Bangladesh University of Engineering and Technology
Department of Computer Science and Engineering

CSE 316

Microprocessors, Microcontrollers, and Embedded Systems Sessional

Experiment 2
Basic Bicolor LED matrix control with ATmega32: static, flash, and rotating display.

GOAL:

To understand basic bicolor LED matrix control in order to generate different characters/symbols.

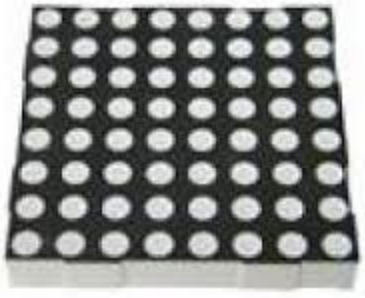
EXPERIMENTAL TOOLS AND MATERIALS:

ATmega32, BiColor LED matrix, USBASP programmer, Trainer Board, Wires, Avr Studio, Extreme Burner.

EXPLANATION OF PRINCIPLES:

The experiment has three parts. First, you have to show a symbol in the LED matrix. **The symbol will be assigned by the lab teachers during the experiment.** In the second part, you will have to flash the same symbol, i.e., for some time the symbol will be shown and then it will disappear for some time before it reappears again, and so on. In the last part of the experiment you will rotate the symbol left/right. **The direction will be assigned by the lab teachers during the experiment.**

The bicolor LED can show symbols either in green or red. **The color will also be assigned during the experiment.** You can connect the bicolor LED matrix to any of the ports of your choice.

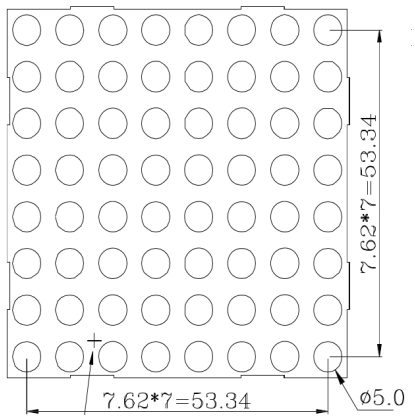


LED Matrix Basics:

The LED matrices used in the labs are 8x8 and bicolor. Each position (i.e., LED) can glow in green or red.

There are 24 pins in total to select which LED(s) we want to light up. 8 of the pins are used to select the row(s), while the remaining 16 pins are used to select the column(s) (The notion of row and columns are only artificial and depends on the way we consider the orientation of the LED matrix. Rotating the matrix by 90 degrees will alter the orientation. However we will use a fix orientation which is described later in this section). Among these 16 pins, 8 pins are used for lighting the LEDs red while the other 8 pins are used if we want to light the LEDs green. LED matrices can be oriented in two

flavors: common (row) anode and common (row) cathode. The difference between these two configurations is how we light up a LED. With common anode orientation, positive voltage is attached to rows and ground to columns. With the common cathode, the connections are reversed. In our experiments we will use the common anode orientation.



The pin diagram of the bicolor LED matrix is given below. The pin orientation depends on how you orient the matrix. Hold the LED matrix in such a way that the pins are towards you and the LEDs are at the opposite side (i.e. you are looking at the back of the matrix). Make sure the pins are at the top and bottom of the matrix rather than on the sides. Finally make sure that the top of the matrix has a couple of outward extensions (bottom will then have inward extensions, the extensions can be seen in the schematic).

Row4	R4	G4	Row3	R3	G3	Row2	R2	G2	Row1	R1	G1
Row8	R8	G8	Row7	R7	G7	Row6	R6	G6	Row5	R6	G5

Text side here - Upside Down

Description of pins:

1) Rows are anode (+ve is to be given). Rows are numbered from up to down.

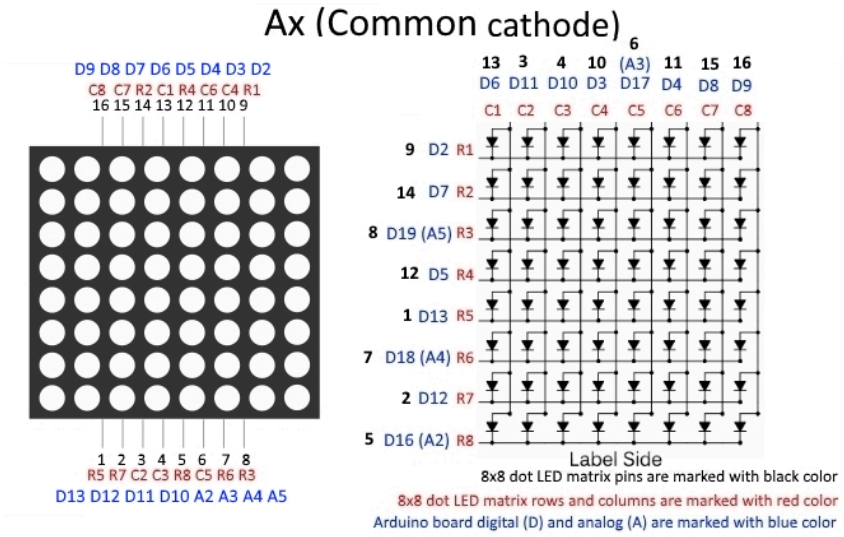
2) Columns are cathode (GND is to be given). Columns are numbered from left to right (Assuming this time you are facing the LED (front side of the matrix)).

3) RowX denotes row number.

4) GX is used to light up green LEDs in column X.

5) RX is used to light up red LEDs in column X.

For example, if we want to light the 2nd column of the 3rd row with red, we will have to connect VCC to the Row3 pin and GND to the R2 pin.



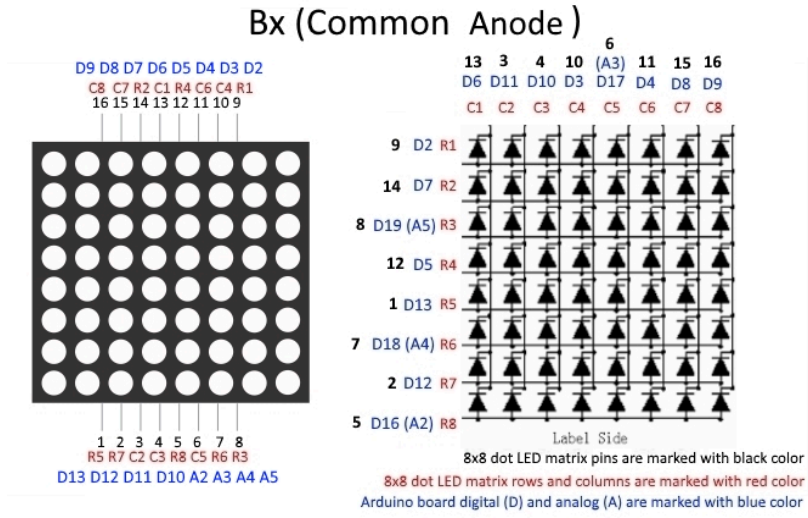


Fig. Single color LED matrix

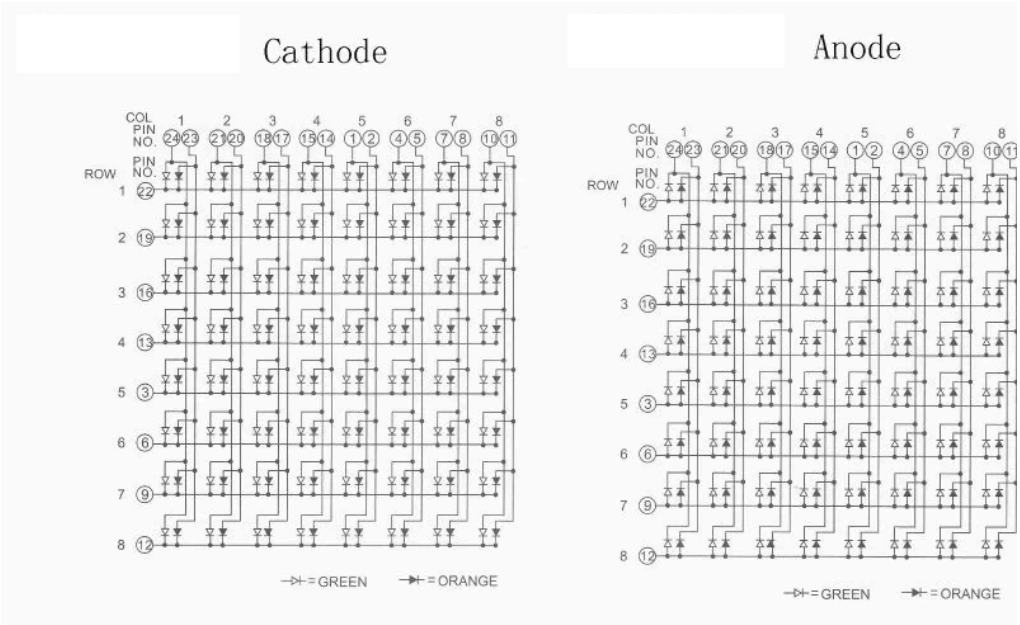


Fig. Bi-color LED matrix

Showing a symbol in dot matrix:

Typically multiplexing is used to display arbitrary patterns with led matrices. Multiplexing is sometimes also called scanning. It scans rows (usually from up to down) and lights selected LEDs only in one row at a time following these steps:

1. Turn on only the topmost row (i.e., provide +5V)
2. Turn on the necessary columns of the topmost row by connecting the corresponding column pins to ground. This lights up the selected LEDs in the topmost row.
3. Do steps 1 and 2 one by one for all rows and keep repeating.

If you do this slowly, you would see the LED rows turning on one after another. However, if it is done fast enough, the human eye will see the whole pattern together. This phenomenon is called persistence of vision.

The amount of time you light up one row, before lighting up the next one is very crucial for the symbol to be displayed properly. Usually you will light up one row and then create a delay. After the delay is over you will light up the next row. If this delay is too long, one will see the rows being light up one after another and the illusion of displaying the whole symbol at once will not work. On the other hand, if the delay is very small, the LEDs will not get enough time to glow fully. Hence, the symbol will look less bright. **In the experiment you will have to find a near optimal delay on trial and error basis, so that the symbol is displayed bright and clear.**

Why multiplexing: There are 64 bicolor LEDs in a dot matrix, without multiplexing we would have needed 128 different pins to operate a single dot matrix!!!! Using multiplexing technique we can still operate it with only 24 pins.

PROCEDURE:

1. Connect the bicolor LED matrix to the ATmega32 according to instructions given by the teachers during the experiment.
2. Write a C program to implement each of the three parts of the experiment, **i.e., static display, flash, and rotating display** one after another. After finishing each part, show it to the lab teacher.

MISCELLANEOUS:

1. Further details on working with LED matrices:
   https://www.appelsiini.net/2011/how-does-led-matrix-work

2. Further details on multiplexing
   https://youtube.com/clip/Ugkxcxmpvr9tH-o0_2AxV42tsGpVe78NJKLv

3. The power provided by USBASP may not be enough to drive the LED matrix. You can consider using external power sources, e.g, the power source of experiment boards you used in DLD experiments and Motor experiment of this course.

4. Some pins of PORTC can not be used for I/O directly. First the JTAG has to be turned off.
   One way is to uncheck the JTAG box while writing the fuse bits, the second is to write a 1 to the JTD bit twice consecutively.
   ```
   MCUCSR = (1<<JTD);
   MCUCSR = (1<<JTD);
   ```
   For more details refer to:
   http://www.avrfreaks.net/forum/jtag-enablingdisabling-atmega32-and-fuse-settings-solved

5. You do not need to bother about FUSE bits. By default it is set to: E199
   If the JTAG Interface is disabled it will be set to: E1D9. You can calculate fuse bit from this

online tool: http://www.engbedded.com/fusecalc/