DRIVER DESIGN FOR MONOCHROME LCDS

Alireza zabihi
khoy, Iran
zabihi.alireza75@gmail.com

Compared to other display technologies, LCD (Liquid Crystal Display) has various benefits like low power consumption, excellent contrast, and information density. In order to achieve a complete user interface in a condensed display space, many LCDs used in calculators, clocks, watches, and other electronic home appliances, in particular, include particular pictograms on their screens. This lcds can be used with some driver ics such as the HT1622 or TM1640. However, in this paper, I will show you how to drive this LCD without a driver, using only your MCU.

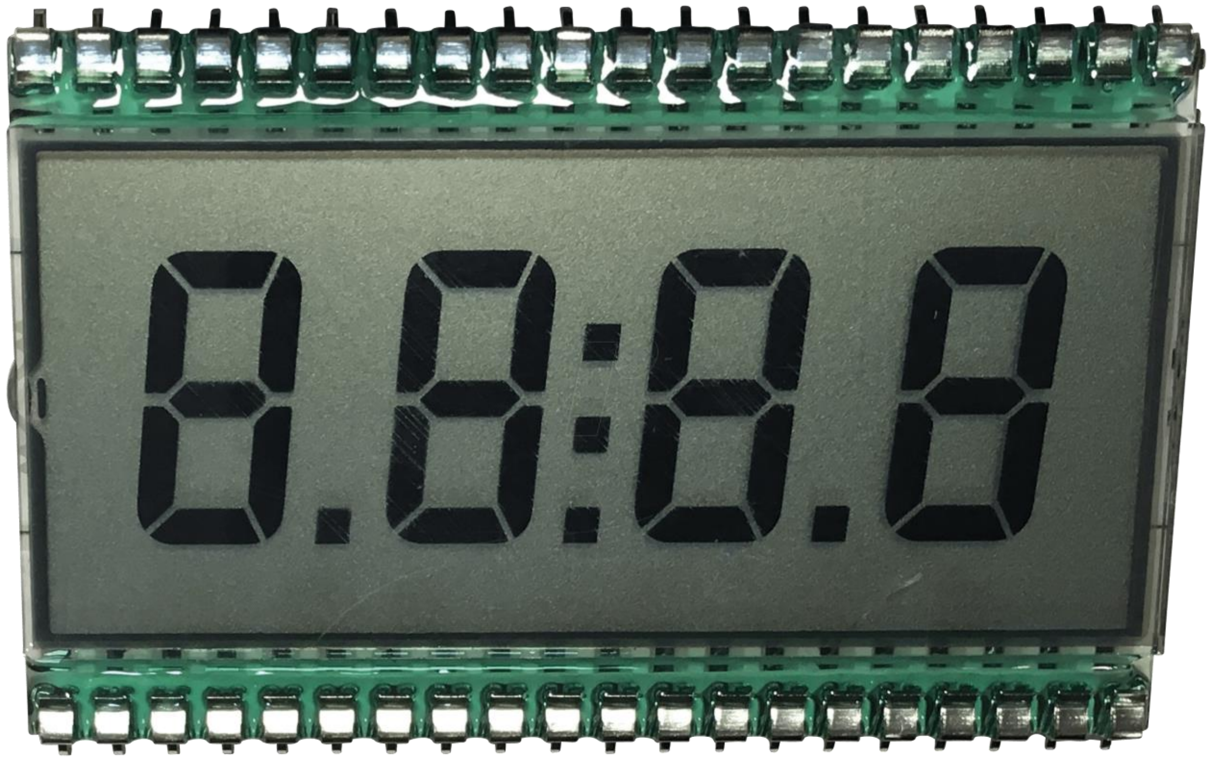


Figure 1. a commercial monochrome lcd

How LCD works

They are made of transparent electrodes on the interior surfaces and contain a simple structure with liquid crystal sandwiched between two parallel flat glasses.

The liquid crystal changes its molecular alignments and optical property characteristics when the proper voltage is applied across its two electrodes, allowing information to be displayed [1]. assuming simple seven-segment displays, which typically have eight pins per segment and some common pins to control which segments you want to display. we are using the same device, Instead of segment and common pins, we have back-plane (BP) and front-plane (FP) . Depending on the type of LCD, one of them is used instead of the common pin and the other is used for display pixels.

HOW TO FIND PINOUT OF LCD.

I will explain two methods for determining the pinout of your LCD. The first method is very useful and simple. Just Apply a square wave of 1 to 6 volts (Depending on your LCD version, the magnitude of the voltage can rise up to 10 volt, “Start with low voltages”) and a frequency of about 100 Hz to each of the pins of lcd. I randomly choose one pin to serve as the ground reference, as seen in Figure 4, and move the signal probe to each of pins. It should be noted that the common pins and segments on the bodies of most LCDs have been separated, making it much easier to find the pins of an LCD. (I figured you didn't have access to your personal LCD's datasheet.).

If any pixel appears on the LCD screen using the proposed method, you have found the common pin. Sometimes you have to look at the LCD in a specific direction.

The image below shows my specific LCD, which is ideal for RF applications.

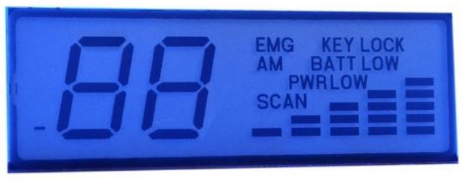


Figure 2.my bare lcd pixels

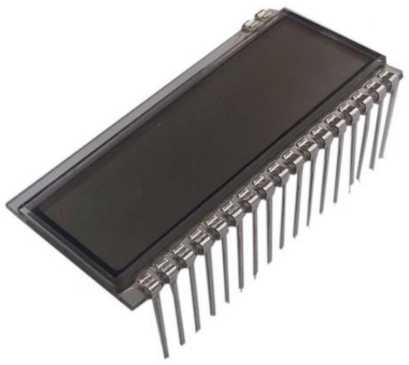


Figure 3. Package in pins of lcd

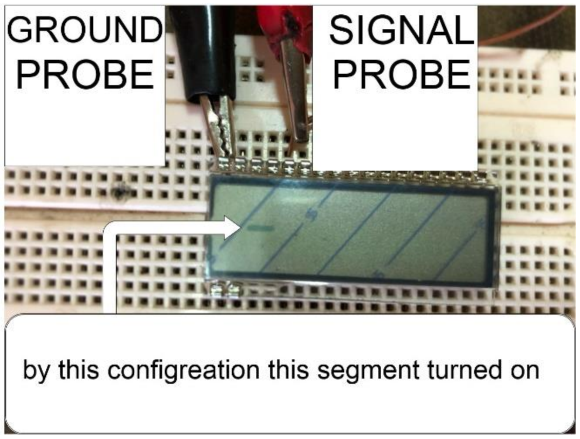


Figure 4.find pinout and pixels of lcd (method one)

Change the position of the probes as shown in Figure 4 to find out the pin configuration of the LCD; it will only take a short time.

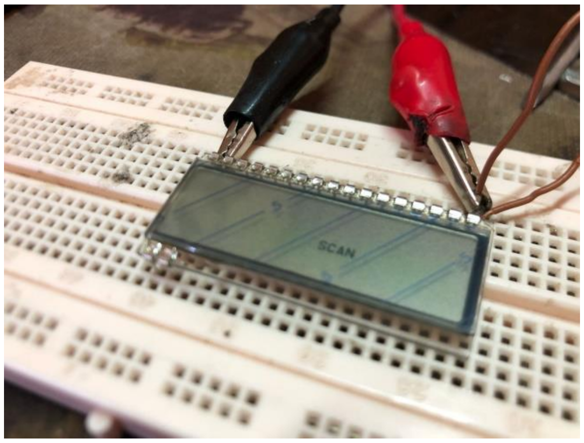


Figure 5.by this configuration the word of scan is appear on lcd screen

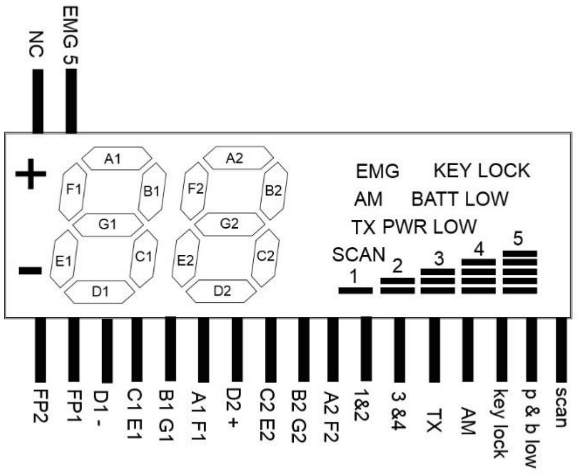


Figure 6 configuration of the pins on my LCD

I have two front planes (FP1 & FP2) as common pins with all other seven segment and pictograms for this specific LCD.

We can turn on all of the LCD's pixels by applying the proper voltage to the FP and BP pins (seven-segment and pictogram pins).for example if we apply appropriate voltage between FP1 and D1 – PIN the pixel of – will turned on and also if we apply appropriate voltage to FP2 and D1 – ping the segment of D1 will be merge on lcd screen.

Figure below shows the common pin between front plane and back plane by different colours. For example, C1 and E1 share a common backplane but have distinct front planes.as well as B1 and G1

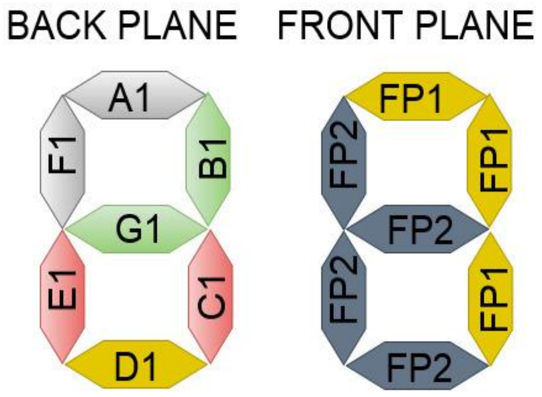


Figure 7.common back plane and fron plane

It should be noted that each and every pixel on an LCD is sandwiched between two planes, known as the back plane and front plane.

Let me explain another way to extract the pins from a LCD before I discuss the driver design for LCD.

This method can work 100% for monochrome LCDs, but the method that was explained should work for 99% of LCDs. You can use the "not" gate to determine the pinout of an LCD by configuring it as shown in Figure 8.

For some LCDs, not gate can be used as a driver (also called static driver) and can be replaced with a "xor" gate to have an enable pin as well (consult "xor" gate truth table).

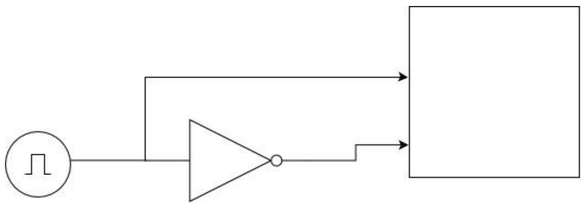


Figure 8. not gate use for find pin out

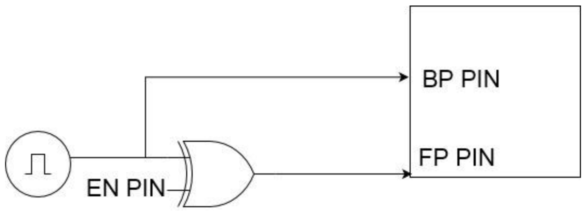


Figure 9.use xor gate as driver

Note: you don't have access to a function generator, you can create a 100hz waveform with your micro controller.

DRIVER DESIGN

If you want to use the static method (xor gate), you must adjust the frequency and voltage amplitude until the desired result is obtained. It may be difficult to achieve the best results in comparison.

in dynamic drive, we need a bit complicated waveform to control.

My LCD has two front plane lines that are shared by all segments (common lines) and four backplane lines that are shared by two of the seven segments (each digit).Figure 6 and 7 depicts this. Your LCD may have a different configuration, such as four common lines and two backplane lines.

So we have two common pins. I named them COM1 and COM2. All of you need to apply different phases of alternating voltage to different common electrodes, and then, by applying voltage to a segment with carefully controlled timing, we can light it if the voltage difference between it and the common electrode is large enough, and turn it off if it is small.

This waveform should be created and applied to common pins.

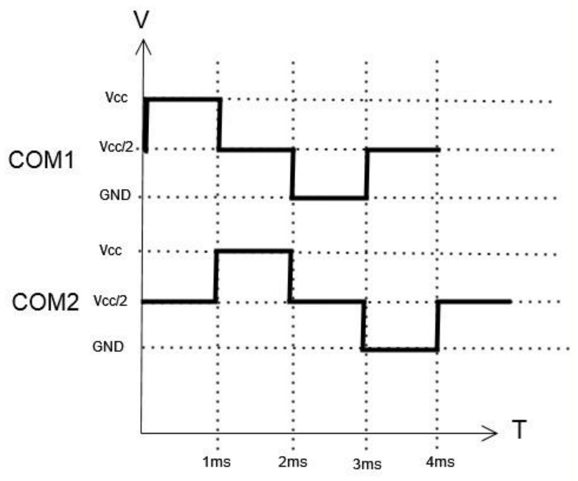


Figure 10.1/2 bias waveform for lcd drive

The number of waveforms is proportional to the number of common pins. Don't worry, creating such a waveform is very simple.

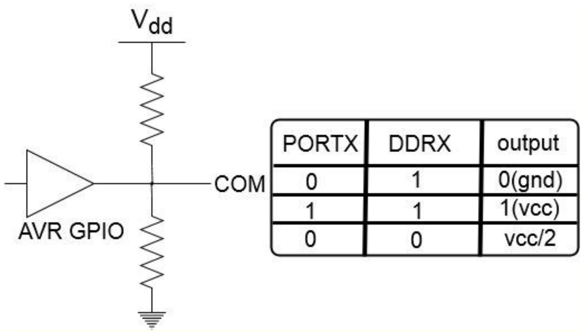
You must control your MCU GPIOS in order to create such a waveform.

Assume the amplitude of the vcc is 5 volts.

When you set a specific GPIO, the output will have a 5v amplitude; when you reset it, the output will be shorted to GND.

If you don't do anything with the GPIO pin or set it to HI-Z, the output will be vdd/2 due to the resistor deviation circuit.

Refer to Figure 11.



PORTX	DDRX	output
0	1	0(gnd)
1	1	1(vcc)
0	0	vcc/2

Figure 11. configuration for make 1/2 bias waveform

To turn segments on, apply signals to BP lines.

If your common pin signal has a 180 degree phase difference from the BP signal during a period (for example, T1), its segment will turn on. If it hasn't, its segment will remain off.

So, in the configuration shown below, the number 5 will be displayed on the LCD.

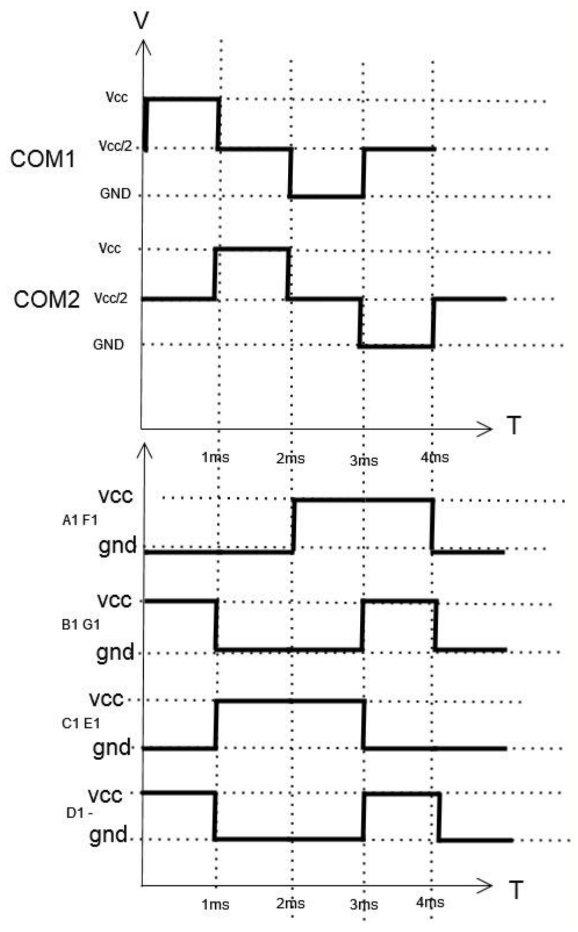


Figure 12. apply appropriate signal to common pins for displayed desired number or character

by the pinout and segments are showed on figure 6 and 7. If I connect an LCD pin to, say, PORTB, and want to display the number 5, I must write IN PERIOD OF 1ms PORTB=0b00001010, IN PERIOD OF 2ms PORTB=0b00000100, IN PERIOD OF 3ms PORTB=0b00000101, and IN PERIOD OF 4ms PORTB=0b00001011.

Check out the Source code at the end of the article.

Target hardware

ATMega8 is the main MCU. It is powered by 8MHz internal oscillator and The LCD is

directly connected to the majority of its
ports.

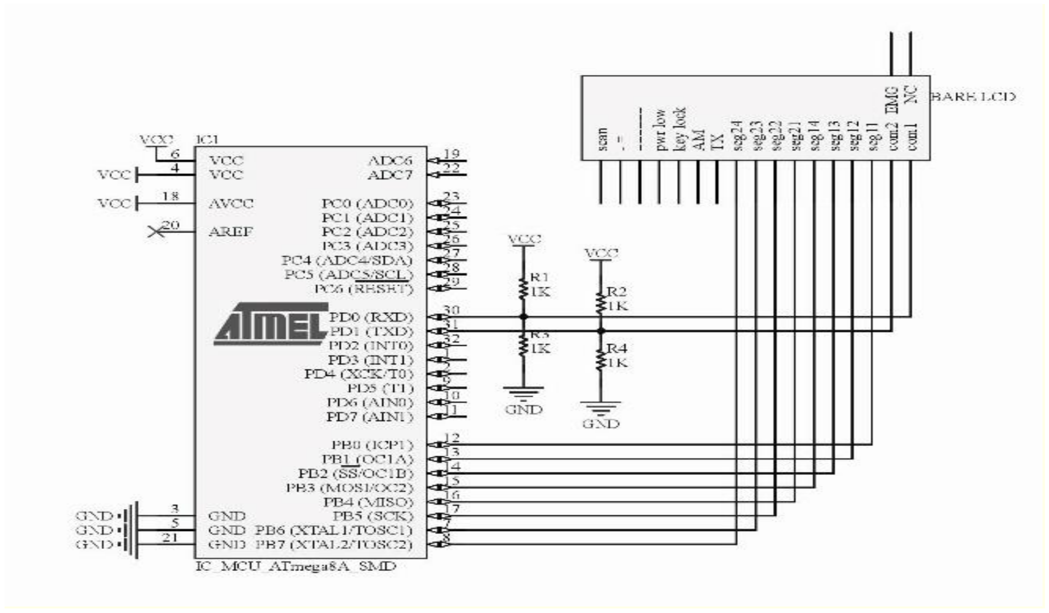


Figure 13. hardware schematic

Source code

```
#include <mega8.h>

#include <delay.h>

unsigned char a,s0,s1,number,t0,t1,temp,temp0,digit2; // We declare some variables that will be used.

// Create two lookup tables to display numbers ranging from 0 to 9. I have two seven segment that why I create four string

unsigned int FP11[]={ 0B00000000 , 0B00001001 , 0B00000010 , 0B00000110 , 0B00001101 ,
0B00000100 , 0B00000000 , 0B00001110 ,0B00000000 , 0B00000100};

unsigned int FP22[]={0B00000101 , 0B00001111 , 0B00001001 , 0B00000001 , 0B00000001 ,
0B00000011 ,  0B00000011 ,0B00000101 , 0B00000001 , 0B000000001};

unsigned int FP1[]={ 0B00000000 , 0B10010000 , 0B00100000 , 0B01100000 , 0B11010000 ,
0B01000000 , 0B00000000 , 0B11100000 ,0B00000000 , 0B01000000};

unsigned int FP2[]={0B01010000 , 0B11110000 , 0B10010000 , 0B00010000 , 0B00010000 ,
0B00110000 ,  0B00110000 ,0B01010000 , 0B00010000 , 0B00010000};
```

```

interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{//generate 1/2 bias  waveform
  a++;  //timer frequency is about 67 HZ and a is increase every 1ms
switch (a){
                        case 1: // I used the parameters in Figure 5 to create the waveform shown in Figure 4.
 DDRD=  (1<<DDD1) | (0<<DDD0);
 PORTD=  (1<<PORTD1) | (0<<PORTD0);
  PORTB=(t0|s0); // Write some values on PORTB, and some digits will have 180 phase differences,
causing some segments to turn on while others remain off.
  break;
                          case 2:
 DDRD=  (0<<DDD1) | (1<<DDD0);
 PORTD=  (0<<PORTD1) | (1<<PORTD0);
   PORTB=(t1|s1); // s1 for the first segment and t1 for the second segment If you only need to use
one segment, remove s1 or t1.
break;
                            case 3:
 DDRD=  (1<<DDD1) | (0<<DDD0);
  PORTD=  (0<<PORTD1) | (0<<PORTD0);
   PORTB=~(t0|s0); // According to the common pins waveform, the logic state of segments must be
reversed.
   break;
                            case 4:
DDRD=  (0<<DDD1) | (1<<DDD0);
  PORTD=  (0<<PORTD1) | (0<<PORTD0);
   PORTB=~(t1|s1);
   a=0;
     break;    }
TCNT0=0x04;  }
```

```
void main(void)
{
// Input/Output Ports initialization
DDRB=0XFF;
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4);
DDRC=(0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (0<<DDC0);
PORTC=(0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) |
(0<<PORTC0);
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (1<<DDD1) |
(1<<DDD0);
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2)
| (0<<PORTD1) | (0<<PORTD0);
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
TCCR0=(0<<CS02) | (1<<CS01) | (1<<CS00);
TCNT0=0x05;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(1<<TOIE0);
#asm("sei")
while (1)
     {
     temp=number //make a copy from the number
     temp=temp%10; // keep segment 1's value between 0 and 9

s0=FP1[temp];   // The value will be displayed on segment 1.
s1=FP2[temp];

if(digit2==10){temp0++; digit2=0;}  keep segment 2 value between 0 and 9
t0=FP11[temp0]; // The value will be displayed on segment 2.
t1=FP22[temp0];

     number++; // increase the value of the test number
     digit2++;
delay_ms(900);
if(number>=99){number=0;}
     }
}
```

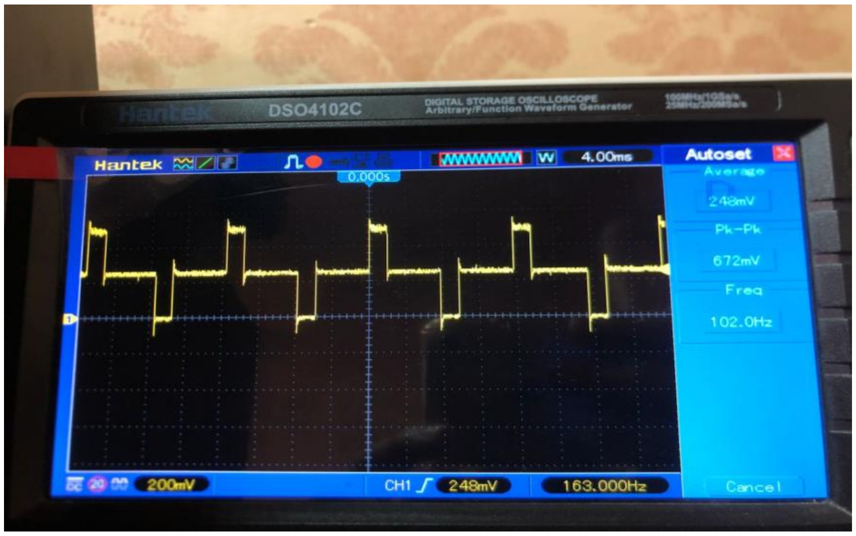


Figure 14. 1/2bias waveform

Conclusion

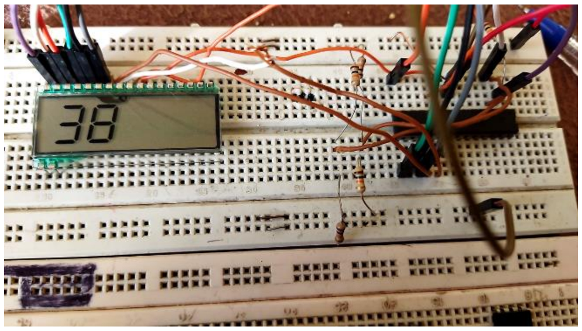


Figure 15.lcd driver work successfully

I believe I covered everything you needed to know about driving these types of LCDs.

Simply look for common pins. apply the 1/2 waveform to the common pins Apply the appropriate voltage to the segment pins (remember, if you want to display segments, the applied voltage must be have 180 degrees phase different from the common pin waveform at a specific time , otherwise that segment will stay off).

In the end, I just write very simple code and hardware to test the proposed drive method; you can add some interfaces such as spi or i2c to communicate with other micros. also you can optimize the code and use it for other MCUs.

I will update the codes and this post. if you encounter any problems or have any suggestions, please contact me at zabihi.alireza75@gmail.com.

The hardware testing video will be posted to YouTube, and I will make a tutorial video for this article.

The link of test video on youtube:

https://youtu.be/gpRRLDh8GfM

GitHub:

https://github.com/zabihi-alireza/driver-design-for-monochrome-lcds.git documents of project

https://github.com/zabihi-alireza keep an eye out for new updates on this project

https://github.com/zabihi-alireza/bare-lcd-schematic.git library package for my LCD

On LinkedIn, I discuss my work experience. You can connect with me on LinkedIn by clicking on this link.:

https://www.linkedin.com/in/alireza-zabihi-b72412241/

other contacts

Instagram:

https://www.instagram.com/eng.zabihi/

Telegram:

https://t.me/engzabihi

Acknowledge

Many thanks and best wishes to Mr.Awawa.Hariko and Mr.Iman Davoudi for their assistance in designing the driver.

You can contact with them by links below:

http://awawa.hariko.com

imi.davoudi@gmail.com