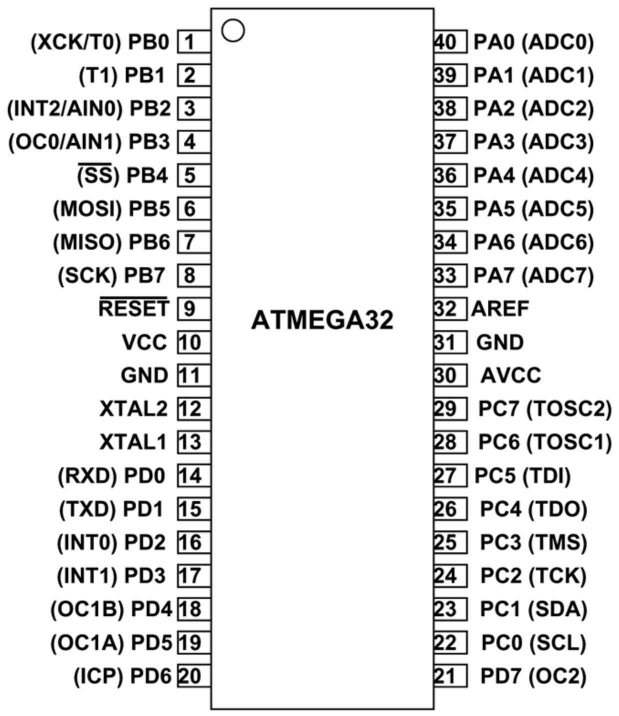
Helwan University
Faculty of Engineering at Helwan
Electronics and Communications Department
Summer Training

Microcontroller Project

Overview:
A microcontroller is a small computer on a single integrated circuit chip. It contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications consisting of various discrete chips.

ATmega32:
The ATmega32 is an 8-bit microcontroller manufactured by Microchip Technology. It is based on the AVR enhanced RISC architecture and has a 32 Kbyte In-System Programmable Flash Program memory, 1024 byte EEPROM, 2 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers and more peripherals.

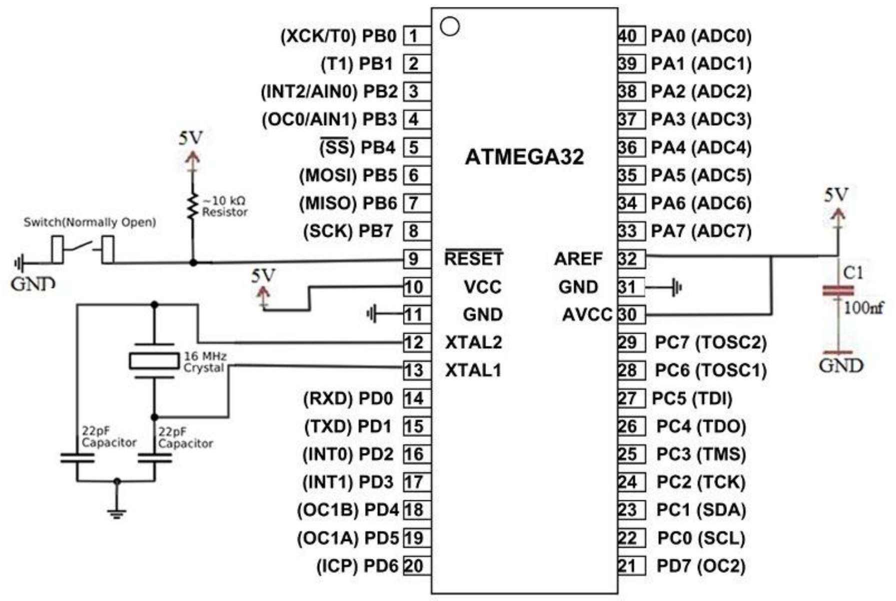
ATmega32 Pin Diagram:



ATmega32 Components:

- 8-bit CPU
- 32 Kbytes Flash Memory
- 1 Kbytes EEPROM
- 2 Kbytes RAM
- Four Parallel 8-bit I/O Ports
- Three Serial I/O Port
- Two 8-bit Timers
- One 16-bit Timer
- 10-bit ADC
- Clock Generator

ATmega32 Power up:



You need to provide the chip with a 2.7 to 5.5V DC voltage source. The 2 most common voltages are 3.3V and 5V. The best way to get a stable 3.3V or 5V voltage is to use a Voltage Regulator. This is a component that basically takes a voltage source of 7 to 12V and converts it down to 5V (or 3.3V) for you.

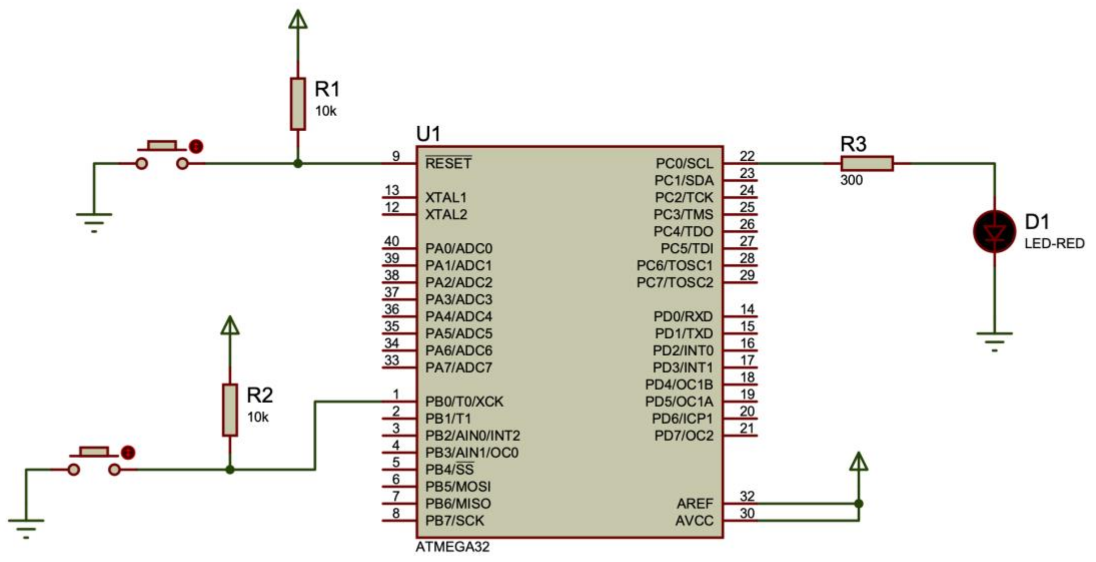
AVCC is the supply voltage pin for the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. The filter is basically the capacitor, C1 in the figure. So, it is recommended to use it just in case you decide to use the ADC.

Crystal Oscillator can be used to provide a clock signal to the ATmega32, or the internal clock source can be used instead.

Project – 1 (Programming I/O Ports)

Task: Control a LED connected to PORTC by pressing switch on PORTB.

Schematic Diagram:



NOTES for Hardware Connections:

- Proteus doesn't include V_{CC} and GND pins of ATMEGA32, you need to connect it manually for breadboard circuit.
- Use Voltage Regulator 7805 to power up the Microcontroller in breadboard.

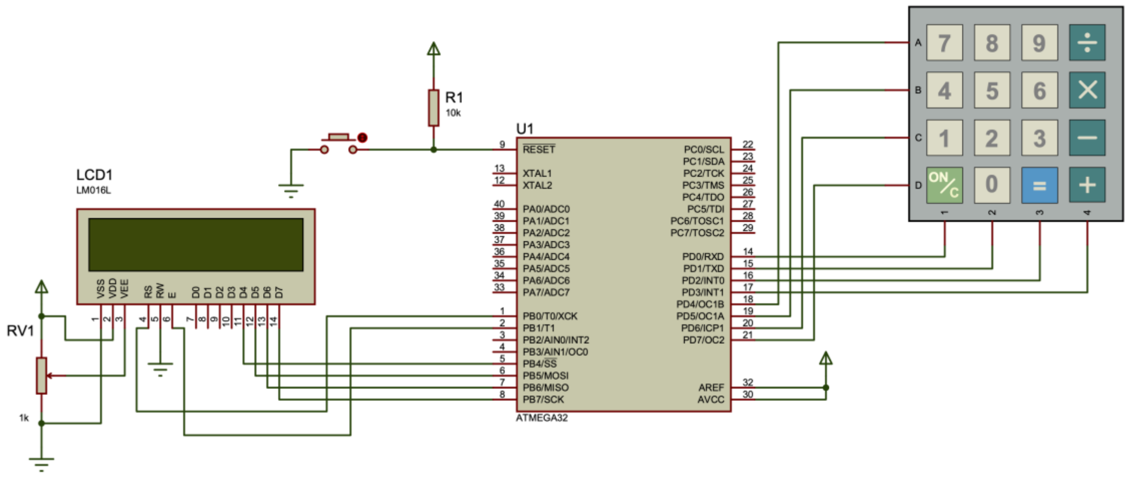
Code: (For Microchip/Atmel Studio)

```
#include <avr/io.h>              // Include avr header file for IO ports

int main(void) {
        unsigned char i;         // temporary variable
        DDRB = 0x00;             // set PORTB as input
        DDRC = 0xFF;             // set PORTC as output

        while (1) {              // Infinite loop
                i = PINB;        // Read input from PORTB
                PORTC = i;       // Send output to PORTC
        }

        return 0;
}
```

Project – 2 (Interfacing keypad with LCD display)

Task: Interface 16x2 LCD and 4x4 keypad with microcontroller to display pressed numbers on LCD.

Schematic Diagram:



NOTES for Hardware Connections:

- Proteus doesn't include V_{CC} and GND pins of ATMEGA32, you need to connect it manually for PCB layout and for breadboard circuit.
- Use Voltage Regulator 7805 to power up the Microcontroller in breadboard and PCB.
- Proteus doesn't include A (Anode) and K (Cathode) pins of LCD, you need to connect it manually for PCB layout and for breadboard circuit. Anode and Cathode pins are used to turn on the backlight led for LCD. A should be connected to V_{CC} and K should be connected to GND.
- Use pin header to represent LCD (16 pins) and keypad (8 pins) in PCB layout.

Code: (For Microchip/Atmel Studio)

```
#define F_CPU 8000000UL            // Define CPU Frequency e.g. here 8MHz
#include <avr/io.h>                // Include avr header file for IO ports
#include <util/delay.h>            // Include Delay header file

#define LCD_Dir  DDRB              // Define LCD data port direction
#define LCD_Port PORTB             // Define LCD data port
#define RS PB0                     // Define Register Select pin
```

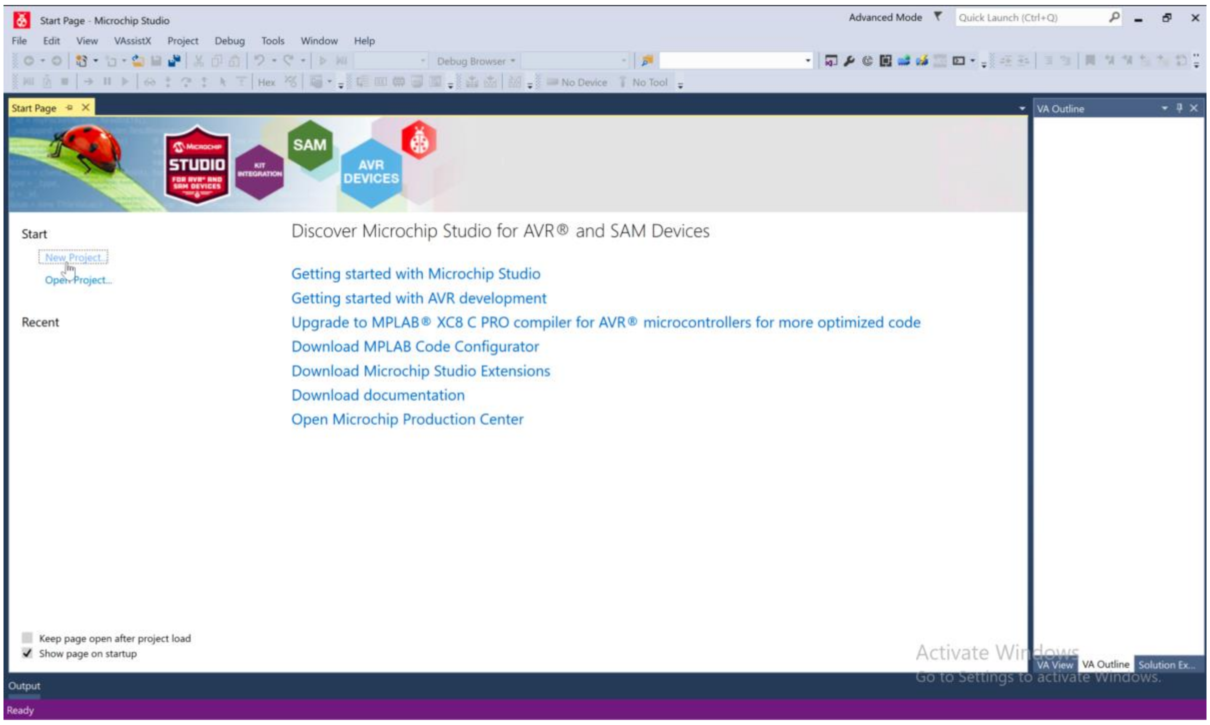
```
#define EN PB1                          // Define Enable signal pin

#define KEY_PRT   PORTD                 // Define keypad output data port
#define KEY_DDR   DDRD                  // Define keypad data port direction
#define KEY_PIN   PIND                  // Define keypad input data port

// Define keypad buttons characters
// TODO: edit characters according to your keypad
unsigned char keypad[4][4] = {  {'7','8','9','/'},
                                {'4','5','6','*'},
                                {'1','2','3','-'},
                                {'c','0','=','+'}
                              };

void LCD_Command(unsigned char cmnd) {
        LCD_Port &= ~ (1<<RS);                          // RS=0, command reg
        _delay_ms(1);
        LCD_Port |= (1<<EN);                            // Enable pulse
        _delay_ms(1);

        LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0);// sending upper nibble
        _delay_ms(1);                                   // delay for processing

        LCD_Port &= ~ (1<<EN);                          // EN=0, command reg
        _delay_ms(1);                                   // delay for processing
        LCD_Port |= (1<<EN);                            // Enable pulse
        _delay_ms(1);                                   // delay for processing

        LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);// sending lower nibble
        _delay_ms(1);                                   // delay for processing
        LCD_Port &= ~ (1<<EN);
        _delay_ms(1);                                   // delay for processing
}

void LCD_Init (void) {                  // LCD Initialize function
        LCD_Dir = 0xFF;                 // Make LCD port direction as o/p
        _delay_ms(20);                  // LCD Power ON delay always >15ms
        LCD_Command(0x02);              // send for 4 bit initialization of LCD
        _delay_ms(20);
        LCD_Command(0x28);              // 2 line, 5*7 matrix in 4-bit mode
        _delay_ms(20);
        LCD_Command(0x0C);          // Display on cursor off
        _delay_ms(20);
        LCD_Command(0x01);          // Clear display screen
        _delay_ms(20);
}

void LCD_Char( unsigned char data ) {
        LCD_Port |= (1<<RS);                            // RS=1, data reg.
        _delay_ms(1);
        LCD_Port|= (1<<EN);
        _delay_ms(1);

        LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0);// sending upper nibble
        _delay_ms(1);                                   // delay for processing

        LCD_Port &= ~ (1<<EN);                          // EN=0, command reg
        _delay_ms(1);                                   // delay for processing
        LCD_Port |= (1<<EN);                            // Enable pulse
        _delay_ms(1);                                   // delay for processing


        LCD_Port = (LCD_Port & 0x0F) | (data << 4); // sending lower nibble
        _delay_ms(1);                                   // delay for processing
        LCD_Port &= ~ (1<<EN);

        _delay_ms(1);                                   // delay for processing
}

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_Clear() {
        LCD_Command (0x01);         // Clear display
        _delay_ms(2);
        LCD_Command (0x80);         // Cursor at home position
}
```

```
unsigned char colloc, rowloc;

char get_pressed_key() {
        while(1) {
                KEY_DDR = 0xF0;            // set port direction as input-output
                KEY_PRT = 0xFF;

                do {
                        KEY_PRT &= 0x0F;  // mask PORT for column read only
                        asm("NOP");
                        colloc = (KEY_PIN & 0x0F); // read status of column
                } while (colloc != 0x0F);

                do {
                        do {
                                _delay_ms(20); // 20ms key debounce time
                                colloc = (KEY_PIN & 0x0F); // read status of column
                        } while(colloc == 0x0F);    // check for any key press

                        _delay_ms (40);             // 20 ms key debounce time
                        colloc = (KEY_PIN & 0x0F);
                } while (colloc == 0x0F);

                // now check for rows
                KEY_PRT = 0xEF;            // check for pressed key in 1st row
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F) {
                        rowloc = 0;
                        break;
                }

                KEY_PRT = 0xDF;            // check for pressed key in 2nd row
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if (colloc != 0x0F) {
                        rowloc = 1;
                        break;
                }
                KEY_PRT = 0xBF;             // check for pressed key in 3rd row
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if(colloc != 0x0F) {
                        rowloc = 2;
                        break;
                }

                KEY_PRT = 0x7F;             // check for pressed key in 4th row
                asm("NOP");
                colloc = (KEY_PIN & 0x0F);
                if(colloc != 0x0F) {
                        rowloc = 3;
                        break;
                }
        }

        if(colloc == 0x0E)
        return(keypad[rowloc][0]);
        else if(colloc == 0x0D)
        return(keypad[rowloc][1]);
        else if(colloc == 0x0B)
        return(keypad[rowloc][2]);
        else
        return(keypad[rowloc][3]);
}

int main(void)
{
        unsigned char key_count = 0;
        _delay_ms(1000);
        LCD_Init();                             // Initialization of LCD
        _delay_ms(1000);
        LCD_Char('T');                          // Display pressed key
        _delay_ms(20);
        LCD_Char('E');                          // Display pressed key
        _delay_ms(20);
        LCD_Char('S');                          // Display pressed key
        _delay_ms(20);
        LCD_Char('T');                                  // Display pressed key
        _delay_ms(2000);
        LCD_Clear();                                    // Clear LCD if c key is pressed
        LCD_Command(0x80);                              // Go to 1st line
        while (1) {
                unsigned char key = get_pressed_key();
                _delay_ms(200);
                key_count++;
                if (key_count == 17) {
                        LCD_Command(0xC0);      // Go to 2nd line
                        } else if (key_count > 32) {
                        LCD_Clear();                    // Clear LCD if c key is pressed
                        LCD_Command(0x80);      // Go to 1st line
                        key_count = 1;
                }
                if (key == 'c') {
                        LCD_Clear();                    // Clear LCD if c key is pressed
                        LCD_Command(0x80);      // Go to 1st line
                        key_count = 0;
                } else {
                        LCD_Char(key);                  // Display pressed key
                }

        }
}
```

Note: LCD and keypad code should be written in separate files, but for simplicity we combined all the files into one file

Microchip/Atmel Studio Project Creation

1. Open Microchip/Atmel Studio and select New Project.

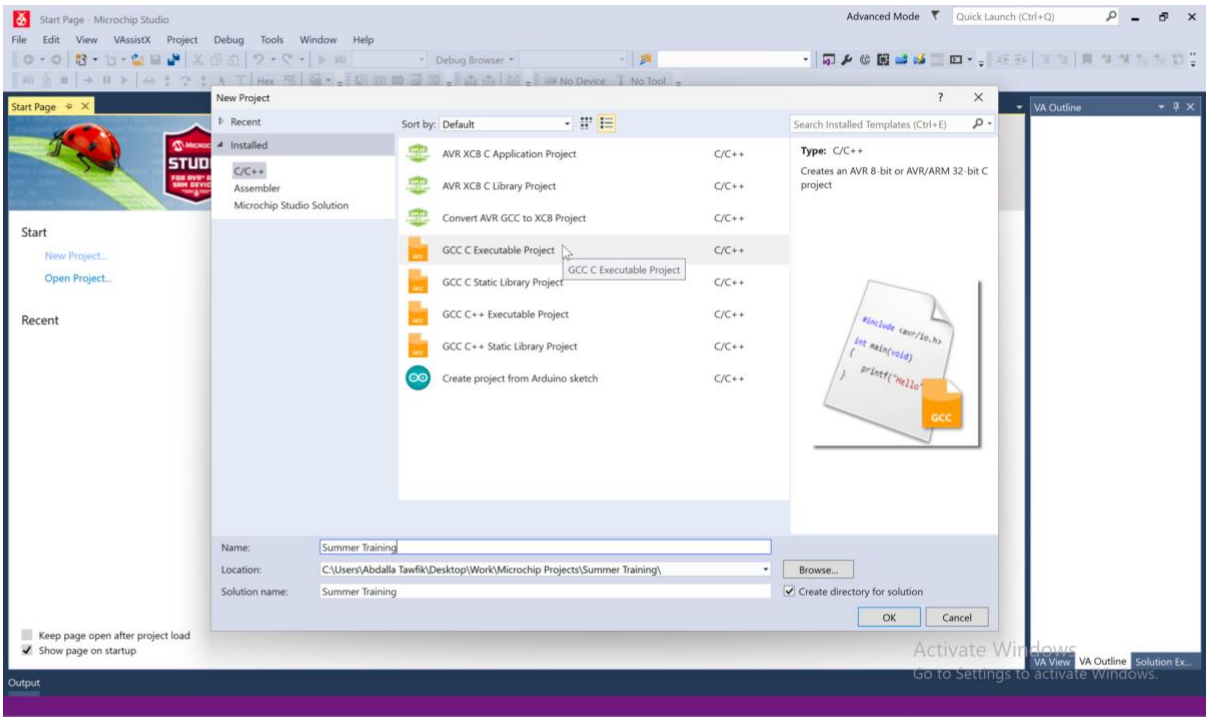


Discover Microchip Studio for AVR® and SAM Devices

Getting started with Microchip Studio
Getting started with AVR development
Upgrade to MPLAB® XC8 C PRO compiler for AVR® microcontrollers for more optimized code
Download MPLAB Code Configurator
Download Microchip Studio Extensions
Download documentation
Open Microchip Production Center

Keep page open after project load
✓ Show page on startup

2. Choose **GCC C Executable Project (C/C++)** and edit the project name and location.

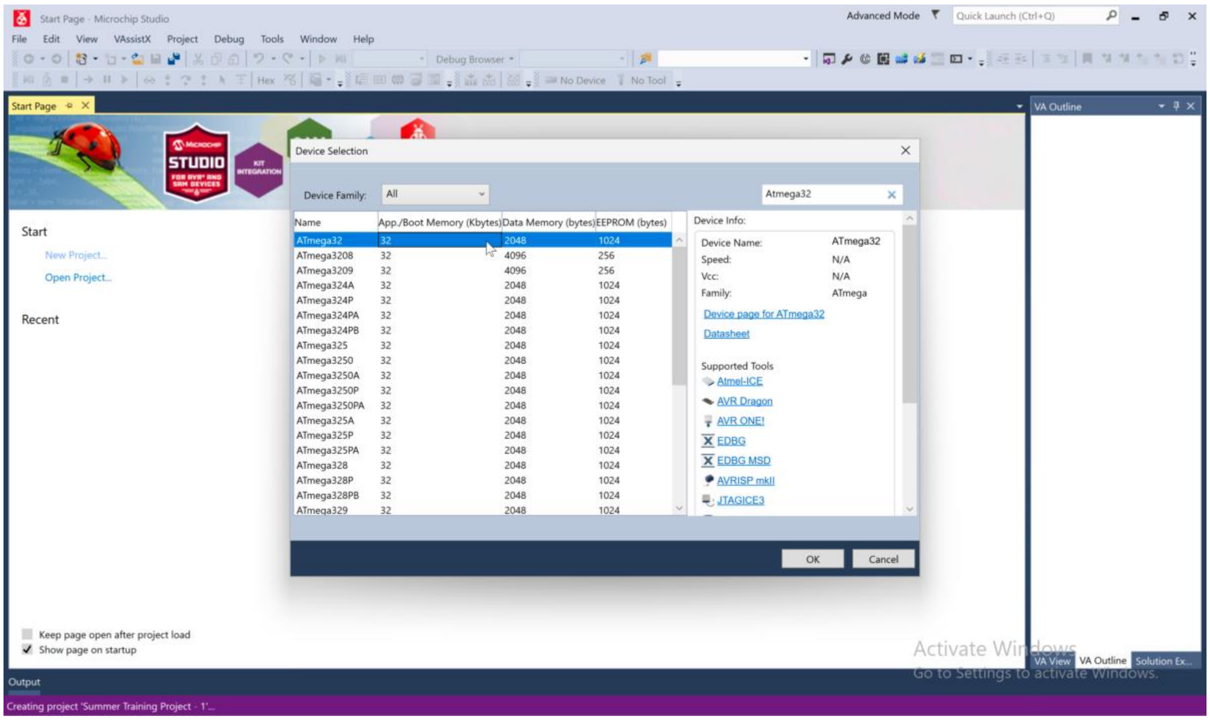


AVR XC8 C Application Project	C/C++
AVR XC8 C Library Project	C/C++
Convert AVR GCC to XC8 Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
Create project from Arduino sketch	C/C++

Type: C/C++
Creates an AVR 8-bit or AVR/ARM 32-bit C project

Name: Summer Training
Location: C:\Users\Abdalla Tawfik\Desktop\Work\Microchip Projects\Summer Training\
Solution name: Summer Training
✓ Create directory for solution

3. Choose ATmega32 from the devices list (You can use the search bar to find it easily).



Device Selection

Device Family: All

Atmega32

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega32	32	2048	1024
ATmega3208	32	4096	256
ATmega3209	32	4096	256
ATmega324A	32	2048	1024
ATmega324P	32	2048	1024
ATmega324PA	32	2048	1024
ATmega324PB	32	2048	1024
ATmega325	32	2048	1024
ATmega3250	32	2048	1024
ATmega3250A	32	2048	1024
ATmega3250P	32	2048	1024
ATmega3250PA	32	2048	1024
ATmega325A	32	2048	1024
ATmega325P	32	2048	1024
ATmega325PA	32	2048	1024
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega328PB	32	2048	1024
ATmega329	32	2048	1024

Device Info:

Device Name:	ATmega32
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Device page for ATmega32

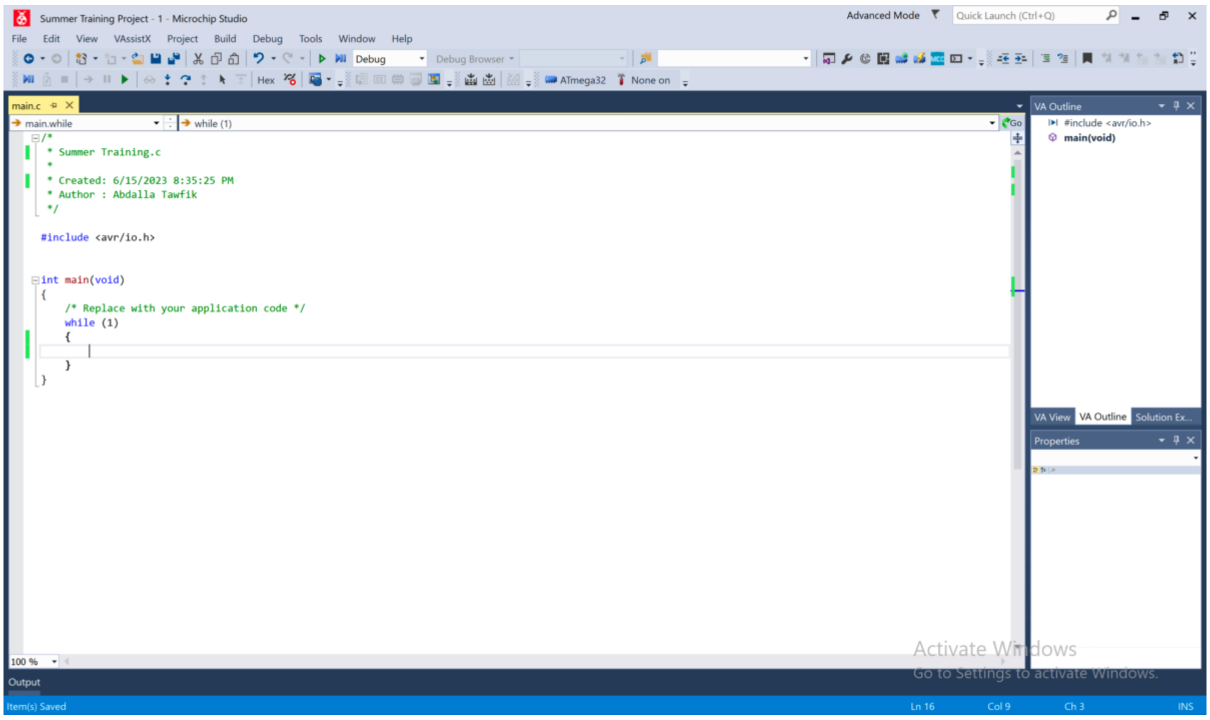
Datasheet

Supported Tools

- Atmel-ICE
- AVR Dragon
- AVR ONE!
- EDBG
- EDBG MSD
- AVRISP mkII
- JTAGICE3

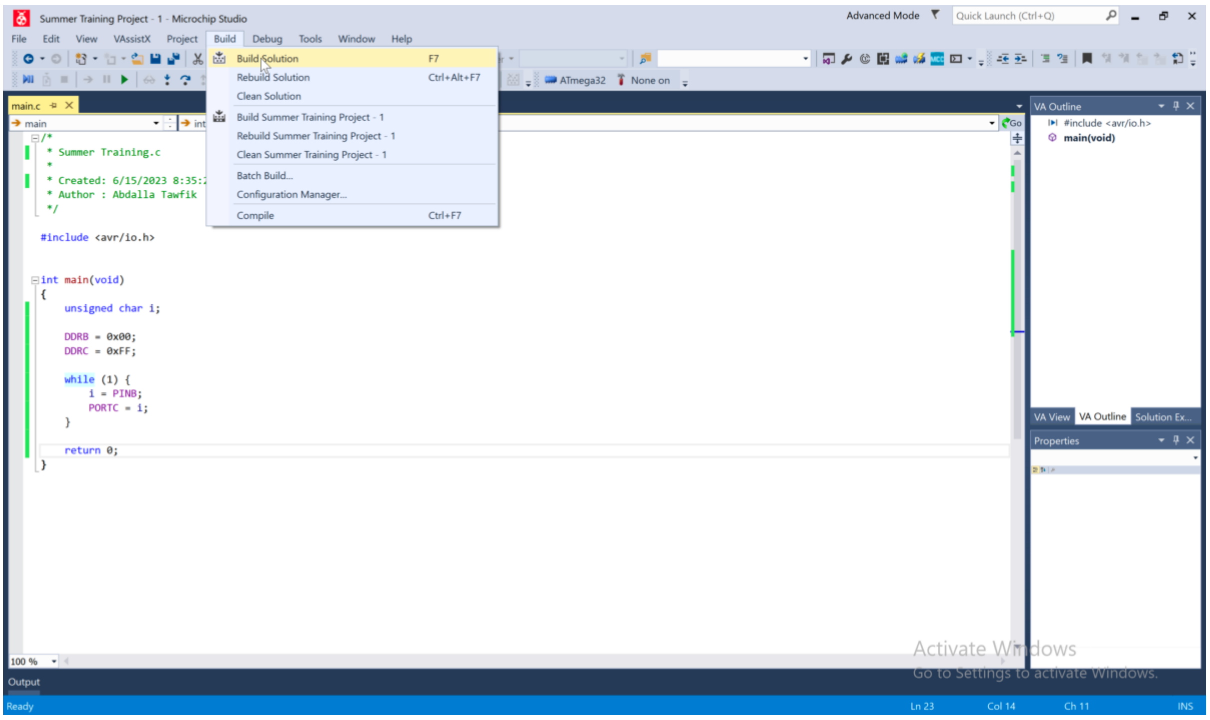
Start Page - Microchip Studio
File Edit View VAssistX Project Debug Tools Window Help
Start Page
Start
New Project...
Open Project...
Recent
Keep page open after project load
Show page on startup
OK
Cancel
Output
Creating project 'Summer Training Project - 1'...
Activate Windows
Go to Settings to activate Windows.

4. The created project should look like this and the main.c file is where you should add your code.



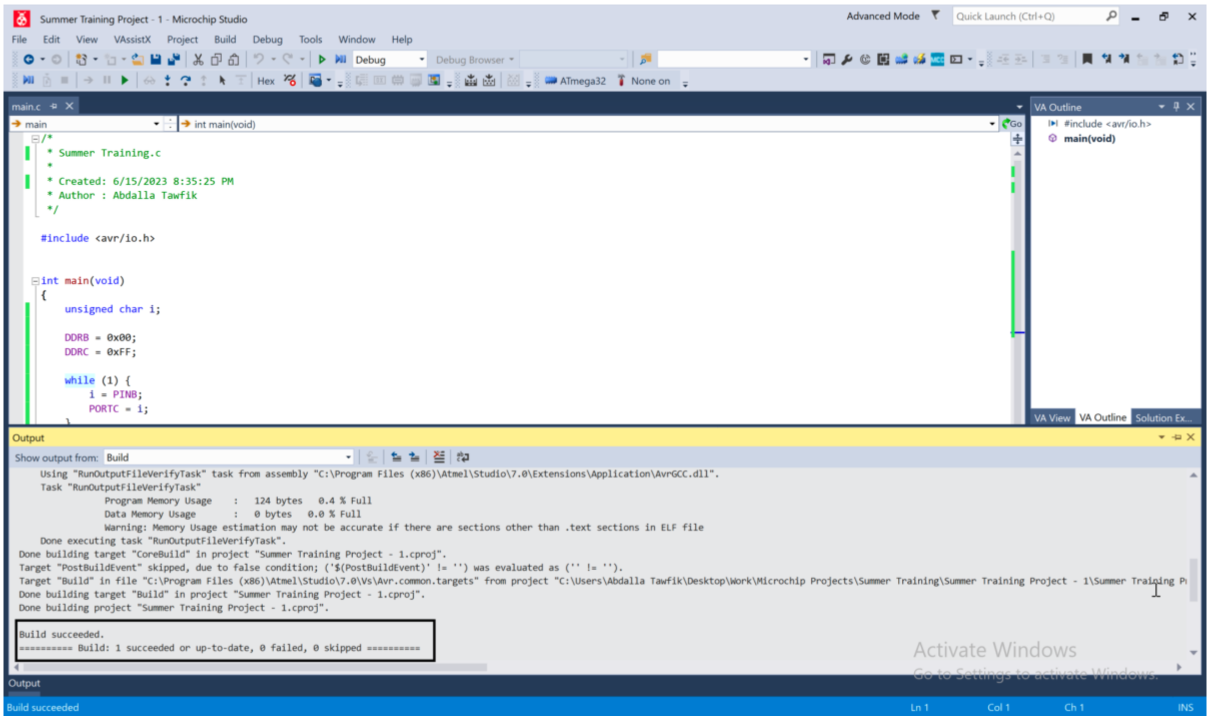
```c
/*
 * Summer Training.c
 *
 * Created: 6/15/2023 8:35:25 PM
 * Author : Abdalla Tawfik
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {

    }
}
```

5. After writing the code choose **Build Solution** from Build menu.



```
/*
 * Summer Training.c
 *
 * Created: 6/15/2023 8:35:2
 * Author : Abdalla Tawfik
 */

#include <avr/io.h>


int main(void)
{
    unsigned char i;

    DDRB = 0x00;
    DDRC = 0xFF;

    while (1) {
        i = PINB;
        PORTC = i;
    }

    return 0;
}
```

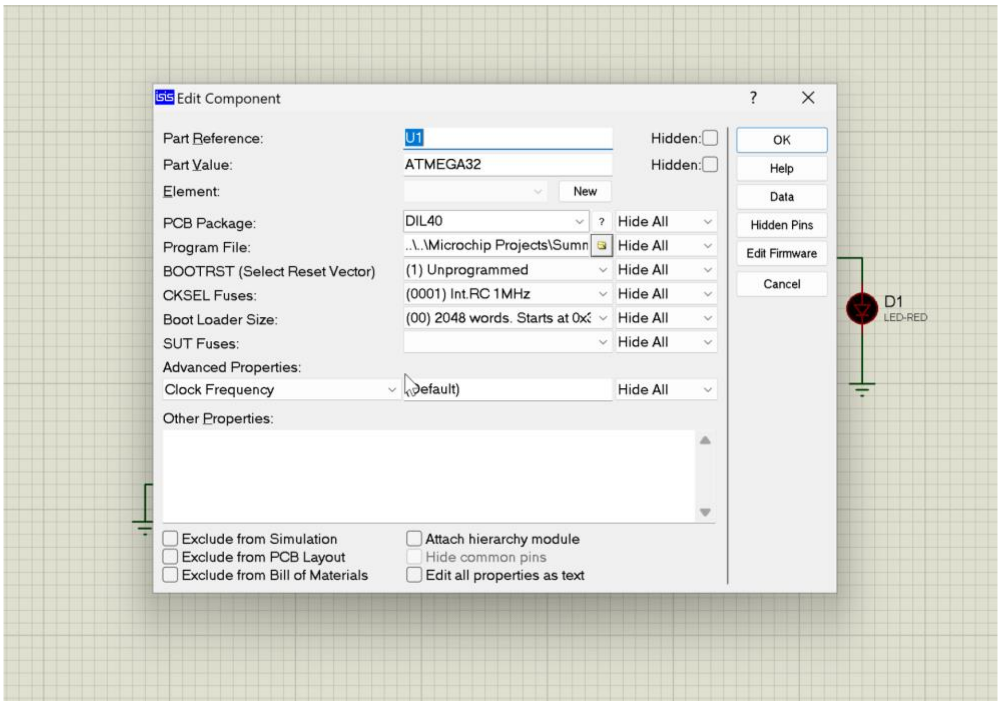
6. Make sure the build is succeeded, then you can find the generated .hex file in {Project Folder}\{Project Name}\Debug folder.



```
/*
 * Summer Training.c
 *
 * Created: 6/15/2023 8:35:25 PM
 * Author : Abdalla Tawfik
 */

#include <avr/io.h>


int main(void)
{
    unsigned char i;

    DDRB = 0x00;
    DDRC = 0xFF;

    while (1) {
        i = PINB;
        PORTC = i;
```

```
    Using "RunOutputFileVerifyTask" task from assembly "C:\Program Files (x86)\Atmel\Studio\7.0\Extensions\Application\AvrGCC.dll".
    Task "RunOutputFileVerifyTask"
            Program Memory Usage    :   124 bytes   0.4 % Full
            Data Memory Usage       :   0 bytes   0.0 % Full
            Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Summer Training Project - 1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Abdalla Tawfik\Desktop\Work\Microchip Projects\Summer Training\Summer Training Project - 1\Summer Training Pr
Done building target "Build" in project "Summer Training Project - 1.cproj".
Done building project "Summer Training Project - 1.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

7. [Proteus] Double click on the ATmega32 microcontroller and in the program file field select the generated .hex file.
   In CKSEL Fuses select Int.RC 1 MHz for project 1 and Int.RC 8 MHZ for project 2.



Edit Component

Part Reference:	U1	Hidden: ☐
Part Value:	ATMEGA32	Hidden: ☐
Element:		New
PCB Package:	DIL40	Hide All
Program File:	..\..\Microchip Projects\Sumn	Hide All
BOOTRST (Select Reset Vector)	(1) Unprogrammed	Hide All
CKSEL Fuses:	(0001) Int.RC 1MHz	Hide All
Boot Loader Size:	(00) 2048 words. Starts at 0x:	Hide All
SUT Fuses:		Hide All
Advanced Properties:		
Clock Frequency	(Default)	Hide All

Other Properties:

☐ Exclude from Simulation ☐ Attach hierarchy module
☐ Exclude from PCB Layout ☐ Hide common pins
☐ Exclude from Bill of Materials ☐ Edit all properties as text

8. [Chip Programming] Use eXtreme Burner – AVR software to program the .hex file to the actual microcontroller.

References:
- AVR Microcontrollers Programming and Applications by Dr. Mohamed Eladawy: https://drive.google.com/open?id=1AbelQWiTPjVwXZvTS8uqvSTBFNyquOok
- Microchip Studio: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio
- eXtreme Burner – AVR: https://extreme-burner-avr.software.informer.com/download/