SSG EMBEDDED SOLUTIONS

SSG

info@ssges.co.in

DEVELOPMENT KIT of AVR DOCUMENTATION

List of Contents

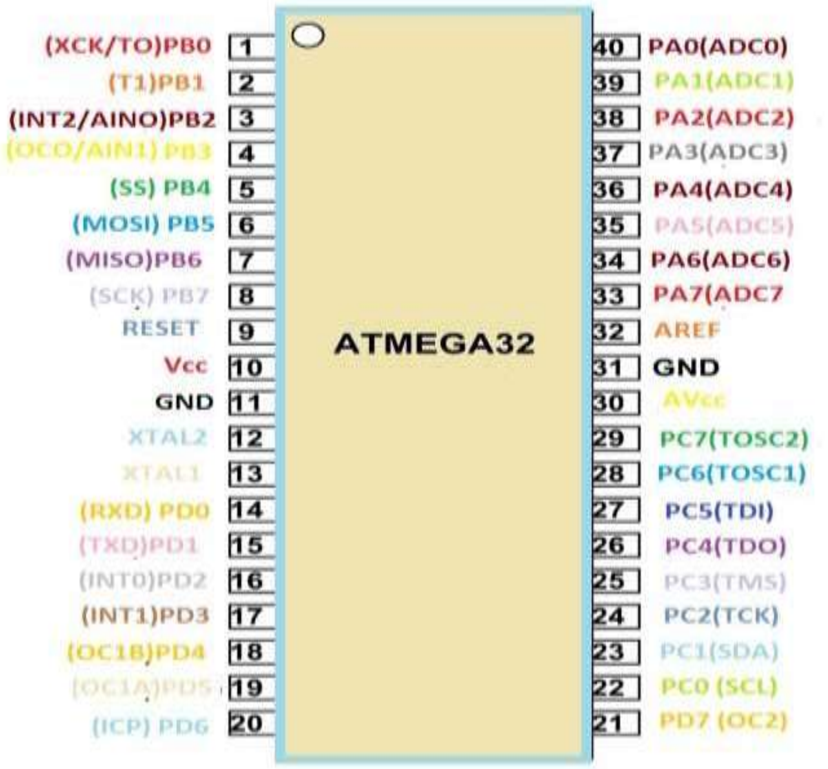
Introduction

ATmega32 is eight-bit higher enactment microcontroller, it is manufactured by an Atmel (it is a designer and builder of different semiconductors materials). It is founded on enriched RISC which stands for (Reduced Instruction Set Computing) design which consists of 131 (one thirty-one) potent commands. Mostly commands implement in one mechanism sequence. The maximum frequency at which it operates sixteen MHz. It delivers a subtle equilibrium among enactment and balance. It is the Pico Power sort of the normal ATmega328 kind which permits it to work below lesser voltage and power necessities, nearly 1.62 volts.

.

This module consists of one-kilo byte EEPROM (Electrically Erasable Programmable Read-Only Memory), two kilobyte SRAM (static RAM), 54 inputs and 69 general persistence output lines, thirty-two universal persistence functioning registers, a JTAG (Joint Test Action Group) interfacing for border scanning and onboard repairing or debugging.

Pin Diagram



Specifications

- 32 x 8 general working purpose registers.
- 32K bytes of in system self programmable flash program memory
- 2K bytes of internal SRAM
- 1024 bytes EEPROM
- Available in 40 pin DIP, 44 lead QTFP, 44-pad QFN/MLF
- 32 programmable I/O lines
- 8 Channel, 10 bit ADC
- Two 8-bit timers/counters with separate prescalers and compare modes
- One 16-bit timer/counter with separate prescaler, compare mode and capture mode.
- 4 PWM channels
- In system programming by on-chip boot program
- Programmable watch dog timer with separate on-chip oscillator.
- Programmable serial USART
- Master/slave SPI serial interface

Application

- It used in different temperature control systems.
- It used in the different analog signal calculation and management techniques.
- It used in different entrenched schemes like chocolate apparatus, peddling mechanism.
- It used for controlling the motor.
- It used for Numerical signal handling.
- It used for Marginal Interfacing scheme.

Material Required

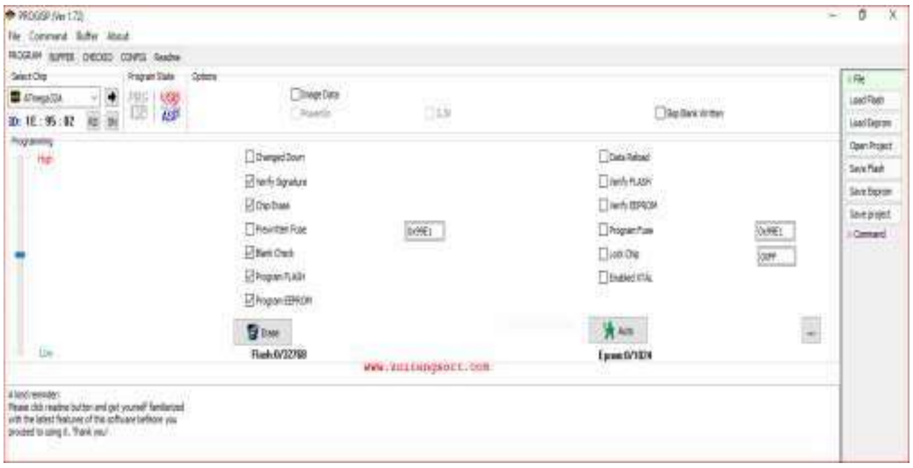
- USB Cable
- Flat Ribbon Cable
- AVR Cable

Usbaspv2011_Proteus8.3 Files

usbasp.2011-05-28.tar.gz (519 kB)

Open AVR folder---→Open Usbaspv2011_Proteus8.3Files folder--→

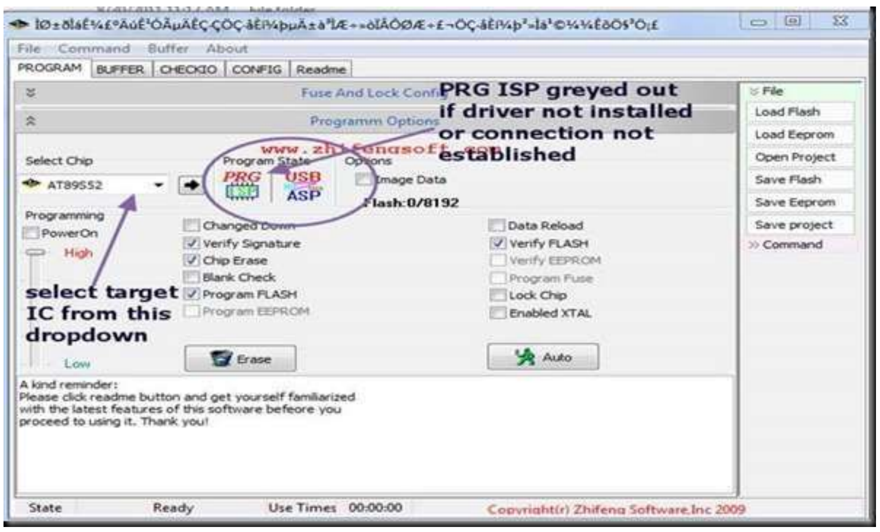
Open progisp(application-577KB) file --→this will provide following window



A kind reminder:
Please click readme button and get yourself familiarized with the latest features of the software before you proceed to using it. Thank you!

Steps:

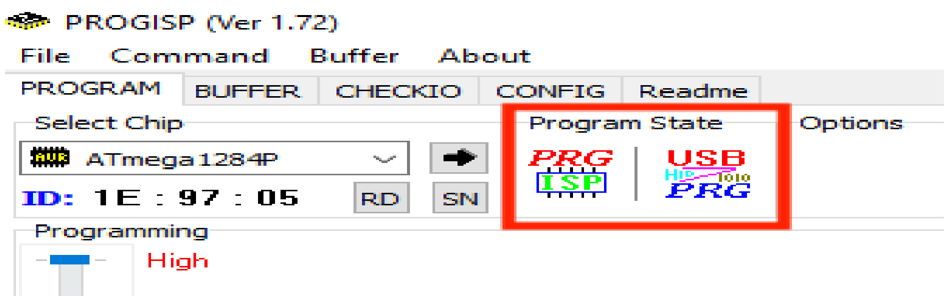
1) At first connect AVR kit by using USB cable-→this will highlight Program State PRGISP



PRG ISP greyed out if driver not installed or connection not established

select target IC from this dropdown

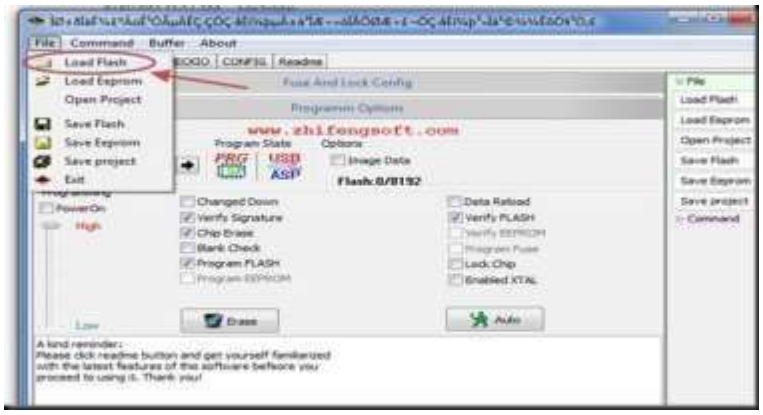
A kind reminder:
Please click readme button and get yourself familiarized with the latest features of this software beefore you proceed to using it. Thank you!



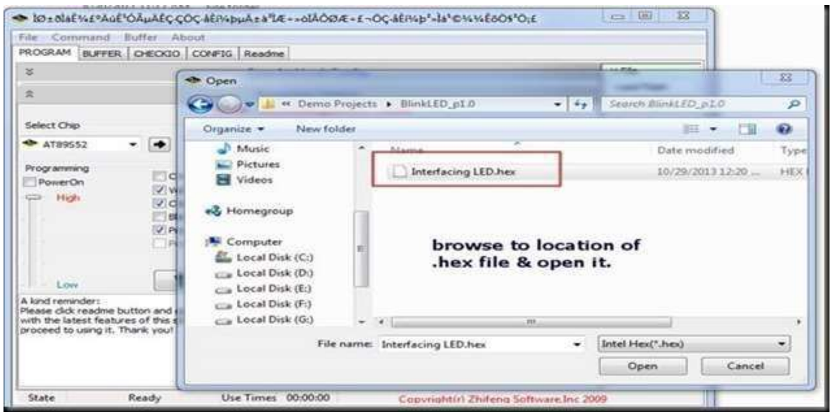
2) Move cursor on select chip--→select proper IC ie Atmega32A

3) Load program from-→Load Flash--→select your Hex file and click on open

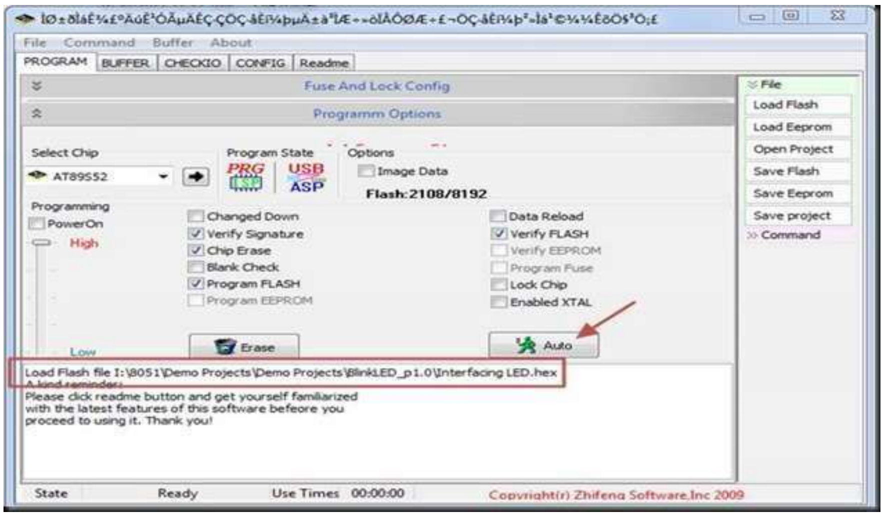
Ut



A kind reminder:
Please click readme button and get yourself familiarized with the latest features of the software befeore you proceed to using it. Thank you!



---then click on Auto

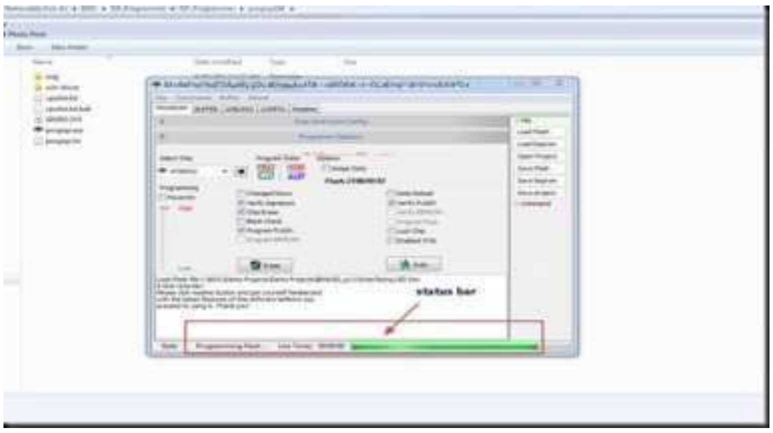


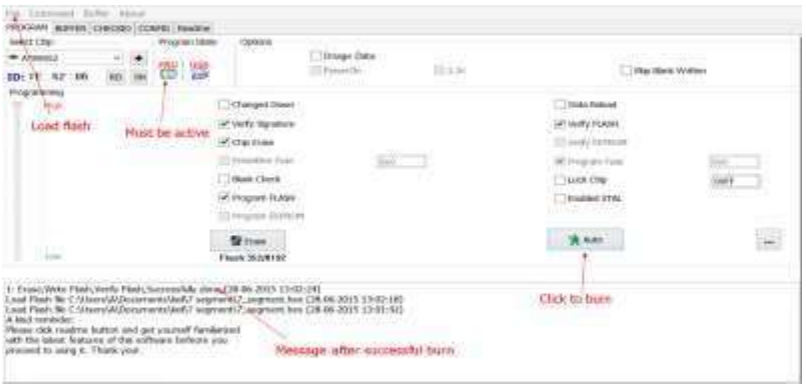
Flash:2108/8192

```
Load Flash file I:\8051\Demo Projects\Demo Projects\BlinkLED_p1.0\Interfacing LED.hex
A kind reminder:
Please click readme button and get yourself familiarized
with the latest features of this software beferore you
proceed to using it. Thank you!
```

State Ready Use Times 00:00:00 Copyright(r) Zhifeng Software,Inc 2009

4) See the output on your kit





Flash 352/8192

```
1: Erase,Write Flash,Verify Flash,Successfully done [28-06-2015 13:02:24]
Load Flash file C:\Users\A\Documents\led\7 segment\7_segment.hex [28-06-2015 13:02:18]
Load Flash file C:\Users\A\Documents\led\7 segment\7_segment.hex [28-06-2015 13:01:51]
A kind reminder:
Please click readme button and get yourself familiarized
with the latest features of this software before you
proceed to using it. Thank you!
```

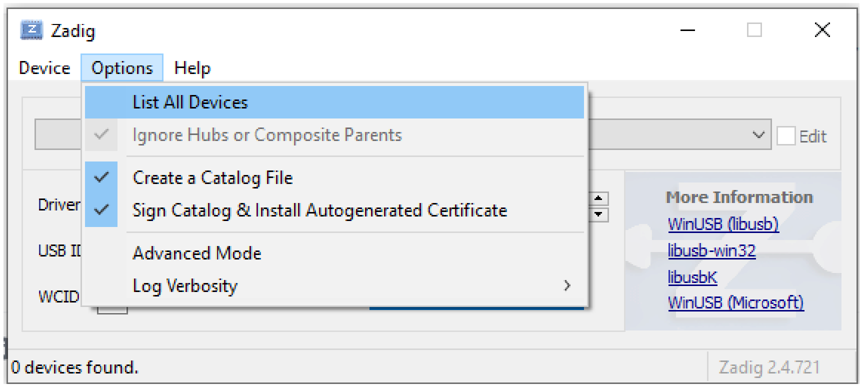
Zadig-2.7 Driver Installation

- Install drivers with Zadig When you plug your RTL device in, for the first time, Windows may request a driver, or automatically install a driver from Microsoft - this is OK as it will be replaced in the next few steps using Zadig.
- Important: Do not install the software on the CD that comes with the device.
- Download the latest version of Zadig from: http://zadig.akeo.ie
- You will need to use 7zip to extract it from the .7z file: http://www.7-zip.org
- Or you can download the version 2.1.1 Zipped from here: http://www.theremino.com/wp-content/uploads/files/zadig_2.1.1.zip
- Run the Zadig.exe file and you should see the following with an empty list.

Step 1: Plug in USBASP

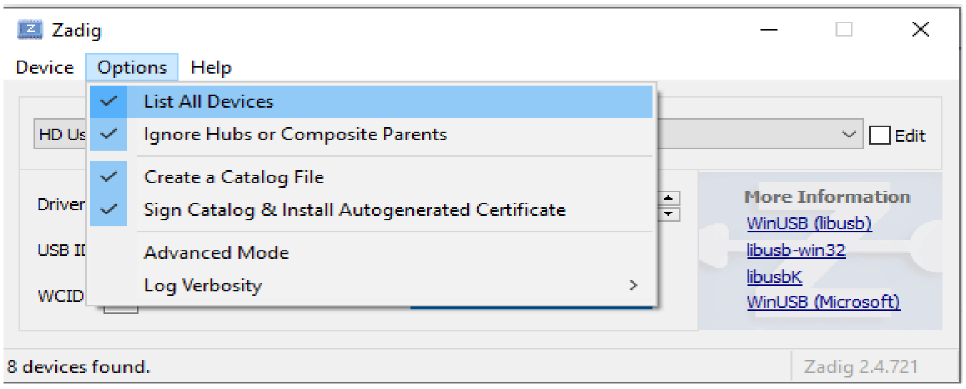
Step 2: Install Zadig

Step 3: Open Option



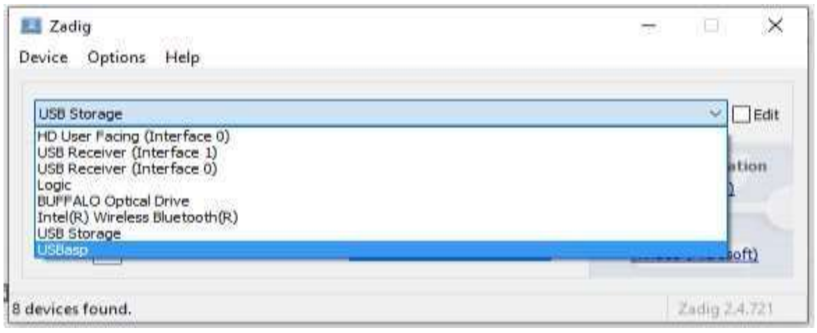
0 devices found.

Step 4: Check List All Devices



8 devices found.

Step 5: Select USB ASP



Zadig

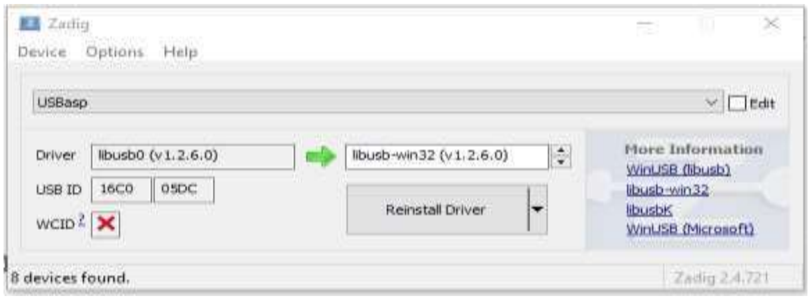
Device Options Help

USB Storage

HD User Facing (Interface 0)
USB Receiver (Interface 1)
USB Receiver (Interface 0)
Logic
BUFFALO Optical Drive
Intel(R) Wireless Bluetooth(R)
USB Storage
USBasp

8 devices found. Zadig 2.4.721

Step 6: Select Libusb-win32



Zadig

Device Options Help

USBasp

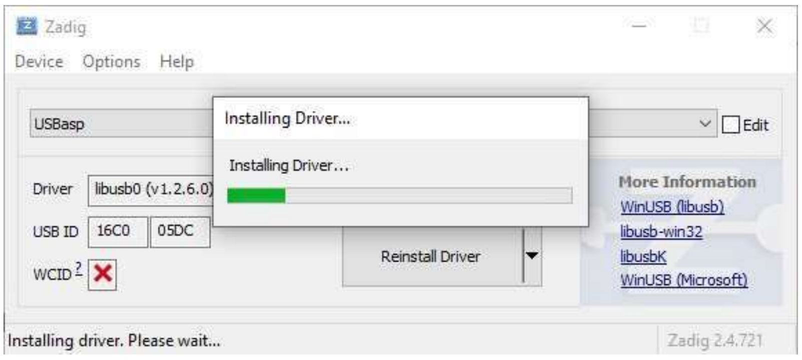
Driver: libusb0 (v1.2.6.0) → libusb-win32 (v1.2.6.0)

USB ID: 16C0 05DC

WCID ?

8 devices found. Zadig 2.4.721

Step 7: Click Reinstall Driver



Zadig

Device Options Help

USBasp

Installing Driver...

Installing Driver...

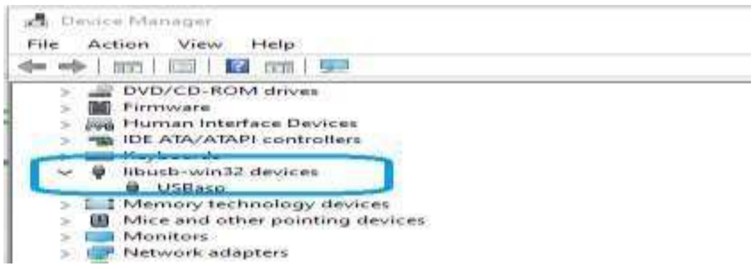
Driver: libusb0 (v1.2.6.0)

USB ID: 16C0 05DC

WCID ?

Installing driver. Please wait... Zadig 2.4.721

Step 8: Check Your Device Manager



Device Manager

File Action View Help

DVD/CD-ROM drives
Firmware
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
libusb-win32 devices
USBasp
Memory technology devices
Mice and other pointing devices
Monitors
Network adapters

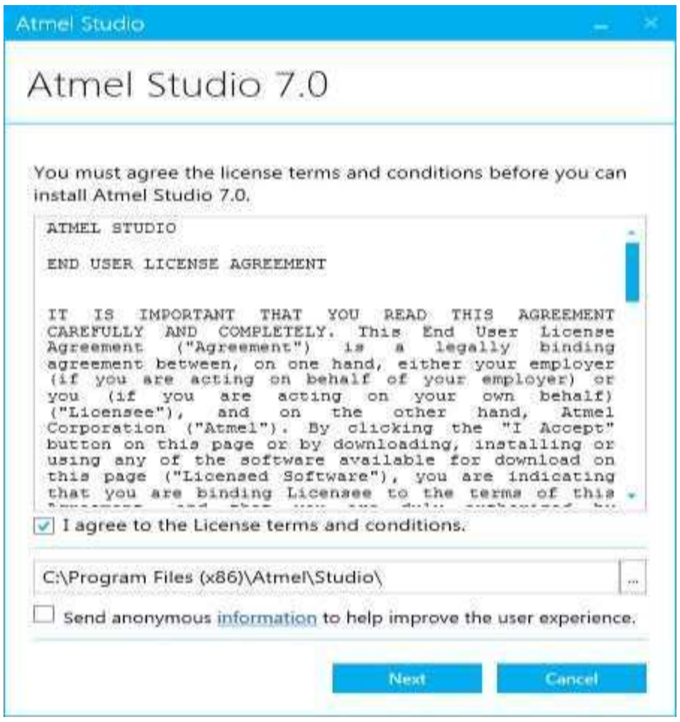
Installer-7.0.2389

1)Download Microchip Studio (Latest):

Microchip Studio for AVR and SAM Devices 7.0.2542 Web Installer

2)Run the Atmel Studio Installer

3)Accept the License Terms. ...


Atmel Studio

Atmel Studio 7.0

You must agree the license terms and conditions before you can install Atmel Studio 7.0.

ATMEL STUDIO

END USER LICENSE AGREEMENT

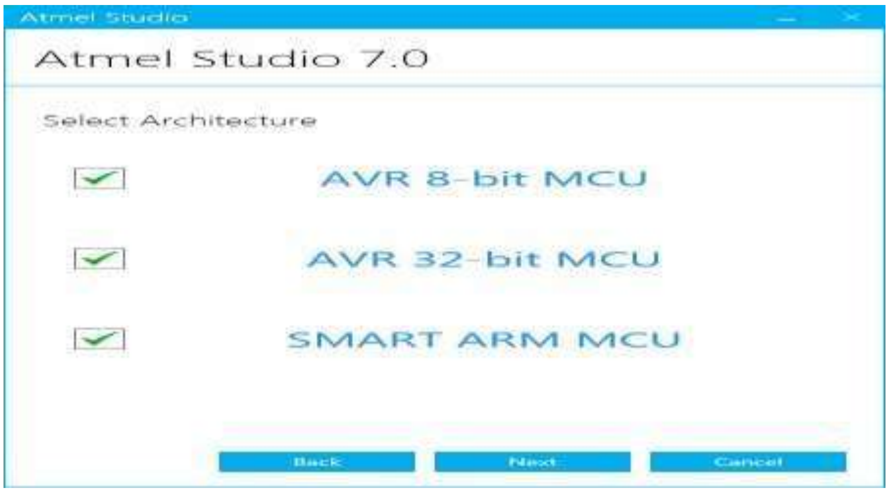
IT IS IMPORTANT THAT YOU READ THIS AGREEMENT CAREFULLY AND COMPLETELY. This End User License Agreement ("Agreement") is a legally binding agreement between, on one hand, either your employer (if you are acting on behalf of your employer) or you (if you are acting on your own behalf) ("Licensee"), and on the other hand, Atmel Corporation ("Atmel"). By clicking the "I Accept" button on this page or by downloading, installing or using any of the software available for download on this page ("Licensed Software"), you are indicating that you are binding Licensee to the terms of this

[✓] I agree to the License terms and conditions.

C:\Program Files (x86)\Atmel\Studio\

[ ] Send anonymous information to help improve the user experience.

4)Select Required Microcontroller Architecture.


Atmel Studio

Atmel Studio 7.0

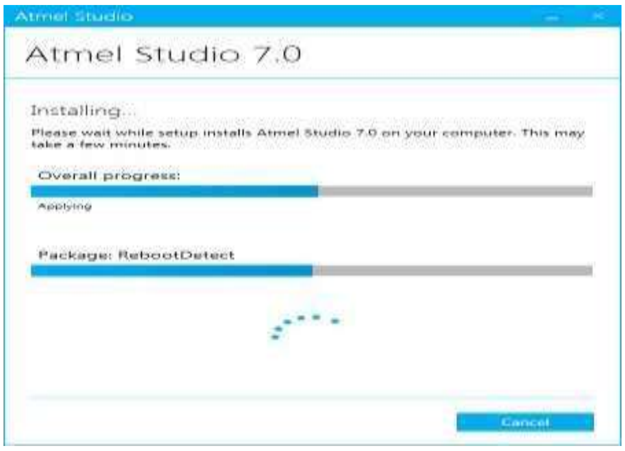
Select Architecture

[✓] AVR 8-bit MCU

[✓] AVR 32-bit MCU

[✓] SMART ARM MCU

1. Software Framework and Example Projects. ...
2. System Validation. ...
3. Installation Started. ...


Atmel Studio

Atmel Studio 7.0

Installing...

Please wait while setup installs Atmel Studio 7.0 on your computer. This may take a few minutes.

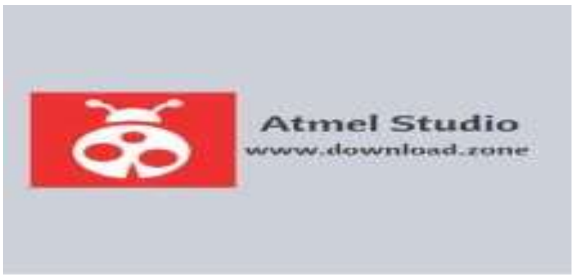
Overall progress:

Applying

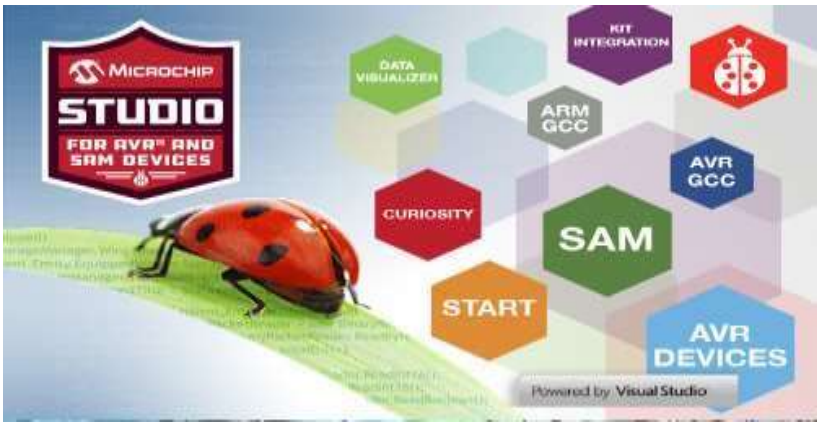
Package: RebootDetect

Cancel

4. System Restart. ...
5. Completing Installation
6. Click following icon



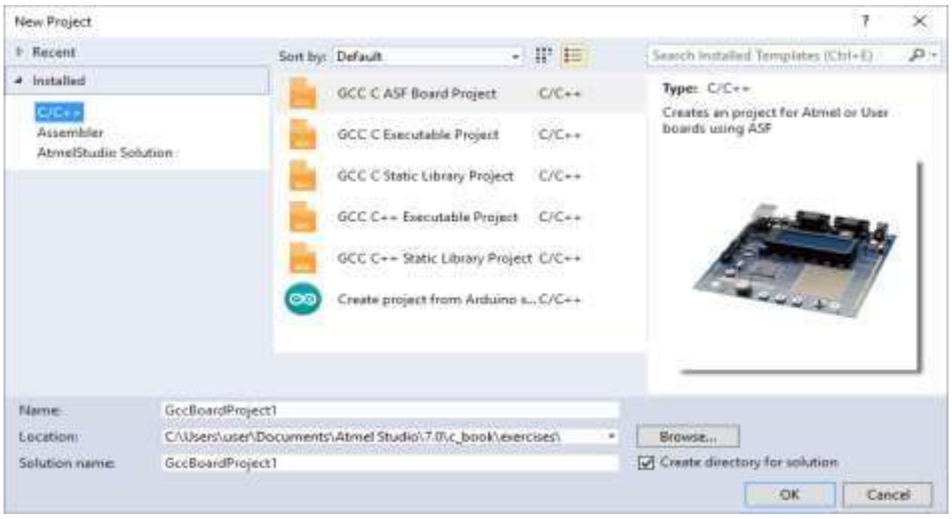
7) Following window will open



8) Select New Project

9) Select GCC C Executable Project   C/C++

10) Select OK



GCC C ASF Board Project C/C++
GCC C Executable Project C/C++
GCC C Static Library Project C/C++
GCC C++ Executable Project C/C++
GCC C++ Static Library Project C/C++
Create project from Arduino s... C/C++

Type: C/C++
Creates an project for Atmel or User boards using ASF

Name: GccBoardProject1
Location: C:\Users\user\Documents\Atmel Studio\7.0\c_book\exercises\
Solution name: GccBoardProject1
☑ Create directory for solution

11) Device Selection

Atmega32A ------→OK

12) Write a program

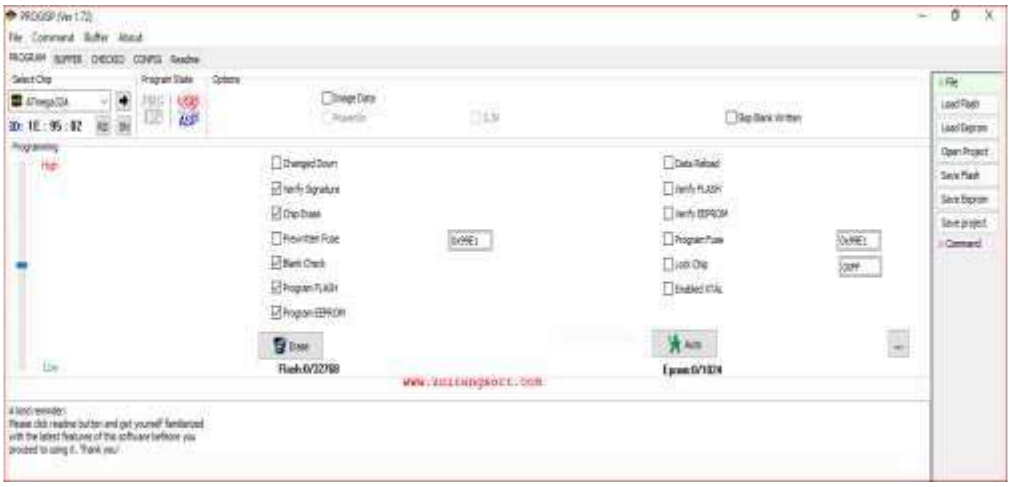
13) Build Solution

Build Succeeded

File is created

14) Go to

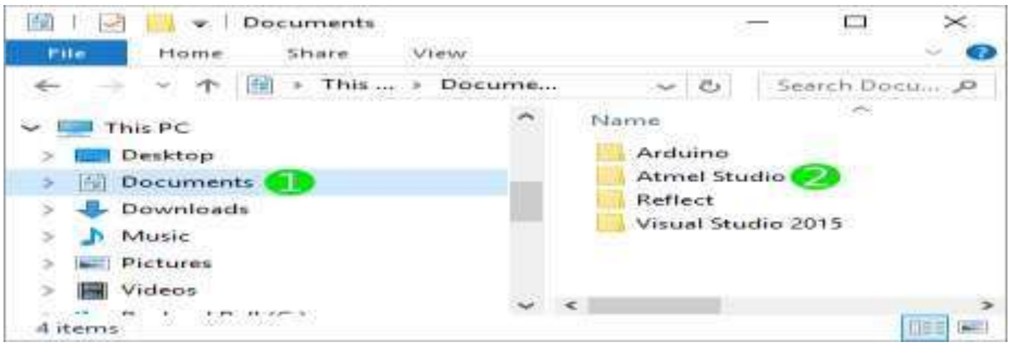
Open progisp(application-577KB) file --→this will provide following window

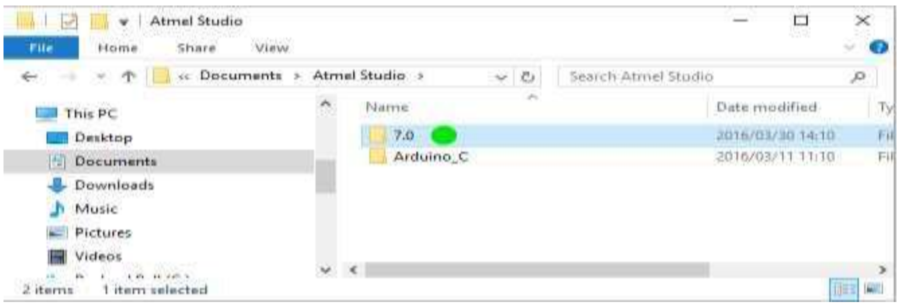


ATmega32A
ID: 1E : 95 : 02
☐ Skip Blank Written
Programming
☐ Changed Down
☑ Verify Signature
☑ Chip Erase
☐ Prewritten Fuse 0x99E1
☑ Blank Check
☑ Program FLASH
☑ Program EEPROM
☐ Data Reload
☐ Verify FLASH
☐ Verify EEPROM
☐ Program Fuse 0x99E1
☐ Lock Chip 0xFF
☐ Enabled XTAL
Flash:0/32768
Eprom:0/1024

A kind reminder:
Please click readme button and get yourself familiarized with the latest features of the software before you proceed to using it. Thank you!

Load program from-→Load Flash--→select your Hex file as follow

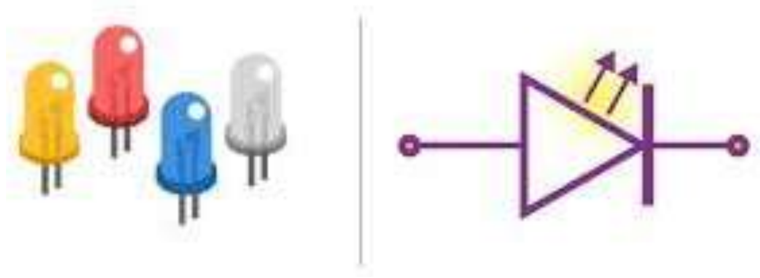




Document---→Atmel Studio7-----→7.0----→Gcc Aplication1---→Gcc Aplication---------→main

LED

It is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased. Remote controls generate invisible light. A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.



These are the applications of LEDs:

- Digital computers and calculators.
- Traffic signals and Burglar alarms systems.
- Camera flashes and automotive heat lamps
- Picture phones and digital watches.

LED Blinking Code

```c
/*
 * ATmega32_LED_Blinking.c

 */

#define F_CPU 1000000UL /* Define CPU frequency here 8MHz */
#include <avr/io.h>
#include <util/delay.h>


int main(void)
{

    DDRC = 0xF0;    /* Make all pins of PORTD as output pins */

    while (1)       /* Blink PORTD infinitely */
    {

        PORTC = 0x00;
        _delay_ms(500); /* Delay of 500 milli second */
        PORTC = 0xF0;
        _delay_ms(500);
    }
}
```

LED chaser Code

```c
/*
 * ATmega32_LED_Blinking.c

 */

#define F_CPU 1000000UL /* Define CPU frequency here 8MHz */
#include <avr/io.h>
#include <util/delay.h>


int main(void)
{

    DDRC = 0b11110000;
    _delay_ms(500);;    /* Make all pins of PORTB as output pins */

    while (1)       /* Blink PORTB infinitely */
    {

    PORTC = 0b10000000;// pin 0 of port c set HIGH
    _delay_ms(100); /* Delay of 100 milli second */
    PORTC = 0b01000000;// pin 1 of port c set HIGH
    _delay_ms(100); /* Delay of 100 milli second */
    _delay_ms(100); /* Delay of 100 milli second */
    PORTC = 0b00100000;// pin 2 of port c set HIGH
    _delay_ms(100); /* Delay of 100 milli second */
    PORTC = 0b00010000;// pin 3 of port c set HIGH
    _delay_ms(100); /* Delay of 100 milli second */

    }
}
```

Buzzer

A **buzzer** is an electronic device that generates sound by converting electrical energy into sound energy. It typically consists of a piezoelectric crystal, which expands and contracts when an alternating current is applied to it, creating sound waves.

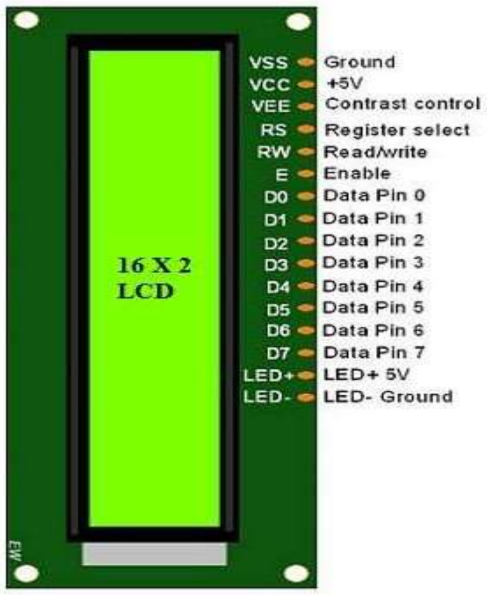
Buzzers are commonly used in a wide range of applications such as alarms, timers, and warning systems. They can also be used in electronic devices such as mobile phones, computers, and other electronic devices to generate different sounds and tones.

Buzzer Code

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main()
{
    PORTD=0x00;
    DDRD=0xfd;
    while(1)
    {
        PORTD=0x00;
        _delay_ms(1000);
        PORTD=0xfd;
        _delay_ms(1000);
    }
}
```

LCD

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.



The features of this LCD mainly include the following.

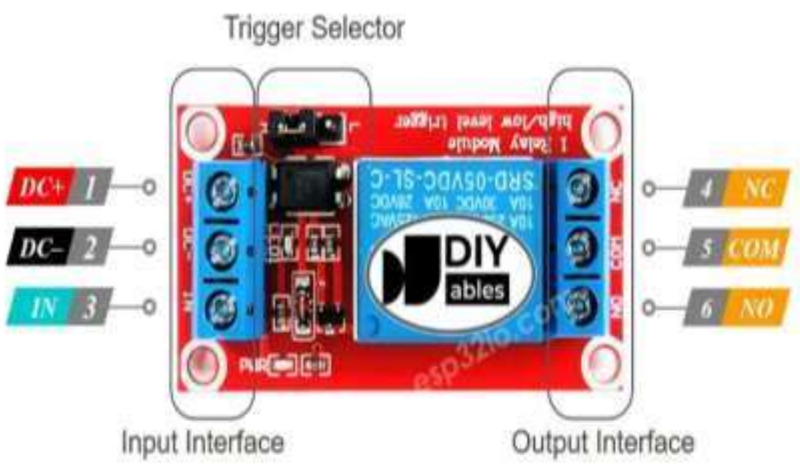
- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit

LCD Code

```
/*
 * GccApplication6.c
 *
 * Created: 12/14/2022 3:32:25 PM
 * Author : Praful
 */

//#include <avr/io.h>


/*
   LCD16x2 4 bit ATmega16 interface
   http://www.electronicwings.com
*/

//#ifndef F_CPU
#define F_CPU 1000000UL // 16 MHz clock speed
//#endif
#define D4 eS_PORTB4
#define D5 eS_PORTB5
#define D6 eS_PORTB6
#define D7 eS_PORTB7
//#define RS eS_PORTB0
//#define EN eS_PORTB2
//#define E eS_PORTB3
#define Back_light PORTB
//#define EN PB2
//#define F_CPU 8000000UL                  /* Define CPU Frequency e.g. here
its 8MHz */
        /* Include inbuilt defined Delay header file */
//#include "lcd.h" //Can be download from the bottom of this article
#include <avr/io.h>                     /* Include AVR std. library file */
#include <util/delay.h>
#define LCD_Dir DDRB                    /* Define LCD data port direction */
#define LCD_Port PORTB                  /* Define LCD data port */
#define RS PB0                          /* Define Register Select (data
reg./command reg.) signal pin */
#define EN PB2                          /* Define Enable signal pin */
#define BL PB3

void LCD_Command( unsigned char cmnd )
{
    LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
    LCD_Port &= ~ (1<<RS);              /* RS=0, command reg. */
    LCD_Port |= (1<<EN);                /* Enable pulse */
    _delay_us(1);
    LCD_Port &= ~ (1<<EN);

    _delay_us(200);

    LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
    LCD_Port |= (1<<EN);
    _delay_us(1);
    LCD_Port &= ~ (1<<EN);
    _delay_ms(2);
}


void LCD_Char( unsigned char data )
{
    LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
    LCD_Port |= (1<<RS);                /* RS=1, data reg. */
    LCD_Port|= (1<<EN);
    _delay_us(1);
    LCD_Port &= ~ (1<<EN);

    _delay_us(200);

    LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
    LCD_Port |= (1<<EN);
    _delay_us(1);
    LCD_Port &= ~ (1<<EN);
    _delay_ms(2);
}

void LCD_Init (void)                    /* LCD Initialize function */
{

    LCD_Dir = 0xFF;                     /* Make LCD command port direction as
o/p */
    _delay_ms(20);                      /* LCD Power ON delay always >15ms */
    LCD_Port |= (1<<BL);
    LCD_Command(0x33);
    LCD_Command(0x32);                  /* send for 4 bit initialization of
LCD  */
    LCD_Command(0x28);                  /* Use 2 line and initialize 5*7
matrix in (4-bit mode)*/
    LCD_Command(0x0c);                  /* Display on cursor off*/
    LCD_Command(0x06);                  /* Increment cursor (shift cursor to
right)*/
    LCD_Command(0x01);                  /* Clear display screen*/
    _delay_ms(2);
    LCD_Command (0x80);                    /* Cursor 1st row 0th position */
}


void LCD_String (char *str)                /* Send string to LCD function */
{
    int i;
    for(i=0;str[i]!=0;i++)                 /* Send each char of string till the
NULL */
    {
        LCD_Char (str[i]);
    }
}

void LCD_String_xy (char row, char pos, char *str) /* Send string to LCD with
xy position */
{
    if (row == 0 && pos<16)
    LCD_Command((pos & 0x0F)|0x80);        /* Command of first row and required
position<16 */
    else if (row == 1 && pos<16)
    LCD_Command((pos & 0x0F)|0xC0);        /* Command of first row and required
position<16 */
    LCD_String(str);                       /* Call LCD string function */
}

void LCD_Clear()
{
    LCD_Command (0x01);                    /* Clear display */
    _delay_ms(2);
    LCD_Command (0x80);                    /* Cursor 1st row 0th position */
}

int main()
{

    LCD_Init();                            /* Initialization of LCD*/

    LCD_String("SSG");            /* Write string on 1st line of LCD*/
    LCD_Command(0xc0);                     /* Go to 2nd line*/
    LCD_String("Hello World");             /* Write string on 2nd line*/
    while(1);
```

Relay

Relays are the switches that aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energized with the open contact. However, if it is closed (NC), the relay isn't energized given the closed contact. However, when energy (electricity or charge) is supplied, the states are prone to change.



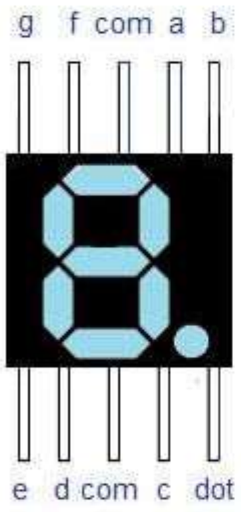
Relays are normally used in the control panels, manufacturing, and building automation to control the power along with switching the smaller current values in a control circuit. However, the supply of amplifying effect can help control the large amperes and voltages because if low voltage is applied to the relay coil, a large voltage can be switched by the contacts.

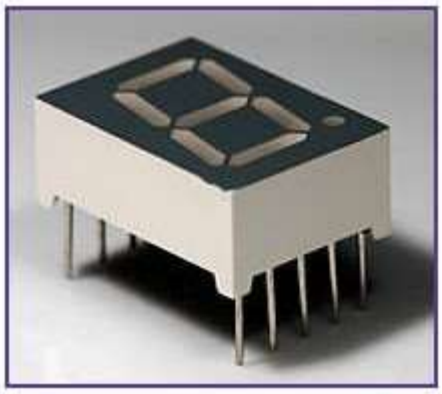
Relay Code

```c
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>


int main()
{
    PORTB=0x0f;
    DDRB=0x0b;
    while(1)
    {
        PORTB=0x00;
        _delay_ms(1000);
        PORTB=0x0b;
        _delay_ms(1000);
    }
}
```

Seven Segment Display

Seven segment displays are important display units in Electronics and widely used to display numbers from 0 to 9. It can also display some character alphabets like A,B,C,H,F,E etc. It's the simplest unit to display numbers and characters. It just consists 8 LEDs, each LED used to illuminate one segment of unit and the 8th LED used to illuminate DOT in 7 segment display. We can refer each segment as a LINE, as we can see there are 7 lines in the unit, which are used to display a number/character. We can refer each line/segment "a,b,c,d,e,f,g" and for dot character we will use "h". There are 10 pins, in which 8 pins are used to refer a,b,c,d,e,f,g and h/dp, the two middle pins are common anode/cathode of all he LEDs. These common anode/cathode are internally shorted so we need to connect only one COM pin.





There are two types of 7 segment displays: Common Anode and Common Cathode:

Common Anode: In this all the Negative terminals (cathode) of all the 8 LEDs are connected together (see diagram below), named as COM. And all the positive terminals are left alone.

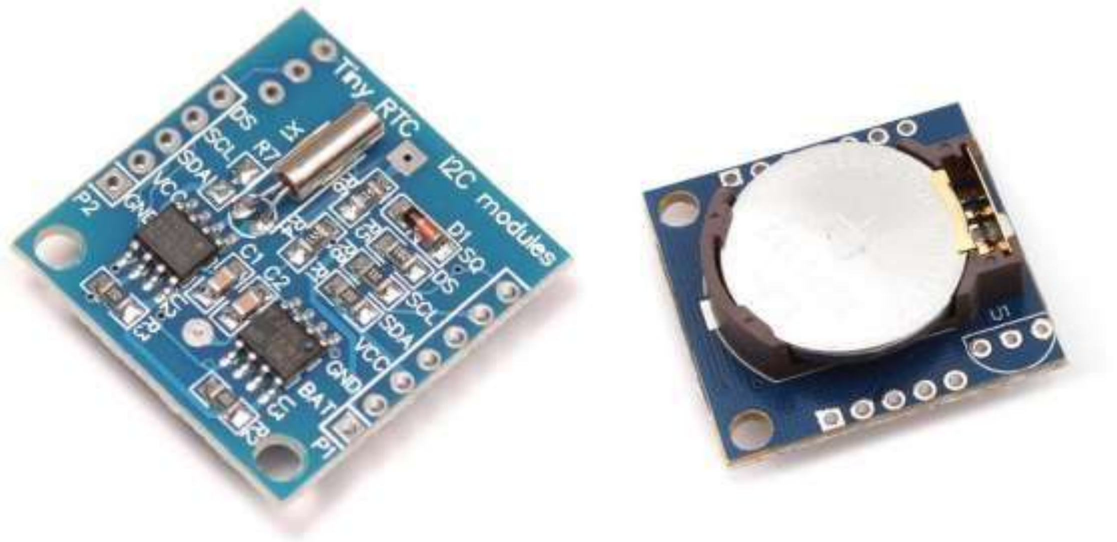
Common Cathode: In this all the positive terminals (Anodes) of all the 8 LEDs are connected together, named as COM. And all the negative thermals are left alone.

Seven Segment Display Code

```c
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define LED_direction DDRC      /* define LED Direction */
#define LED_PORT PORTC          /* define LED PORT */

int main(void)
{
    LED_direction |= 0x0f;      /* define LED port direction is output */
    LED_PORT = 0x0f;
    char array[]={0,1,2,3,4,5,6,7,8,9};
    /* write BCD value for CA display from 0 to 9 */
    while(1)
    {
        for(int i=0;i<10;i++)
        {
            LED_PORT = array[i];/* write data on to the LED port */
            _delay_ms(1000); /* wait for 1 second */
        }
    }
}
```

RTC

- Real Time Clock (RTC) is used to track the current time and date. It is generally used in computers, laptops, mobiles, embedded system applications devices, etc.
- In many embedded systems, we need to put time stamps while logging data i.e. sensor values, GPS coordinates, etc. For getting timestamps, we need to use RTC (Real Time Clock).
- Some microcontrollers like LPC2148, LPC1768, etc., have on-chip RTC. But in other microcontrollers like PIC, and ATmega16/32, do not have on-chip RTC. So, we should use an external RTC chip



Specification of DS1307

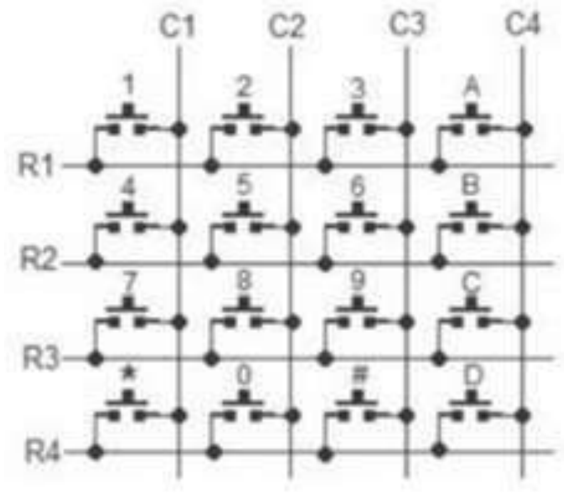
- I2C Interface: Standard I2C interface, with 7-bit addressing
- SRAM Memory: 56 bytes of battery-backed SRAM

RTC Code

```
/*
 * avr_lcd_1.c
 *
 * Created: 10-Oct-23 9:25:00 AM
 * Author : user
 */

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

#include <lcd.h>
#include "clock.h"


void Wait()
{
    uint8_t i;
    for(i=0;i<20;i++)
        _delay_loop_2(0);
}


void write_new_time(void)
{
    LCDClear();
    LCDWriteString("press1toWrtNew");
    LCDWriteStringXY(0,1,"newtime, 2toExit");

    while(1)
    {
        int i;
        i = PINA;
        i = i & 0x0f;
        if (i == 0x07)
        {
            LCDClear();
            LCDWriteString("writing_newTime");
            PORTC=0xef;
            uint8_t hours = 4, minutes = 52, seconds = 0, meridian = 1;//
change these values if you want to reset the time

            SetHour(hours);
            SetMinute(minutes);
            SetSecond(seconds);
            SetAmPm(meridian);
            _delay_ms(3000);
            PORTC=0xff;
            return;
        }
        if (i == 0x0b)
        {
            PORTC=0xdf;
            LCDClear();
            LCDWriteString("SwitchingToRead");
            LCDWriteStringXY(0,1,"mode ... ");
            _delay_ms(3000);
            PORTC=0xff;
            return ;
            //goto back2main;
        }
        _delay_ms(100);
    }

}
int main()
{
    DDRA=0x00;
    PORTA =0xff;
    DDRC=0xff;
    PORTC=0xff;

    //Wait Util Other device startup
    _delay_ms(500);

    //Initialize the LCD Module
    LCDInit(LS_NONE);
    DDRB|=(1<<PB3);
    PORTB|=(1<<PB3);
    LCDWriteString("LCD_initialized!");
    _delay_ms(3000);
    //ClockInit();
    //LCDWriteStringXY(0,1,"clock_init");
    //_delay_ms(1000);
    //Initialize the Clock Module

    if(ClockInit()==0)
    {
        //If we fail to initialize then warn user
        LCDClear();
        LCDWriteString("Error !");
        LCDWriteStringXY(0,1,"DS1307 Not Found");

        while(1); //Halt
    }

    write_new_time();

    char Time[12];  //hh:mm:ss AM/PM

    //Now Read and display time

    while(1)
    {

        GetTimeString(Time);
        LCDClear();
        LCDWriteString("AVR_Rocks!!!");
        LCDWriteStringXY(3,1,Time);

        LCDGotoXY(17,1);

        _delay_ms(500);
    }


}
```

Keypad

we use single I/O pin of a microcontroller unit to read the digital signal, like a switch input. In few applications where 9, 12, 16 keys are needed for input purposes, if we add each key in a microcontroller port, we will end up using 16 I/O ports. This 16 I/O ports are not only for reading I/O signals, but they can be used as peripheral connections too, like ADC supports, I2C, SPI connections are also supported by those I/O pins. As those pins are connected with the switches/keys, we can’t use them but only as I/O ports.



Keypad Code

```
// This code demonstrates the 4x4 keypad array interfaced with ATmega32a �C
at PORTB,
// Since, the output response of the key_presses are being displayed on the
hardwired 7 segment display which is using BCD decoder,
// it can only display the output in proper form upto 0-9 digits, the rest of
the key_responses are displayed in terms of symbols as follows:
/*
            DCBA        BCD_output_responses on 7 seg display
            0000            0
            0001            1
            0010            2
            0011            3
            0100            4
            0101            5
            0110            6
            0111            7
            1000            8
            1001            9
                             _
            1010            |_   (A)
                             _
            1011             _|  (B)

            1100            |_|  (C)
                             _
            1101            |_
                             _   (D)

            1110            |_   (E)
                            |_

            1111                 (F)


*/
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

uint8_t GetKeyPressed()
{
    uint8_t r,c;
    PORTB|= 0XF0;
    for(c=0;c<4;c++)
    {
        DDRB&=~(0XFF);
        DDRB|=(0X01 << c);
        for(r=0;r<4;r++)
        {
            if(!(PINB & (0X10 << r)))
            {
                return (r+(c*4));
            }
        }
    }
    return 0XFF;//Indicate No key pressed
}
int main(void)
{
    DDRC|=((1<<PC0)|(1<<PC1)|(1<<PC2)|(1<<PC3));
    PORTC&=~((1<<PC0)|(1<<PC1)|(1<<PC2)|(1<<PC3));
    char array_1[]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
    uint8_t key;
    while(1)
    {
        key=GetKeyPressed(); //Get the keycode of pressed key
        PORTC=array_1[key];
    }
}
```