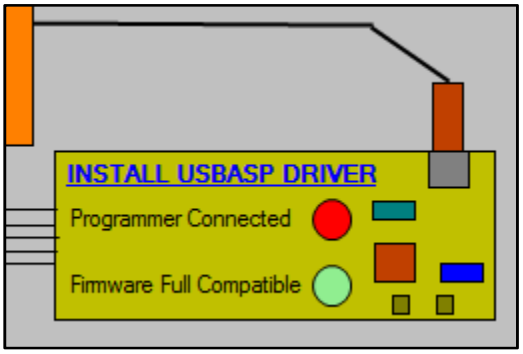
For the list of the software above, we have no responsibility to teach users how to use;users must study themselves in order to use it.

2.2.1 AVRpal

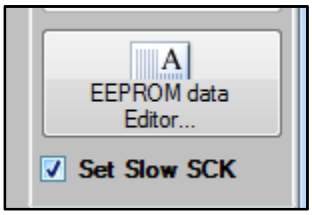
1. At the time I am writing this manual, I have V3.1 in my hand.



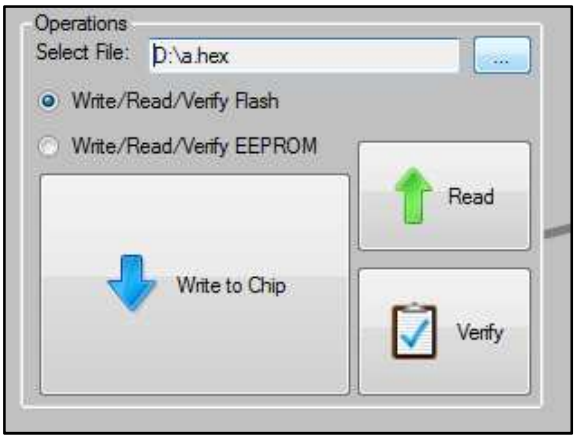
2. If AVRpal.net detect you programmer, then “Programmer Connected” and “Firmware Full Compatible” LED will lit.



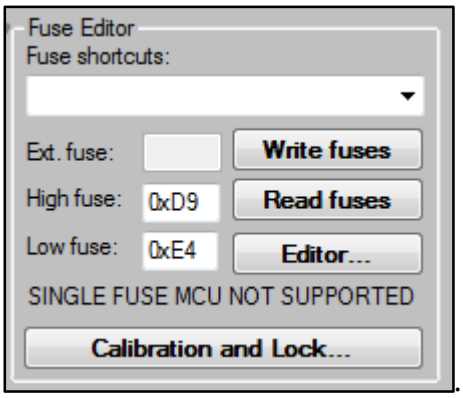
3. Now click on the “Black Box” (which actually represents IC -_-), and software will detect which mcu ispresent in the ZIF socket. Make sure you checked “Set Slow SCK”. A detail on “Slow SCK” isdiscussed in “Fuse Bytes” section.



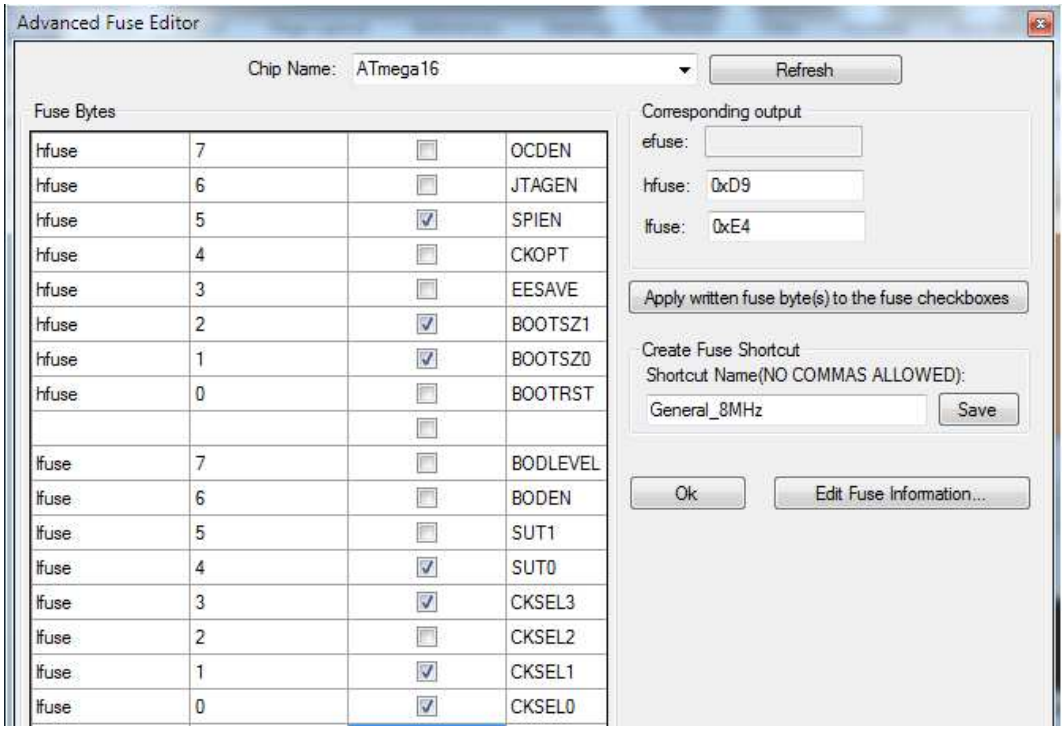
4. So, it detects my mcu (ATmega16). To load “Hex” file browse it from “Select File”.



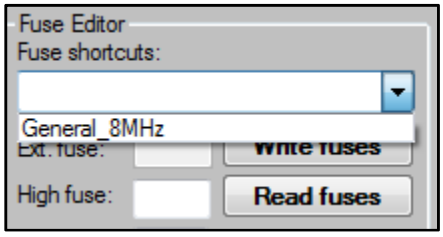
5. You can also read/write fuses. Take great care when you change the fuse bits. If you do not know anything about fuse bits, please read “Fuse Bytes” section first



6. By using fuse “Editor” you can save fuse byte for specific mcu.For instance, I frequently use L=0xE4 / H=0xD9 fuse bytes for ATmega16A. So I named it “General_8MHz” and then press “Save”.



7. Now if you click on the “Fuse Shortcuts” drop down menu, you’ll see “General_8MHz”. If you select it, corresponding fuse bytes will load. If there is another mcu in ZIF socket other than ATmega16A, it won’t be available, which will preventfrom writing wrong fuse bytes!!



2.2.2 eXtreme Burner

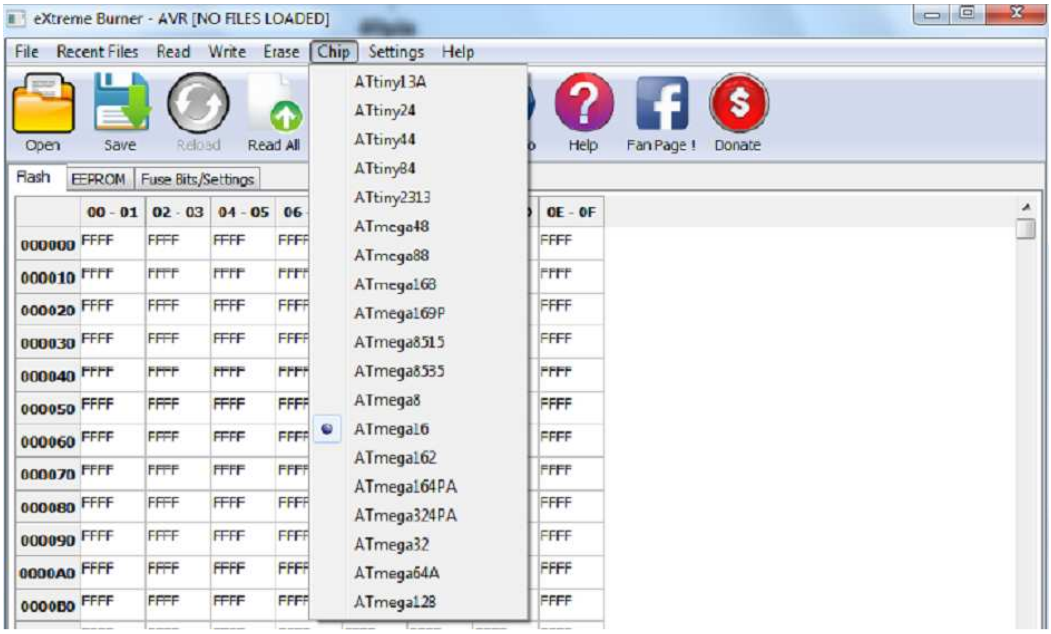
No need to install Driver for USBasp, its included in the Extreme Burner Installation. Programming Steps are:

1) Plug the AVR Programmer into your Windows PC.

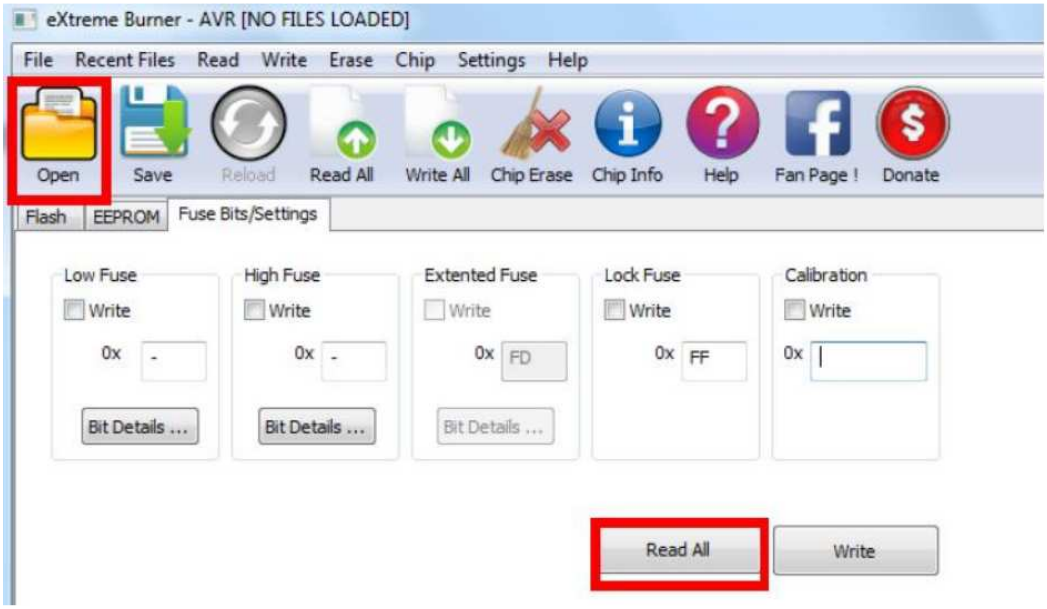
2) Install the “Extreme Burner V1.2”. Driver will be installed with the installation process.

3) Open Extreme Burner.

4) Under drop-down menu choose your desired chip.



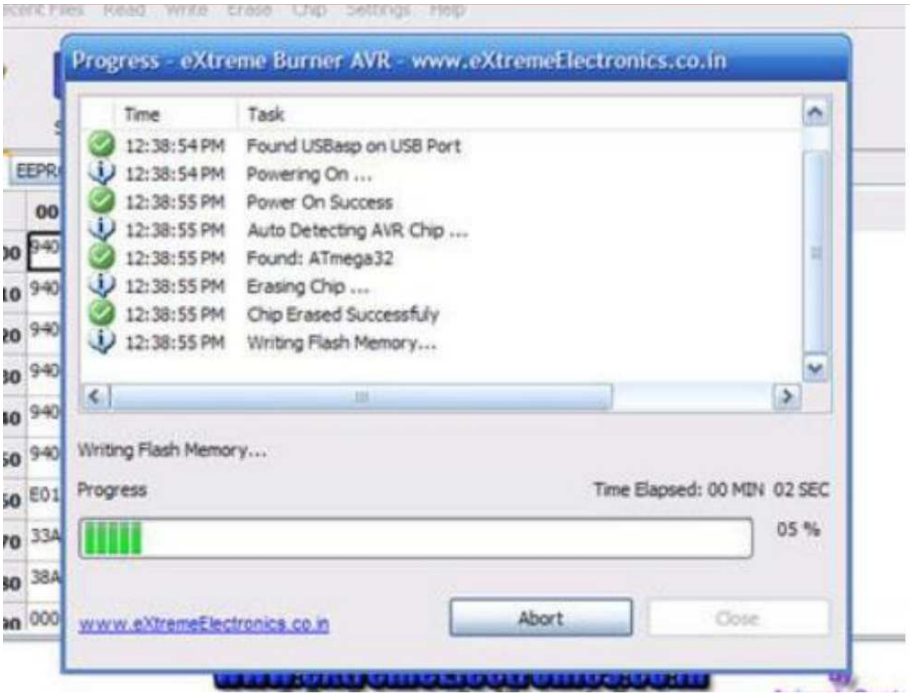
5) Click on the Fuse Bits/Settings tab and choose “Read All” for read your chip’s Fuse Bytes.



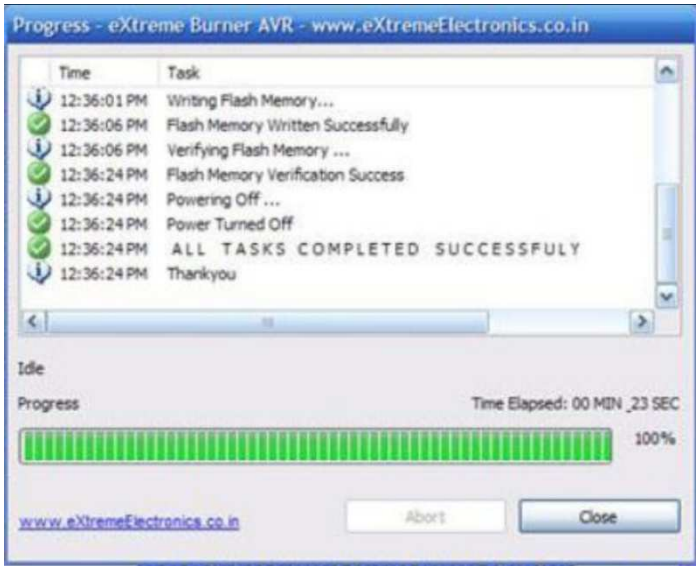
6) If you want to write Fuse Bytes into your chip, write down it on the blank boxes, check the “write”, and then press write. For more on Fuse Bytes check the “Fuse Bytes” chapter. Also you can visit: http://www.engbedded.com/fusecalc/

7) For loading Hex file press “Open” and browse the hex file.

8) From the drop-down menu choose Write > Flash



9) If programming is successful you will see just like below:



To know more about this software, you can visit the official site (eXtreme Electronics): http://extremeelectronics.co.in/avr-tutorials/gui-software-for-usbasp-based-usb-avr-programmers/

Buckle up, because now we are going to introduce you with the features of the boards.

3. Board Features

3.1 AVR Programmer

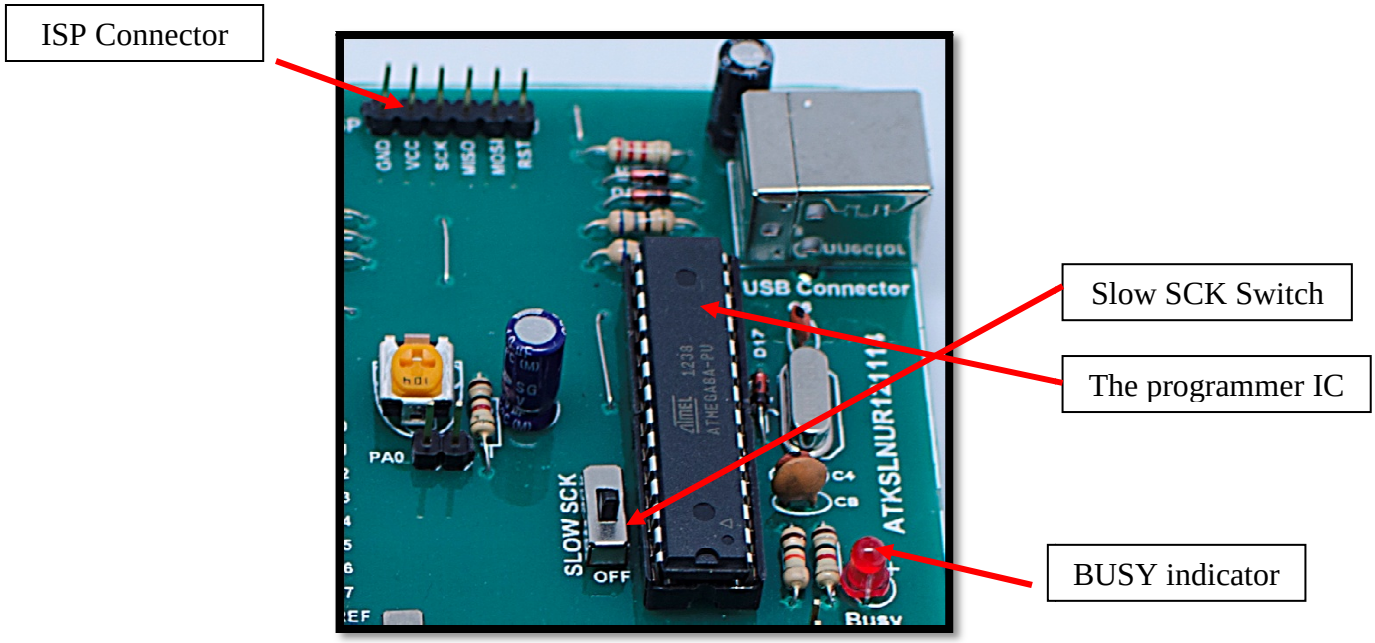


Fig: Programmer unit of AVR Trainer Kit

The on board programmer is shown in the above figure. The ISP connectors are available for burning mcu outside of the board. SLW_SCK is easily accessible by a slide switch (required for eXtreme Burner). To know more about Slow SCK go through 'Fuse Bytes' chapter.

3.2 Oscillator

There is a non-soldered, replaceable crystal oscillator on the board used as an external clock source. Its maximum value depends on the maximum frequency of the microcontroller. The board comes with the 16MHz crystal.

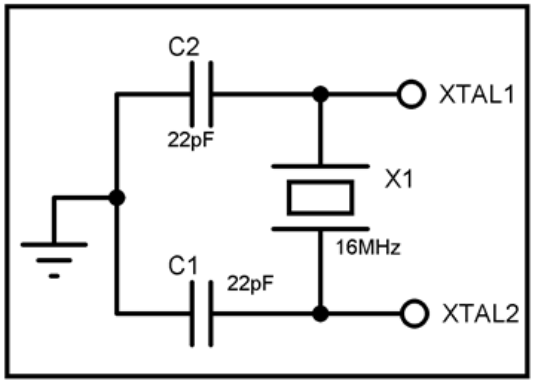
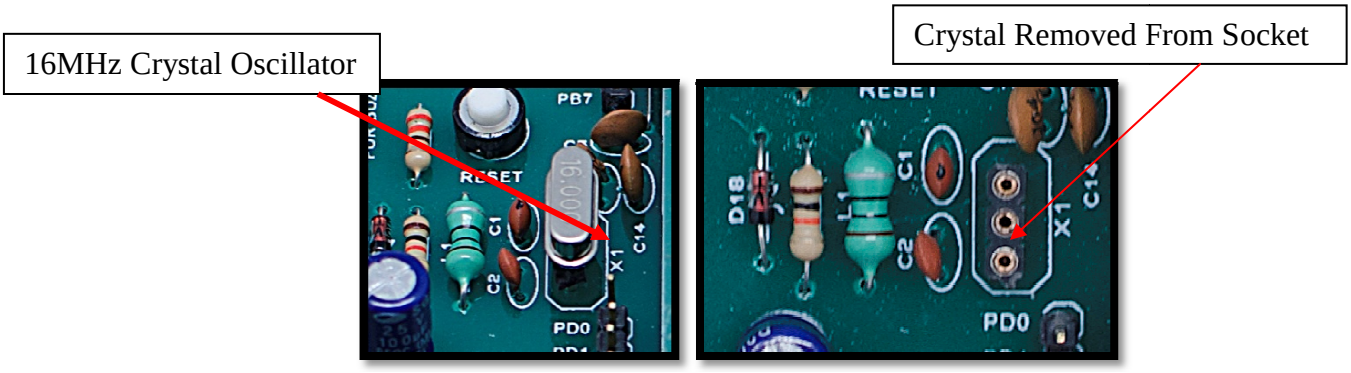


Fig: Schematic of crystal oscillator of AVR Trainer Kit



3.3 Power Supply

The AVR Trainer Kit development board can use two power supply sources:

1. +5V power supply through the USB programming cable.
2. External power supply connected to a DC socket mounted on the development board.

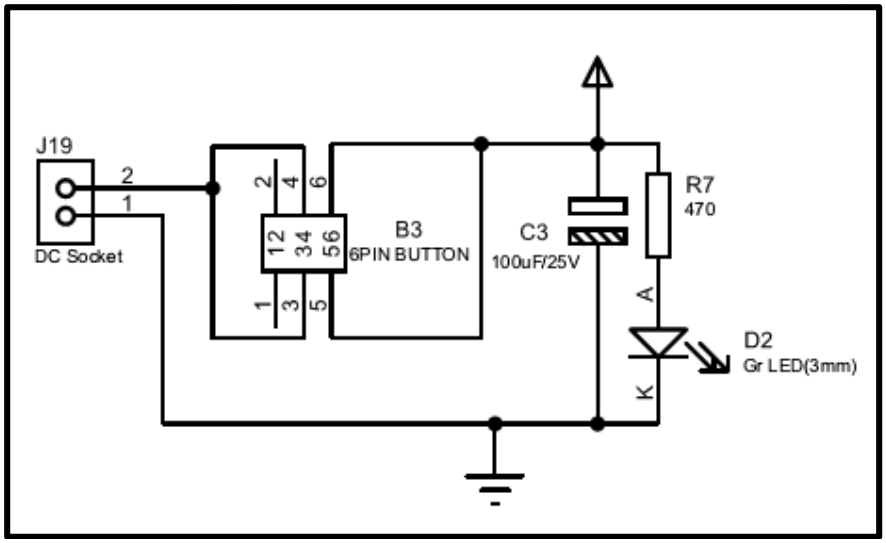
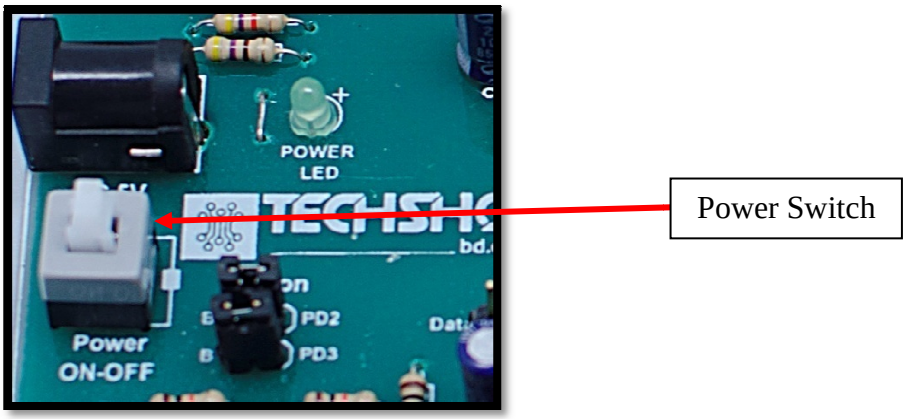


Fig: Schematic of external supply unit

NOTE: The USB supply of the board come directly from USB port of computer; it is advised not to use this power source to power application circuit or device. Wrong connection such as wrong polarity, wrong voltage, shorted might permanently damage your computer.

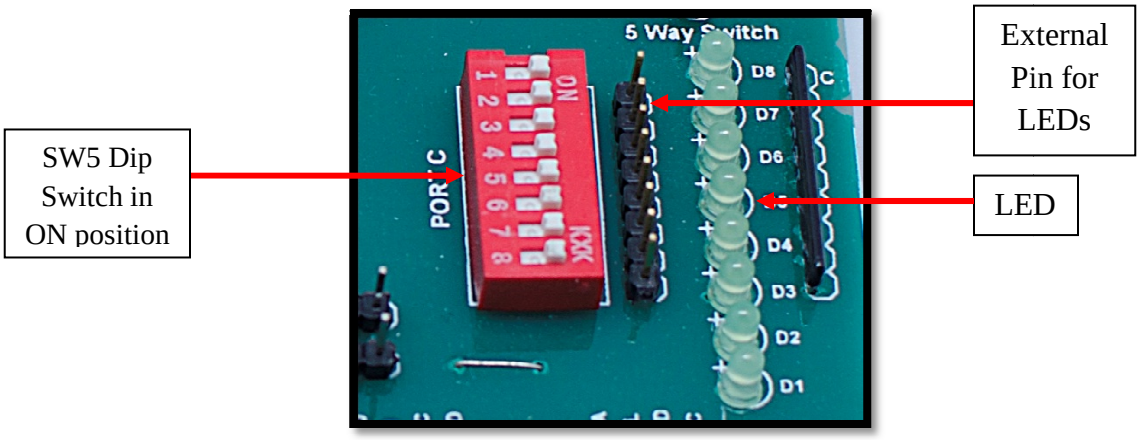
NOTE: Use 4.5-5.3V as an external source. Voltage more than that might burn your components. Take extra caution when using external source. Check out techshopbd.com for appropriate 5v adapter.



3.4 LED Interfacing

LED (Light-Emitting Diode) is a highly efficient electronic light source. A common LED diode voltage is approximately 2.5V, while the current varies from 0.2mA to 20mA, depending on the type of the LED. The AVR Trainer Kit™ board uses low-current LEDs with typical current consumption of 0.2mA or 0.3mA. Board contains 8 LEDs which can be used for visual indication of the logic state on PORT pins. An active LED indicates that a logic high (1) is

present on the pin. In order to enable PORTC LEDs, it is necessary to enable the corresponding DIP switches on SW5 (figure below).



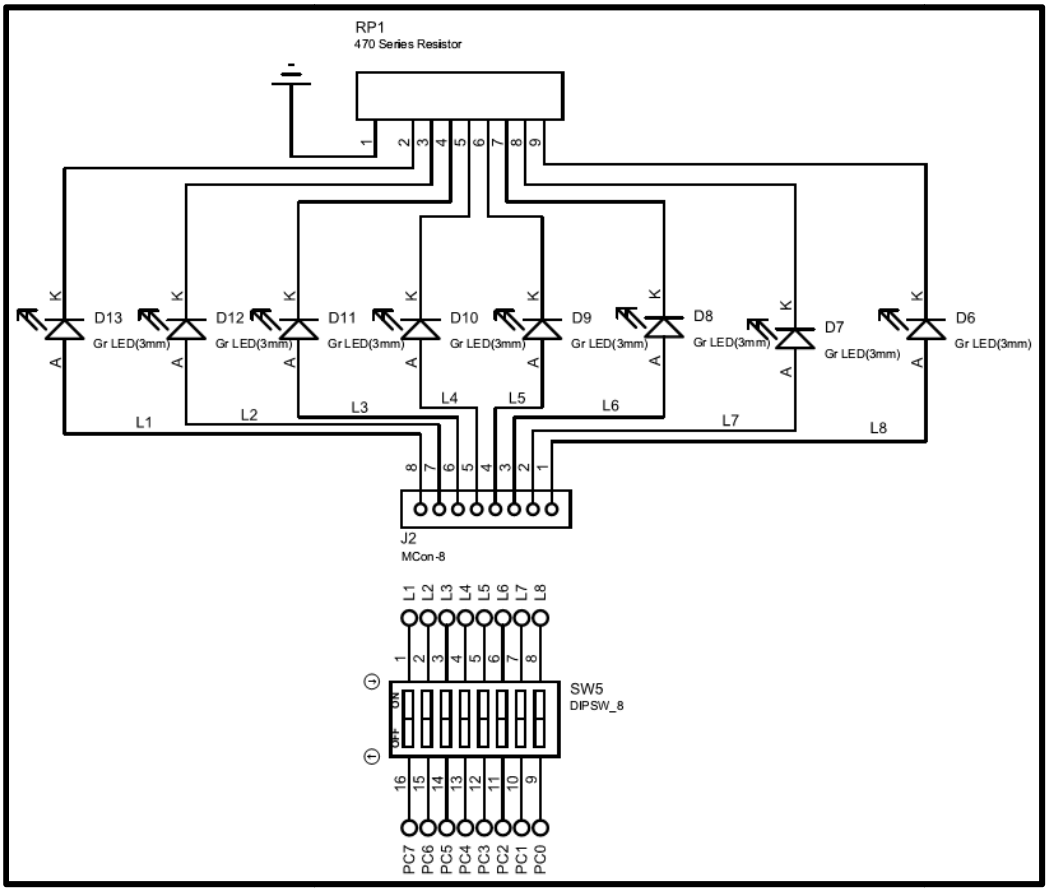


Fig: Schematic of LED unit of AVR Trainer Kit

There is a male connector (J2) present in the board in case you want to check logic state for other PORTs instead of PORTC. Like the figure below, just connect the connector(supplied with the

kit) with your desired PORT and J2. Also don't forget to slide the SW5 switches into OFF position (**unlike** figure below) or there might be logic contention between PORTs.

Also don't forget to disable the LCD's Dip switch, because they too use some of PORTC's pin.

Now load "Led_blinking.hex" (…`\AVR Trainer Kit\Example\LED blinking`) and LED will start blinking.

NOTE: All the example code is written in "MikroC Pro for AVR" for ATmega32A. Also simulation file in "Proteus" is provided too.

3.5 Push Button Interfacing

If you want to play with the digital input of microcontroller, we have two push buttons tied to ground. They are connected with PORTD2 and PORTD3 i.e. with two external interrupts. Buttons are externally pulled up to Vcc.

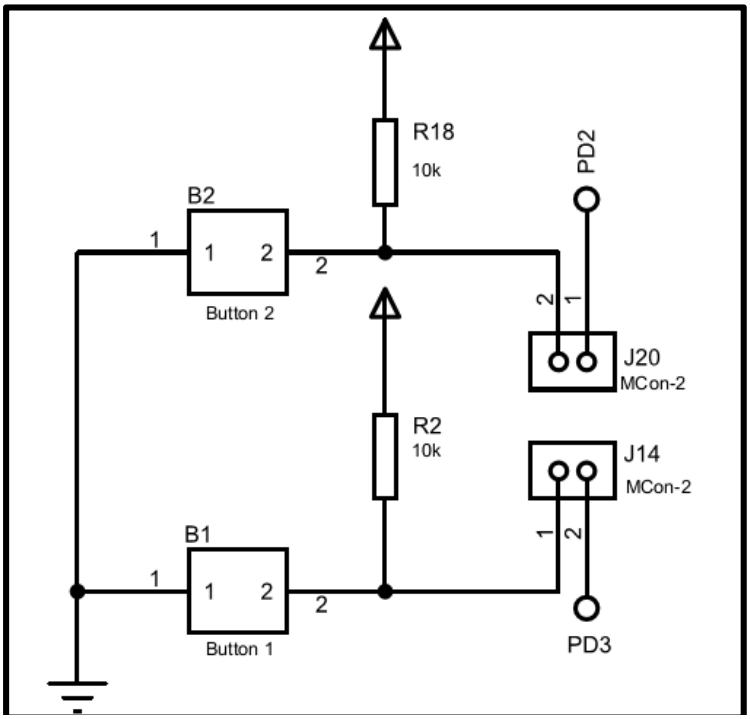
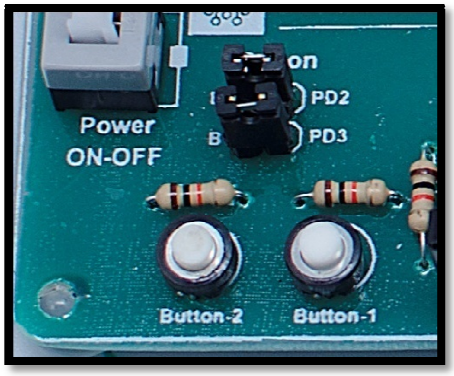
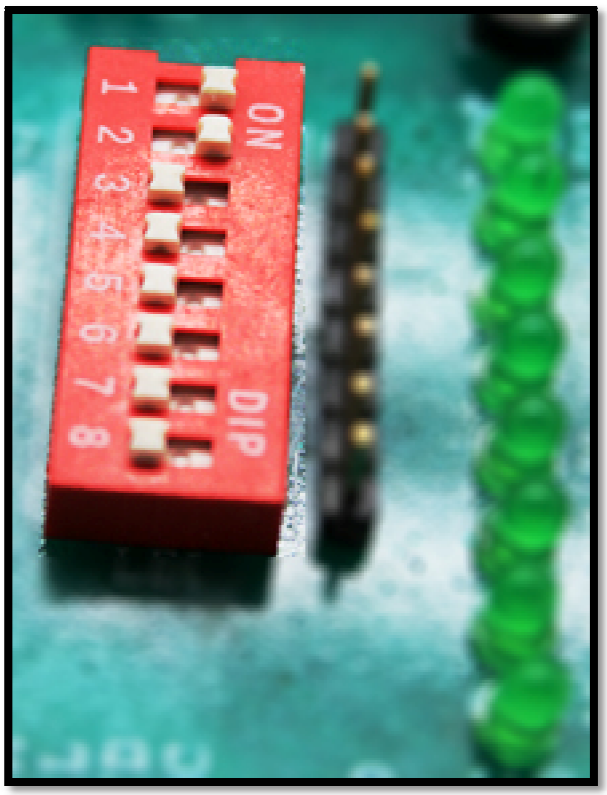


Fig:Schematic of push button in AVR Trainer Kit

Place the jumper like the figure below:



Also don't forget to Enable PORTC0 and PORTC1 LED switch (SW5).



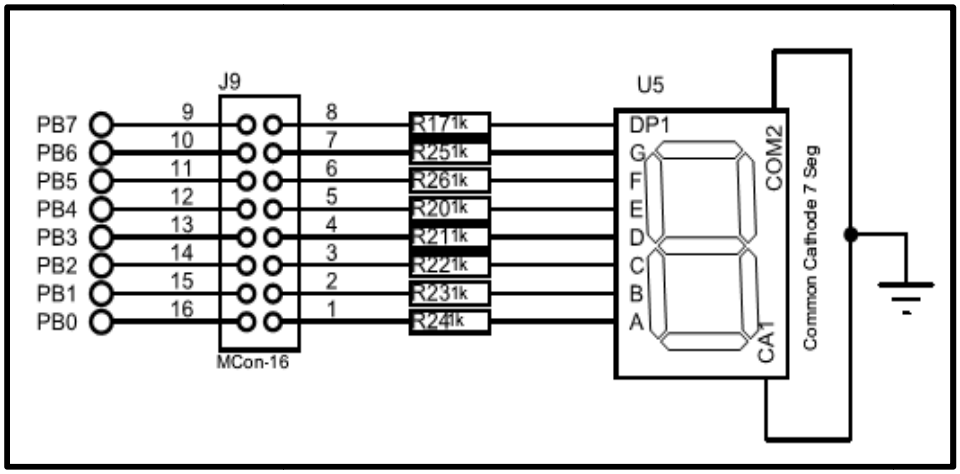
Load the "Push_Button.hex" (…`\AVR Trainer Kit\Example\Push Button`). Now if you press the buttons, corresponding LED will lit.

NOTE: All the example code is written in "MikroC Pro for AVR" for ATmega32A. Also simulation file in "Proteus" is provided too.

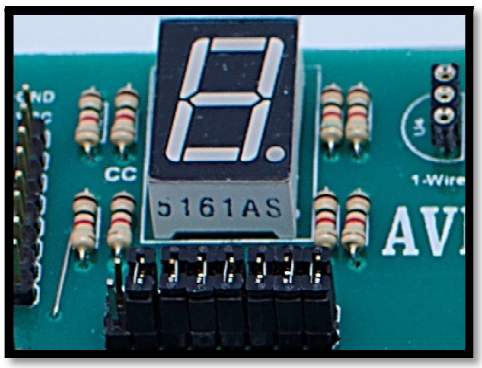
3.6 Seven Segment Display Interfacing

One seven segment digit consists of 7+1 LEDs which are arranged in a specific formation which can be used to represent digits from 0 to 9 and even some letters. One additional LED is used for

marking the decimal dot, in case you want to write a decimal point in the desired segment. AVR Trainer Kit™ contains one (Common Cathode) of these digits. Eight data lines are connected to PORTB.



To enable place jumpers like figure below:



Make sure PORTB isn't connected with other peripheral, like buzzer/SPI/ISP/1-wire etc. Load "seven_segment_display.hex" (…`\AVR Trainer Kit\Example\Seven Segment Display`) and a counter from 0-9 will start immediately.

3.7 ADC Interfacing

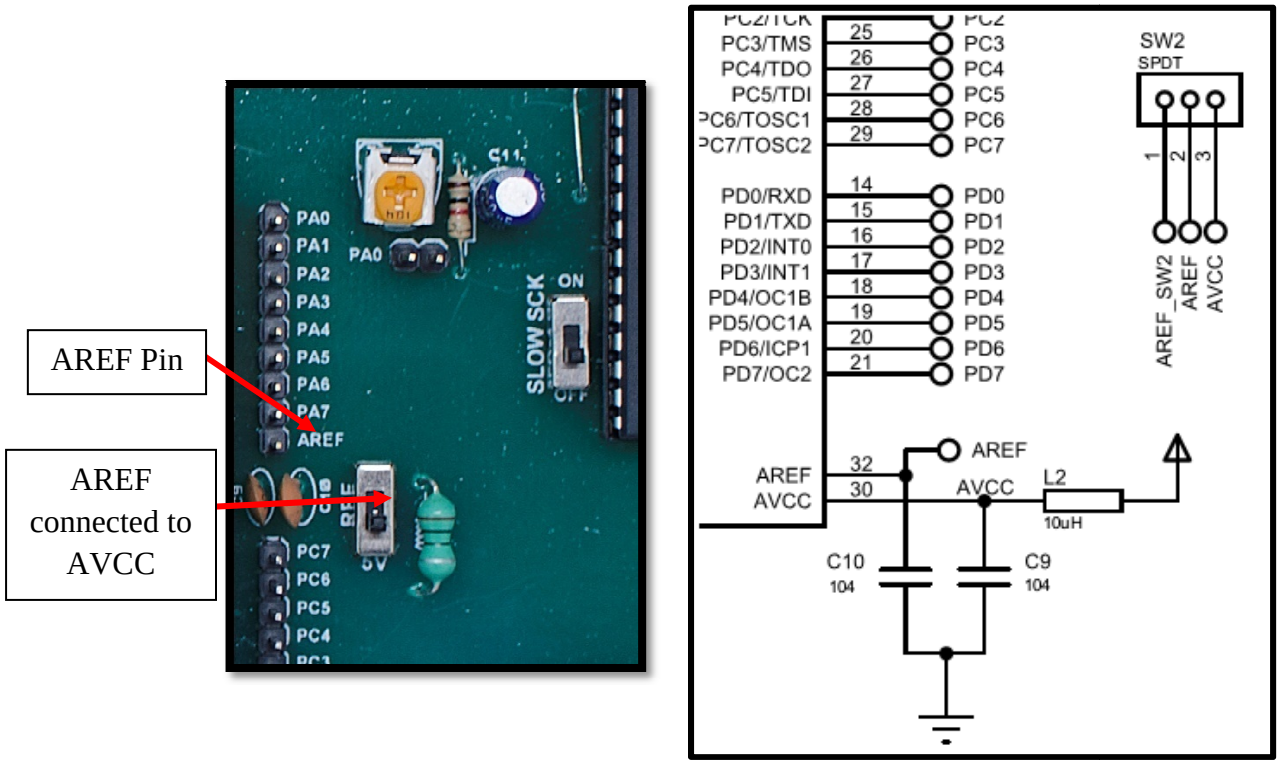
Digital signals have two discrete states, which are decoded as high and low, and interpreted as logic 1 and logic 0. Analog signals, on the other hand, are continuous, and can have any value within defined range. A/D converters are specialized circuits which can convert analog signals (voltages) into a digital representation, usually in form of an integer number. The value of this number is lineary dependent on the input voltage value.

Most microcontrollers now days internally have A/D converters connected to one or more input pins. Some of the most important parameters of A/D converters are conversion time and resolution. Conversion time determines how fast an analog voltage can be represented in form of a digital number. This is an important parameter if you need fast data acquisition.

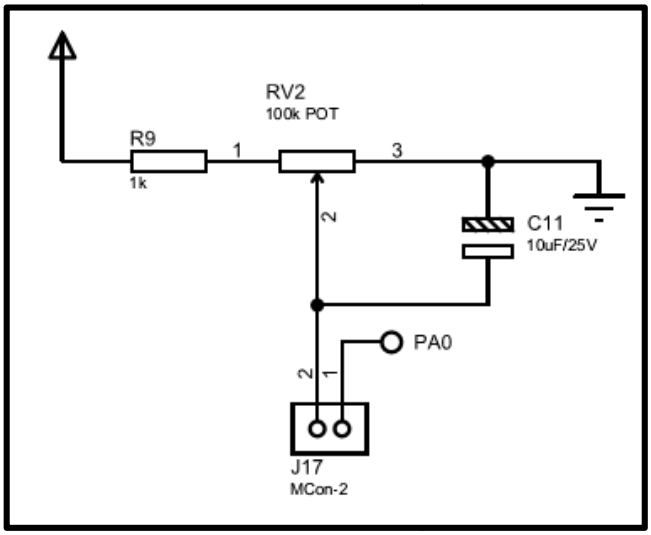
The other parameter is resolution. Resolution represents the number of discrete steps that supported voltage range can be divided into. It determines the sensitivity of the A/D converter. Resolution is represented in maximum number of bits that resulting number occupies. Most AVR microcontrollers have 10-bit resolution, meaning that maximum value of conversion can be represented with 10 bits, which converted to integer is 2^{10}=1024. This means that supported voltage range, for example from 0-5V, can be divided into 1024 discrete steps of about 4.88mV.

AVR Trainer Kit™ v2 provides an interface in form of potentiometers for simulating analog input voltages that can be routed to any of the 8 supported analog input pins (default channel 0).

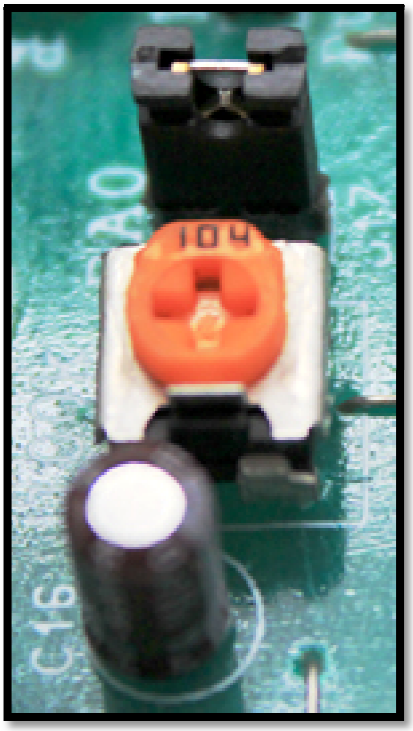
The right figure shows the connection of AVCC and AREF. AVCC is tied to Vcc. And AREF is tied with a slide switch (SW2). You can either choose to connect AREF with AVCC or any external voltage you want. If you want to connect external voltage, slide up the switch and connect the external reference voltage at AREF pin.

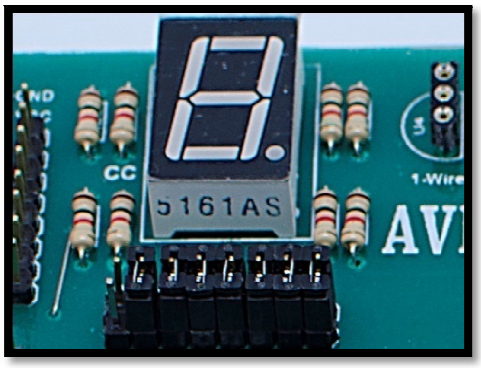


A 1K resistor is connected in series with 100k POT to refrain the direct connection of Vcc with ADC pin. The capacitor C11 will stabilize the voltage across ADC pin. By default and in our example code we connect the POT with Channel 0 (i.e. PA0) but you can always route to any other ADC pin.



Connect the jumper J17 as shown in figure. Connect AREF with AVCC. Our sample code will use 7-segment display to show the ADC value. So connect those jumpers too. Load "ADC_Interfacing.hex" (…\AVR Trainer Kit\Example\ADC). Now rotate the POT and see the voltage change in 7-segment display.





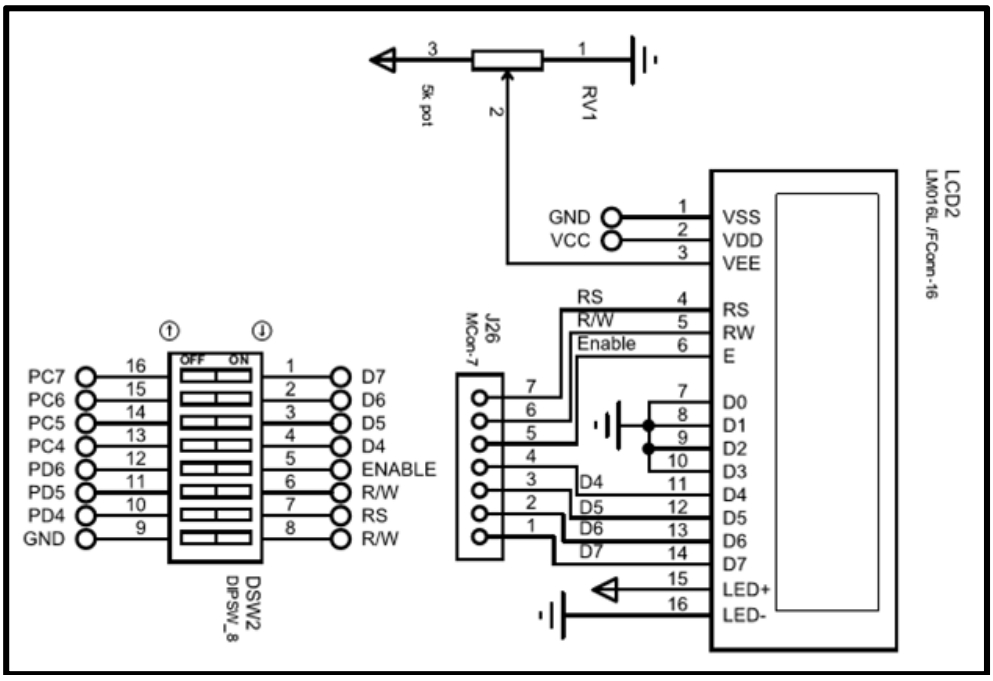
3.8 LCD Interfacing

Liquid Crystal Displays or LCDs are cheap and popular way of representing information to the end user of some electronic device. Character LCDs can be used to represent standard and custom characters in the predefined number of fields. AVR Trainer Kit™ v2 provides the connector and the necessary interface for supporting 2x16 (or 4x20) character LCDs in 4-bit mode. This type of display has two rows consisted of 16 character fields. Each field is a 7x5 pixel matrix.

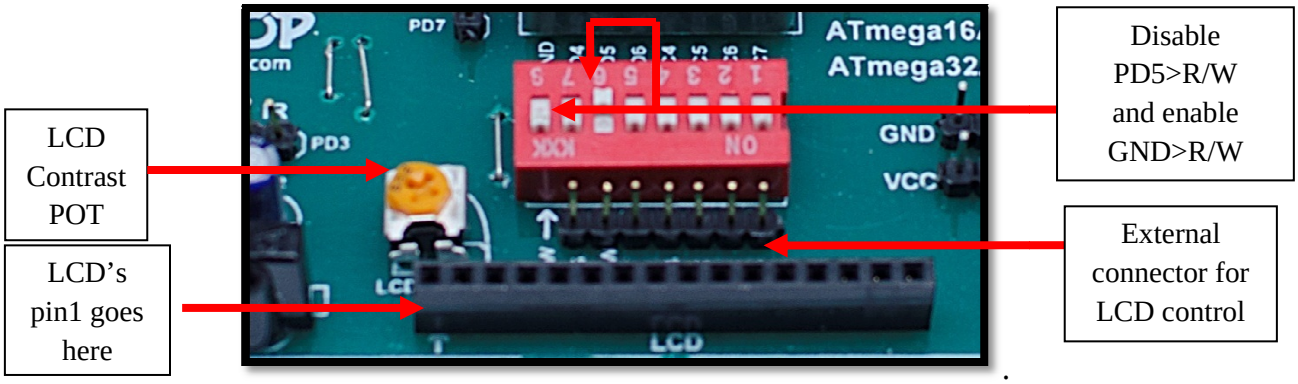
NOTE: Make sure to turn off the power supply before placing LCD onto the board. Otherwise your display can be permanently damaged.

Connector pinout explained

GND & VCC - Display power supply lines.
V_{EE} - LCD contrast.
RS- Register Select Signal line.
R/W- Determines whether display is in Read or Write mode. You can permanently connect it with GND, leaving the display in Write mode all the time.
E - Display Enable line
D0–D3 - Display is supported in 4-bit data mode, so these pins connected to GND.
D4–D7 - Upper half of the data byte
LED+- Backlight LED anode, connected with V_{cc}.
LED-- Backlight LED cathode, connected with GND.



LCD Contrast can be adjusted with RV1. Through Dip Switch (DSW2) you can easily connect LCD with your MCU. The controlling pins go with PORTD4-6. If you want to permanently connect R/W with GND then leave (OFF) PD5>R/W switch and enable GND>R/W (shown below).

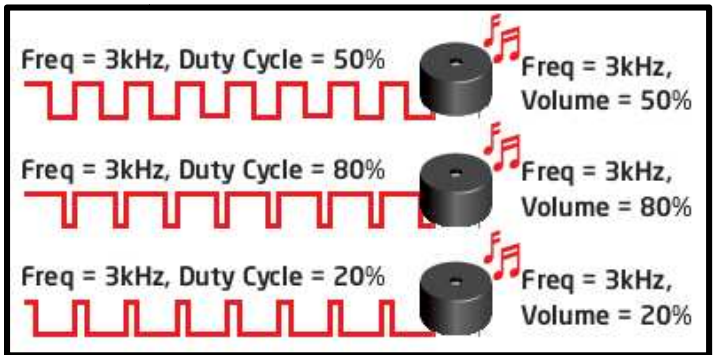


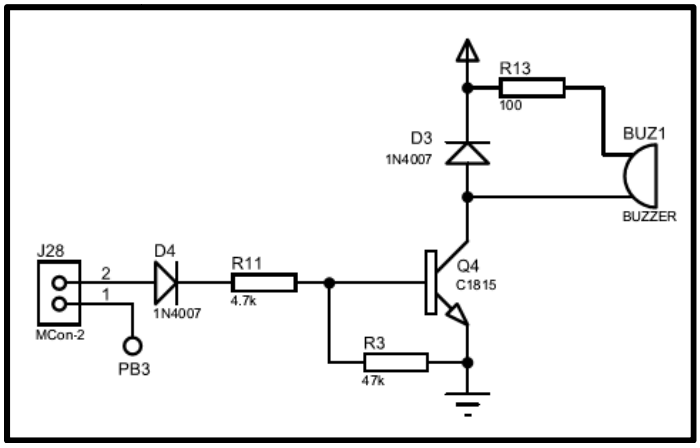
Also a male connector (J26) is available in case you want to connect other mcu pin to LCD. Place a LCD (make sure all the powers are off). Connect USB cable and load "LCD_Interfacing.hex" (…`\AVR Trainer Kit\Example\LCD Interfacing`). Power the board (better use a 5v adapter) and you'll see moving text on the LCD.

3.9 Buzzer Interfacing

Piezoelectricity is the charge which accumulates in certain solid materials in response to mechanical pressure, but also providing the charge to the piezoelectric material causes it to physically deform. One of the most widely used applications of piezoelectricity is the production of sound generators, called piezo buzzers. **Piezo buzzer** is an electric component that comes in different shapes and sizes, which can be used to create sound waves when provided with analog electrical signal.

AVR Trainer Kit™ v2 comes with piezo buzzer which can be connected to PB3 microcontroller pin, which is determined by the position of J20 jumper. Buzzer is driven by transistor Q4. Microcontrollers can create sound by generating a PWM (Pulse Width Modulated) signal – a square wave signal, which is nothing more than a sequence of logic zeros and ones. Frequency of the square signal determines the pitch of the generated sound, and duty cycle of the signal can be used to increase or decrease the volume in the range from 0% to 100% of the duty cycle. You can generate PWM signal using AVR's Timer module, which is usually available in most AVR microcontrollers, or by writing custom software which emulates the desired signal waveform (like in the example we'll show later).





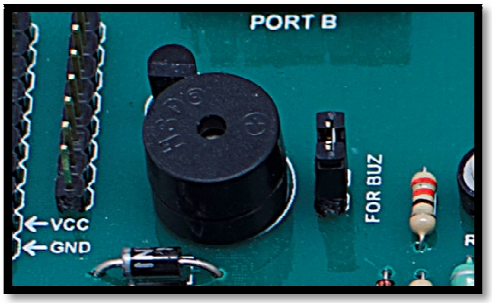


Fig:Buzzer interfacing with AVR Trainer Kit

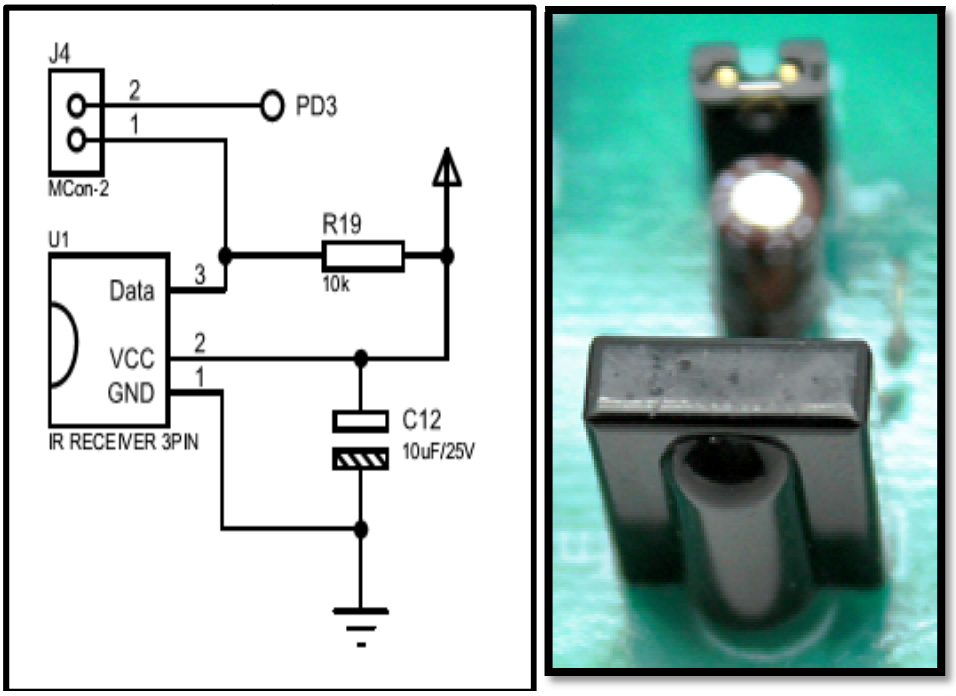
Place the jumper (J20) as shown in figure. Buzzer input will connect with PORTB3. So make sure this pin isn't connecting with other (like 7 segment etc.). Our Buzzer example also needs LCD, so enable the dip switch (DSW2) of the LCD as mentioned at section "3.8 LCD Interfacing" and a push button on PD2, so place the push button (B2) jumper too. Load "Buzzer.hex" (…`\AVR Trainer Kit\Example\Buzzer`).Now power the board (a 5v adapter will be preferable). If you press the B2, music will start to play with the lyrics on the LCD.

3.10 Joystick Interfacing

Joysticks are one of the interesting things to interface. Our joystick will give you 5 positions; Up, Down, Left, Right and Center. The common is connected to ground i.e. they will be active low. Jumper (J3) will connect this joystick with PORTA3-7. No pull ups are present, so you have to use PORTA's internal pull-up option.

3.11 Infrared Interfacing

Want to control your things wirelessly? One of the usual formats is infrared. Use your existing DVD player or TV's remote. To help you with the infrared interfacing we provide an infrared sensor on the board with necessary passive components.



3.12 1-Wire communication:

AVR Trainer Kit supports one wire communication through the connector U4.To do so, you have to set a jumper in J44 position.

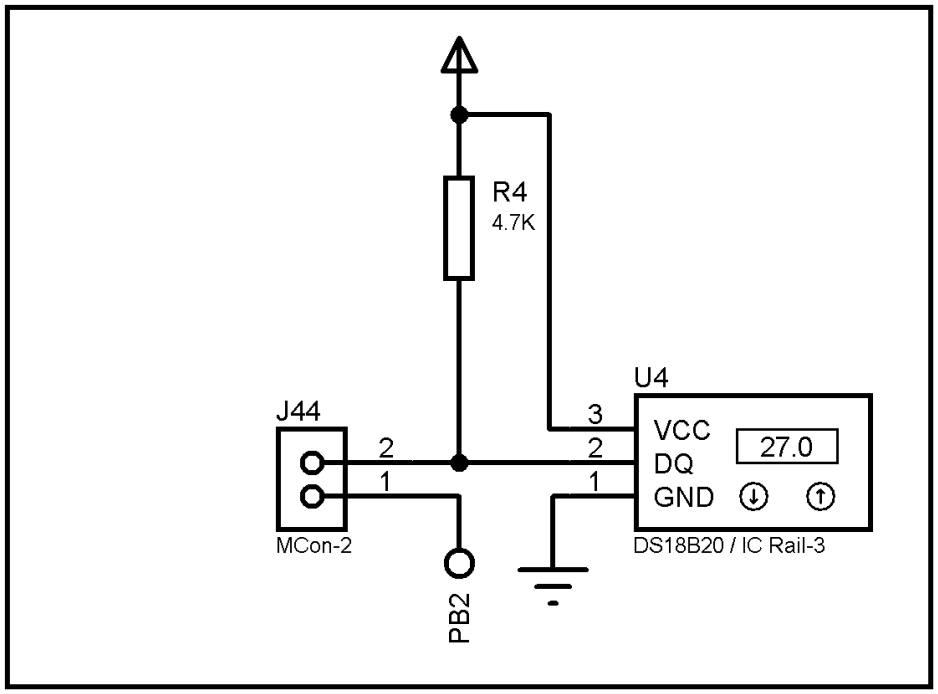


Fig: The schematic of 1-Wire communication pinout of AVR Trainer Kit

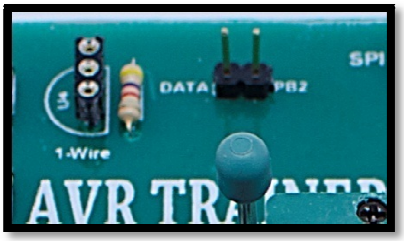


Fig: The 1-Wire communication pinout of AVR Trainer Kit

4. ISP PINOUT:

The ISP PINOUT of AVR Trainer Kit has GND, VCC, SCK, MISO, MOSI and RST (RESET) pins. With these pins you can programme any microcontroller outside the trainer board.

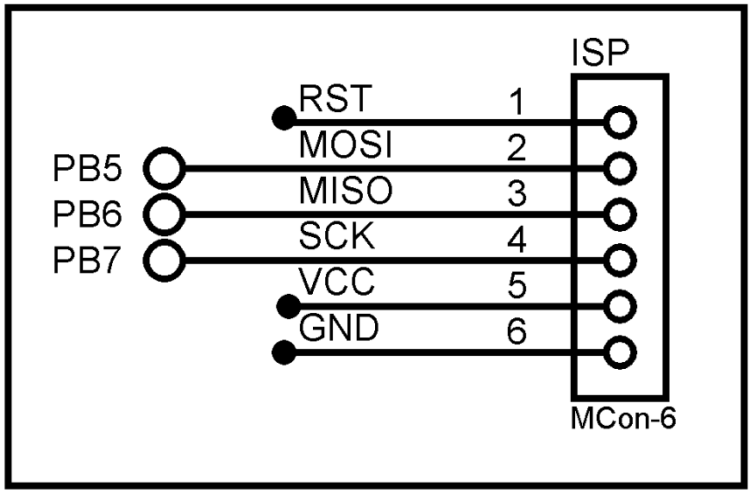


Fig: Schematic of ISP Pinout of AVR Trainer Kit

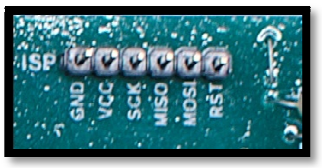


Fig: ISP Pinout of AVR Trainer Kit

5.SPI PINOUT:

The Serial Parallel Interface (SPI) allows high speed-synchronous data transfer between the AVR microcontrollers and peripheral devices or between several AVR devices. The SPI pinout of AVR Trainer Kit features 6 easily accessible pins for SPI.VCC,GND, MISO, MOSI, SCK,SS/.By these pins you can connect any SPI supported device to the microcontroller.

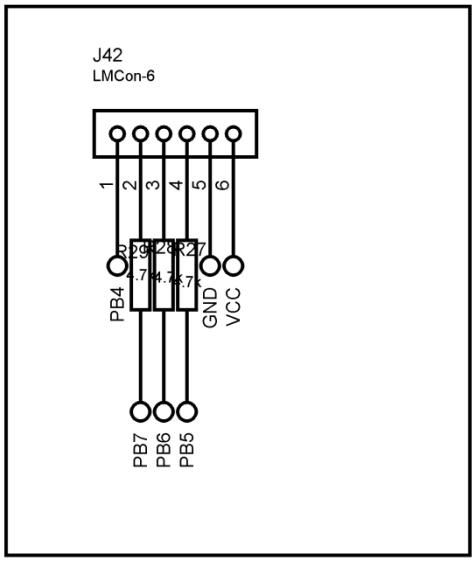


Fig:schematic of SPI Pinout of AVR Trainer Kit

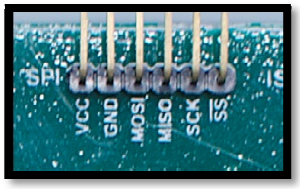


Fig: SPI Pinout of AVR Trainer Kit

6. USART PINOUT:

When USART module is used, it's possible to connect the microcontroller of the development board to any external device that supports USART communication.

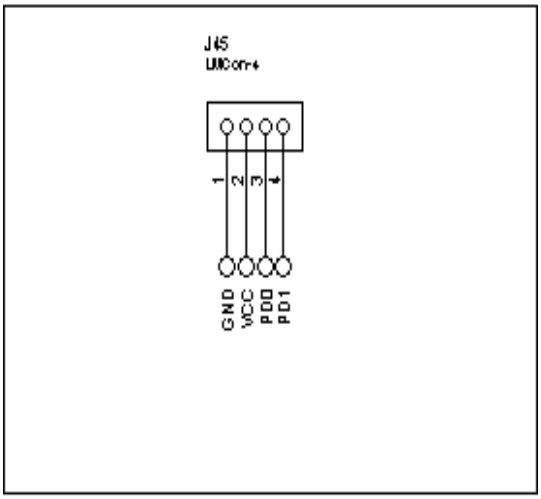


Fig:schematic of USART Pinout of AVR Trainer Kit

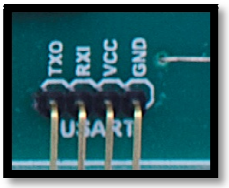


Fig:schematic of USART Pinout of AVR Trainer Kit

7. I2C PINOUT:

The I2C PINOUT of AVR Trainer Kit features VCC, GND, TX1 and RX0 pins side by side so that you can easily connect the microcontroller to any I2C supported device.

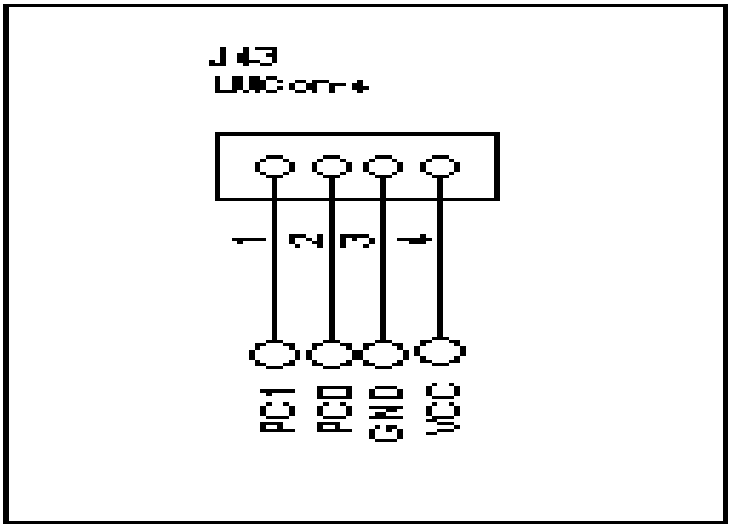


Fig: schematic of I2C Pinout of AVR Trainer Kit

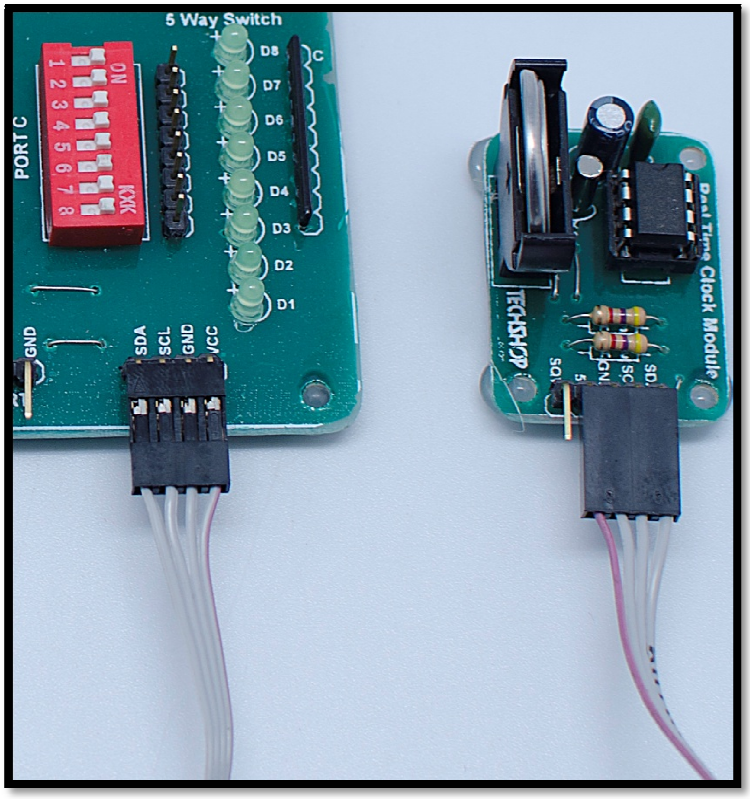


Fig: Connecting RTC module with I2C pins of AVR Trainer Kit

8 Fuse Bytes

I guess many of you were confused when programming AVR fuse bytes. I get many newbie questions like I programmed AVR but it doesn't work. 90% of them always set wrong fuse bytes and make them DEAD or unusable. In this tutorial first I will discuss about fuse bits and then how to active so called "DEAD" chips.

8.1 Introduction

Fuses are an extremely important part of programming a chip, but are rarely explained thoroughly. You only need to set them once, but if you don't do it right, it's a disaster!

You know about flash, EEPROM and RAM as parts of the chip. What I did not mention is that there are also 3 bytes of permanent (by permanent I mean that they stick around after power goes out, but you can change them as many times as you'd like) storage called the fuses. The fuses determine how the chip will act, whether it has a bootloader, what speed and voltage it likes to run at, etc. Note that despite being called 'fuses' they are re-settable and don't have anything to do with protection from overpowering (like the fuses in a home).

The fuses are documented in the datasheets, but the best way to examine the fuses is to look at a fuse calculator such as in "MikroC" compiler. Please collect the latest version. When I am writing this tutorial I have version 5.6 in my hand.

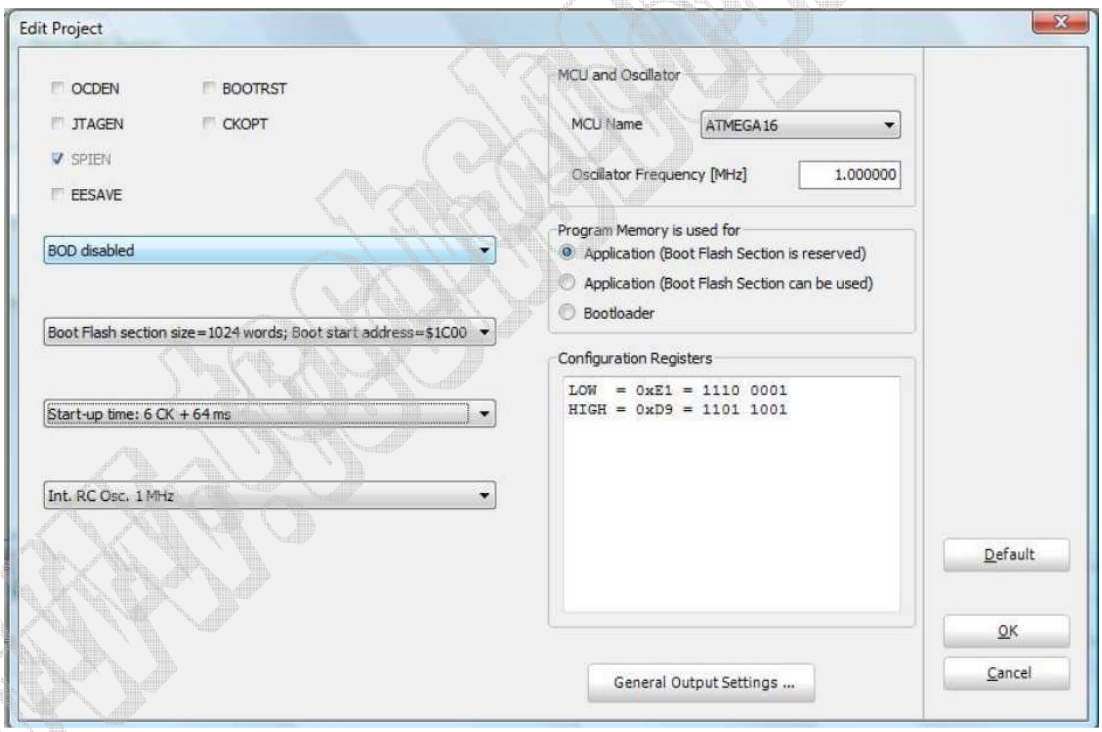
If you want to enable something in AVR what you do? Most probably you set corresponding bit as '1', right? For Fuse bit its opposite. Here „1" means un-programmed or disable and '0' means programmed or enable.

Always remember these.

Fuse bit = 0 => fuse bit is PROGRAMMED

Fuse bit = 1 => fuse bit is UN-PROGRAMMED

Open a project. I am assuming that you are using ATmega16. Go to "Project Menu" and click on "Edit Project" (Shifht+Ctrl+E). A window will open just like below:



SPIEN bit is for programming. It should be enabled for loading program into chip. If you disable it you can't program any more. So 'MikroC' makes this fixed and you can't change it. If you check other bits (Like: JTAGEN) you will see the corresponding bits in "Configuration Registers" box (bottom right) changed to Zero. For more about other fuse bits browse ATmega16's datasheet, page 255. I'd like to mention one thing before I go to next stage. By default in ATmega16, JTAG is enabled by default. That's why you can't use PORTC2-5 as digital I/O. So you have to disable it.

8.2 Brown Out Detect (BOD)

The first drop down menu is “Brown Out Detect”. These fuses set what voltage to turn the **Brown out** protection circuitry on. A brownout for a chip means that the power voltage is too low for it to run reliably at the speed of the clock.

For example, the ATTiny2313 can run as fast at 20MHz but only if the power voltage is between 4.5V and 5.5V. If the voltage is lower than that, it may behave erratically, erasing or overwriting the RAMand EEPROM. It may also start running random piece of the flash program. To keep it from doing that, set the brownout voltage to 4.3V, then if the voltage dips, the chip will turn off until the voltage returns. It will then reset and start over.

If the chip is meant to run at 5V, set the brown-out to 4.3V. If the chip can run at voltage as low as 3.3V you can set the brown-out to 1.8V. If the chip is a 'low voltage compatible' chip such as the attiny2313V (which can run at voltage as low as 1.8V if it’s clocked at 4MHz or less) then you can set the brownout to 1.8V.

For simplicity, disable it.

8.3 Clock Selection

The 2nd option is how the chip is clocked. Every CPU uses a clock, in general one assembly code instruction is run at every clock cycle. The one in your PC has a clock that runs at 1GHz or higher. This little chip runs much slower. If you look at the menu you'll see a huge list of options, but looking carefully you'll see there are two groupings, the **Clock Source** and the **Startup Time**.

The Clock Source can be either of the following:

```
External Clock, Internal 8MHz clock, Internal 4MHz clock,
Internal 128KHz clock, External Crystal (0.4-0.9 MHz), External
Crystal (0.9MHz - 3.0MHz), External Crystal (3.0MHz - 8.0MHz)
or External Crystal (8.0MHz +)
```

External Clock means that a square wave is being input into the **CLOCK-IN** pin. This is pretty rare unless you have a clock generating chip. Don't use this unless you're sure you mean to.

Internal Clock means that there’s a little oscillator inside the chip, it’s not very precise but good for most projects that don’t have fine timing issues. The clock varies with temperature and the power supply voltage. You can choose from a 1MHz, 2MHz, 4MHz or 8MHz clock. Having an internal oscillator means we don't need to wire up a crystal and we can use the clock pins for our own nefarious purposes.

External Crystal, If you need a special clock rate, like 3.58MHz or 12MHz or a high precision clock that won't drift with the temperature, you'll want an external crystal or oscillator.

Crystal Oscillator

Ceramic Oscillator

Clock settings have following options:

Beginners usually use internal 1MHz to 8 MHz. Sochoose internal oscillator. For this tutorial I choose Int. RC Osc. 1 MHz.

8.4 StartupTime

The Startup time can be either of the following: 6CK + 0 ms, 6CK + 4 ms, 6CK + 64 ms.

The Startup Time is just how long the clock source needs to be stable from when power is first applied. Always go with the longest setting 6CK + 64ms unless you know for a fact your clock source needs less time and 64ms is too long to wait.

The “Start-up Time” menu has following options. Choose the 64ms:

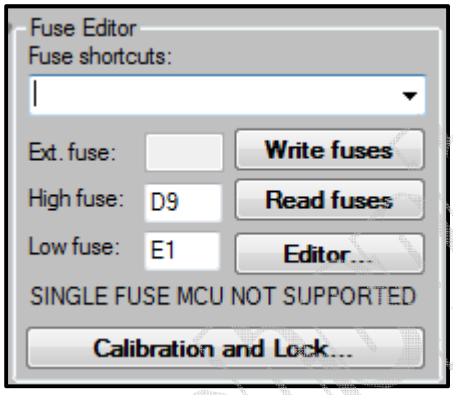
By default, chips that come from the factory have the Internal 1 MHz clock with 6CK + 0ms Startup. So, in configuration register box I see the fuse bits are: LOW: 0xE1 HIGH: 0xD9

It’s time to load this bit into chip.

8.5 Writing Fuse Bytes (By AVRpal.net**):**

If your chip is brand new, then by default its clock frequency is set into internal 1 MHz. So, in AVRpal.net you have to check "Set Slow SCK" box. If you change the fuse bits for higher frequency then you can either "check" or "uncheck" Slow SCK, your choice. The difference is, when "Slow SCK" is checked the program will load slowly and vice versa. REMEMBER, if clock set to 1 MHZ, you have to check "Slow SCK" box. Otherwise it won't work.

So write fuse bytes in corresponding box and press 'Write fuses'.



9. Trouble Shooting!

I think the above tutorial will help you set fuse bits properly. But though what if you set wrong fuse bit? (Most of the time it happens with pony prog user). Here comes the solution:

If your programmer does not detect you chip:

1) First check, is "SLOW SCK" checked (in AVRpal)? If not, then put 'Check' and try again.

2) If till problem remains then put a crystal (12 or 16MHz, I prefer 16MHz) between XTAL1 and XTAL2 pin (if you are using ISP connector to program), with "SLOW SCK" checked. Most of the time problem will solve in this stage.

3) If till problem exist then maybe you disabled SPIEN bit, which is unlikely, a rare situation, but definitely it isn't for wrong clock fuse bit. If that is the situation then you need high voltage programmer (like TopWin). Also you can use "Fuse bit Doctor" (Google it!).

4) Check the Target mcu's Vcc and GND pin; see if they get between 4.5 to 5.5V.

5) 'Short Test'of MOSI, MISO, SCK, RST pin between your programmer and your chip.

I hope the five steps above will solve your problem.

10. Warranty

Product warranty is valid for 6 months. Warranty only applies to manufacturing defect. Damage caused by misuse is not covered under warranty. Warranty does not cover freight cost for both ways.

Servicing is free for life time.

Thank you for using TechShop's Product.