

**TEAM MEMBERS**

**Mentors:**

* Tirth Jain
* Saniya Kureshi
* Pankaj Borade

**Teammates:**

* Manish Vanjara
* Yash Shah
* Prachi Gupta
* Jinhal Maheshwari
* Aseem Sugandhi
* Arjama Mohanty
* Neeleshwari Rohit

**LED MATRIX MULTIPLEXING**

To display desired patterns in an 8x8 LED matrix controlling with two shift registers.

**INDEX**

| **SR. NO** | **CONTENT** | **PAGE NO.** |
| --- | --- | --- |
| 1. | Introduction | 5 |
| 2. | Description of components | 6 |
| 3. | Shift Register 74HC595 | 12 |
| 4. | Proteus and Atmel Studio | 17 |
| 5. | Timeline of events | 24 |
| 6. | Difficulties faced during event and their solutions | 28 |
| 7. | Circuit Diagram | 33 |
| 8. | Project Code | 34 |
| 9 | Bibliography | 48 |

**INTRODUCTION**

Multiplexing is a technique used to connect devices – typically LEDs (for displays) – in a matrix of addressable rows and columns. The advantage is simplification of hardware due to the reduced number of pins required. Multiplexed displays using seven-segment LEDs remain popular due to low cost and high brightness. In this project, we have used the technique of multiplexing in LEDs using Atmega 32.

For this project, we use two 74HC595 shift registers and connect them to Atmega 32 and an 8x8 LED matrix.These shift registers are meant to control the LEDs. The program is written in Atmel Studio in embedded C language and the virtual circuit is made on Proteus Professional 8.

The display output on the LED is coded in the form of binary or hexa-decimal. Basic looping and delay has been used in the code to generate the desired output on the 8x8 LED matrix. Stationary or moving alphabets or any other designs and patterns can be displayed on the LED matrix through this project guidelines.

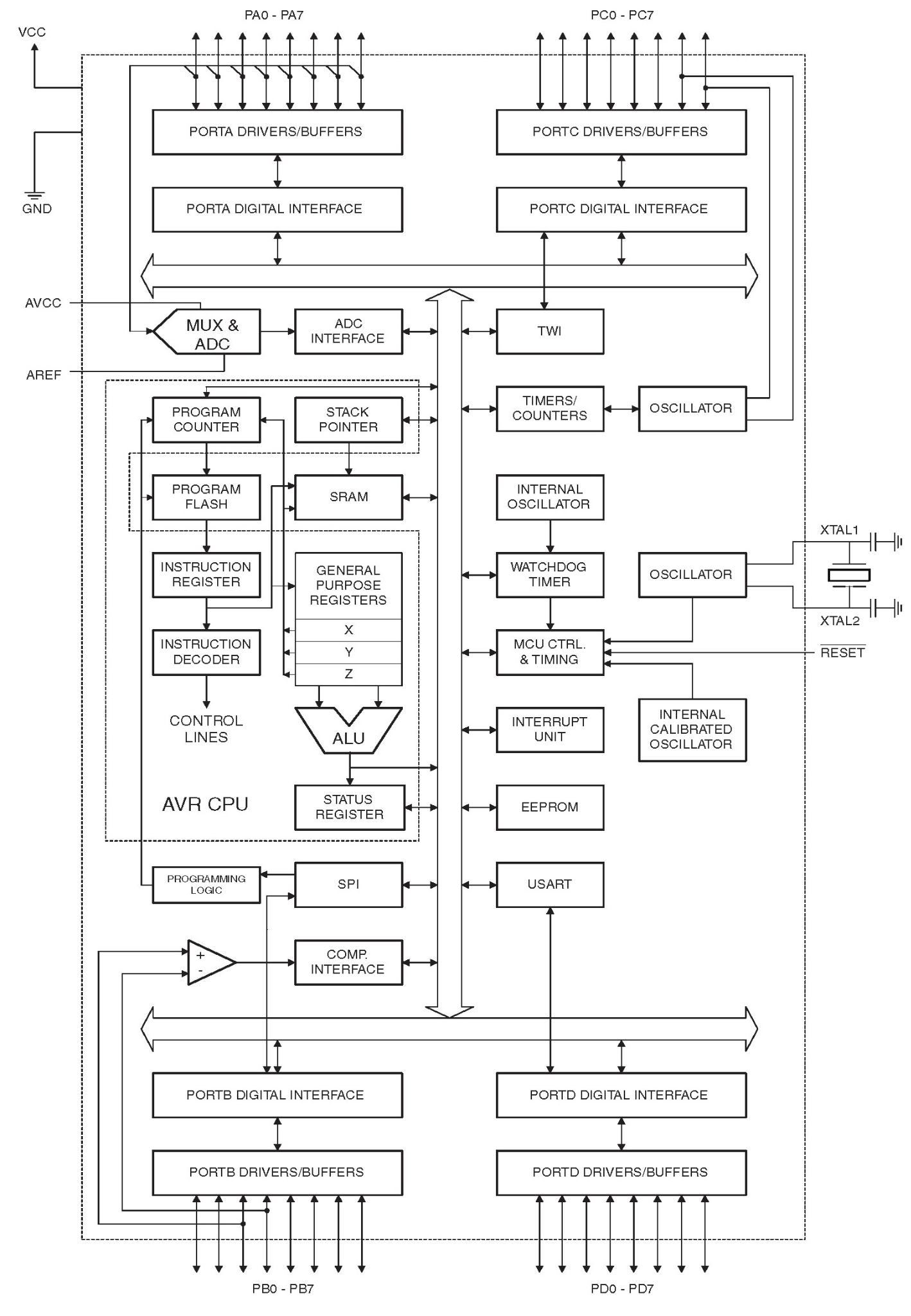
**COMPONENTS REQUIRED AND THEIR DESCRIPTION**

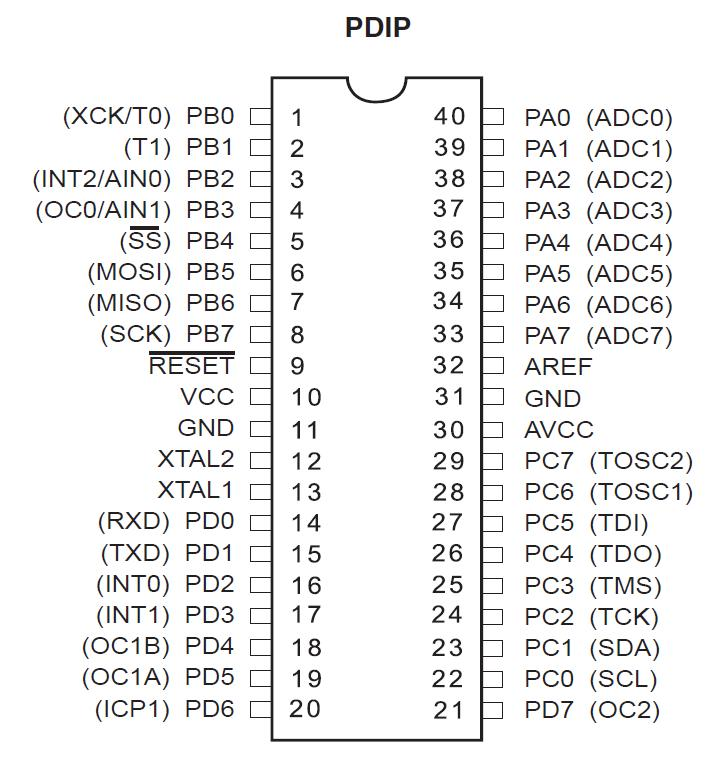
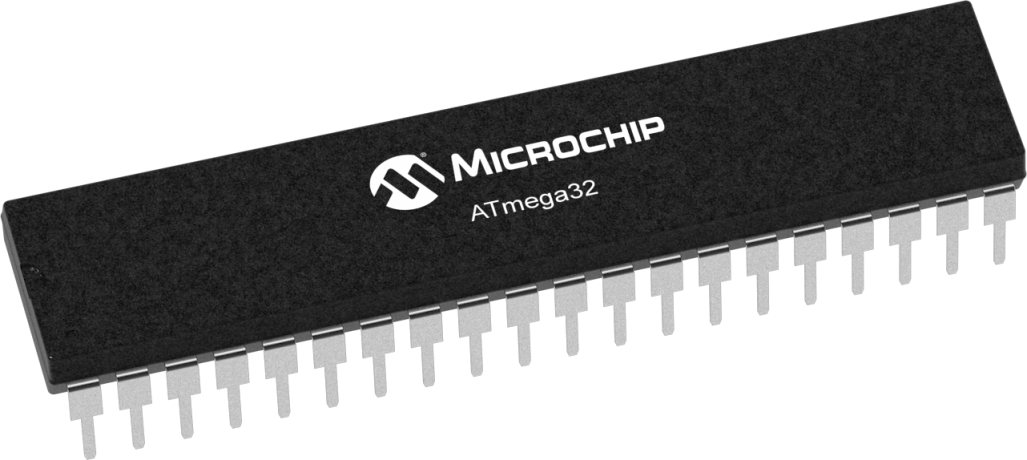
ATMega32

The AVR microController is based on the advanced Reduced Instruction Set Computer (RISC) architecture. ATmega32 microController is a low power CMOS technology based controller. Due to RISC architecture AVR microcontrollers can execute 1 million instructions per second if the cycle frequency is 1 MHz provided by a crystal oscillator.

**Features -**

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

**Functional block diagram -**

**Pin description -**

| vcc | Digital supply voltage. |
| --- | --- |
| GND | Ground. |
| Port A (PA7..PA0) | Port A serves as the analog inputs to the A/D Converter.  Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used.  Port pins can provide internal pull-up resistors (selected for each bit).The Port A output buffers have symmetrical drive characteristics with both high sink and source capability.  When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.  The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running. |
| Port B (PB7..PB0) | Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).  The Port B output buffers have symmetrical drive characteristics with both high sink and source capability.  As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated.  The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.  Port B also serves the functions of various special features of the ATmega32. |
| Port C (PC7..PC0) | Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).  The Port C output buffers have symmetrical drive characteristics with both high sink and source capability.  As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated.  The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.  If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.  The TD0 pin is tri-stated unless TAP states that shift out data are entered.  Port C also serves the functions of the JTAG interface and other special features of the ATmega32 . |
| Port D (PD7..PD0) | Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit).  The Port D output buffers have symmetrical drive characteristics with both high sink and source capability.  As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated.  The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.  Port D also serves the functions of various special features of the ATmega32. |
| RESET | Reset Input.  A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running.  The maximum pulse length is 1.5µs .  Shorter pulses are not guaranteed to generate a reset. |
| XTAL1 | Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. |
| XTAL2 | Output from the inverting Oscillator amplifier |
| AVCC | AVCC is the supply voltage pin for Port A and 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. |
| AREF | AREF is the analog reference pin for the A/D Converter |

Shift register (74HC595):

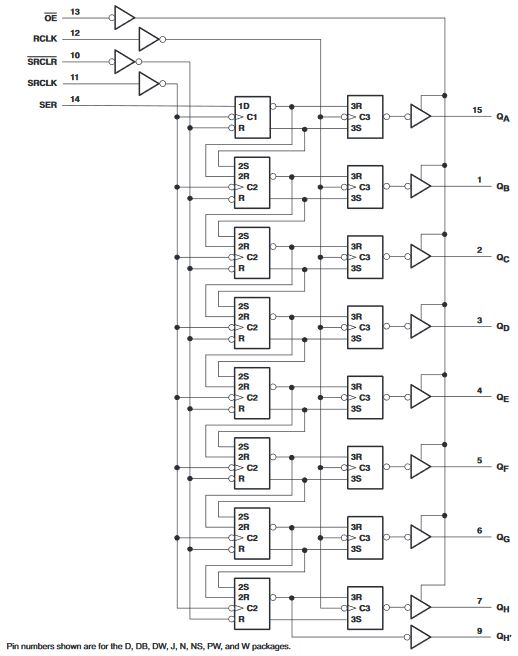
74HC595 is a shift register which works on **Serial IN Parallel OUT** protocol.It receives data serially from the microcontroller and then sends out this data through parallel pins.

Shift registers are often used for the purpose of saving pins on the microcontroller, as every microcontroller has a limited number of I/O pins (GPIO).

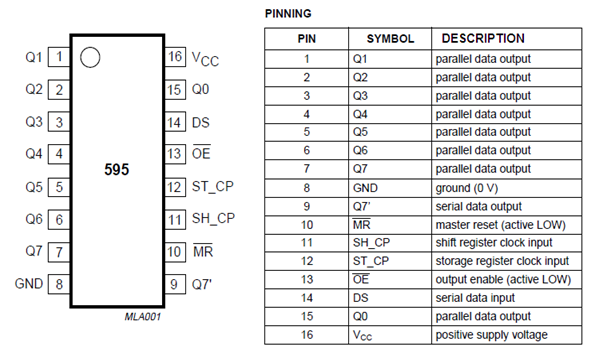
## **How does the 74HC595 Shift Register work?**

* The 595 has two registers (which can be thought of as “memory containers”), each with just 8 bits of data. The first one is called the Shift Register. The Shift Register lies deep within the IC circuits, quietly accepting input.
* Whenever we apply a clock pulse to a 595, two things happen:
* The bits in the Shift Register move one step to the left. For example, Bit 7 accepts the value that was previously in bit 6, bit 6 gets the value of bit 5 etc.
* Bit 0 in the Shift Register accepts the current value on the DATA pin. At the rising edge of the pulse, if the data pin is high, then a 1 gets pushed into the shift register. Otherwise, it is a 0.
* On enabling the Latch pin, the contents of Shift Register are copied into the second register, called the Storage/Latch Register. Each bit of the Storage Register is connected to one of the output pins QA–QH of the IC, so in general, when the value in the Storage Register changes, so do the outputs.

**Functional block diagram-**



**Pin Configuration and Functions -**



8X8 LED Matrix:

The 8×8 LED matrix module is a very useful and low-cost way to add display in your electronic circuits. It has 16 pins to achieve a different combination to ON/OFF the LEDs. It fits properly into any standard solderless breadboard.. It is a Great display for electrical and test equipment. The Exact dimensions of this module are shown in the Schematic image.

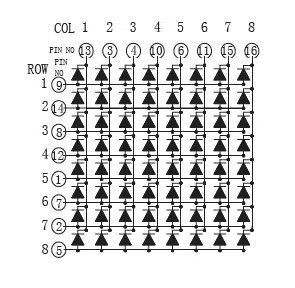
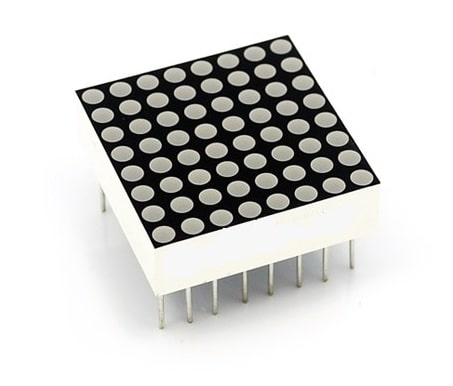
These Displays are always in high demand in the DIY community as well as in Industrial Projects due to their multipurpose uses and cheap cost.

##### **Applications-**

1. Bar-Graph Displays
2. 7-Segment Displays
3. Industrial Controllers
4. Electronic Panel Meters
5. LED Matrix Displays
6. PIXEL gaming.
7. Character design.
8. Measuring instruments.
9. Hobby projects.
10. Display of symbols, simple graphics and texts.

#### **Features-**

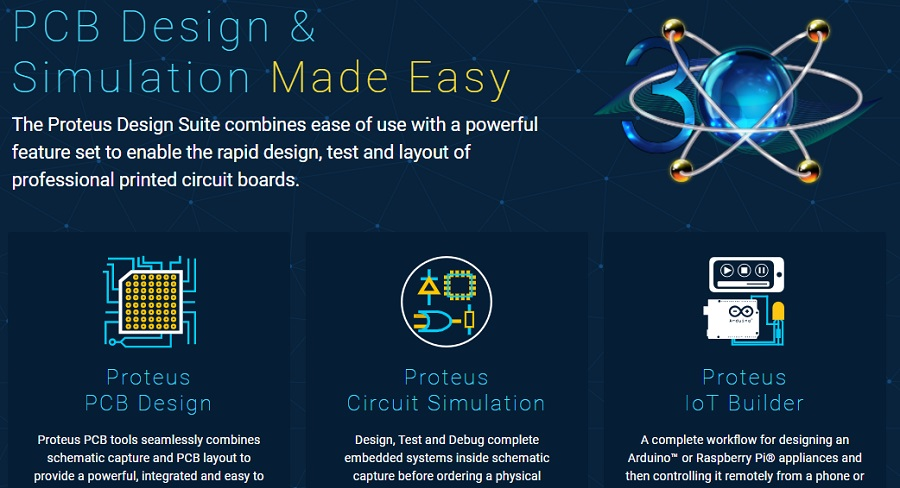
1. Easy to Install on Breadboard.
2. Bright LED lights.
3. Compact Design.
4. LED Size: 3mm.
5. Configuration: Common Anode.

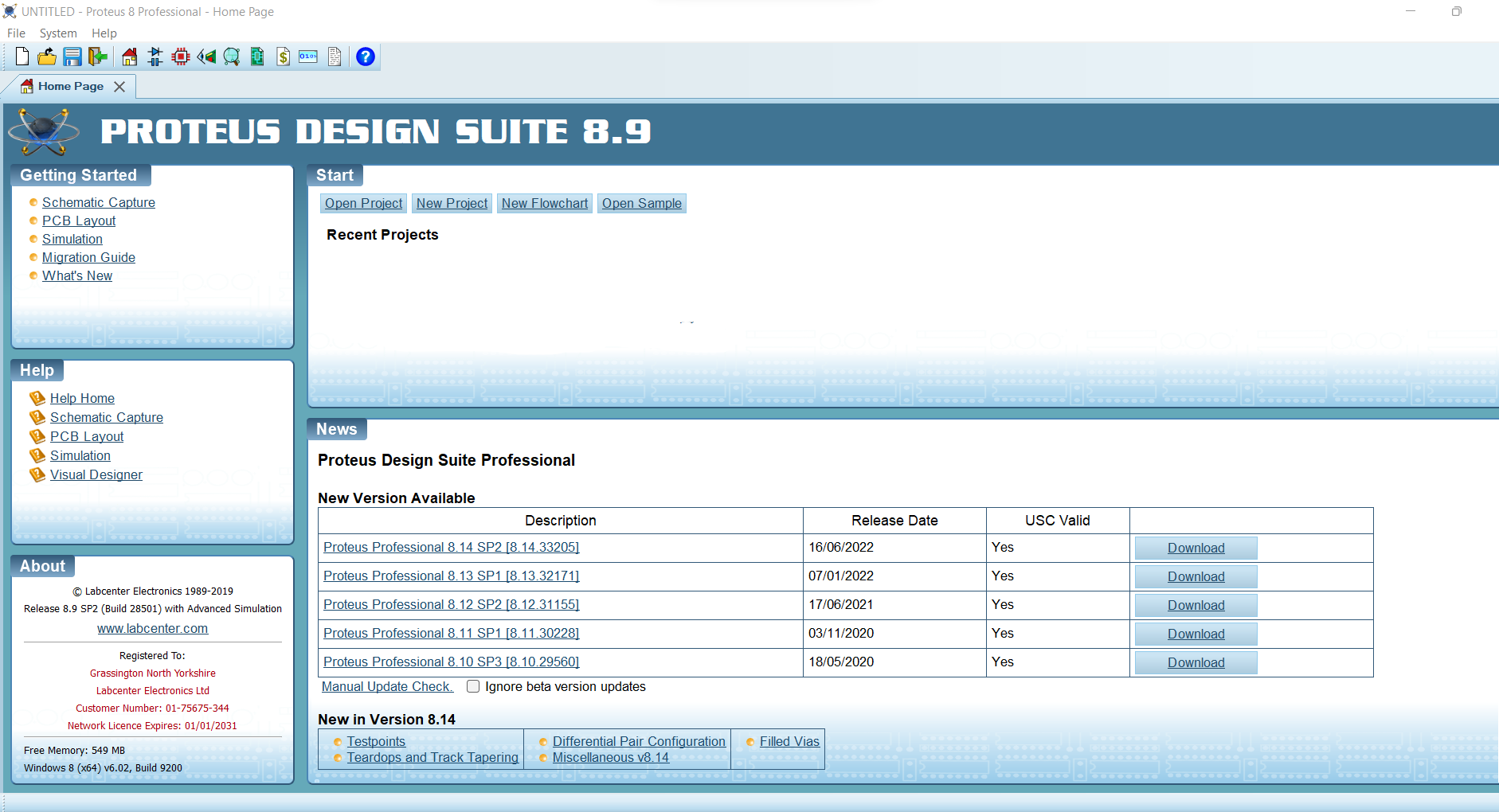
 

**SOFTWARES REQUIRED**

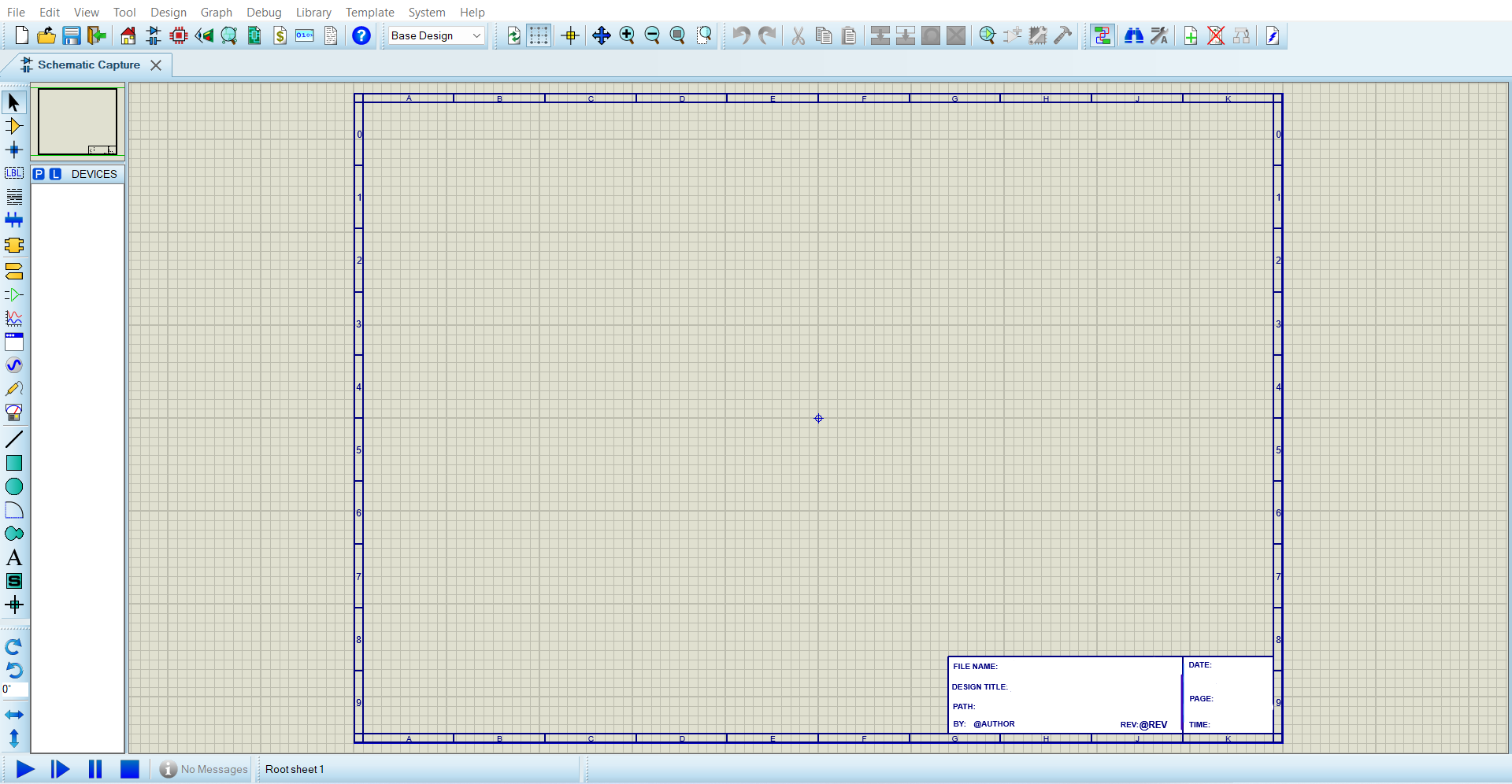
Proteus:

* Proteus is used to simulate, design and draw electronic circuits. It was invented by the **Labcenter electronics.**
* By using proteus you can make two-dimensional circuits designs as well.
* With the use of this engineering software, you can construct and simulate different electrical and electronic circuits on your personal computers or laptops.
* The use of proteus for any electronic circuit project makes that project cost-effective and less error due to schematic construction on the proteus.
* Designing of circuits on the proteus takes less time than practical construction of the circuit.
* The possibility of error is less in software simulation such as loose connection that takes a lot of time to find out connection problems in a practical circuit.
* Circuit simulations provide the main feature that some components of circuits are not practical then you can construct your circuit on proteus.
* There is zero possibility of burning and damaging any electronic component in proteus.
* The electronic tools that are very expensive can easily get in proteus such as an oscilloscope.
* Using proteus you can find different parents of circuits such as current, a voltage value of any component and resistance at any instant which is very difficult in a practical circuit.

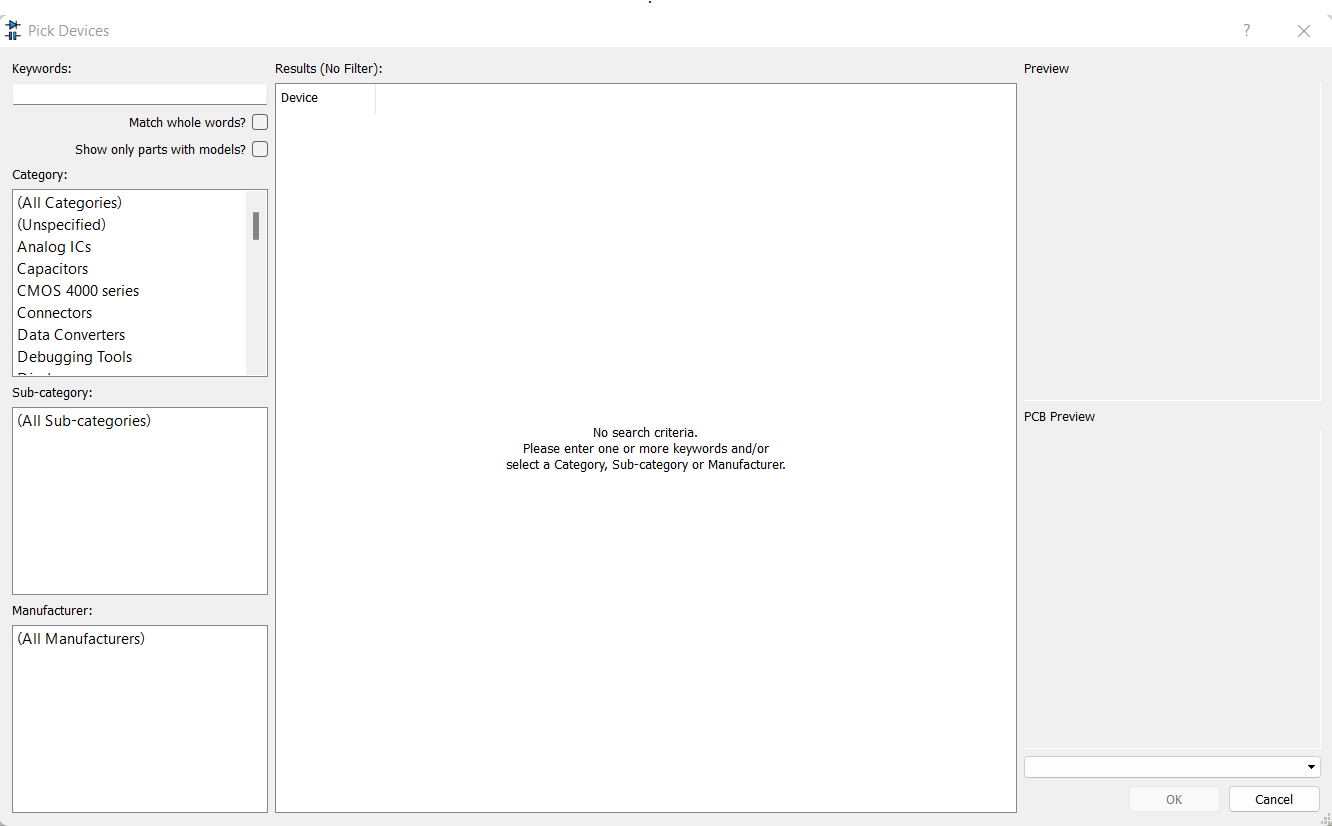
**Images of Proteus**



**Home Page**



**Area of working**



**Window where we can select Devices**

Atmel studio:

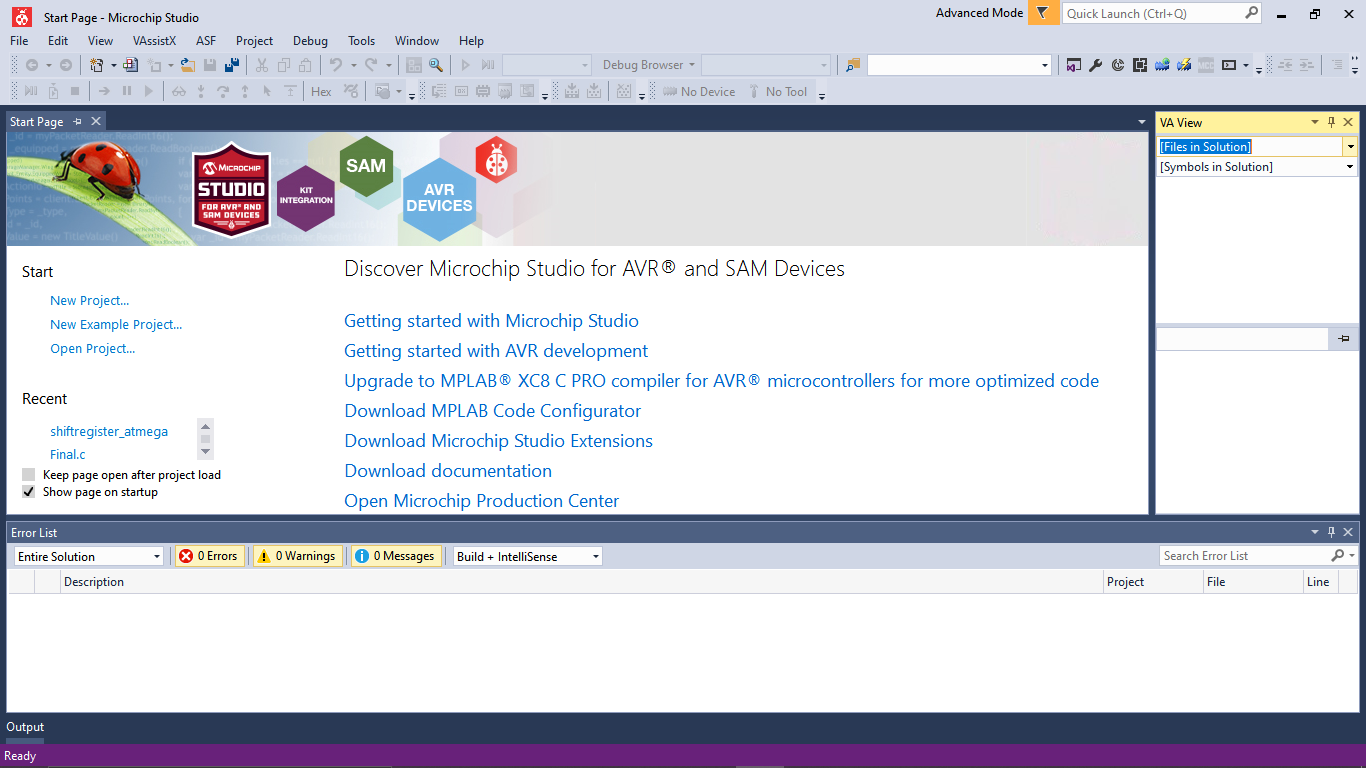
Microchip Studio is an **Integrated Development Environment** (IDE) for developing and debugging AVR® and SAM microcontroller applications. It merges all of the great features and functionality of Atmel Studio into Microchip’s well-supported portfolio of development tools to give you a seamless and easy-to-use environment for writing, building and debugging your applications written in C/C++ or assembly code. Microchip Studio can also import your Arduino® sketches as C++ projects to provide you with a simple transition path from makerspace to marketplace.

You can use Microchip Studio with the debuggers, programmers and development kits that support AVR and SAM devices. Extend your development environment with Microchip Gallery, an online app store for Microchip Studio plug-ins developed by Microchip as well as third-party tool and embedded software vendors.

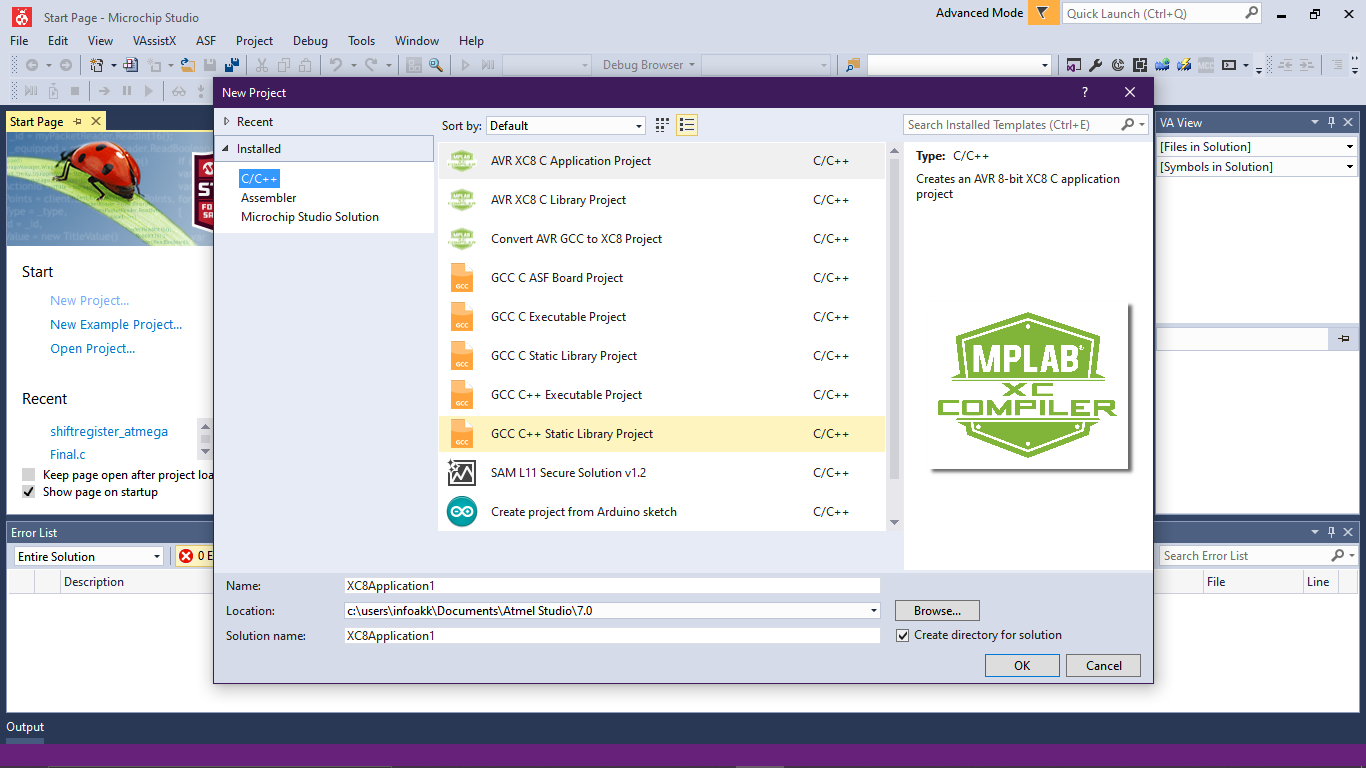
## **Key Features-**

* Support for 500+ AVR and SAM devices.
* MPLAB® XC8 compiler support.
* Vast source code library including drivers, communication stacks, 1,600+ project examples with source code, graphic services and touch functionality through Advanced Software Framework (ASF).
* IDE extensions through Microchip Gallery, the online app store for development tools and embedded software from Microchip and third parties.
* Wireless Composer to configure and test the performance of wireless designs.
* Integrated compiler to write and debug C/C++ and assembly code.
* Advanced debugging features including complex data breakpoints, nonintrusive trace support (SAM3 and SAM4 devices), statistical code profiling, interrupt trace/monitoring, polled data tracing (Arm® Cortex®-M0+ based devices), real-time variable tracking with optional timestamping.
* Integrated editor with visual assist.
* Project wizard that allows projects to be created from scratch or from a large library of design examples.
* In-system programming and debugging that provides an interface to all Microchip in-circuit programmers and debuggers.
* Transparent debug views into CPU and peripherals for easy code development and debugging.
* Full-chip simulation for an accurate model of CPU, interrupts, peripherals and external stimuli.

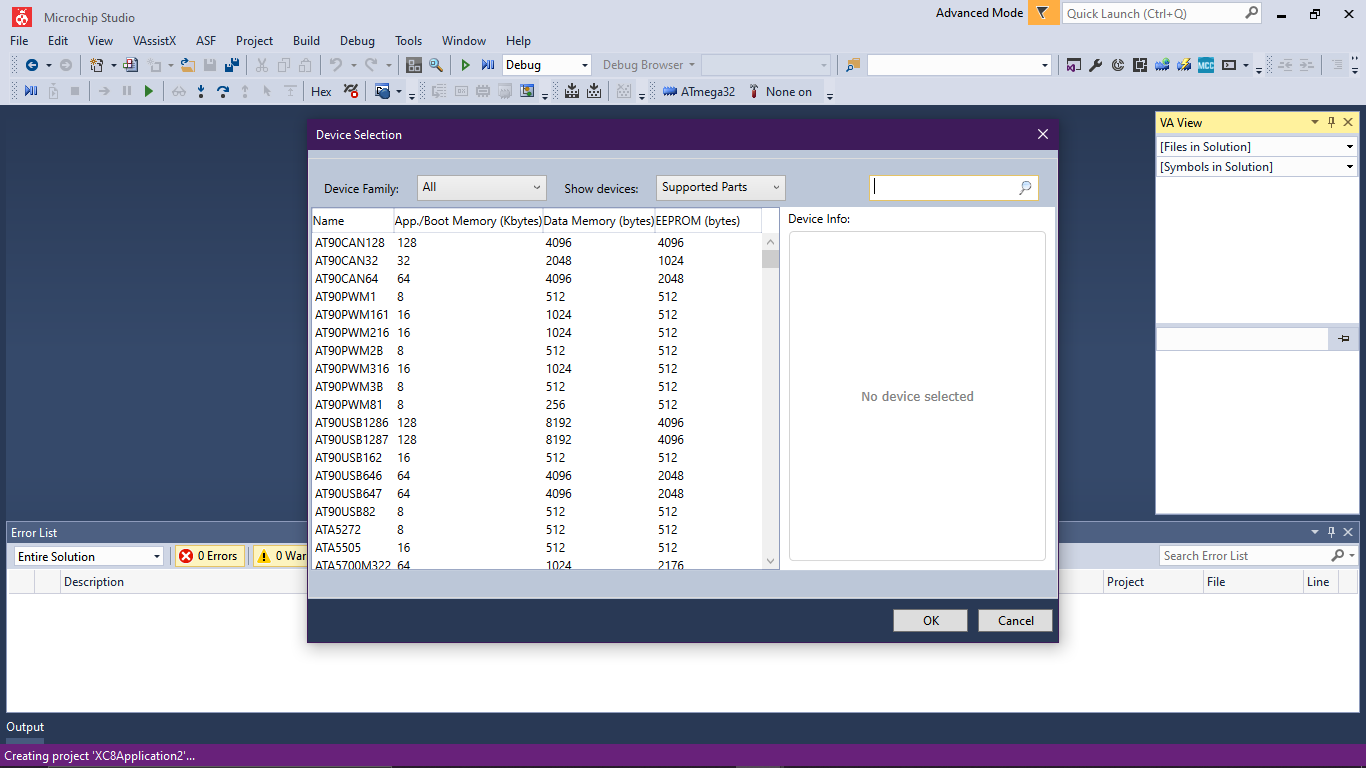




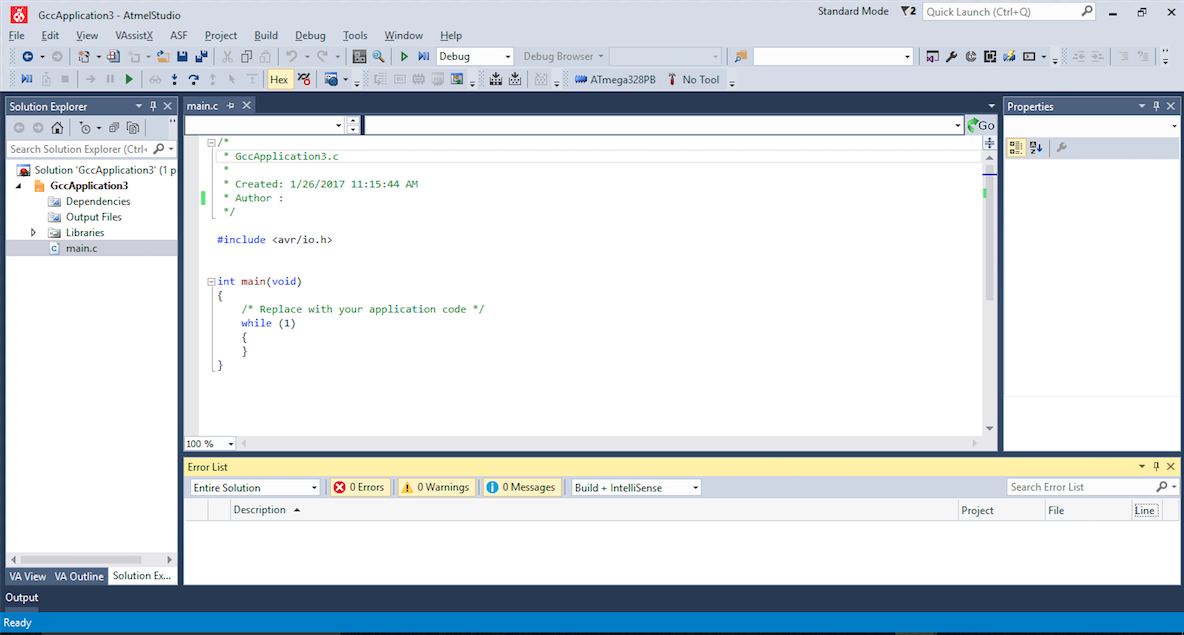
**Homescreen**

****

**Creating new file**

****

**Selecting device to be programmed**



**Working Area**

**TIMELINE OF EVENTS**

* **24th June 2022**

The team members and the mentors had a discussion on Arduino, alternatives methods to increase the number of possible inputs in the circuit. Important discussion on matrix multiplexing and shift register was also done to give a basic idea to the team about the project.

* **25th June 2022**

The team member with the mentor had a discussion about 8-bit shift register, atmega32 and atmega328p. There were also discussions about Difference between Microcontroller and microprocessor and LED brightness if the delay is very small.This way major start was done for the project.

* **26th June 2022**

Doubts regarding proteus installations were solved. We were instructed to go through the datasheet of ATMega32 and get knowledge about pins, ports and DDR.

* **27th June 2022**

We had a discussion on another method of plotting graphs of output waveforms using Oscilloscope and Probes on Proteus.

* **29th June 2022**

We Discussed about internal architecture and its types - Harvard architecture, von-Neumann architecture, RISC architecture.

About bootloader - its code and working, when a computing system is initially powered on, the first piece of code to be loaded and run is the boot loader.

About RAM - SRAM , DRAM; ROM - EEPROM..

Block diagram of atmega32 was discussed in detail which includes the use of different pins for their specific work, from where we get output/input,and the connections made for getting desired outcome.

Bitwise operators types and their use for different types of operations which makes our programming easier and compact which will be more understandable.

Difference between bitwise operators and logical operators was covered.

Information about pin port and ddr in which DDR is used for setting the direction (INPUT or OUTPUT) of gpio pins. PIN is read only and is used for checking the status of any input pin. PORT is used for enabling/ disabling the internal pull-up resistor in case of input pin and it is also used for setting the pin as HIGH/ LOW in case of output pin.

Storage of code in atmega32 by writing a program and storing it in atmega32 for getting desired output according to requirement.

|  |
| --- |

* **30th June 2022**

We discussed and presented our previous task i.e. to blink each led in a loop using Atmega32 and about why to avoid using Port C. We also discussed why DRAM is used in laptops.

* **1st July 2022**

Doubts regarding previous tasks were discussed on blinking led using shift register.

* **3rd July 2022**

Discussion on I2C communication (which is used to send and receive data) was done by the team. The working of SCL and SDA pins was discussed, which help in communication between the master and the slave pins and their functioning on how data is written and read by the pins in the form of 0 and 1.

* **5th July 2022**

The Team discussed the I2C protocol of communication and Half duplex and Full duplex way of communication . We also discussed how to use MAX7219 IC and the workingand the working of the Shift register. There was also a topic about TWI and the connection between it and pin PC0 and PC1.

* **8th July 2022**

We discussed about how to do multiplexing and tried to implement it with help of ATMega32 and 8\*8 LED matrix.We learned how to obtain binary equivalent of any pattern by using code generator.We also looked into a general code which can be used with any desired pattern by just changing values of a single array.

* **9th July 2022**

Evaluation of previous task which was to show moving characters in 8\*8 led matrix multiplexing by giving direct output through atmega32 to 8x8 led matrix was done.The problems faced during the implementation of code in the given task were solved and proper explanation was given. Information regarding the writing of code and use of delay in the proper way to make our multiplexing more effective was helpful in smooth displaying of characters.

* **10th July 2022**

Mentors evaluated our previous task of showing 2 moving characters on the matrix and discussed about the problems faced during the implementation of code in the given task.Former information about presentation was provided by the mentors.We also discussed about Circuit interfacing ATMega32 with 2 shift registers and 8\*8 matrix.

* **11th July 2022**

We performed the task of glowing multiple LED bulbs one-by-one using ATMega32 and shift register by writing the code in Atmel Studio during the meet.The mentor guided the team and clarified their doubts regarding the project.

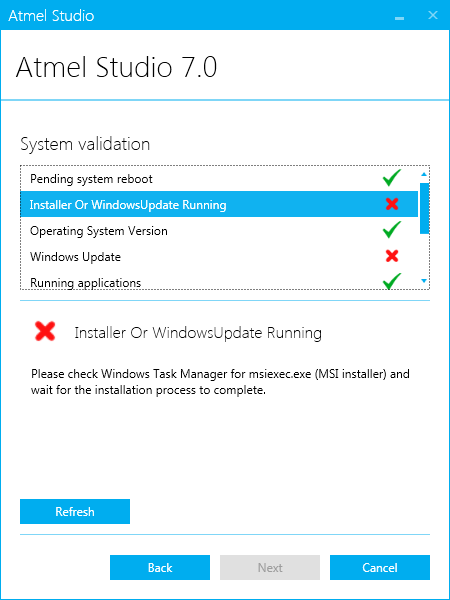
* **13 July 2022**

We were asked to solve problems regarding smoothness in multiplexing of our matrix. Mentors introduced us with another function of multiplexing. Further we had discussion about documentation work.

**DIFFICULTIES FACED DURING EVENT AND ITS SOLUTIONS**

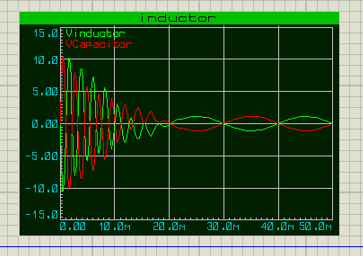
* ***Initially,there were issues in installation of Atmel studio and proteus software.***

Installation of atmel studio software was troubling in window 11, but later on mentors directed us and were able to install the software easily. And installation became easier as appropriate links were shared by the mentors.



* ***In plotting graphs and obtaining waveforms of a simple circuit.***

Getting familiar with atmel studio and proteus software helped us to overcome the problems which were faced by us while making a basic circuit ,writing code for led blinking and so on.

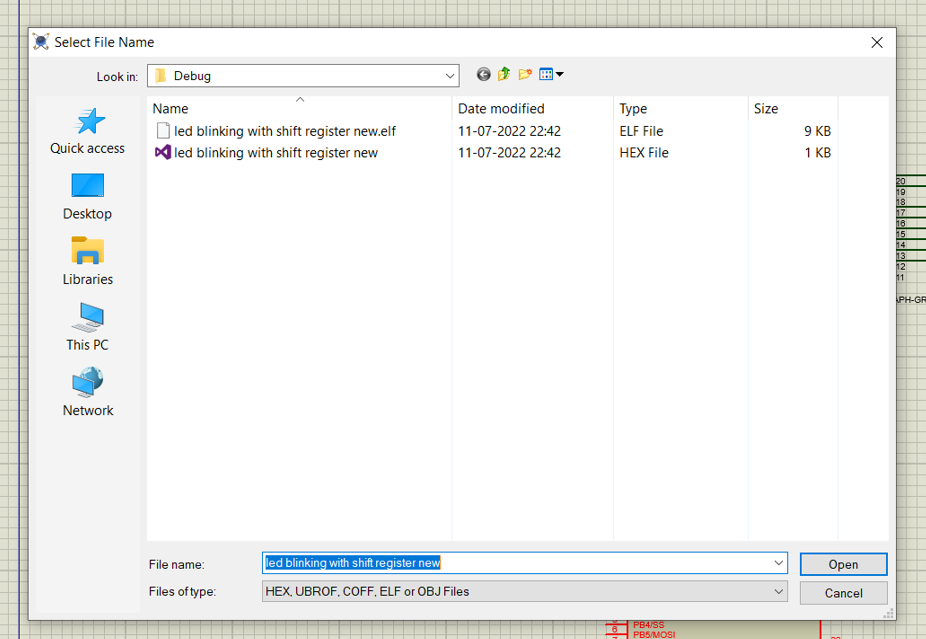


* ***In writing code so as to run our program accordingly to blink one LED consecutively at a time in a series of LEDs.***

We used atmel to write code for our circuit.By understanding bitwise operators we were able to achieve this task efficiently.

* ***In uploading code to ATMega32 from Atmel studio - where we have written our code.***

When we built our code in atmel it generated a hex extension file.In proteus by clicking on atmega 32 and on adding the program file we were able to connect atmel studios and proteus successfully.



* ***To code in atmel studio to blink a series of led by using shift register.***

We understood the working of shift registers and referred materials available on the internet. After reading that we were able to write a code which provided input to the shift register.

* ***In understanding the use of shift register 74HC595 over MAX7219 IC in led matrix multiplexing.***

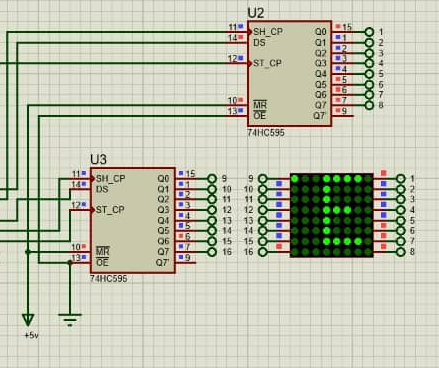
Interfacing of atmega32 with MAX7219IC is much more complicated and difficult to understand, while in case of shift register 74HC595, the interfacing is easier. That’s the reason for using shift registers in led matrix multiplexing.

* ***To decrease the number of port pins of atmega required for led matrix multiplexing.***

We use a shift register to decrease the number of atmega32 pins used while doing led matrix multiplexing.This is done because we can use the rest of pins for other purposes .

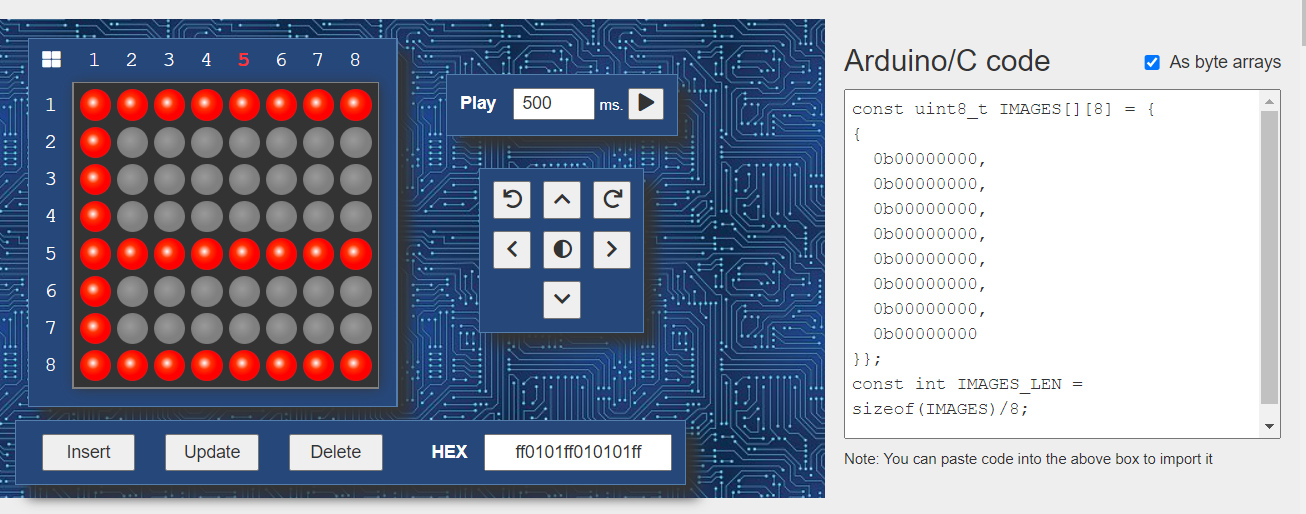
* ***Complication of circuit due to overlapping of wires.***

We used debug to connect two points without using wires which decreased the complexity of circuit as well as it appeared to be more organized.



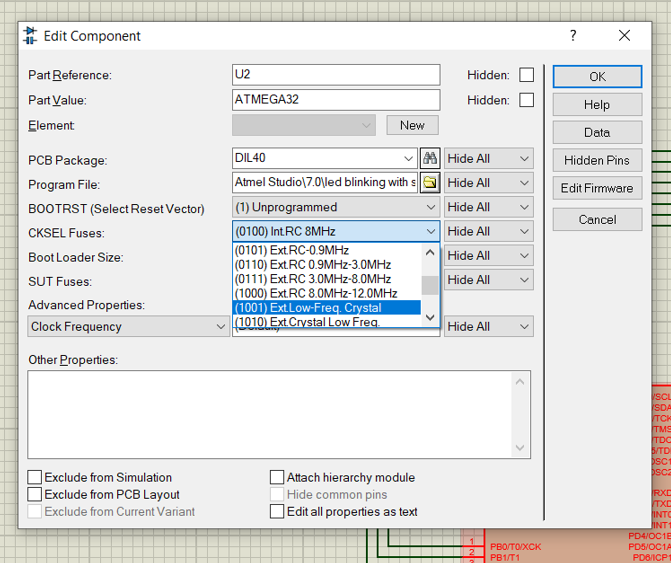
* ***Creating binary code for characters as it was time consuming.***

To overcome this problem we used an online led matrix font generator .We succeeded to create different positions of characters just by one click which eased our task.

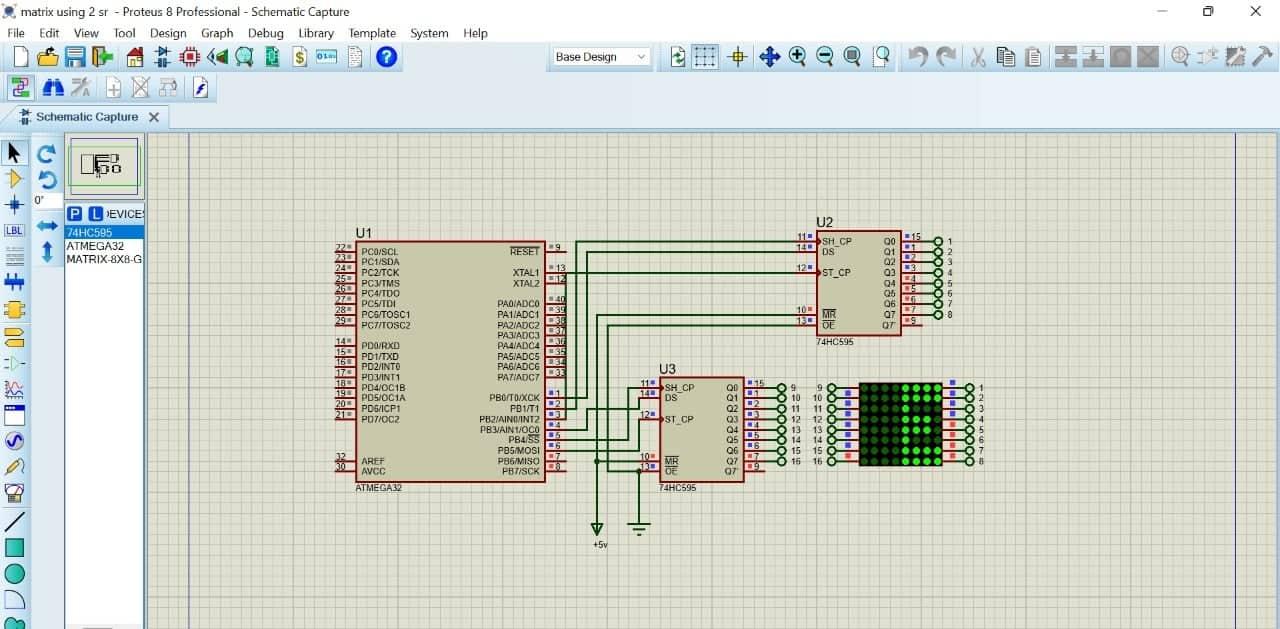


* ***To solve the issue of blinking of letters while displaying on a matrix.***

We conquered this difficulty by adjusting delay time and clock speed.Use of delay in the proper way to make our multiplexing more effective which helps in the smooth displaying of characters.



**CIRCUIT DIAGRAM**

****

**PROJECT CODE**

#include <avr/io.h>

#define F\_CPU 16000000UL

#include <util/delay.h>

#define HC595\_PORT PORTB

#define HC595\_DDR DDRB

#define HC595\_DS\_POS1 PB0 //Data pin (DS) pin location

#define HC595\_SH\_CP\_POS1 PB1 //Shift Clock (SH\_CP) pin location

#define HC595\_ST\_CP\_POS1 PB2 //Store Clock (ST\_CP) pin location

#define HC595DataHigh1() (HC595\_PORT|=(1<<HC595\_DS\_POS1))

#define HC595DataLow1() (HC595\_PORT&=(~(1<<HC595\_DS\_POS1)))

void HC595DCL1(uint8\_t data)

{

//Make the Data(DS), Shift clock (SH\_CP), Store Clock (ST\_CP) lines output

HC595\_DDR|=((1<<HC595\_SH\_CP\_POS1)|(1<<HC595\_ST\_CP\_POS1)|(1<<HC595\_DS\_POS1));

{

for(uint8\_t i=0;i<8;i++) //data

{

if(data & 0b10000000)

{

HC595DataHigh1();

}

else

{

HC595DataLow1();

}

HC595\_PORT|=(1<<HC595\_SH\_CP\_POS1);//HIGH //clock

HC595\_PORT&=(~(1<<HC595\_SH\_CP\_POS1));//LOW

data=data<<1; //Now bring next bit at MSB position

}

HC595\_PORT|=(1<<HC595\_ST\_CP\_POS1);//HIGH //latch

\_delay\_loop\_1(1);

HC595\_PORT&=(~(1<<HC595\_ST\_CP\_POS1));//LOW

\_delay\_loop\_1(1);

}

}

#define HC595\_DS\_POS2 PB3 //Data pin (DS) pin location

#define HC595\_SH\_CP\_POS2 PB4 //Shift Clock (SH\_CP) pin location

#define HC595\_ST\_CP\_POS2 PB5 //Store Clock (ST\_CP) pin location

#define HC595DataHigh2() (HC595\_PORT|=(1<<HC595\_DS\_POS2))

#define HC595DataLow2() (HC595\_PORT&=(~(1<<HC595\_DS\_POS2)))

void HC595DCL2(uint8\_t data)

{

//Make the Data(DS), Shift clock (SH\_CP), Store Clock (ST\_CP) lines output

HC595\_DDR|=((1<<HC595\_SH\_CP\_POS2)|(1<<HC595\_ST\_CP\_POS2)|(1<<HC595\_DS\_POS2));

{

for(uint8\_t i=0;i<8;i++) //data

{

if(data & 0b10000000)

{

HC595DataHigh2();

}

else

{

HC595DataLow2();

}

HC595\_PORT|=(1<<HC595\_SH\_CP\_POS2);//HIGH //clock

HC595\_PORT&=(~(1<<HC595\_SH\_CP\_POS2));//LOW

data=data<<1; //Now bring next bit at MSB position

}

HC595\_PORT|=(1<<HC595\_ST\_CP\_POS2);//HIGH //latch

\_delay\_loop\_1(1);

HC595\_PORT&=(~(1<<HC595\_ST\_CP\_POS2));//LOW

\_delay\_loop\_1(1);

}

}

void Wait()

{

for(uint8\_t i=0;i<30;i++)

{

\_delay\_loop\_2(0);

}

}

void main()

{

DDRB=0xff;

PORTB=0xff;

uint8\_t dotControl[8]={

0b00000001,

0b00000010,

0b00000100,

0b00001000,

0b00010000,

0b00100000,

0b01000000,

0b10000000};

uint8\_t letterP[29][8]={{

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000

},{

0b00000001,

0b00000001,

0b00000001,

0b00000001,

0b00000001,

0b00000001,

0b00000001,

0b00000001

},{

0b00000011,

0b00000011,

0b00000010,

0b00000011,

0b00000011,

0b00000010,

0b00000011,

0b00000011

},{

0b00000111,

0b00000111,

0b00000100,

0b00000111,

0b00000111,

0b00000100,

0b00000111,

0b00000111

},{

0b00001111,

0b00001111,

0b00001000,

0b00001110,

0b00001110,

0b00001000,

0b00001111,

0b00001111

},{

0b00111110,

0b00111110,

0b00100000,

0b00111100,

0b00111100,

0b00100000,

0b00111110,

0b00111110

},{

0b01111100,

0b01111100,

0b01000000,

0b01111000,

0b01111000,

0b01000000,

0b01111100,

0b01111100

},{

0b11111001,

0b11111001,

0b10000000,

0b11110000,

0b11110000,

0b10000000,

0b11111000,

0b11111000

},{

0b11110010,

0b11110011,

0b00000001,

0b11100000,

0b11100000,

0b00000000,

0b11110000,

0b11110000

},{

0b11100100,

0b11100110,

0b00000011,

0b11000001,

0b11000000,

0b00000000,

0b11100000,

0b11100000

},{

0b11001000,

0b11001100,

0b00000110,

0b10000011,

0b10000001,

0b00000001,

0b11000001,

0b11000001

},{

0b10010000,

0b10011000,

0b00001100,

0b00000111,

0b00000011,

0b00000011,

0b10000011,

0b10000011

},{

0b00100000,

0b00110000,

0b00011001,

0b00001111,

0b00000110,

0b00000110,

0b00000110,

0b00000110

},{

0b01000000,

0b01100001,

0b00110011,

0b00011110,

0b00001100,

0b00001100,

0b00001100,

0b00001100

},{

0b10000001,

0b11000011,

0b01100110,

0b00111100,

0b00011000,

0b00011000,

0b00011000,

0b00011000

},{

0b00000010,

0b10000110,

0b11001100,

0b01111000,

0b00110000,

0b00110000,

0b00110000,

0b00110000

},{

0b00000100,

0b00001100,

0b10011000,

0b11110000,

0b01100000,

0b01100000,

0b01100000,

0b01100000

},{

0b00001001,

0b00011001,

0b00110001,

0b11100001,

0b11000001,

0b11000001,

0b11000001,

0b11000001

},{

0b00010011,

0b00110011,

0b01100010,

0b11000011,

0b10000011,

0b10000010,

0b10000011,

0b10000011

},{

0b00100111,

0b01100111,

0b11000100,

0b10000111,

0b00000111,

0b00000100,

0b00000111,

0b00000111

},{

0b01001111,

0b11001111,

0b10001000,

0b00001110,

0b00001110,

0b00001000,

0b00001111,

0b00001111

},{

0b10011111,

0b10011111,

0b00010000,

0b00011100,

0b00011100,

0b00010000,

0b00011111,

0b00011111

},{

0b00111110,

0b00111110,

0b00100000,

0b00111000,

0b00111000,

0b00100000,

0b00111110,

0b00111110

},{

0b01111100,

0b01111100,

0b01000000,

0b01110000,

0b01110000,

0b01000000,

0b01111100,

0b01111100

},{

0b11111000,

0b11111000,

0b10000000,

0b11100000,

0b11100000,

0b10000000,

0b11111000,

0b11111000

},{

0b11110000,

0b11110000,

0b00000000,

0b11000000,

0b11000000,

0b00000000,

0b11110000,

0b11110000

},{

0b11100000,

0b11100000,

0b00000000,

0b10000000,

0b10000000,

0b00000000,

0b11100000,

0b11100000

},{

0b11000000,

0b11000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b11000000,

0b11000000

},{

0b10000000,

0b10000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b10000000,

0b10000000

}};

int j=0;

while (1)

{

for (int k=0;k<29;k++) //loop changing time

{

for (int i=0;i<105;i++) //loop for stabilizing same frame for extended period of time

{

HC595DCL2(dotControl[j]);

HC595DCL1(~letterP[k][j]);

j+=1;

if (j>7)

{j=0;}

\_delay\_ms(1.5);

}

\_delay\_ms(1.5);

}

}

}

**PROJECT RESULT**

<https://drive.google.com/file/d/1Z1XXk6u6bwGrkCFHEgpxjGjYYpM_ZdR6/view?usp=sharing>

**BIBLIOGRAPHY**

* **Link for Downloading Proteus**

<https://drive.google.com/file/d/1bKfTUkJstJ5I3h16DkSMrEglp_xVoz-0/view?fbclid=IwAR2r7od4ckL2CyTwmXxoNcKEbkRxNczDF-BDjHWDEIunIRZqqdGslXuH-3A>

* **Link for Downloading Atmel Studios**

<https://ww1.microchip.com/downloads/aemDocuments/documents/DEV/ProductDocuments/SoftwareTools/as-installer-7.0.2594-full.exe>

* **74HC595 Serial Shift Register Interfacing with Pic Microcontroller**

<https://microcontrollerslab.com/74hc595-shift-register-interfacing-pic-microcontroller/?fbclid=IwAR1uxgIfjy22CVbM9HuMSLhs30YJi-9sp8cYD5xfViK7-N_xqrn06ZSG4Bw>

* **Atmega 8 Connection to 74HC595**

<https://electronicsmaker.com/em/admin/pdfs/free/ATMEGA.pdf?fbclid=IwAR0JEn-s80Iec6DXaVZDv5x0_j7W67BIBHi1zKGUBACB-UW6VVY-Rcd_3KQ>

* **MAX7219 IC : Pin Configuration & Its Working**

<https://www.elprocus.com/max7219-ic/?fbclid=IwAR0qa9rtWaVZ1hS_e6fleMEejAaDT6psEWLqXfXKyc9yywQ7Nu8faTH417A>

* **74HC595 Shift Register & Its Working**

<https://www.elprocus.com/74hc595-shift-register/?fbclid=IwAR2OZ6mEnn4woAlbu1M90c1DVrIwoh659Za31kzmxGZfd-iszq0NYsPvbbM>

* **Bit – Shifting and Brightness Control with 74HC95**

<https://mywtmb.wordpress.com/2020/05/09/example-post-2/?fbclid=IwAR1KHgiDmZHLMcLfiUgYnuLiSFiqdUp7bUt_0ufh0dIkOfpClPiUtVUq5Qw>

* **Basics of the I2C communication protocol**

<https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/?fbclid=IwAR0HiaV_pxdHFpOirWNAgPOzl3znfnM2D2RftyS_bgvCCm4OOomIwj84S58>

* **ATMega 32 Datasheet**

<https://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf>

* **ATMega32 and a dot matrix display interfacing**

<https://www.youtube.com/watch?v=m0uHIXuihro>

* **Online LED Hex code generator**

<https://www.riyas.org/2013/12/online-led-matrix-font-generator-with.html?fbclid=IwAR1n1vPdE0_sYnUek0pv0StiM-WXbPmMN8leB27vHvypJharb-U4orzXrIY>

* **Shift Registers with AVR Micro – AVR**

<https://extremeelectronics.co.in/avr-tutorials/using-shift-registers-with-avr-micro-avr-tutorial/?fbclid=IwAR2ugBX-Dt2lFJN-5DGwA9hN_LQhHXIIRU_zlbuPhzHWfGQpCxGqI5zy6oE>

* **Interfacing ATMega32 to 74HC595 shift register**

<https://aki-technical.blogspot.com/2020/05/interfacing-atmega32-to-74hc595-shift.html?fbclid=IwAR3YbFIizUjJY9Td8Rh2ljKX15v4isQvw8CNRlZTOWgN47TW1y41MK7hDjw>