آزمایشگاه ریزپردازنده

سرکار خانم دکتر زارعی

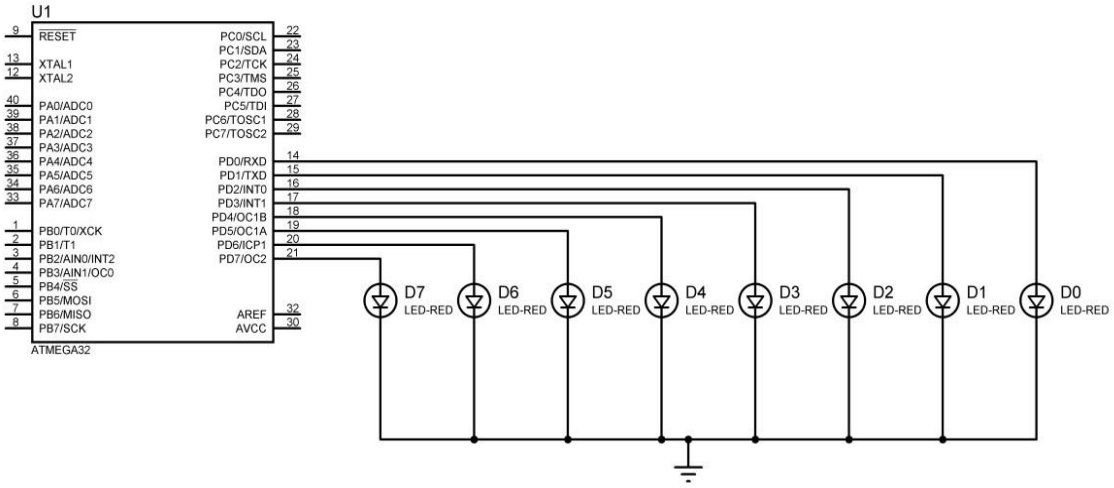
آزمایش ۱ – پیاده سازی فلاشر

۱-۱ آزمایش اول

برنامه زیر را در محیط نرم افزار Code Vision AVR نوشته و پس از بررسی درستی برنامه نوشته شده، آن را کامپایل کرده و در محیط نرم افزار Proteus مدار نشان داده شده در شکل ۱-۱ را شبیه سازی نمایید. سپس به کمک Programmer برنامه خود را به IC میکروکنترلر AVR منتقل کرده و با استفاده از مجموعه آموزشی موجود در آزمایشگاه، مدار نشان داده شده در شکل ۱-۱ را پیاده سازی کنید.

```
#include <mega32.h>
#include <delay.h>
#define xtal 8000000

int i;
void main (void)
{
        DDRD = 0XFF;
        While(1)
        {
                for(i = 1; i <= 128; i = i*2)
                {
                        PORTD = i;
                        delay_ms(100);
                 }
for(i = 64; i > 1; i = i/2)
     {
          PORTD = i;
          delay_ms(100);
     }
  }
}
```



شکل ۱-۱: شمای مداری آزمایش اول

۱-۲ آزمایش دوم

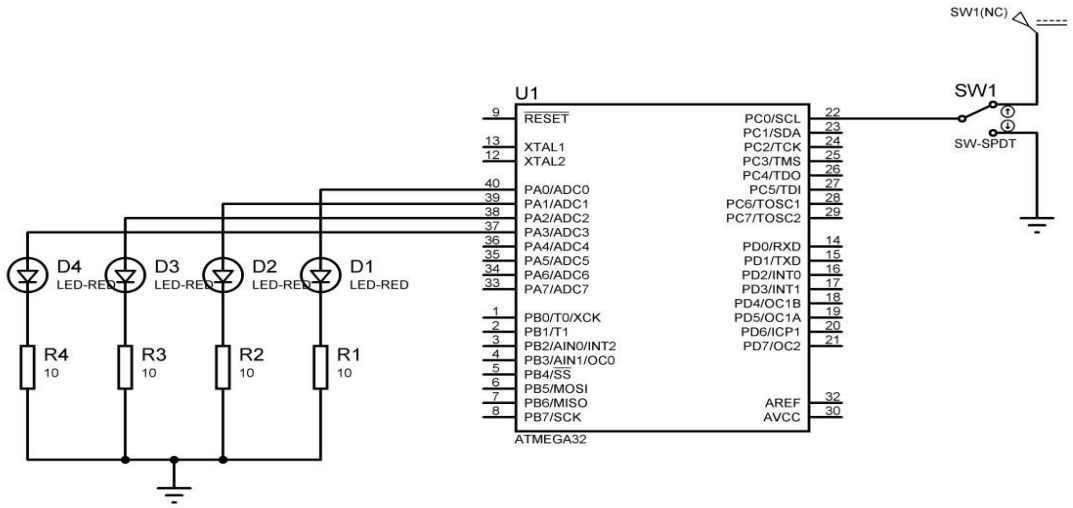
برنامه ای بنویسید که در مدار شکل ۱-۱، LED ها با الگوی نشان داده شده در جدول ۱-۱، روشن و خاموش شوند. مدت زمان تأخیر بین هر دو لحظه متوالی را ۱۰۰ میلی ثانیه درنظر بگیرید. برنامه را به گونه ای بنویسید که این عمل تا بینهایت تکرار شود. برنامه خود را با استفاده از یکی از دستورالعمل های حلقه که فکر می کنید مناسب باشد، بنویسید.

جدول ۱-۱: الگوی روشن و خاموش شدن LEDها

	D0	D1	D2	D3	D4	D5	D6	D7
لحظه ۱	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	**ON**
لحظه ۲	OFF	**ON**	OFF	OFF	OFF	OFF	**ON**	OFF
لحظه ۳	OFF	OFF	**ON**	OFF	OFF	**ON**	OFF	OFF
لحظه ۴	OFF	OFF	OFF	**ON**	**ON**	OFF	OFF	OFF
لحظه ۵	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
لحظه ۶	OFF	OFF	OFF	**ON**	**ON**	OFF	OFF	OFF
لحظه ۷	OFF	OFF	**ON**	OFF	OFF	**ON**	OFF	OFF
لحظه ۸	OFF	**ON**	OFF	OFF	OFF	OFF	**ON**	OFF
لحظه ۹	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	**ON**

۱-۳ آزمایش سوم

۱- برنامه ای بنویسید که در مدار شکل۱-۲، اگر کلید متصل به PortC.0 در وضعیت یک منطقی (متصل به ۵ ولت باشد)، آنگاه معادل باینری اعداد صفر تا ۱۵ به صورت شمارش از صفر تا ۱۵ بر روی LED های متصل به PORT A ارسال شود و اگر در وضعیت صفر منطقی (متصل به زمین) باشد، عملی انجام نشود. این برنامه را با استفاده از حلقه ها بنویسید و آنرا به گونه ای بنویسید که عمل شمارش تا بینهایت تکرار شود. همچنین زمان تأخیر بین هر دو شمارش متوالی را ۱۰۰ میلی ثانیه درنظر بگیرید.



شکل ۱-۲: شمای مداری مربوط به پرسش ۱

آزمایش ۲ – چگونگی استفاده از 7-Segment و ارتباط آن با میکروکنترلر

۲-۱ آزمایش او ل

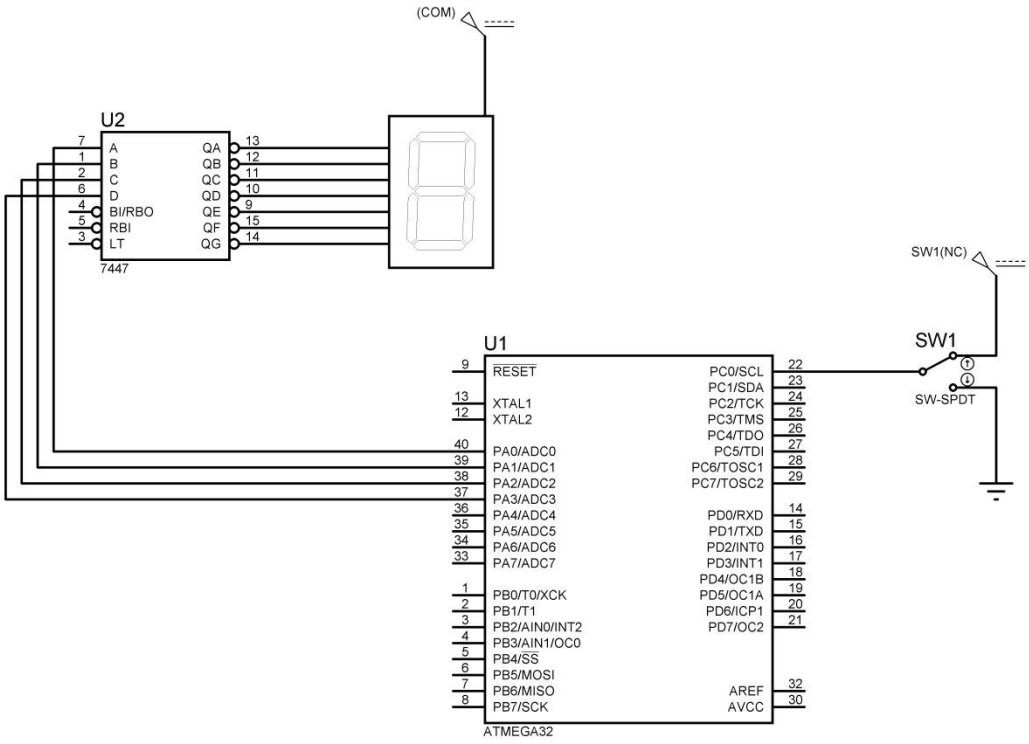
برنامه ای بنویسید که وضعیت پایه صفر از Port C را بررسی کند و در صورت یک بودن آن، شمارش را صفر تا ۹ و در صورت صفر بودن، شمارش را از ۹ تا صفر انجام دهد. سپس حاصل را بر روی یک 7-Segment که توسط یک IC دیکدر 7447 به Port A متصل شده است، نمایش دهد.

توجه: برای شبیه سازی از یک 7-Segment آند مشترک استفاده کنید.

توجه: IC دیکدر 7447 یک دیکدر BCD به 7-Segment است.

در این IC، عدد BCD مورد نظر به ورودی های A، B، C و D اعمال می شود که در آن A بیت کم ارزش است و خروجی متناظر با آن در پایه های a تا g قرار می گیرد. در عمل، پایه های خروجی توسط مقاومت های 150 Ω به 7-Segment متصل می شوند.

شمای مداری این بخش از آزمایش در شکل ۲-۱ نشان داده شده است.



شکل ۲-۱: شمای مداری آزمایش ۱

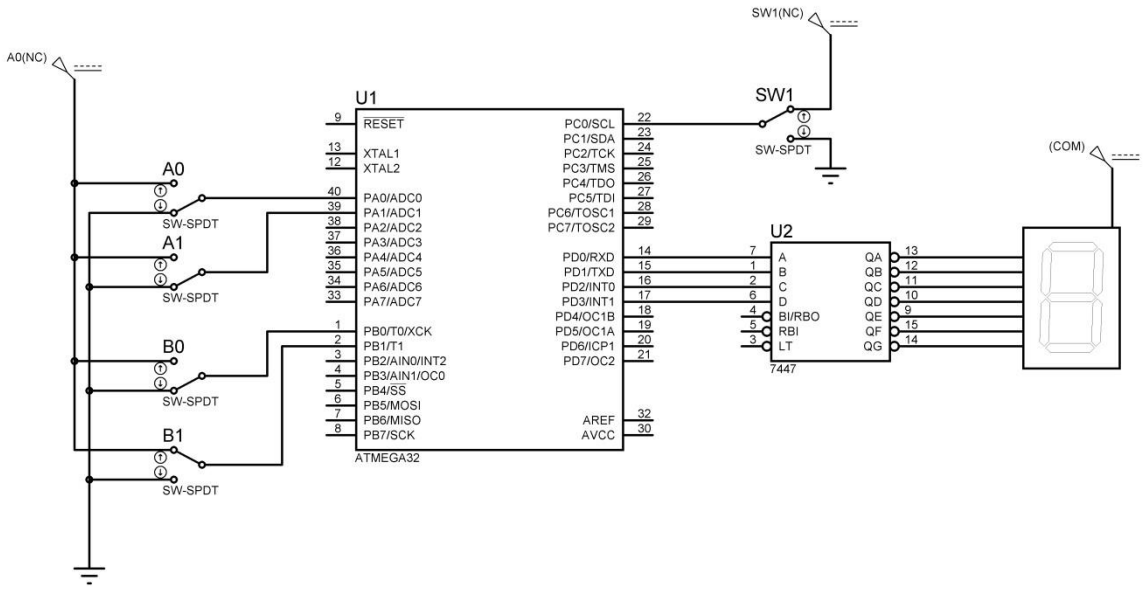
۲-۲ آزمایش دوم

برنامه ای بنویسید که داده ورودی به پایه های صفر و یک از PortA و داده ورودی به پایه های صفر و یک از PortB را برحسب وضعیت پایه صفر از PortC با یکدیگر جمع یا در هم ضرب نماید؛ به این صورت که اگر پایه صفر از PortC یک باشد دو عدد را با هم جمع و اگر صفر باشد دو عدد را در هم ضرب کند. بدیهی است که دو عدد ورودی دو بیتی

بوده و بیت کم ارزش آنها به پایه های صفر از PortB و PortC متصل است. سپس نتیجه عمل جمع یا ضرب بر روی یک 7-Segment که توسط یک IC دیکدر 7447 به PortD متصل شده است، نمایش داده شود.

توجه: برنامه را به گونه ای بنویسید که اگر در هر لحظه از اجرای شبیه سازی وضعیت Port C.0 یا اعداد ورودی به Port A و Port B تغییر کند، متناسب با آن، خروجی نمایش داده شده نیز تغییر کند.

شمای مداری این بخش از آزمایش در شکل ۲-۲ نشان داده شده است.



شکل ۲-۲: شمای مداری آزمایش ۲

** **نکته مهم:** در این آزمایش از دیکدر 7447 برای تبدیل عدد BCD به معادل 7-Segment آند مشترک استفاده شده است. اما ممکن است حالاتی رخ دهند که در آن ها نتوانید از دیکدر استفاده نمایید. در این صورت بایستی اعداد معادل 7-Segment را بر روی پورت میکروکنترلر قرار دهید.

آزمایش ۳ – اتصال LCD به میکروکنترلرهای AVR

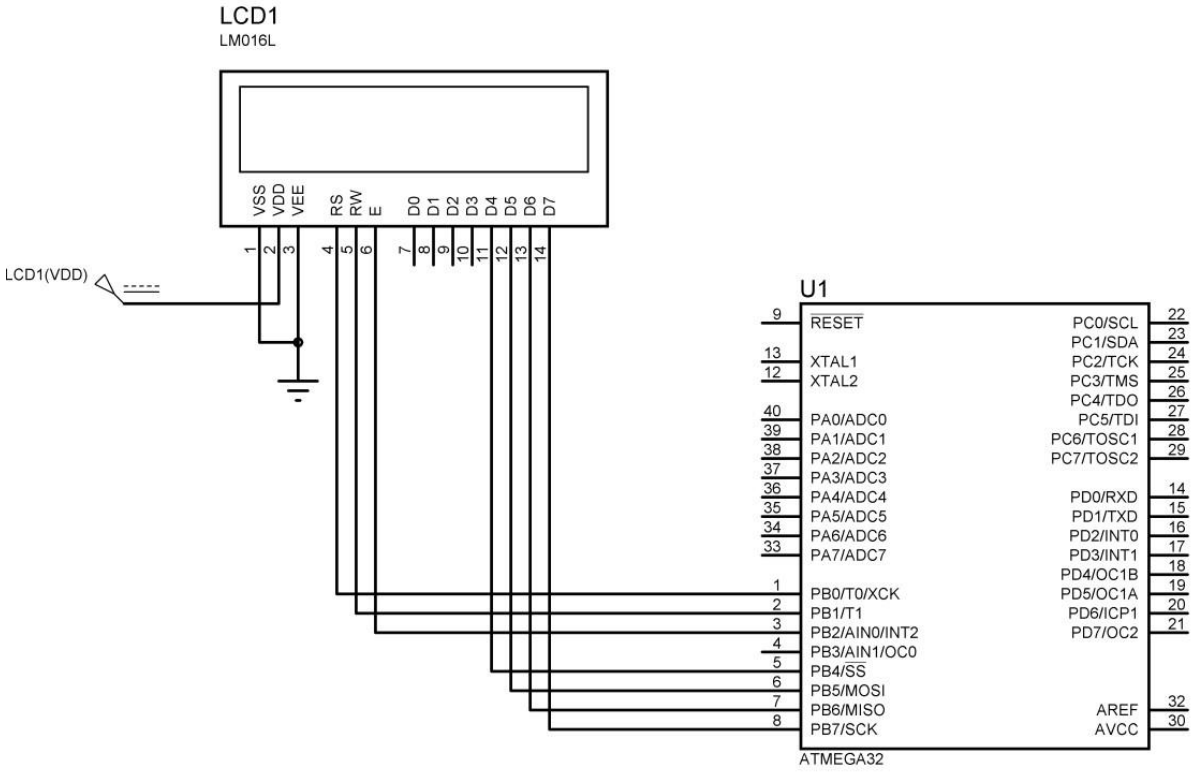
۳-۱ مقدمه

در سال های اخیر، برای نمایش اطلاعات میکروکنترلرها بیشتر از LCD استفاده می شود. دلیل آن، بهای پایین و امکان نمایش حروف، رقم و کاراکترهای گرافیکی است.

اطلاعات بر روی LCD ممکن است در یک، دو یا چهار سطر نوشته شود. این اطلاعات را می توان به صورت ۸ بیتی یا ۴ بیتی برای LCD ارسال نمود و با فرمان هایی که برای آن پیش بینی شده است، در هر لحظه می توان آن را پاک نمود، محل نمایش اطلاعات را تغییر داد، حروف را از طرف راست به چپ یا برعکس نوشت و بالاخره حروف یا مکان نما را روشن یا خاموش نمود.

۳-۲ آزمایش اول

در این آزمایش می خواهیم با اســـتفاده از میکروکنترلر ATmega32 برنامه ای بنویســـیم که توســـط آن، پیغام Microcontroller Course بر روی یک LCD نمایش داده شود. شمای مداری این آزمایش در شکل ۳-۱ نشان داده شده است.



شکل ۳-۱: شمای مداری آزمایش ۳

برای یادآوری و آشنایی بیشتر با دستورات LCD، برنامه این آزمایش در ادامه آورده شده است:

```
#include <mega32.h>
#include <stdio.h>
#include <delay.h>

#asm
.equ __lcd_port=0x18;PORTB
#endasm
#include <lcd.h>
 void main (void){
PORTB = 0x00;
DDRB = 0x00;
                  lcd_init(16);
lcd_clear();          lcd_gotoxy(0,0);
lcd_putsf("microcontroller”);
         lcd_gotoxy(5,1);
lcd_putsf("course”);
}
```

این برنامه را به دقت بررسی نموده و آزمایش را انجام دهید.

۳-۳ آزمایش دوم

در همان مدار نشان داده شده در شکل ۳-۱ برنامه ای بنویسید که رشته Microcontroller Course را به طور متناوب و به صورت حرف به حرف و با تأخیر زمانی اندکی بین نمایش حروف، بر روی LCD نمایش دهد.

۳-۴ آزمایش سوم

برنامه ای بنویسید که در هر لحظه، داده متصل به Port D را بخواند و بر روی LCD نمایش دهد.

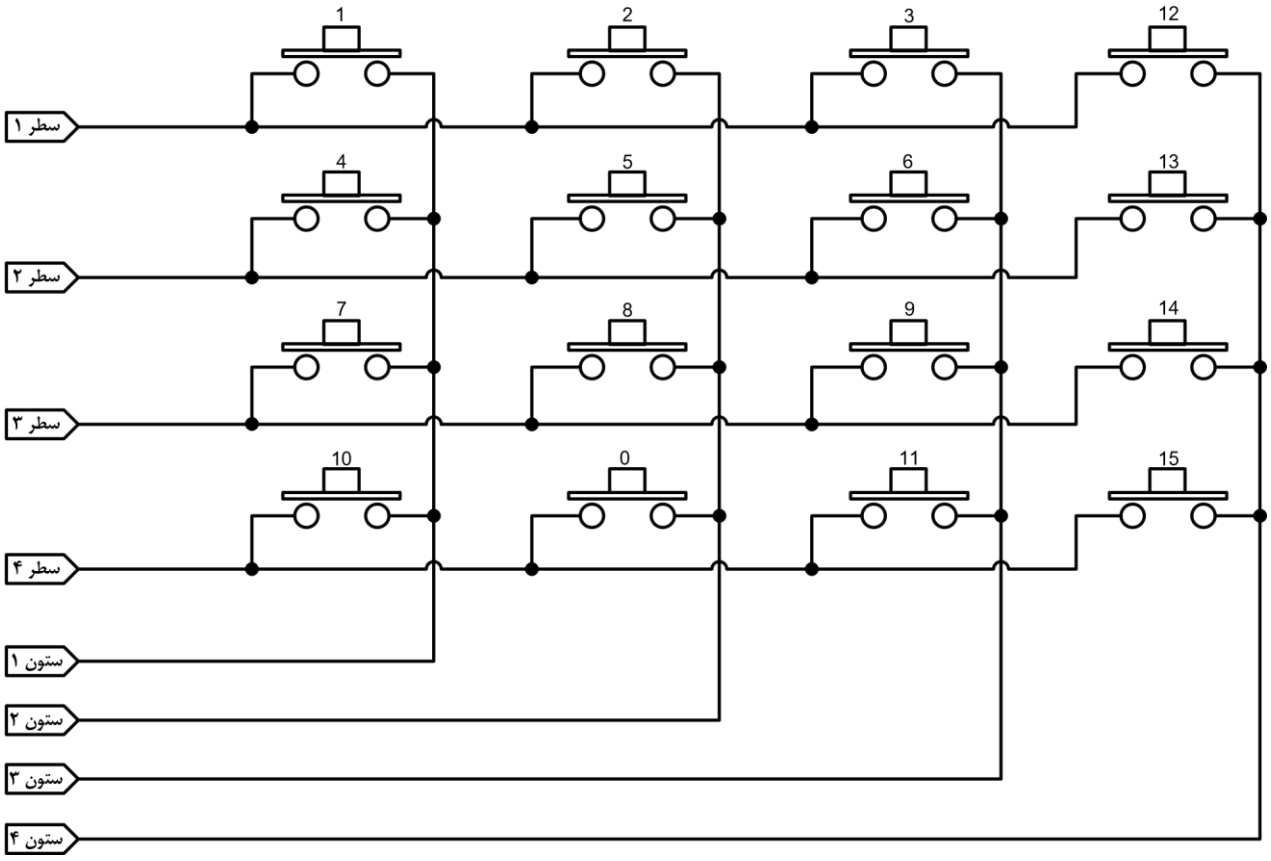
آزمایش ۴ – اتصال صفحه کلید به میکروکنترلرهای AVR

۴-۱ مقدمه

یکی از وسایلی که برای وارد کردن اطلاعات در میکروکنترلرها به کار می رود، صفحه کلید می باشد. ساختار صفحه کلید به صورت ماتریسی از سطر و ستون می باشد که با فشار هر کلید، یک سطر به یک ستون متصل می شود و در غیر این صورت، اتصالی بین سطر و ستون وجود ندارد.

برای خواندن از صفحه کلید توسط میکروکنترلر، معمولاً از روش اسکن صفحه کلید برای تشخیص کلید فشرده شده استفاده می شود. همانگونه که در شکل ۴-۱ نشان داده شده است. می توان ستون ها را با یک مقاومت به سطح ولتاژ Vcc متصل نمود تا میکروکنترلر هنگامی که کلیدی فشرده نشده است، سطح منطقی High را بخواند. سطرها با توجه به الگوی چرخشی←1110←1101←1011 ←1110 ... اسکن می شوند. زمانی میکروکنترلر می تواند در پایه یکی از ستون ها صفری تشخیص دهد که کلید فشرده شده، آن ستون را به سطر صفر شده وصل کند و با توجه به شماره سطر و ستون صفر شده، می توان کلید فشرده شده را تشخیص داد.

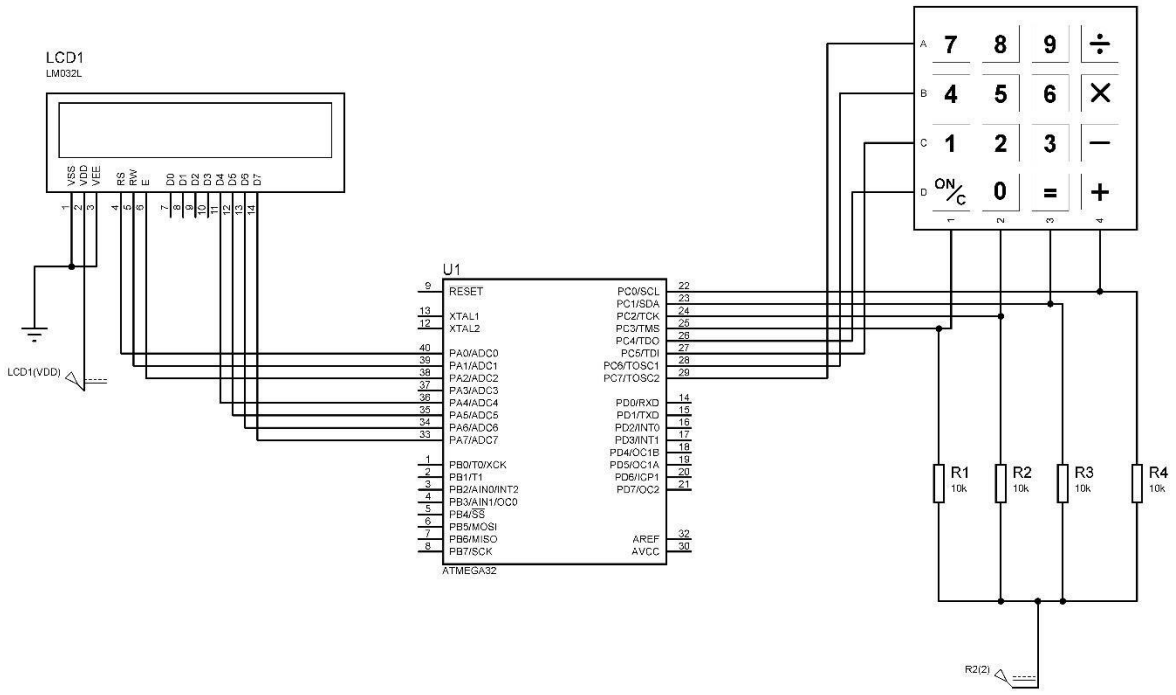
برای روشن شدن روش بالا، فرض کنید سطرهای یک صفحه کلید 4×4 به چهار بیت کم ارزش PortC و ستون های آن به چهار بیت پر ارزش PortC متصل شده اند. همچنین ستون ها توسط مقاومت هایی به Vcc وصل شده اند. حال اگر سطرها را صفر کنیم و هر یک از کلیدهای یک سطر را فشار دهیم، اتصال بین یک سطر و ستون برقرار می شود و ستون مربوطه نیز صفر می شود. به عنوان مثال، در شکل ۴-۱ اگر سطر یک را برابر صفر کرده باشیم، در این صورت اگر هر یک از کلیدهای 1، 2، 3 یا 12 را فشار دهیم، یکی از ستون های 1 تا 4 نیز مساوی صفر می شود؛ بنابراین اگر سطرها یا چهار بیت پر ارزش PortC یعنی PC4 تا PC7 را صفر کنیم، لذا با فشار یک کلید، ستون 1 برابر با صفر و بقیه ستون ها مساوی 1 می شوند یعنی مقدار ستون های PC0 تا PC3 برابر با 0111 می شود به این ترتیب کد کلیدهای سطر یک برابر با 00000111 باینری یا 0x07 در مبنای شانزده می باشد. حال با توجه به ستون صفر شده می توان تشخیص داد که کدام یک از کلیدهای سطر یک فشار داده شده است.



شکل ۴-۱: صفحه کلید ۴×۴

۴-۲ آزمایش اول

برنامه ای بنویسید که کلید فشرده شده در یک صفحه کلید 4×4 متصل به PortC را بر روی یک LCD متصل به PortA نمایش دهد. شمای مداری این آزمایش در شکل ۴-۲ نشان داده شده است.



شکل ۴-۲: شمای مداری آزمایش ۴

برنامه ای که می توان برای اسکن صفحه کلید به کار برد، در ادامه آورده شده است:

```
#include <mega32.h>
#include <stdio.h>
#include <delay.h>

unsigned char scan_key(void); unsigned char code[4][4]={
{7,4,1,10},{8,5,2,0},{9,6,3,11},{12,13,14,15}
};
.
.
.
.
unsigned char scan_key(void)  {
unsigned char i,data,num_key,temp;
 num_key=0xff;
temp=0x70;
for(i=0;i<4;i++){
  PORTC = temp;
delay_ms(5);     data=PINC &
0x0f;   if(data==0x07)
num_key=code[0][i];
if(data==0x0B)
num_key=code[1][i];
if(data==0x0D)
num_key=code[2][i];
if(data==0x0E)
num_key=code[3][i];
  temp= ((temp>>=1)  | 0x80) & 0xF0 ;

 }
 return num_key; }
```

این برنامه را به دقت بررسی نموده و از آن به عنوان یک تابع در برنامه خود استفاده کنید به این صورت که مقدار بازگردانده شده توسط تابع مذکور را به عنوان شماره کلید فشرده شده به کار ببرید.

۳-۴ پرسش

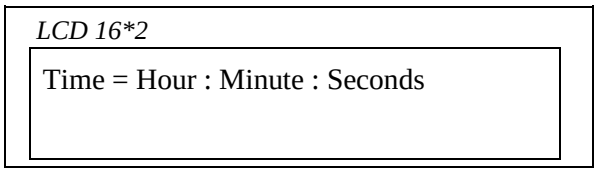
۱- همین برنامه را برای یک صفحه کلید 4×3 بنویسید.

۲- با استفاده از صفحه کلید 4×4 برنامه ای بنویسید که دو عدد را گرفته و با توجه به کلید فشرده شده در ستون چهارم (یعنی کلیدهای تقسیم، ضرب، تفریق و جمع) عمل متناظر با هر کلید را انجام داده و با فشردن کلید = نتیجه محاسبه را بر روی LCD نمایش دهد. همچنین اگر کلید ON/C فشار داده شود، صفحه نمایش LCD پاک شود.

آزمایش ۵ – ساعت دیجیتال به کمک میکروکنترلر AVR

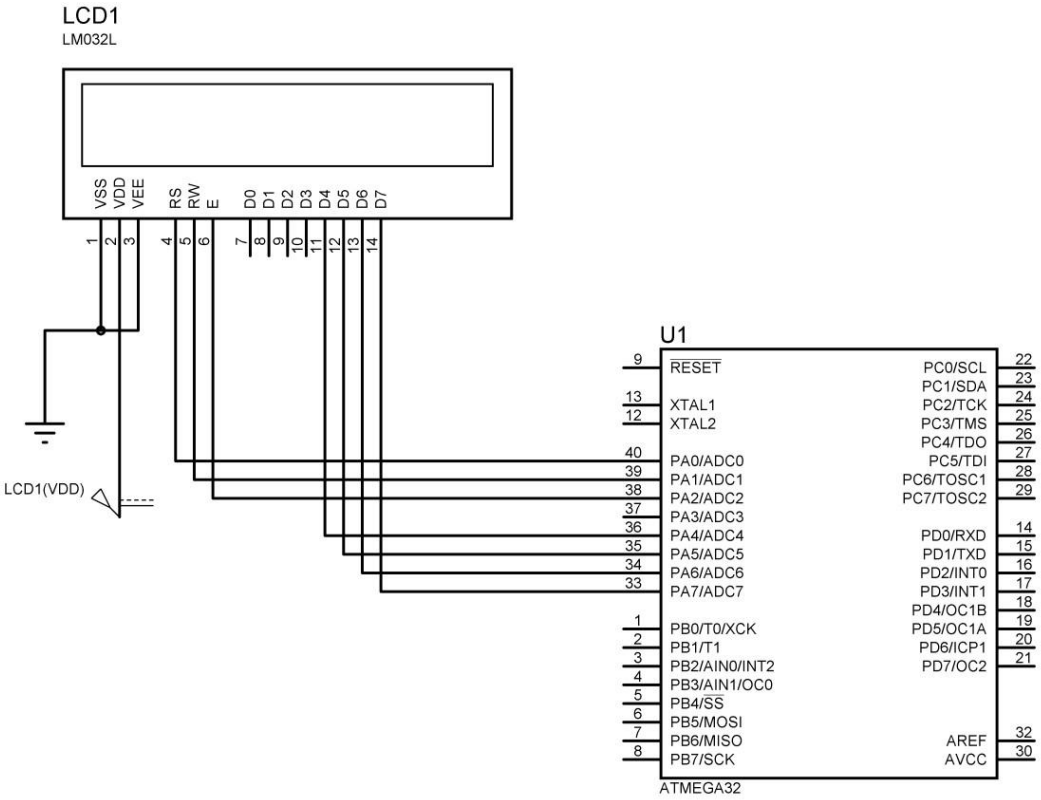
۵-۱ آزمایش اول

برنامه ای بنویسید که زمان به صورت ساعت، دقیقه و ثانیه بر روی یک LCD متصل به PortA نمایش داده شود و با شمارش و افزایش یک واحدی ثانیه شمار، دقیقه و ساعت نیز update شوند. برنامه را به گونه ای بنویسید که زمان به صورت نشان داده شده در شکل ۵-۱ بر روی LCD نمایش داده شود.



شکل ۵-۱: چگونگی نمایش زمان بر روی LCD

برای ایجاد تأخیر، از دستور delay_ms() استفاده کنید. شمای مداری این آزمایش در شکل ۵-۲ نشان داده شده است.



شکل ۵-۲: شمای مداری آزمایش ۵

۵-۲ آزمایش دوم

برنامه آزمایش اول را به گونه ای بنویسید که در سطر دوم LCD شماره روز و ماه نیز نمایش داده شود.

۵-۳ پرسش

۱- برنامه ساعت دیجیتال را به گونه ای بنویسید که قبل از شروع به کار ساعت، تنظیم اولیه ساعت، دقیقه، ثانیه، روز و ماه توسط یک صفحه کلید انجام شود.

آزمایش ۶ – آشنایی با تایمر و کانتر در میکروکنترلر AVR

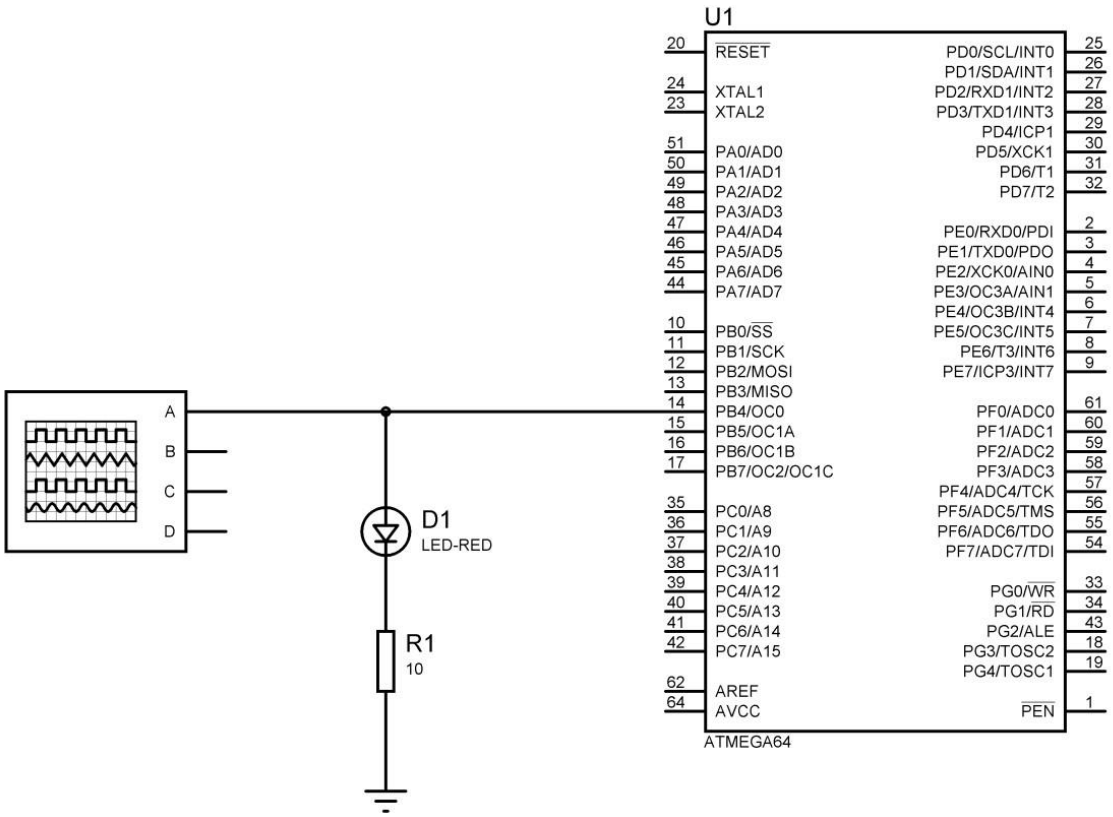
۶-۱ مقدمه

تایمر/کانتر یکی از بخش های مهم میکروکنترلرها می باشد. در بیشتر مواقع لازم که تعدادی وقایع خارجی (با سرعت بالا) شمارش شود و یا گاهی لازم است که در یک زمان خاص و دقیق، کاری صورت گیرد. تنها توسط تایمر/کانتر ها می توان این کارهای دقیق و با سرعت بالا را انجام داد.

میکروکنترلرهای AVR حداکثر دارای شش عدد تایمر، کانتر هشت بیتی و شانزده بیتی هستند. برخی از آن ها دارای عملکرد ساده و برخی دیگر دارای امکانات بیشتر نظیر تولید موج PWM، حالت مقایسه CTC، حالت تسخیر، عملکرد غیر همزمان و ... می باشند.

۶-۲ آزمایش اول

ابتدا مدار شکل ۶-۱ را ببندید.



شکل ۶-۱: مدار قسمت اول آزمایش ۶

با استفاده از برنامه زیر می توان هر 1 ms پایه OC0 (PORTB.4 در ATmega64) را مکمل کرد. فرکانس پالس ساعت سیستم برابر با 8 MHz فرض شده است.

```
#include <mega64.h>
void main(void){
DDRB=0x10;
TCNT0=0x00;
OCR0=0x7C;
TCCR0=0x1C;
while(1);
}
```

در برنامه بالا مراحل زیر انجام میشود:

۱- تعیین بیت چهارم PortB به عنوان خروجی جهت فعال شدن OC0 به صورت خروجی.

۲- مقداردهی اولیه در رجیستر تایمر صفر (TCNT0) با مقدار 00H.

۳- مقداردهی رجیستر مقایسه خروجی با 7CH (124).

۴- تنظیم فرکانس پالس ساعت تایمر صفر روی 125 KHz (با فرض فرکانس پالس ساعت سیستم 8 MHz) و قرار دادن تایمر صفر در حالت عملکرد مقایسه و تنظیم عملکرد شکل موج روی پایه OC0 به صورت مکمل (Toggle) و شروع شمارش تایمر.

۶-۳ آزمایش سوم

با استفاده از تایمر صفر برنامه ای بنویسد که اعداد صفر تا 9 را به ترتیب بر روی یک 7segment و با فاصله زمانی 0/25 ثانیه نشان دهد. فرکانس پالس ساعت را برابر با 1 MHz و N را برابر با 1024 در نظر بگیرید.

۶-۴ پرسش

۱- در برنامه قسمت دوم آزمایش، منظور از خطای ذکر شده چیست و چگونه می توان آن را برطرف کرد؟ (برنامه آن را بنویسید).

۲- منظور از بافر مضاعف چیست؟

۳- در برنامه ساعت دیجیتال در آزمایش ۵ برای ایجاد تأخیر به جای استفاده از دستور delay_ms() از تایمر استفاده کنید.