

# ATmega 16 Development Kit User Guide

*November, 2015*

# Table of Contents

---

Chapter 1: Overview .....	1
In this guide.....	1
Chapter 2: Setting up the IDE and running the first project.....	3
What You Need for a Working System.....	3
Installing the software.....	3
Connecting the battery.....	3
Steps to create a project in Atmel Studio 4.....	4
Your first project.....	5
Troubleshooting.....	7
Chapter 3: Features of the board.....	8
PORTs.....	8
I/O Devices.....	10
Board power supply.....	11
RS232 serial communication.....	11

# Chapter 1:

## Overview

ATmega 16 Microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The ATmega16 is open-source, which means the hardware is reasonably priced and the development software is free. This guide is for students and hobbyists alike who are confronting the ATmega16 for the first time.

### In this guide

This guide covers the ATmega16 development board offered by Grape Embedded Solutions, a good choice for students and educators and will help with the initial setup for doing a project. With this development board, you can kick-start your prototyping or even develop a product. Write the code, upload/burn it, create interface circuits to read switches and other sensors, control motors etc. with very little effort and give life to your project.

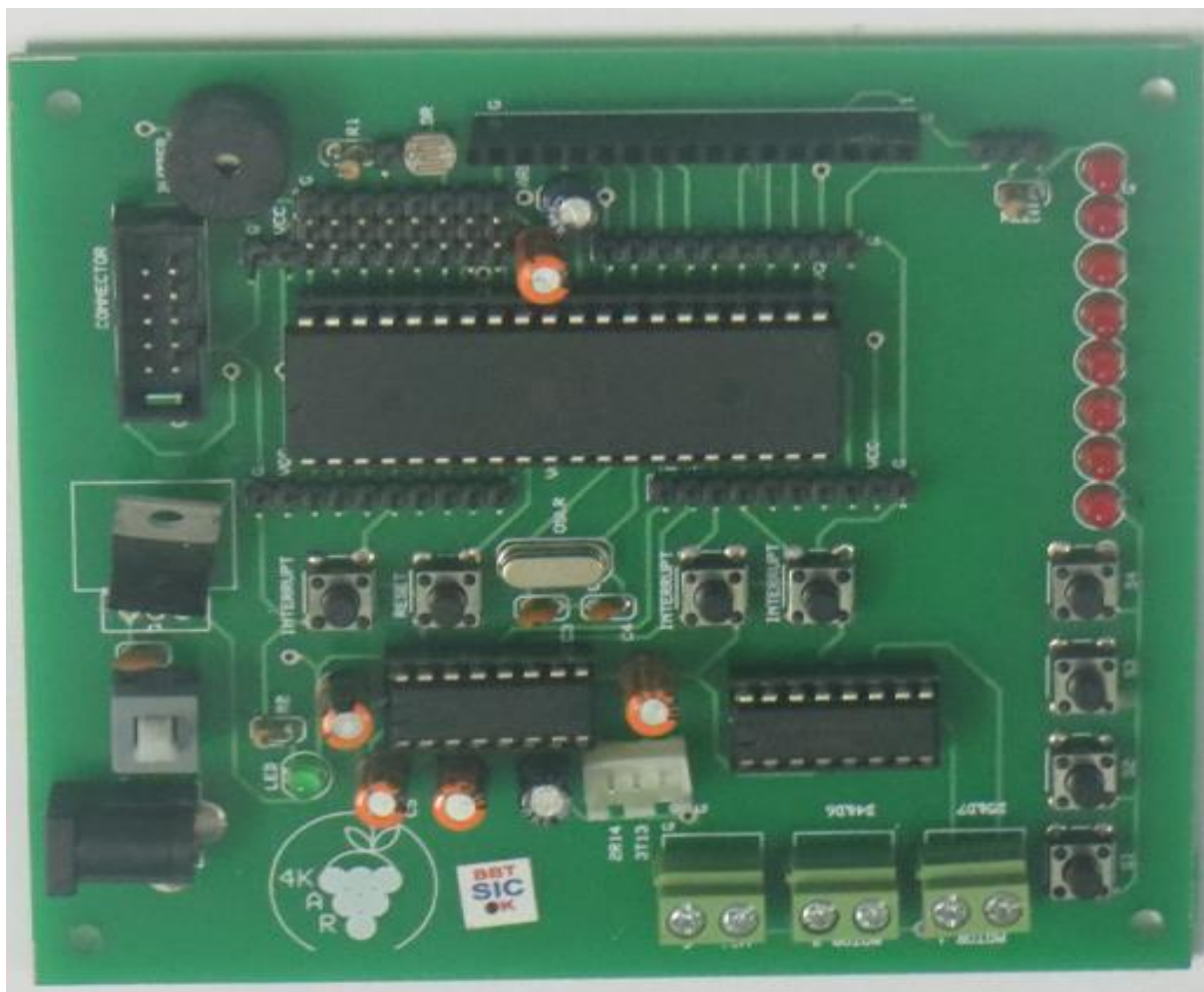


Figure 1. The development board

The board features an 8 bit Atmel ATmega16 microcontroller operating at 5 V with 1KB of SRAM, 16 KB of flash memory for storing programs and 512B of EEPROM for storing parameters. The clock speed is 16 MHz, which translates to about executing about 300,000 lines of C source code per second. The board has 32 digital I/O pins and 8 analog input pins. There is a USB connector for 'talking' to the host computer and a DC power jack for connecting an external power source, for example a 9 V battery, when running a program while not connected to the host computer. When connected to the host, the host provides the power for the board via the USB cable. Male headers are provided for interfacing to the I/O pins using header connectors. A Motor Driver supplies the motors with the extra current required, else the current is provided by the microcontroller. On-board connections are provided for LCD. In addition there are 8 LEDs for indicating status of ports or to display patterns which is usually the 'hello world' project in the world of micro-controllers. The next chapters will explain how to set up your first project and run the basic 'hello world' program (LED blinking).

**Note:** This board is also compatible with ATmega 32 and ATmega 644.

# Chapter 2:

## Setting up the IDE and running the first project

This section will explain how to setup the Integrated Development Environment (IDE) and how to run your first project on the development board.

### What You Need for a Working System

1. ATmega16 development kit by Grape Embedded Solutions.
2. 9V battery or external power supply (for stand-alone operation)
3. Host PC running the (AVR Studio 4 & Khazama AVR Programmer) development environment. Versions exist for Windows, Mac and Linux. Download links:  
<http://khazama.com/project/programmer/>  
<http://www.atmel.com/tools/STUDIOARCHIVE.aspx>

### Installing the Software

Download and install AVR Studio 4 from Atmel's official website. It is free to download and use. The latest version is AVR Studio 7 but Atmel Studio 4 is powerful enough with a familiar UI and is the most widely used IDE when it comes to online tutorials. Also download and install Khazama AVR Programmer for burning hex code after building the embedded C source code.

### Connecting a Battery

For stand-alone operation the board is powered by a battery rather than through the USB connection to the computer. While the external power can be anywhere in the range of 6-24V (for example, you could use a car battery), a standard 9 V battery is convenient. While you could connect the leads of a battery snap into the  $V_{in}$  and Gnd connections on the board, it is better to solder the battery snap leads to a DC power plug and connect to the power jack on the board.

**Warning:** Watch out for the polarity as you connect the battery to the snap as reverse orientation could blow your board.

## Steps to create a project in Atmel Studio 4:

1. Start Atmel Studio 4. The initial screen is as shown below:

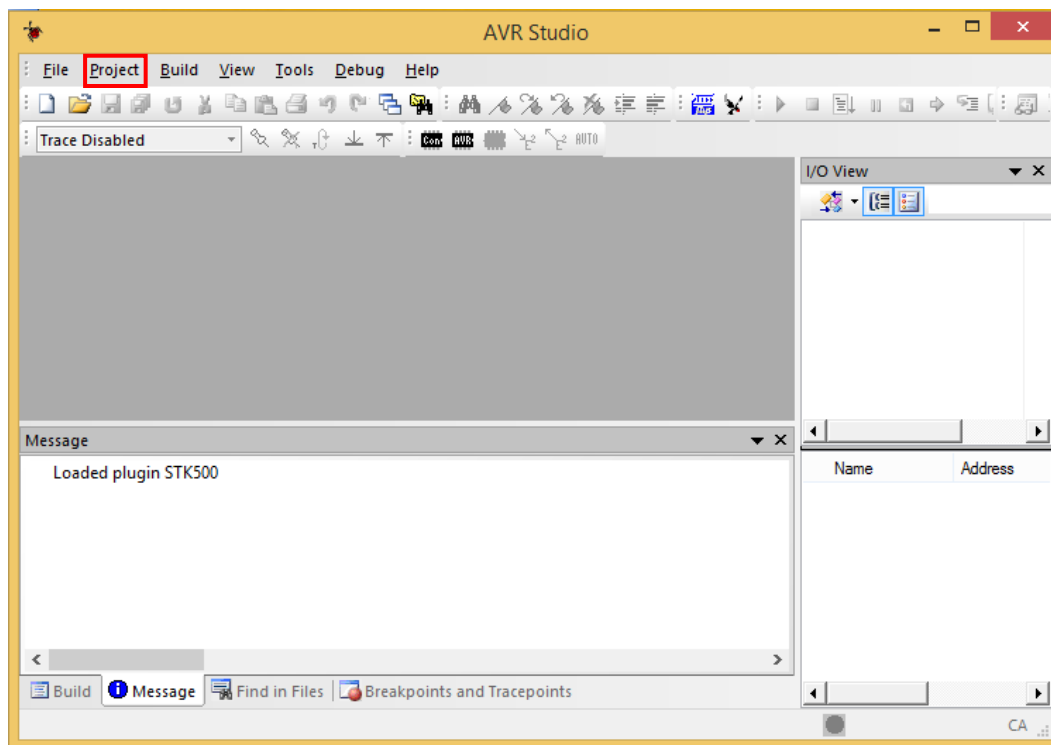


Figure 2. IDE initial window

2. Click on the Projects -> New Projects. You will be navigated to the next screen as shown below. Select AVR GCC and give a name to your project (in this case - Test\_project) and mention path/location to store the file (where you want your project to be stored). Then click on 'Next'.

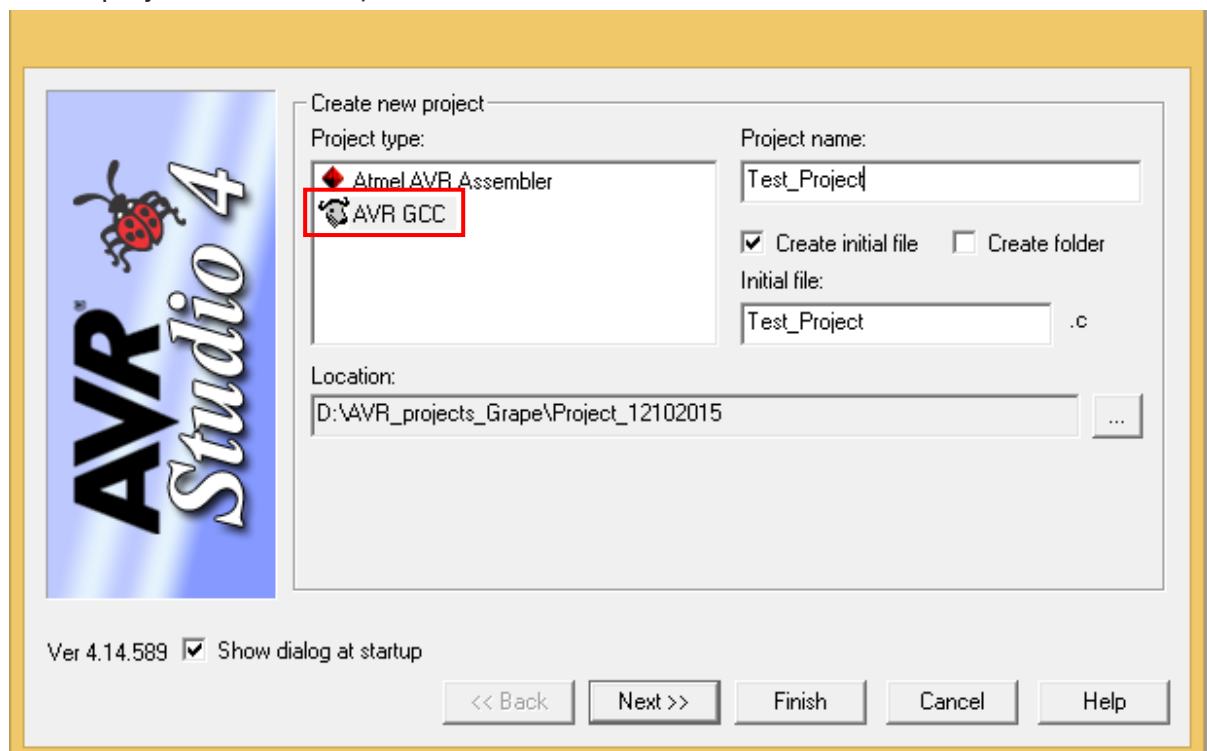


Figure 3. New project window

3. Select AVR Simulator as your debugging platform and the device as ATmega16. Click on finish and now you have setup your project and you can go ahead with your first project.

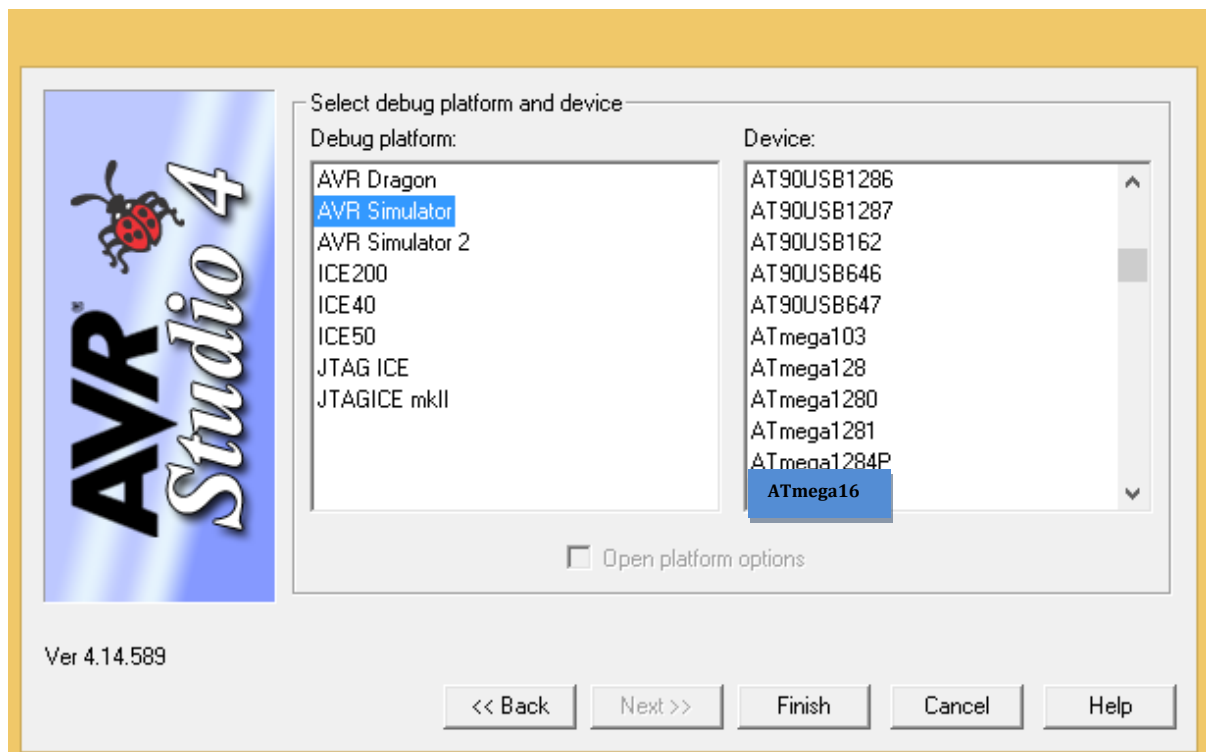


Figure 4. Debugging platform and device choosing window

## Your first project:

Connect your ATmega16 development board to the Burner board and this in turn, to the computer with the USB cable. If there was already a program burned into the Development board, it will run.

**Warning:** Do not put your board down on a conductive surface; the pins on the back will get shorted and damage the board!

Start the AVR development environment. In AVR-speak, programs are called “Projects”, but here we will just call them programs. In the editing window that comes up, enter the following program, paying attention to where semi-colons appear at the end of command lines. Remember, each project can contain only one source (.c) file. Delete any multiple source files. However, you can include any number of header files.

```
#include<avr/io.h>
#include<stdio.h>
#include<util/delay.h>
```

```
int main()
```

```

{

    DDRC=255;          //Declare the whole PORT-C as output
    PORTC=0;           //Ground PORT-C; essentially switching off the LEDs
    while(1)
    {
        PORTC= ~(PORTC);
        _delay_ms(1000);
    }
return 0;
}

```

The editor window will look something like this:

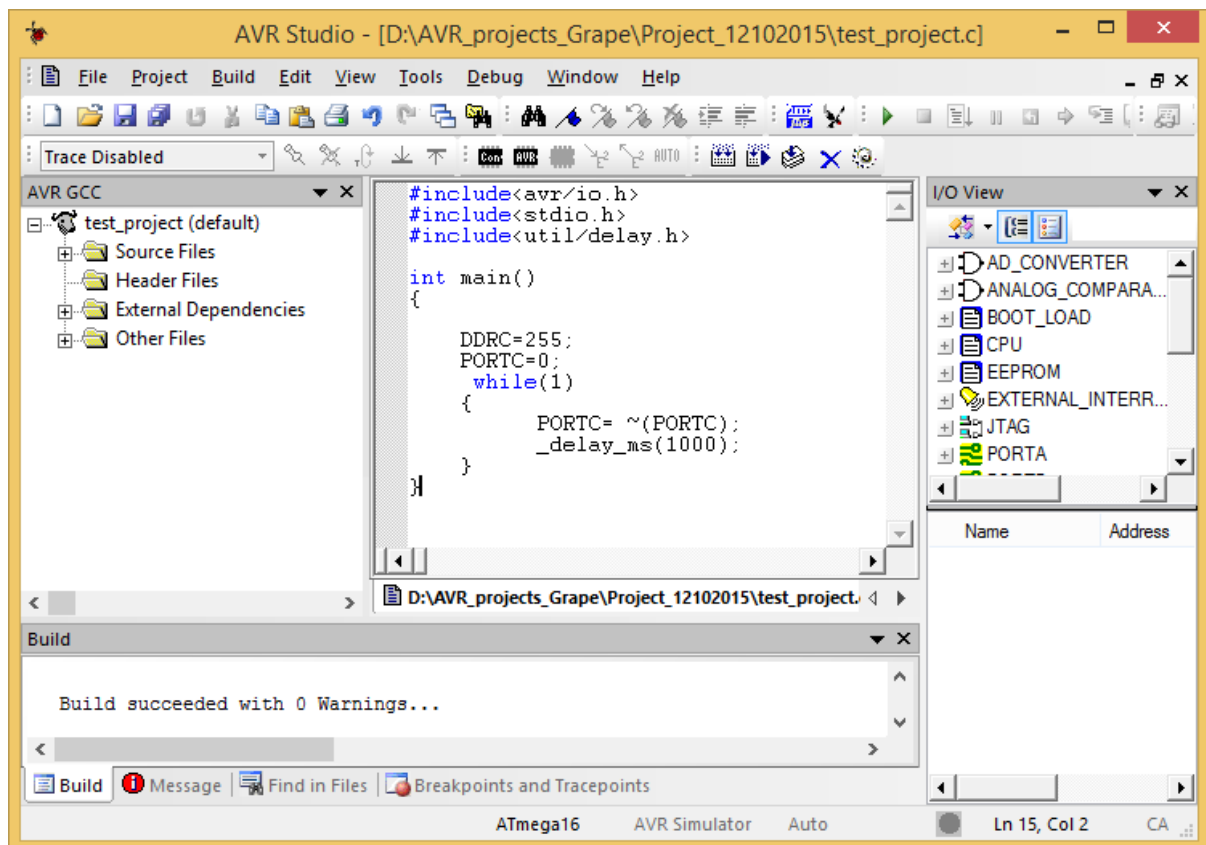



Figure 5. Editor window

Click the button  in the tool-bar or F7 to build the program in order to generate a loadable hex file to ATmega16. This hex file will be present in the project folder. After building the program the hex file is burned onto the ATmega16 using Khazama AVR Programmer. The window will look something like this.



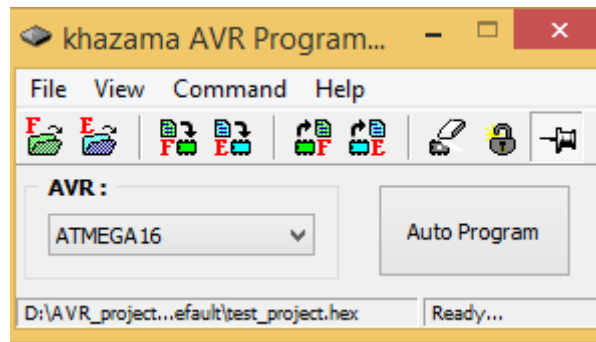




Figure 6. Khazama AVR Programmer

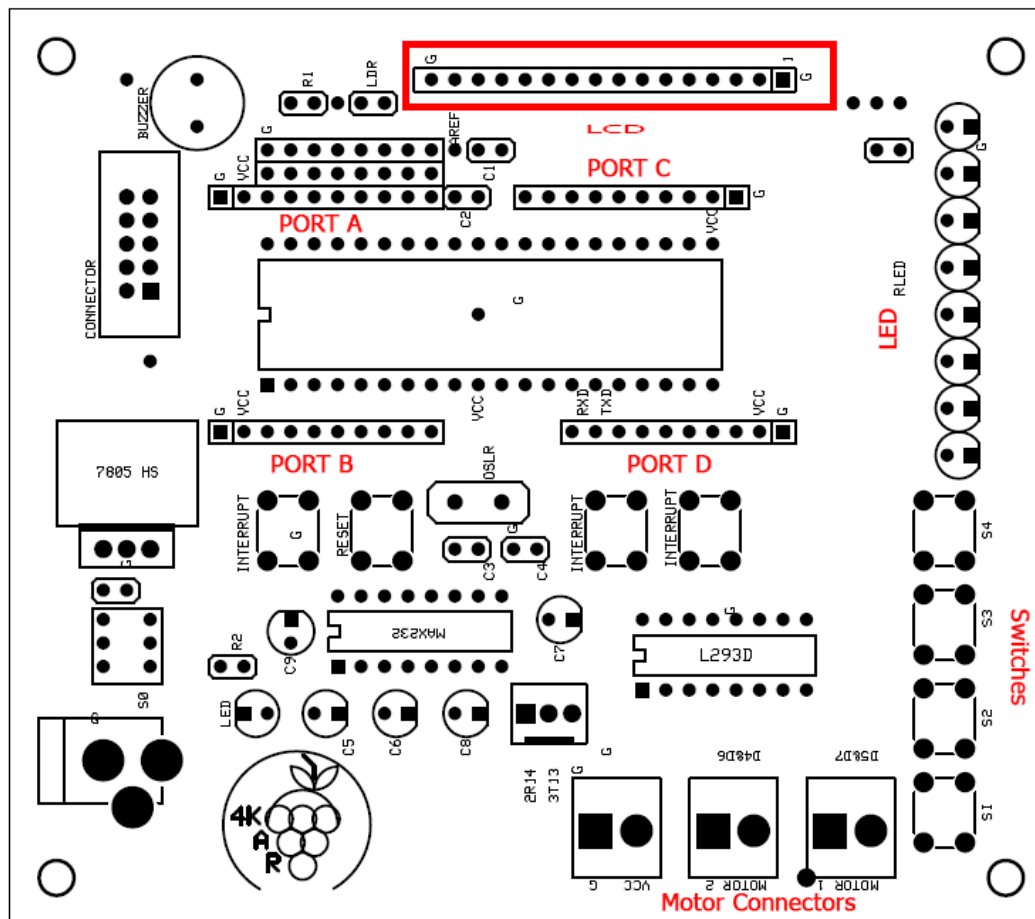
Click the  button to select the hex file which you want to upload to ATmega16. Then  
 Click the  button to upload the hex file on to ATmega16. Congratulations; you have created and run your first ATmega16 program!

## Troubleshooting

If there is a syntax error in the program caused by a mistake in typing, an error message will appear in the bottom of the program window. Generally, staring at the error will reveal the problem. If you continue to have problems, feel free to contact us by mail - [connect2grape@gmail.com](mailto:connect2grape@gmail.com)

# Features of the Board

This chapter explains the functionalities of various components, ports and how to use the development board in different modes/configurations.



**Figure 7. Component layout of the development board**

## PORTs

**PORT A:** Port A is an 8 bit bi-directional I/O port with internal pull-up resistors when ADC feature is not enabled. One of the important features of ATmega 16 can be enabled on this Port, i.e. Port A can be used as an 8 channel ADC which means 8 analog to digital conversions can take place concurrently. The resolution of the ADC is 10 bits which can be adjusted as per requirement by setting bits on ADLAR register. ADC can be enabled by setting respective bits on ADMUX registers. In our development Port A can be used as ADC or I/O port as per the user's requirement.

**PORT B:** Port B is an 8 bit bi-directional I/O port with internal pull-up resistors. Port B also facilitates alternative functionalities as listed below.

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	SS (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/output)

In case of the ATmega 16 development board by Grape Embedded Solutions, in order to provide multiplexing, the 4 bits of the Port, namely PB0, PB1, PB3 and PB4 are connected to switch 1-4, and PB2 is provided with a switch to provide an external interrupt in case of user requirements.

**PORT C:** Port C is an 8 bit bi-directional I/O port with internal pull-up resistors. Port C also facilitates the function of JTAG interface. If the JTAG function is enabled, the pull-up resistors on pins PC2, PC3 and PC5 will be active even if a reset occurs. In case of ATmega 16 development board of grape, PORTC is connected to 8-LEDs, which can be used as output and also to a 16x2 LCD. PORTC can drive either LCD or LED but not both simultaneously. The selection between LCD and LED is done by alternatively connecting the on-board jumper between the three male pins as shown in the images in the previous section of this chapter.

**PORT D:** Port D is an 8 bit bi-directional I/O port with internal pull-up resistors. Port D also facilitates alternative functionalities as listed below.

Port Pin	Alternate Functions
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

In case of ATmega 16 development board by Grape Embedded Solutions, in order to provide multiplexing, the 4 bits of the Port, namely PB4, PB5, PB6, and PB7 are connected to motors via L293D which helps the user to connect motors to the board. The serial communication pins are connected to the on-board RS232 to facilitate serial communication. The external interrupt pins are connected to interrupt switches (to provide external interrupt).

## I/O devices

**Buzzer:** It is an audio signalling device and the one used here is piezo-electric. Typical use case could be to sound an alarm, output of timer or confirmation of user input such as a switch etc. To use it,

**Light Dependent Resistor (LDR):** A photoresistor or light-dependent resistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity. To take the reading from the LDR i.e. to use it as an input device, connect the pin between the LDR and the resistor to any pin of a PORT declared as an input.

**Motor Driver:** The motor driver used here is L293D, capable of driving two motors at the same time. L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors. A separate power supply (4.5 V to 36 V) is needed to power the motors as the board power supply will not suffice. The supply and motor connectors have been shown in the silkscreen diagram in Figure 7. For more details on L293D, going through datasheets available in the web is encouraged.

**Switches:** Four momentary switches have been provided for applications requiring switch interfacing. Switches 1 to 4 are connected to PORT B 1 to 4 respectively. If these PORT pins are configured as inputs, the switches can be used.

**Liquid Crystal Display (LCD):** The LCD used here is JHD162A. It is a 16X2 display device i.e. it can display sixteen characters in each of the two rows. It is connected to PORT C. The whole PORT C has to be configured as output for LCD interfacing. For more details on JHD162A, going through datasheets available in the web is encouraged.



Figure 10. Correct connection method of the LCD. Take care that the LCD is not right/left shifted with respect to the connectors. Press down the LCD on the female connectors properly.

**Warning:** Make sure the LCD is connected properly to the 16 female slots provided (See pic below). If the LCD is not connected properly there are chances of damaging the LCD or the development board.

**Light Emitting Diodes (LEDs):** They are connected to PORT C. The whole PORT C needs to be configured as output to operate these. LEDs and LCD will not work together as they are connected to the same PORT i.e. PORT C. Their operation is mutually exclusive. So a jumper connector is provided to choose between the two. Refer Figure 8 and 9 for the connection method.

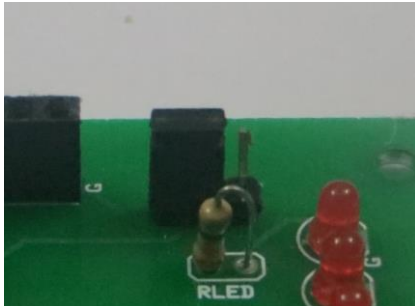


Figure 8. Jumper position for LCD selection

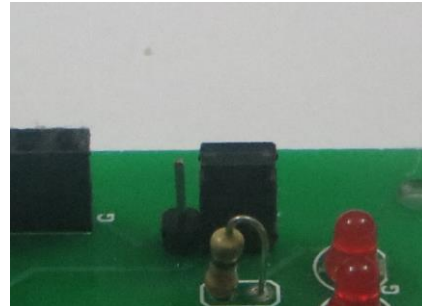


Figure 9. Jumper position for LED selection

## On-Board Power supply

In order to facilitate/provide external - mobile power to the micro-controller and the onboard peripherals there is a Standard DC socket for powering the board where a standard 9V battery can be connected. Since the microcontroller and the ATmega16 development board are designed to work on 5V, this DC power socket is connected to the voltage regulator IC 7805 which steps down the input DC voltage from the battery voltage to the microcontroller specific voltage i.e. 5V.

**Warning:** Make sure the battery is connected only when the Burner (USBasp) is disconnected from the development board or from your PC/Laptop.

## RS232 Serial Communication (MAX232IC)

### RS-232 Basics

RS-232 (Recommended Standard – 232) is a standard interface approved by the Electronic Industries Association (EIA) for connecting serial devices. In other words, RS-232 is a long established standard that describes the physical interface and protocol for relatively low-speed serial data communication between computers and related devices. RS-232 is the interface that your computer uses to “talk” to and exchange data with your modem and other serial devices.

In RS232 there are two data lines RX and TX. TX is the wire in which data is sent out to other device. RX is the line in which other device put the data which it needs to be sent to the device. As there is no "clock" line for synchronization accurate timing is required so transmissions are carried out with certain standard speeds. The speeds are measured in bits

per second. Number of bits transmitted is also known as **baud rate**. Some of the standard speeds are 1200,2400,4800,9600 bauds per second.

### MAX232IC

ATmega16 development board offered by Grape Embedded Solutions contains an onboard MAX232 IC which facilitates serial communication with other microcontroller or PC. MAX232 ICs were invented by Maxim. These IC packages are used to convert TTL/CMOS logics to RS232 logic directly. Below is the circuit diagram of the MAX232 IC.

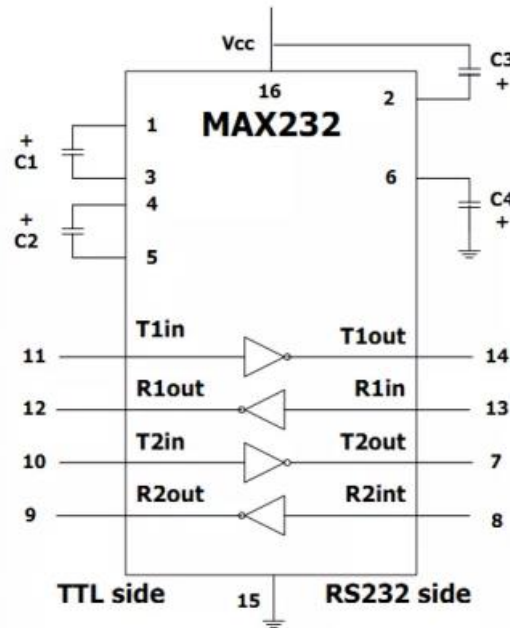


Figure 10. Pin out of MAX232

MAX232 is used to convert TTL to RS232, and vice-versa as shown in the above circuit diagram. MAX232 is basically a 16 pin IC with the transmitter pins connected to the microcontroller and the port such that the input transmitter pin get TTL input from the Microcontroller and the output transmitter pin supply output to the RS232 port. The receiver pins are connected to the RS232 port such that the input receiver pin receive RS232 standard input from the PC port and the output receiver pin supplies the TTL input to the Microcontroller. Thus the transmitter takes input from the Microcontroller and gives output to the RS232 port whereas the receiver takes input from the RS232 port and gives output to the Microcontroller.



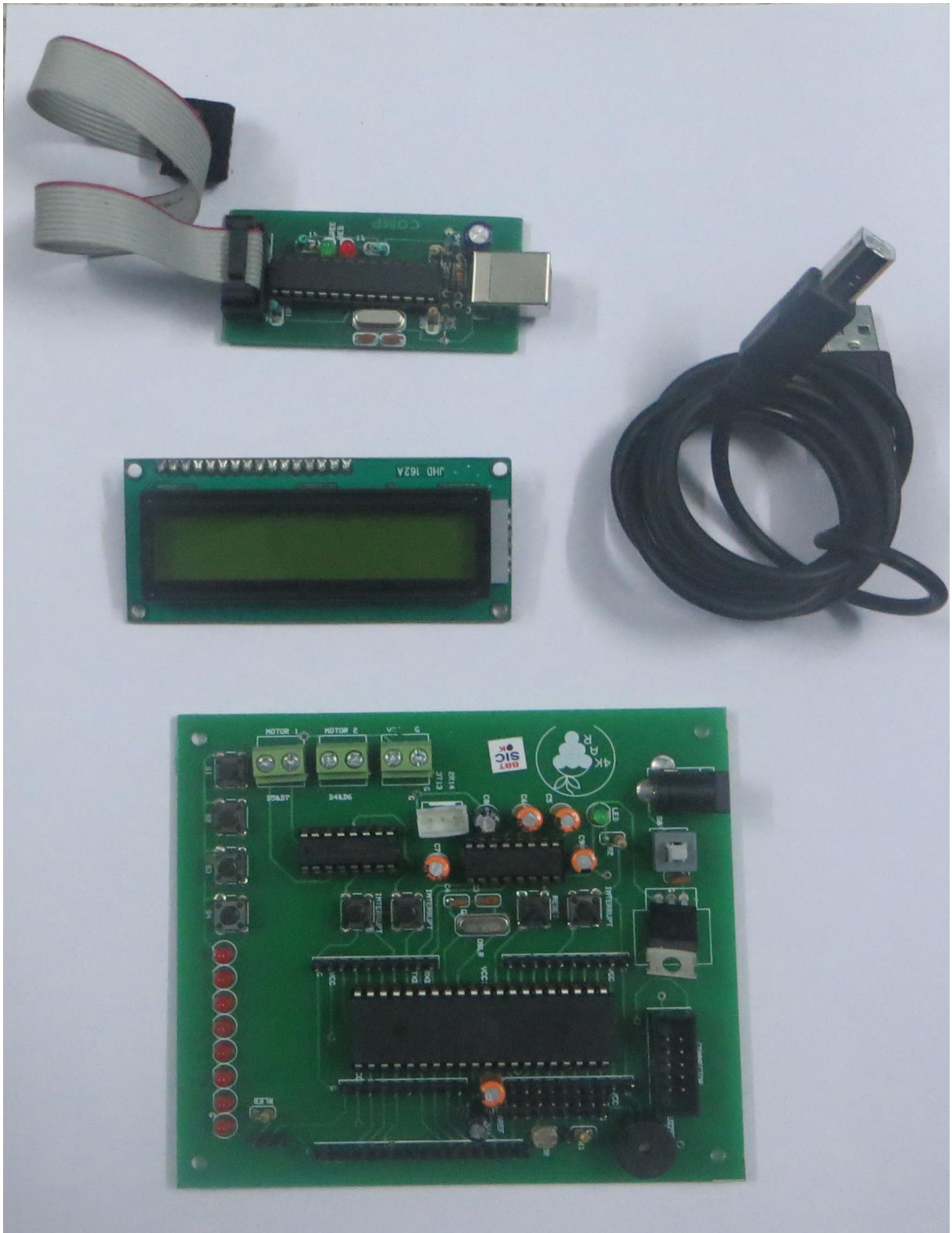


Figure 11. The full development kit by Grape Embedded Solutions