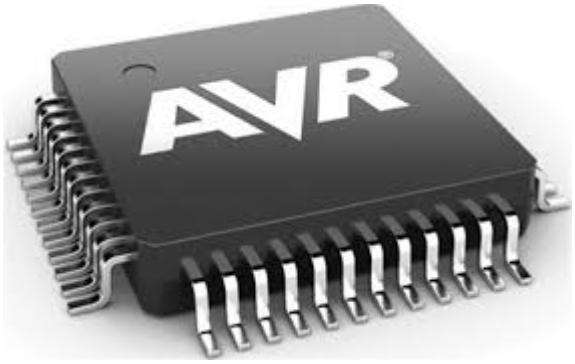
وزارت علوم ، تحقیقات و فناوری
دانشگاه جهرم
دانشکده فنی و مهندسی

آزمایشگاه ریزپردازنده

(با رویکرد برنامه‌نویسی به زبان بیسیک و استفاده از نرم‌افزار BASCOMAVR)



گردآورنده :

محمّدعلی شفیعیان

آزمایشگاه ریزپردازنده

آزمایش شماره ۲

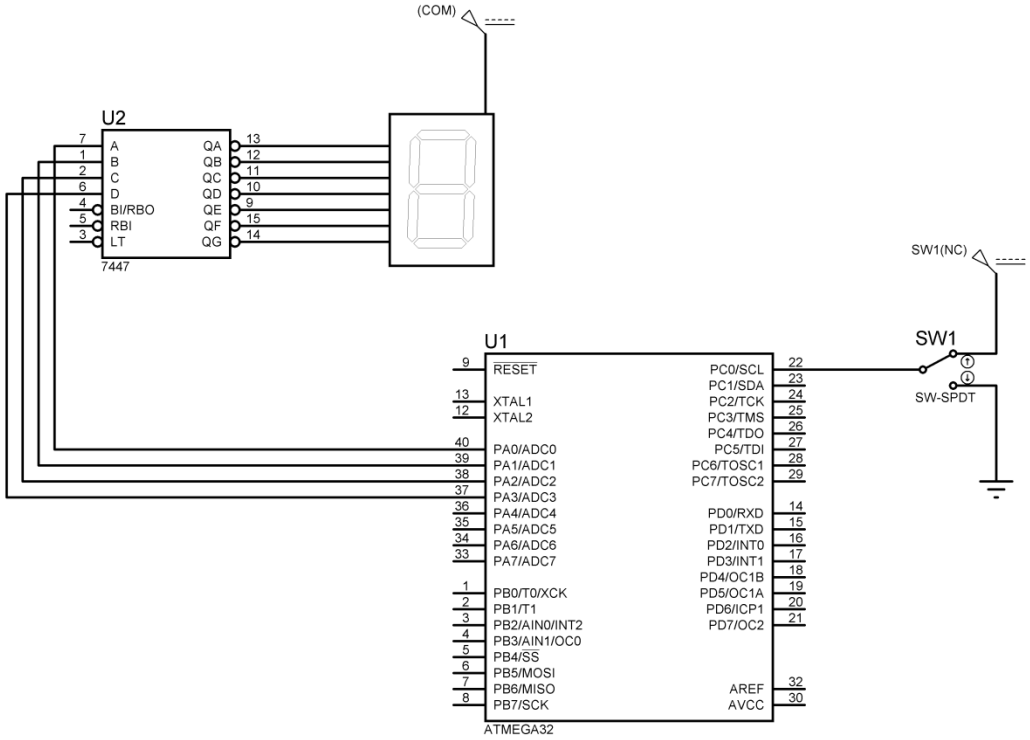
نحوه استفاده از *7-Segment* و ارتباط آن با میکروکنترلر

آزمایش ۲–۱

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 ۷۴۴۷ به *Port A* متصل شده است، نمایش دهد.

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

توجه : *IC* دیکدر ۷۴۴۷ 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 ۱۵۰ Ω به *7-Segment* متصل می شوند. شمای مداری این بخش از آزمایش در شکل ۱ نشان داده شده است.



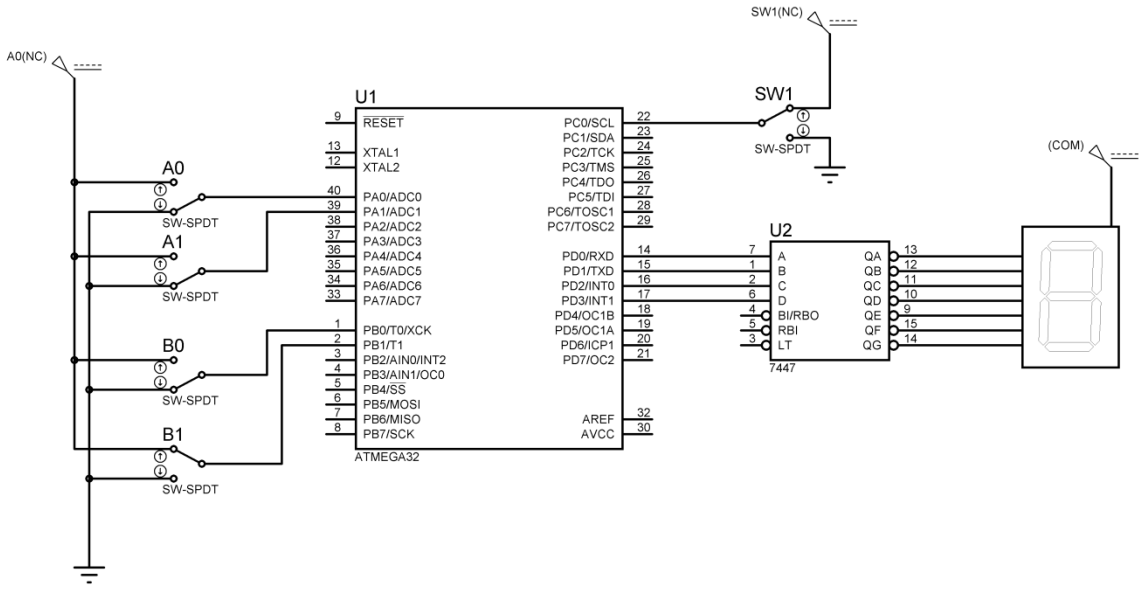
شکل ۱ : شمای مداری آزمایش ۲–۱

آزمایش ۲–۲

برنامه ای بنویسید که داده ورودی به پایه های صفر و یک از *Port A* و داده ورودی به پایه های صفر و یک از *Port B* را برحسب وضعیت پایه صفر از *Port C* با یکدیگر جمع یا در هم ضرب نماید؛ به این صورت که اگر پایه صفر از *Port C* 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 *Port B* و *Port C* متصل است. سپس نتیجه عمل جمع یا ضرب بر روی یک *7-Segment* که توسط یک *IC* دیکدر ۷۴۴۷ به *Port D* 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 نشان داده شده است.



شکل ۲ : شمای مداری آزمایش ۲–۲

پرسش :

۱ـ با توجه به آزمایش ۲–۲ برنامه ای بنویسید که با توجه به وضعیت *Port C.0* دو عدد ۴ بیتی ورودی به *Port A* و *Port B* را با هم جمع یا در هم ضرب نماید و نتیجه را (که ممکن است عددی دو رقمی باشد) بر روی دو عدد *7-Segment* متصل به *Port D* نمایش دهد. همچنین اگر حاصل ضرب دو عدد بزرگتر از ۹۹ بود، بر روی دو *7-Segment* عدد یا حرفی نمایش داده نشود.

آزمایشگاه ریزپردازنده

آزمایش شماره ۳

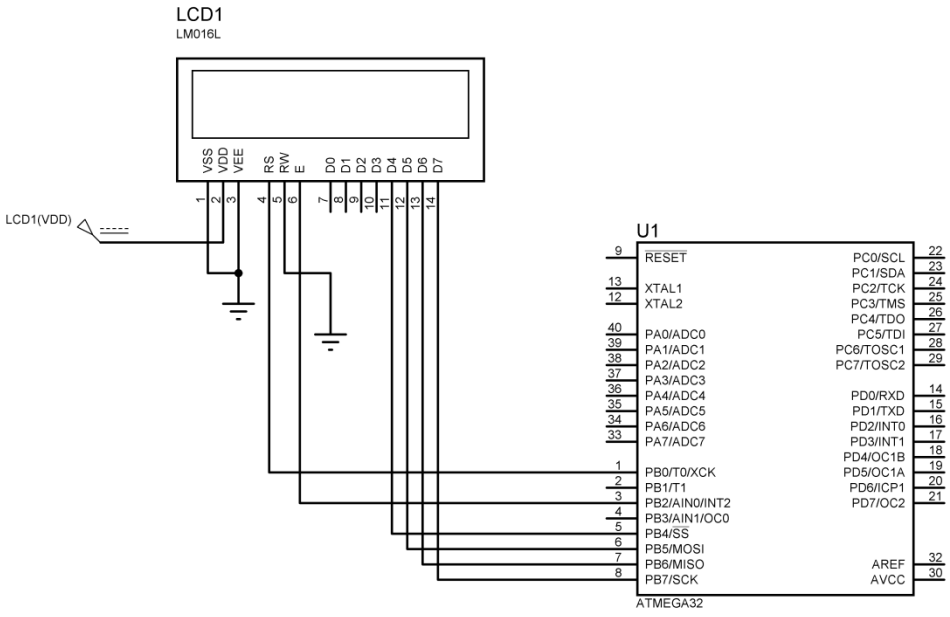
اتصال *LCD* به میکروکنترلرهای *AVR*

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 بالأخره حروف یا مکان نما را روشن یا خاموش نمود.

آزمایش ۳-۱

در این آزمایش می خواهیم با استفاده از میکروکنترلر *ATmega32* برنامه ای بنویسیم که توسط آن، پیغام *Microcontroller* بر روی یک *LCD* نمایش داده شود. شمای مداری این آزمایش در شکل ۱ نشان داده شده است.



شکل ۱ : شمای مداری آزمایش ۳-۱

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

```
$regfile = "M32def.dat "
$crystal = 8000000

Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 , E = Portb.2 , Rs = Portb.0
Config Lcd = 16 * 2

Cls
Home
Lcd "Microcontroller"
End
```

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

آزمایش ۳–۲

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

آزمایش ۳–۳

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

پرسش :

۱ـ برنامه ای بنویسید که یک رشته (مانند نام و نام خانوادگی خودتان) را به صورت متحرک بر روی *LCD* نمایش دهد به این صورت که رشته مذکور از سمت راست *LCD* وارد شود و از راست به چپ *LCD* را طی کند و از سمت چپ خارج شود.

آزمایشگاه ریزپردازنده

آزمایش شماره ۴

اتصال صفحه کلید به میکروکنترلرهای *AVR*

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. در شکل ۱ نمای داخلی یک صفحه کلید ۴×۴ نشان داده شده است. اسکن صفحه کلید در *BASCOM* ساده است و تنها کافی است صفحه کلید خود را طبق مدار شکل ۳ به یکی از پورت های میکرو متصل نموده و توسط دستور *CONFIG KBD* آن را پیکربندی کنید. یک صفحه کلید توسط دستور زیر پیکربندی می شود :

```
CONFIG KBD = PORTx , DEBOUNCE = value [,DELAY = value]
```

که در آن *PORTx* مشخص‌کننده پورتی است که صفحه کلید به آن متصل می‌شود. *DEBOUNCE* به‌صورت پیش‌فرض ۲۰ است و می‌تواند بیشینه مقدار ۲۵۵ را داشته باشد. *DELAY* نیز پارامتری اختیاری است و مشخص‌کننده تأخیر بر حسب میلی‌ثانیه است که در زمان خواندن کلید توسط دستور *GETKBD ( )* ایجاد می‌شود. این گزینه برای کاهش نویزهای محیط ایجاد شده است و به‌طور مثال، مقدار *DELAY=100* می‌تواند این مشکل را برطرف کند.

پیکربندی فوق را می توان به صورت ساده شده زیر نیز نوشت ولی باید برای ایجاد تأخیر در زمان اسکن صفحه کلید، از دستوات تأخیر استفاده نمود :

```
CONFIG KBD = PORTx
```

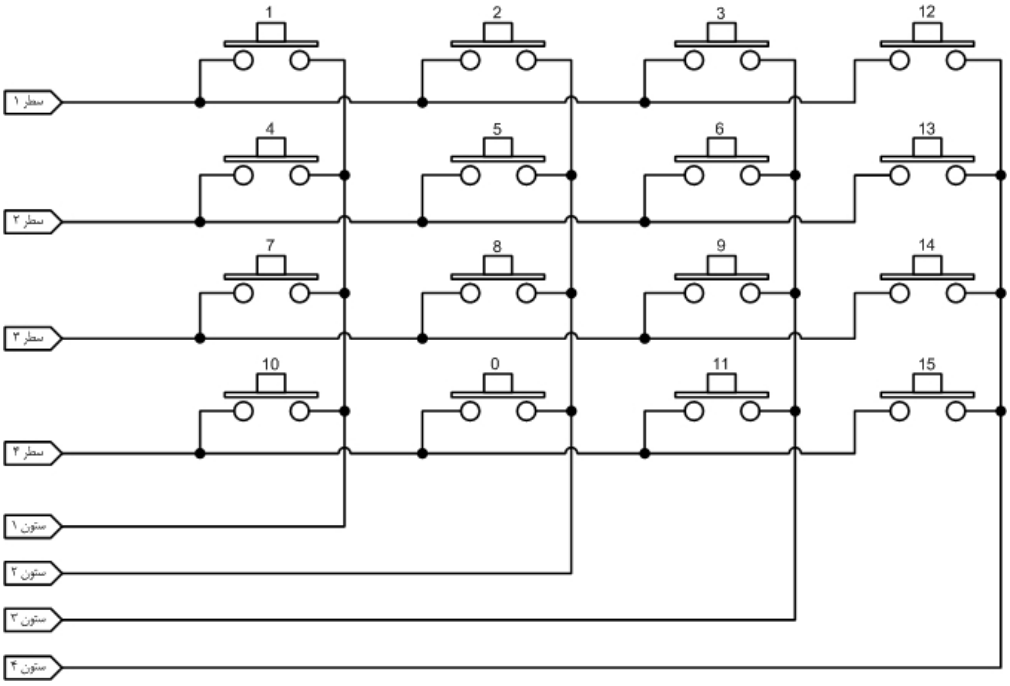
همچنین زمانی که نیاز باشد صفحه کلیدی با ۶ سطر را بخوانید، می توان از دستور زیر به جای دستور بالا استفاده نمود :

```
CONFIG KBD = PORTx , DEBOUNCE = value , ROWS = 6 , ROW5 = PINX.Y , ROW6 = PINA.B
```

در این حالت، سطر پنجم به پایه *PINX.Y* وسطر ششم به پایه *PINA.B* اتصال می یابد؛ به عنوان مثال :

```
CONFIG KBD = PORTC , DEBOUNCE = 50 , ROWS = 6 , ROW5 = PIND.6 , ROW6 = PIND.7
```

با توجه به پیکربندی بالا، سطر پنجم و ششم به ترتیب به پایه‌های *PIND.6* و *PIND.7* متصل می‌شوند.



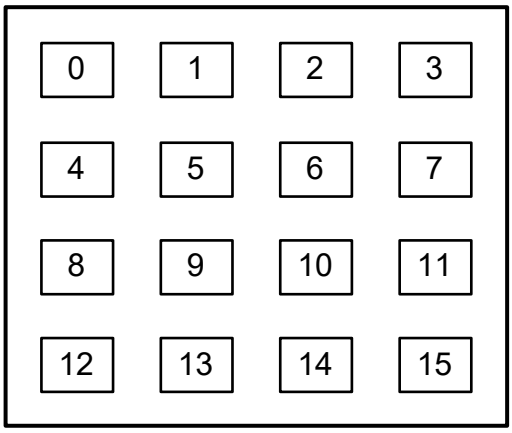
شکل ۱ : صفحه کلید ۴×۴

ـ دستور *GETKBD ()*

توسط این دستور، می توان صفحه کلید را خواند که صورت کلی آن به‌صورت زیر است :

```
SOURCE = GETKBD ()
```

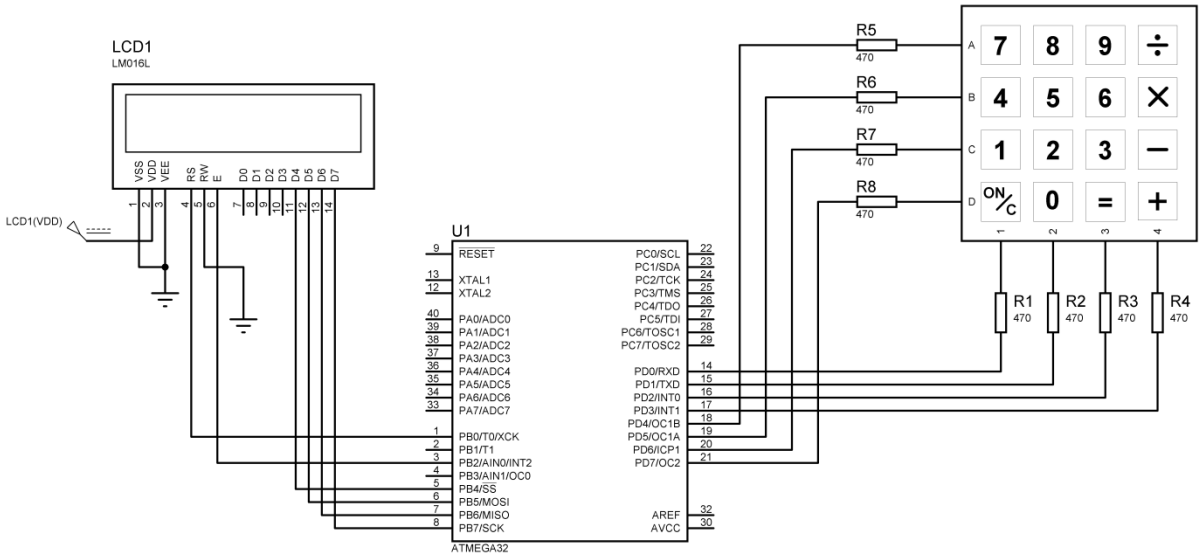
با اجرای این دستور، میکرو صفحه کلید را خوانده و عدد متناظر با کلید فشرده شده را در متغیر *SOURCE* قرار می دهد. این دستور زمانی که کلیدی فشرده نشده باشد، عدد ۱۶ را برمی‌گرداند. با استفاده از جدول *Lookup* می‌توان عدد به‌دست آمده از صفحه کلید را به مقدار دلخواه خود تبدیل نمود. در صورت اتصال صفحه کلید طبق مدار شکل ۳، عدد متناظر با کلید فشرده شده توسط دستور *GETKBD ( )* به صورت شکل ۲ خواهد بود.



شکل ۲ : عدد متناظر با کلید فشرده شده توسط دستور *( ) GETKBD*

آزمایش ۴–۱

برنامه ای بنویسید که کلید فشرده شده در یک صفحه کلید ۴×۴ متصل به *PortD* را بر روی یک *LCD* متصل به *PortB* نمایش دهد. شمای مداری این آزمایش در شکل ۳ نشان داده شده است.



شکل ۳ : شمای مداری آزمایش ۴–۱

برنامه ای که می توان برای اسکن و خواندن صفحه کلید و نمایش عدد به دست آمده روی *LCD* به کار برد، در ادامه آورده شده است :

```
$regfile = "M32def.dat "
$crystal = 8000000

Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 , E = Portb.2 , Rs = Portb.0
Config Lcd = 16 * 2
Config Kbd = Portd , Debounce = 50 , Delay = 20
```

```
Dim A As Byte
Main:
    A = Getkbd ()
    If A > 15 Then Goto Main
    Cls
    Home
    Lcd A
JMP Main
End
```

این برنامه را به دقت بررسی نموده و بررسی نمایید که عدد نمایش داده شده بر روی *LCD* با اعداد مورد انتظار در شکل ۲ یکسان است یا خیر. اگر ترتیب اتصال سطر یا ستون‌ها به پایه‌های میکرو را تغییر دهیم، چه اتفاقی می‌افتد ؟

آزمایش ۴–۲

با استفاده از جدول *Lookup* برنامه ای بنویسد که در مدار شکل ۳ اگر یکی از اعداد فشرده شود، عدد متناظر روی *LCD* نمایش داده شود. برنامه را به گونه ای بنویسید که کلید تقسیم به معنای *Backspace*، کلید ضرب به معنای *Space*، کلید تفریق به معنای ابتدای *LCD*، کلید جمع به معنای انتهای خط اول، کلید مساوی به معنای ابتدای خط دوم و کلید *ON/C* به معنای *CLS* (پاک کردن صفحه نمایش *LCD*) باشد.

پرسش :

۱ـ برنامه آزمایش ۴–۱ را برای یک صفحه کلید ۴×۳ بنویسید.

۲ـ با استفاده از صفحه کلید ۴×۴ 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.

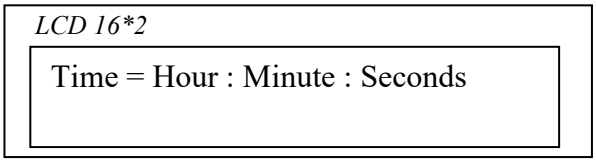
آزمایشگاه ریزپردازنده

آزمایش شماره ۵

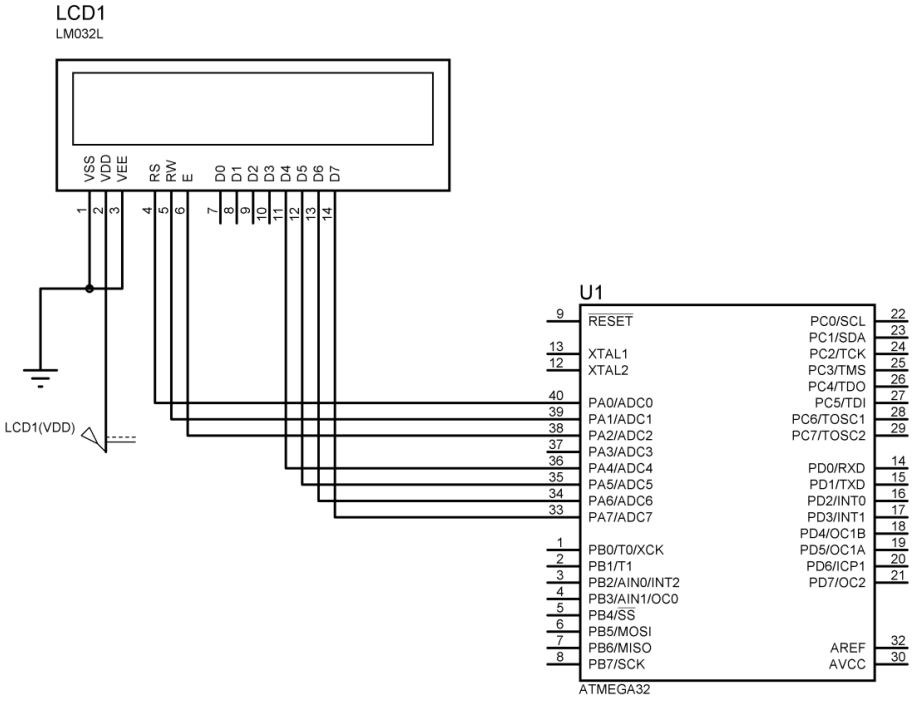
ساعت دیجیتال به کمک میکروکنترلر *AVR*

آزمایش ۵–۱

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 زیر بر روی *LCD* نمایش داده شود :



برای ایجاد تأخیر، از دستور *waitms()* استفاده کنید. شمای مداری این آزمایش در شکل ۱ نشان داده شده است.



شکل ۱ : شمای

مداری آزمایش ۵–۱

آزمایش ۵–۲

برنامه آزمایش ۵–۱ را به گونه ای بنویسید که در سطر دوم LCD شماره روز و ماه نیز نمایش داده شود.

پرسش ها :

۱ـ 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.

آزمایشگاه ریزپردازنده ها

آزمایش شماره ۶

آشنایی با تایمر و کانتر در میکروکنترلر *AVR*

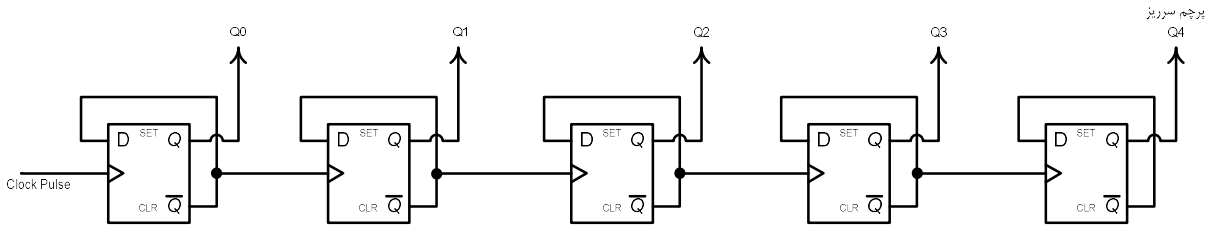
مقدمه

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 آنها دارای عملکرد ساده و برخی دیگر دارای امکانات بیشتر نظیر تولید موج *PWM* ، حالت مقایسه *CTC* ، حالت تسخیر، عملکرد غیر همزمان و ... می باشند.

۱ ـ تئوری عملکرد تایمر/کانتر

تایمر و کانترها از تعدادی فلیپ فلاپ که به صورت پشت سر هم قرار گرفته و به صورت یک شمارنده آسنکرون عمل می کنند، تشکیل شده اند (شکل ۱).



شکل ۱ : شمارنده آسنکرون

در این شمارنده پایه های *Q3-Q0* به عنوان خروجی استفاده می شود. با فرض وارد نمودن مقدار *0000* و با اعمال پالس ساعت، شمارنده شروع به شمارش می نماید و حداکثر تا مقدار *1111* بالا می رود. پس از این حالت و با اعمال پالس ساعت بعدی، بیت سرریز (*Q4*) فعال می شود که نشان دهنده این است که شمارنده به حداکثر مقدار خود رسیده است.

هنگامی که پالس ساعت اعمالی به این شمارنده یک مقدار مشخص داشته باشد، می توان با در نظر گرفتن مدت زمان شمارش از مقدار *xxxx* تا *1111* و سپس سرریز شدن شمارنده، زمان های معینی را ایجاد نمود. در این حالت شمارنده به صورت تایمر عمل می کند. به عنوان نمونه برای شکل ۱ اگر پالس ساعت اعمالی دارای فرکانس *Hz* ۲ باشد و شمارنده نیز با مقدار پیش فرض *0000* در نظر گرفته شود، حداکثر ۸ ثانیه طول می کشد تا شمارنده سرریز شود.

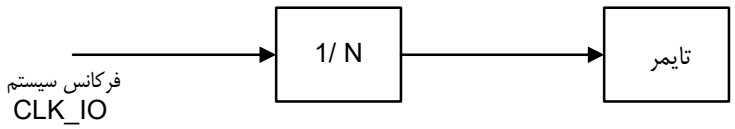
هنگامی که منبع پالس ساعت اعمالی از یک منبع منظم و معین نباشد (منبع خارجی)، در این صورت می توان از شمارنده برای شمارش وقایع خارجی نظیر عبور تعداد قطعات از جلوی یک سنسور نوری استفاده نمود که در این حالت شمارنده به صورت کانتر به کار گرفته می شود.

۲ ـ نحوه محاسبه زمان بندی برای تایمرها

برای محاسبه زمان بندی مورد نظر در تایمرها باید مراحل زیر انجام پذیرد :

۱ ـ فرکانس داخلی پالس ساعت سیستم را در نظر گرفت.

۲ ـ ضریب تقسیم پالس ساعت سیستم برای تولید پالس ساعت تایمر توسط بیت های *CS00*، *CS01* و *CS02* را تعیین نمود (شکل ۲).



شکل ۲ : تعیین ضریب تقسیم پالس ساعت تایمر

۳ ـ مقدار مطلوب را رجیستر *TCNTx* قرار داد (*x* شماره تایمر مورد نظر است).

مثال ۱ : با فرض نوسان ساز داخلی *MHz* ۱ و ضریب تقسیم *N*=۳۲ و *TCNT0=0AH* مقدار زمان تا فعال شدن پرچم سرریز را محاسبه کنید.

فرکانس پالس ساعت تایمر = $\frac{1\ MHz}{32} = 31.25\ KHz$

مدت زمان یک شمارش = $\frac{1}{32.25\ KHz} = 32\ \mu s$

تعداد پالس برای سرریز شدن تایمر = $256 - TCNT0 = 256 - 10 = 246$

مدت زمان شمارش تایمر = $246 \times 32\ \mu s = 7.872\ ms$

مثال ۲ : با فرض استفاده از کریستال خارجی $16\ MHz$ برای تولید مدت زمان $10\ ms$ چه عددی را باید در تایمر یک ($TCNT1$) قرار داد ؟

با فرض انتخاب ضریب تقسیم ۶۴ (N=۶۴) داریم :

فرکانس پالس ساعت تایمر = $\frac{16\ MHz}{64} = 250\ KHz$

مدت زمان یک شمارش = $\frac{1}{250\ KHz} = 4\ \mu s$

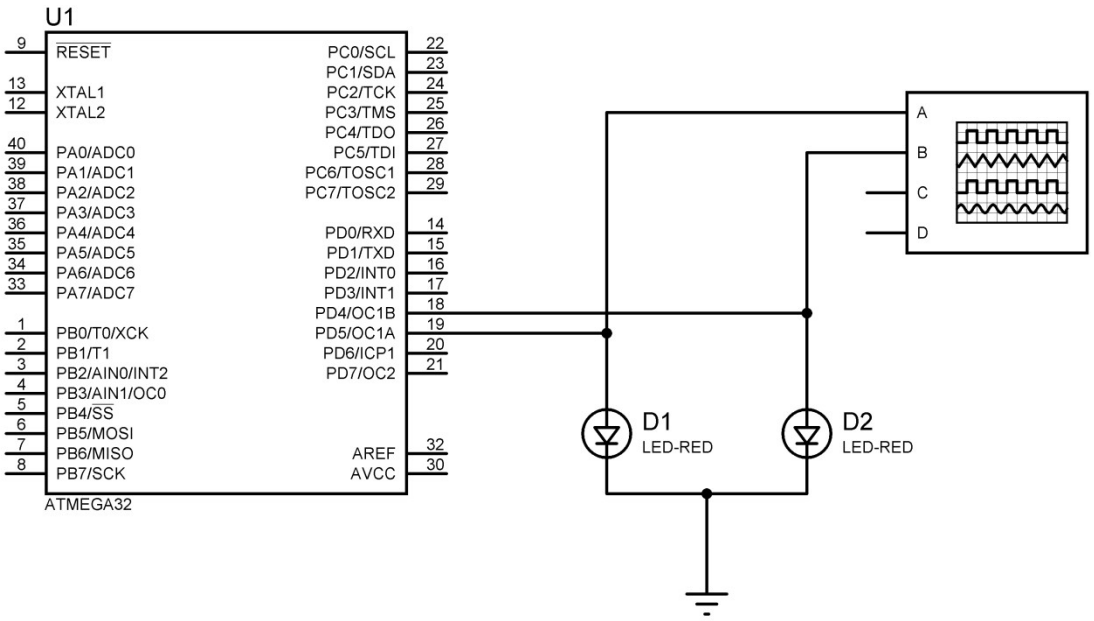
تعداد پالس لازم برای مدت $10\ ms$ = $\frac{10\ ms}{4\ \mu s} = 2500$

عدد لازم برای $TCNT1$ = $65536 - 2500 = 63036 = F63CH$

شرح آزمایش

ـ قسمت اول :

ابتدا مدار شکل ۲ رابیندید.



شکل ۲ : مدار قسمت اول آزمایش

۲ ـ با استفاده از برنامه زیر می توان یک موج PWM ایجاد نمود. فرکانس خروجی آن را به دست آورید

```
$regfile = "M32def.dat"
$crystal = 8000000

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Up , Compare B Pwm = Clear
Down , Prescale = 8
```

```
Config Pind.4 = Output
Config Pind.5 = Output

Do
  Pwm1a = 100
  Pwm1b = 200
Loop

End
```

ـ قسمت دوم :

با استفاده از تایمر یک و در مد *PWM* برنامه ای بنویسد که اعداد صفر تا ۹ را به ترتیب بر روی یک *7segment* و با فاصله زمانی ۰/۲۵ ثانیه نشان دهد.

پرسش :

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

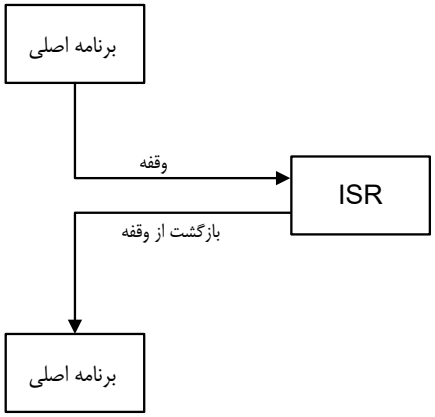
آزمایشگاه ریزپردازنده ها

آزمایش شماره ۷

آشنایی با وقفه و سازمان وقفه در میکروکنترلر AVR

مقدمه

وقفه، مکانیزمی از میکروکنترلر است که میکروکنترلر را برای پاسخ گویی به برخی از وقایع لحظه ای فعال می کند. وقفه ها نقش مهمی در میکروکنترلرها ایفا می کنند. در زمان رخداد یک وقفه، برنامه در حال اجرا متوقف شده و زیربرنامه وقفه شروع به اجرای برنامه خود می کند و پس از پایان زیربرنامه موردنظر، اجرای برنامه اصلی ادامه می یابد. برنامه ای را که CPU در زمان رخداد یک وقفه به آن رجوع می کند، روال سرویس وقفه (ISR) می نامند. شکل ۱ نحوه عملکرد میکروکنترلر را به هنگام رخداد وقفه نشان می دهد.



شکل ۱ : عملکرد میکروکنترلر هنگام رخداد وقفه ها

۱ ـ مراحل اجرای یک وقفه

میکروکنترلر به محض دریافت یک وقفه، مراحل زیر را انجام می دهد :

۱ ـ دستوری که در حال اجرا می باشد را پایان داده و آدرس دستورالعمل بعدی را در پشته ذخیره می نماید.

۲ ـ رجوع به جدول بردار وقفه (محلی از حافظه) و به دست آوردن آدرس سرویس وقفه (ISR) و پرش به آن آدرس.

۳ ـ شروع مرحله اجرای ISR تا رسیدن به دستور بازگشت از وقفه ($RETI$).

۴ ـ برداشتن آدرس از پشته و قرار دادن آن در شمارنده برنامه (PC) و اجرای ادامه برنامه.

۲ ـ سازمان وقفه در *AVR*

در میکروکنترلرهای *AVR* منابع وقفه با توجه به نوع میکروکنترلر متفاوت است. به عنوان نمونه در میکروکنترلر *ATmega8* تعداد ۹ منبع وقفه و در *ATmega32* تعداد ۲۱ منبع وقفه وجود دارد.

۲-۱ بردار وقفه

هنگام رخداد یک وقفه، آدرسی که در شمارنده برنامه قرار می گیرد را بردار وقفه می نامند. این آدرس، همان آدرس شروع *ISR* مربوط به منبع وقفه است. در میکروکنترلرهای *AVR* برخلاف دیگر خانواده میکروکنترلرها نظیر ۸۰۵۱ که بردارهای وقفه در یک مکان ثابت و مشخص در ابتدای حافظه برنامه قرار داشتند، این قابلیت به کاربر داده شده است که مکان بردار وقفه را بین دو بخش حافظه برنامه کاربردی و *BOOT* حرکت دهد. پس با توجه به این موضوع می توان میکروکنترلرهای *AVR* را به صورت زیر تقسیم بندی نمود :

الف) *AVR* بدون حافظه *BOOT* :

در این نوع میکروکنترلرها، فقط حافظه کاربردی وجود دارد، مانند سری های *ATTiny* و *AT80s*. در این *AVR*ها بردارهای وقفه در یک مکان ثابت و مشخص در ابتدای حافظه برنامه قرار داده شده که در هنگام وقوع وقفه، *CPU* به آن محل از حافظه برنامه رجوع کرده و از آنجا شروع به اجرای برنامه می کند. اگر وقفه ای استفاده نشود، خانه مربوط به هر یک از وقفه ها می تواند به صورت خانه معمولی حافظه برنامه در نظر گرفته شود.

ب) *AVR* با حافظه *BOOT* :

این تقسیم بندی برای میکرو کنترلرهای سری *ATmega* بوده (به غیر از *ATmega103*، *ATmega603* و *ATmega48*) که هم حافظه *BOOT* و هم حافظه کاربردی را پشتیبانی می نمایند.

۳ ـ رجیستر کنترل وقفه عمومی (*GICR*)

این ثبات در شکل ۲ نشان داده شده است.

Bit	7	6	5	4	3	2	1	0
	INT1	*INT0*	*INT2*	-	-	-	*IVSEL*	*IVCE*
Read/Write	*R/W*	*R/W*	*R/W*	*R*	*R*	*R*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

شکل ۲ : رجیستر *GICR*

دو بیت از بیت های مهم این ثبات عبارتند از :

ـ بیت انتخاب بردار وقفه (*IVSEL*)

از این بیت برای انتخاب بردار وقفه استفاده می شود. زمانی که *IVSEL=0* باشد، بردار وقفه در آغاز حافظه کاربردی و اگر *IVSEL=1* باشد، بردار وقفه در آغاز حافظه *BOOT* قرار می گیرد.

ـ بیت فعال ساز تعیین بردار وقفه (*IVCE*)

برای تغییر *IVSEL*، ابتدا باید بیت *IVCE* را یک کرد و سپس مقدار دلخواه را در *IVSEL* قرار داد. بیت *IVCE* به طور خودکار و به صورت سخت افزاری پس از گذشت چهار سیکل صفر می شود.

نکته : بیت های *IVSEL* و *IVCE* در میکروکنترلرهای *ATmega64*، *ATmega169* و *ATmega128* در رجیستر *MCUCR* قرار دارند.

برای جلوگیری از تغییرات غیرعادی در جدول بردار وقفه، مراحل زیر را دنبال نمایید :

۱ ـ فعال کردن بیت *IVCE*

۲ ـ با فعال کردن *IVCE* به مدت چهار سیکل فرصت دارید تا در *IVSEL* مقدار دلخواه خود را بنویسید. پس از این مدت به طور خودکار *IVCE* صفر می شود. در این مدت، وقفه ها به صورت خودکار غیرفعال خواهند بود.

۴ ـ وقفه های خارجی (*External Interrupts*)

میکروکنترلرهای *AVR*، علاوه بر وقفه های داخلی مانند وقفه تایمرها، سریال، مقایسه کننده آنالوگ و ... از وقفه های خارجی نیز پشتیبانی می کنند. این وقفه ها با توجه به نوع میکروکنترلرها می توانند از یک (در میکروکنترلر *ATtiny13*) تا هشت (در میکروکنترلرهای *ATmega103* و *ATmega128*) وقفه خارجی باشند. میکروکنترلر *ATmega32* از سه وقفه خارجی (*INT0*، *INT1* و *INT2*) پشتیبانی می کند.

وقفه های خارجی توسط پین های *INT0-7* راه اندازی می شوند. حتی اگر پین های *INT0-7* به صورت خروجی پیکربندی شده باشند، وقفه ها راه اندازی خواهند شد. وقفه های خارجی می توانند با یک لبه پایین رونده یا بالا رونده و یا یک سطح پایین فعال شوند. یکی از خصوصیات وقفه ها تشخیص غیرهمزمان آنها می باشد که از این خصوصیت می توان برای بیدار کردن *CPU* از بخشی از مدهای *sleep* به غیر از مد *Idle* (در این مد، پالس ساعت *I/O* متوقف است) استفاده نمود.

۵ ـ فعال کردن وقفه ها

۵-۱ فعال نمودن وقفه سراسری

پیش از به کارگیری وقفه باید بیت فعالساز وقفه عمومی (*I*) را فعال نمود. این بیت در رجیستر وضعیت (*SREG*) قرار دارد.

Bit	7	6	5	4	3	2	1	0
	I	*T*	*H*	*S*	*V*	*N*	*Z*	*C*
Read/Write	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

I : بیت فعالساز وقفه عمومی

همانگونه که گفته شد، خانواده *AVR* تا هشت (از ۰ تا ۷) وقفه خارجی را پشتیبانی می کند. بنابراین رجیسترهای داخلی مربوط به کنترل و وضعیت هر وقفه در هر میکروکنترلر ممکن است متفاوت با دیگری باشد. در ادامه، به طور نمونه به بررسی رجیسترهای مربوط به وقفه *ATmega32* و *ATmeg128* خواهیم پرداخت که به ترتیب ۳ و ۸ منبع وقفه خارجی را پشتیبانی می کنند. البته اساس کار وقفه در تمام سری های *AVR* یکسان بوده و تنها تفاوت موجود، فقط در تعداد وقفه ها ونام رجیسترها خلاصه می شود.

۵-۲ رجیستر کنترل (*MCUCR*)

رجیستر کنترل *MCU* (*Master Control Unit*) شامل بیت های کنترل وقفه و عملکردهای *AVR* در حالت کلی می باشد.

Bit	7	6	5	4	3	2	1	0
	SE	*SM2*	*SM1*	*SM0*	*ISC11*	*ISC10*	*ISC01*	*ISC00*
Read/Write	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

ـ بیت های *ISC00* و *ISC01*

از بیت های *ISC00* و *ISC01* جهت تعیین نوع فعالسازی وقفه خارجی روی لبه بالا رونده، لبه پایین رونده، حساس به سطح و یا حساس به هر تغییری بر روی پین *INT0* استفاده می شود، حتی اگر پین های مربوطه به صورت خروجی پیکربندی شده باشند (جدول ۱).

جدول ۱ : وضعیت های مختلف بیت های *ISC00* و *ISC01*

ISC01	*ISC00*	*Description*
0	*0*	*The low level of INT0 generates an interrupt request.*
0	*1*	*Any logical change on INT0 generates an interrupt request.*
1	*0*	*The falling edge of INT0 generates an interrupt request.*
1	*1*	*The rising edge of INT0 generates an interrupt request.*

مقدار موجود روی پایه *INT0* قبل از تشخیص لبه ها، نمونه برداری می شوند. اگر وقفه لبه یا *Toggle* انتخاب شده باشد، پالس هایی که مدت زمان آنها بیش از سیکل پالس ساعت طول بکشد، وقفه را تولید خواهد کرد و در پالس های کوتاه، ضمانتی برای اجرای وقفه وجود ندارد. همچنین اگر وقفه سطح انتخاب شود، این سطح باید تا زمان اجرای دستورالعمل جاری تحت تولید وقفه پایین نگه داشته شود.

ـ بیت های *ISC10* و *ISC11*

وظیفه این بیت ها همانند بیت های *ISC00* و *ISC01* اما برای *INT1* است.

ـ بیت های *SM2-0*

از این بیت ها برای انتخاب مد *sleep* استفاده می شود.

ـ بیت *SE*

از این بیت برای فعال کردن مد *sleep* استفاده می شود.

نکته : رجیسترهای *EICRA* و *EICRB* در *ATmega128* همان وظیفه رجیستر *MCUCR* در *ATmega32* را برعهده دارند.

Bit	7	6	5	4	3	2	1	0
EICRA	ISC31	ISC30	ISC21	ISC21	ISC11	ISC10	ISC01	ISC00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

۳-۵ رجیستر وقفه و کنترل *AVR* (*MCUCSR*)

Bit	7	6	5	4	3	2	1	0
	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	See Bit Description				

ـ بیت *ISC2*

از این بیت برای تعین نوع فعالسازی وقفه خارجی ۲ در *ATmega32* استفاده می شود.

نکته : وقفه *INT2* در *Atmega32* فقط با لبه فعال خواهد شد.

اگر در *ISC2* صفر نوشته شود، *INT2* توسط یک لبه پایین رونده فعال و اگر در *ISC2* یک نوشته شود، *INT2* توسط یک لبه بالارونده فعال می شود.

پالس های تولیدکننده وقفه باید بیش از *ns* ۵۰۰ طول بکشد و در پالس های کوتاه تر از آن، ضمانتی برای تولید وقفه وجود ندارد.

نکته : هنگام تغییر بیت *ISC2* ممکن است وقفه ای رخ دهد؛ بنابراین پیشنهاد می شود برای جلوگیری از این حالت، ابتدا بیت *INT2* در رجیستر فعالساز وقفه (*GICR*) را غیرفعال کنید. سپس بیت *ISC2* را تغییر دهید و مجدداً بیت *INT2* در رجیستر *GICR* را فعال نمایید.

۵-۴ رجیستر کنترل وقفه عمومی (*GICR*)

به کمک این رجیستر می توان وقفه های خارجی را در *ATmwga32* فعال یا غیرفعال نمود.

Bit	7	6	5	4	3	2	1	0
	INT1	*INT0*	*INT2*	-	-	-	*IVSEL*	*IVCE*
Read/Write	*R/W*	*R/W*	*R/W*	*R*	*R*	*R*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

بیت *INT0* : بیت فعالساز وقفه خارجی صفر

بیت *INT1* : بیت فعالساز وقفه خارجی یک

بیت *INT2* : بیت فعالساز وقفه خارجی دو

نکته : رجیستر *EIMSK* در *ATmega128* همان وظیفه *GICR* در *Atmega32* را برعهده دارد.

Bit	7	6	5	4	3	2	1	0
EIMSK	*INT7*	*INT6*	*INT5*	*INT4*	*INT3*	*INT2*	*INT1*	*INT0*
Read/Write	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

۵-۵ رجیستر پرچم وقفه عمومی (*GIFR*)

در زمان وقوع وقفه، پرچم مربوط به آن در این رجیستر فعال می شود.

Bit	7	6	5	4	3	2	1	0
	INTF1	*INTF0*	*INTF2*	-	-	-	-	-
Read/Write	*R/W*	*R/W*	*R/W*	*R*	*R*	*R*	*R*	*R*

Initial Value	0	0	0	0	0	0	0	0

ـ بیت *INTF1* (پرچم وقفه خارجی ۱)

زمانی که یک تغییر سطح روی پین *INT1* رخ دهد، بیت *INTF1* یک خواهد شد و اگر بیت *I* در *SREG* و بیت *INT1* در *GICR* یک باشند، *CPU* به محل بردار وقفه پرش خواهد کرد. این پرچم در حین اجرای زیرروال وقفه پاک می شود. زمانی که *INT1* به صورت یک وقفه سطح پیکربندی شده باشد، به صورت خودکار صفر می شود.

ـ بیت *INTF2* (پرچم وقفه خارجی ۲)

عملکرد این بیت همانند *INTF1* است.

نکته : رجیستر *EIFR* در *ATmega128* همان وظیفه *GIFR* در *ATmega32* را بر عهده دارد.

Bit	7	6	5	4	3	2	1	0
EIFR	*INTF7*	*INTF6*	*INTF5*	*INTF4*	*INTF3*	*INTF2*	*INTF1*	*INTF0*
Read/Write	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*	*R/W*
Initial Value	0	0	0	0	0	0	0	0

در *AVR*ها امکان تعیین اولویت وقفه وجود ندارد و هر وقفه ای که در آدرس پایین تری قرار دارد، از اولویت بالاتری برخوردار می باشد.

۶ ـ برنامه نویسی وقفه ها در *Codevision*

برای استفاده از وقفه در *Codevision* از کلمه کلیدی *interrupt* به فرم زیر کمک می گیریم :

```
interrupt [Vector number] void int_expresion (void) {
    برنامه سرویس وقفه
}
```

عبارت *Vector number* شمارنده بردار وقفه مورد نظر بوده و از روی جدول بردار وقفه به دست می آید؛ به عنوان نمونه [۱] برای بردار *Reset* و [۲] برای بردار *INT0* و ... . در جدول ۲ بردارهای وقفه برای میکروکنترلر *ATmega32* آورده شده است.

عبارت *int_expresion* نام تابع سرویس دهنده وقفه بوده که می تواند توسط برنامه نویس نوشته شود.

نکته : قبل از به کارگیری این تابع باید رجیسترهای مربوط به وقفه را تنظیم نموده تا وقفه ها فعال شوند. همچنین تابع وقفه چیزی را باز نمی گرداند.

جدول ۲ : بردارهای وقفه برای *ATmega32*

Vector No.	***Program Address***	***Source***	***Interrupt Description***
1	*$000*	*RESET*	*External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset*
2	*$002*	*INT0*	*External Interrupt Request 0*
3	*$004*	*INT1*	*External Interrupt Request 1*
4	*$006*	*INT2*	*External Interrupt Request 2*
5	*$008*	*TIMER2 COMP*	*Timer/Counter2 Compare Match*
6	*$00A*	*TIMER2 OVF*	*Timer/Counter2 Overflow*
7	*$00C*	*TIMER1 CAPT*	*Timer/Counter1 Capture Event*
8	*$00E*	*TIMER1 COMPA*	*Timer/Counter1 Compare Match A*
9	*$010*	*TIMER1 COMPB*	*Timer/Counter1 Compare Match B*
10	*$012*	*TIMER1 OVF*	*Timer/Counter1 Overflow*
11	*$014*	*TIMER0 COMP*	*Timer/Counter0 Compare Match*
12	*$016*	*TIMER0 OVF*	*Timer/Counter0 Overflow*
13	*$018*	*SPI, STC*	*Serial Transfer Complete*
14	*$01A*	*USART, RXC*	*USART, Rx Complete*
15	*$01C*	*USART, UDRE*	*USART Data Register Empty*
16	*$01E*	*USART, TXC*	*USART, Tx Complete*
17	*$020*	*ADC*	*ADC Conversion Complete*
18	*$022*	*EE_RDY*	*EEPROM Ready*
19	*$024*	*ANA_COMP*	*Analog Comparator*
20	*$026*	*TWI*	*Two-wire Serial Interface*
21	*$028*	*SPM_RDY*	*Store Program Memory Ready*

شرح آزمایش

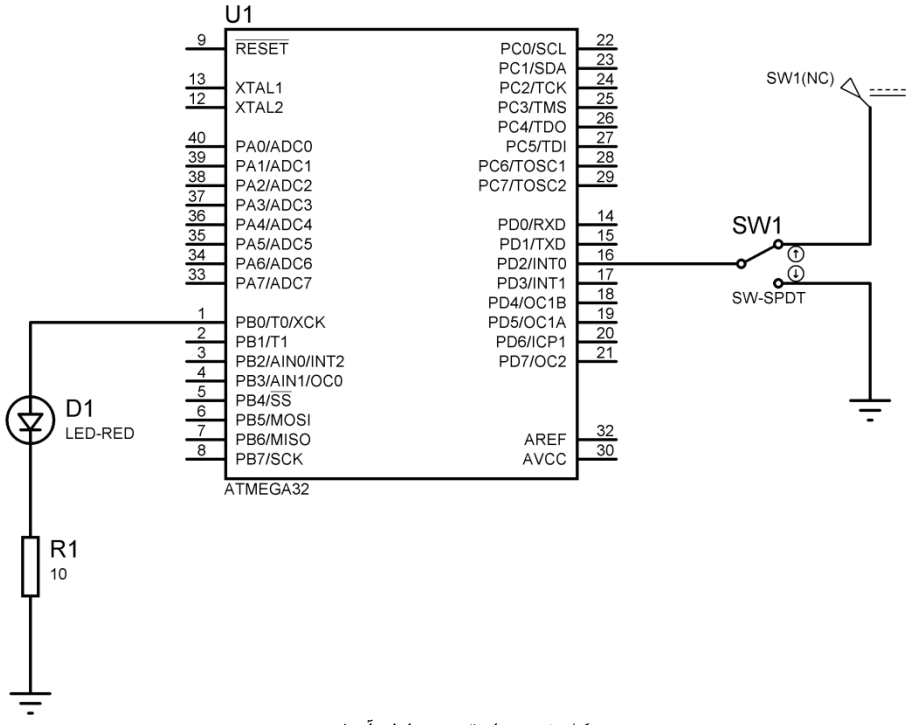
ـ قسمت اول :

می خواهیم برنامه ای را بنویسیم که هنگام وقوع وقفه خارجی صفر، *LED* متصل به *PortB.0* به مدت ۱ ثانیه روشن و ۲ ثانیه خاموش شود. توجه شود که از میکروکنترلر *ATmega32* استفاده نموده ایم. برای این منظور می توان برنامه زیر را نوشت :

```
#include <mega32.h>
#include <delay.h>
#define LED PORTB.0

void main(){
      DDRB=0xFF;              PORTB به صورت خروجی
      PORTB=0x00;
      GICR=0x40;              فعال نمودن وقفه خارجی صفر
      MCUCR=0x03;             فعال شدن وقفه با لبه بالارونده
      GIFR=0x40;
      #asm("sei");            فعال کردن وقفه سراسری
      while(1);               در انتظار وقفه
}
//-----------------------------
interrupt [2] void ext_int0 (void)
{
       #asm("cli");           غیر فعال کردن وقفه سراسری
       LED=1;
       delay_ms(1000);
       LED=0;
       delay_ms(2000);
       #asm("sei");           فعال کردن وقفه سراسری
}
```

این برنامه را به دقت بررسی نمایید و با بستن مدار نشان داده شده در شکل ۳ طرز کار آن را مشاهده و درستی برنامه را تحقیق نمایید.



شکل ۳ : مدار قسمت اول آزمایش

ـ قسمت دوم :

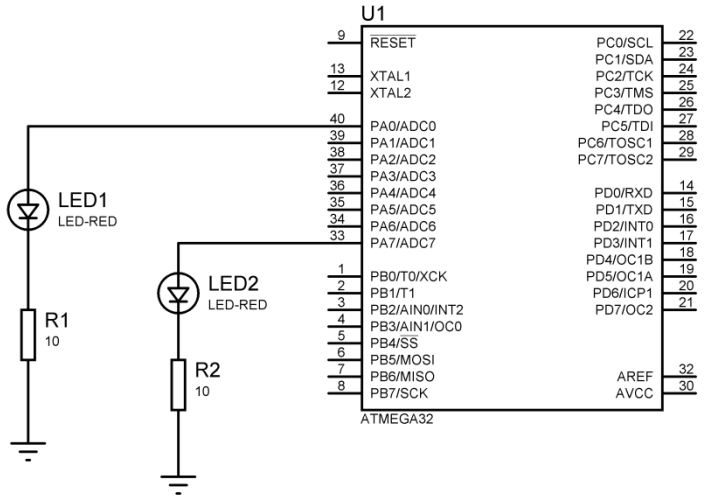
در این قسمت، می خواهیم یک وقفه داخلی را ایجاد کنیم. برای این منظور فرض می کنیم که تایمر صفر در حالت عملکرد *CTC* فعال شده باشد. می خواهیم هرگاه در مقایسه بین عدد موجود در رجیستر *TCNT0* و رجیستر *OC0* مطابقت وجود داشت، یک وقفه از نوع تایمر صفر فعال شود و *LED* متصل به *PortB.7* را برای *ms* ۱ روشن و دوباره خاموش نماید. باید توجه داشت که برای استفاده از وقفه تایمر صفر در حالت عملکرد *CTC* حتماً بایستی بیت *OCIE0* در رجیستر *TIMSK* یک شود. برنامه این قسمت از آزمایش در ادامه آورده شده است.

```
#include <mega32.h>
#include <delay.h>
#define xtal 8000000
#define LED1 PORTA.0
#define LED2 PORTA.7

void main(void){
      DDRA=0xFF;
      PORTA=0x00;
      TIMSK=0x02;
      LED1=1;
```

```
        TCNT0=0x00;
        OCR0=0xFF;
        TCCR0=0x1C;
        #asm("sei");
        while (1);
}
//---------------------------------
interrupt [11] void tim0_comp(void)
{
        #asm("cli");
         LED1=0;
         LED2=1;
         delay_ms(100);
         LED2=0;
         LED1=1;
         delay_ms(100);
         #asm("sei");
}
```

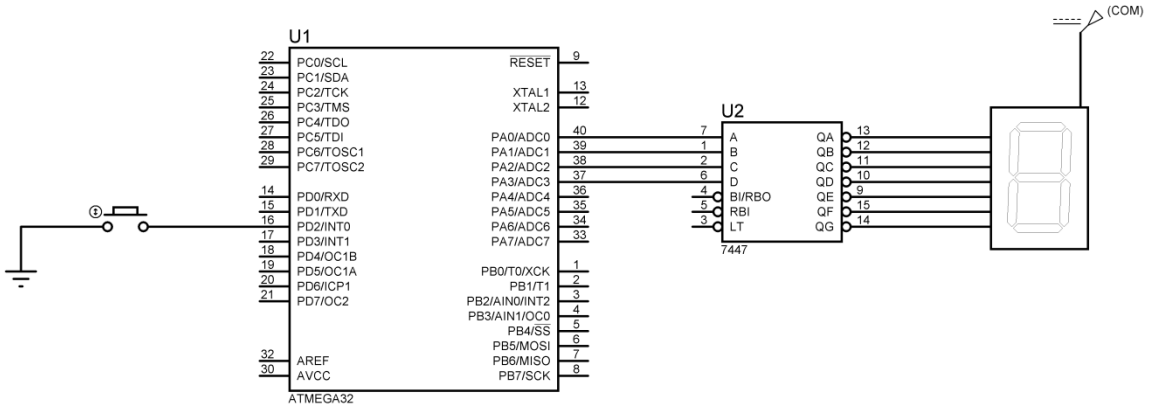
این برنامه را نیز به دقت مورد بررسی قرار داده و با بستن مدار نشان داده شده در شکل ۴ طرز کار آن را مشاهده و درستی برنامه را تحقیق نمایید.



شکل ۴ : مدار قسمت دوم آزمایش

ـ قسمت سوم :

برنامه ای بنویسد که اعداد صفر تا ۹ را به ترتیب بر روی یک *7segment* نشان دهد، اما هرگاه سوییچ متصل به *PortD.2* فشار داده شد، شمارش آن برعکس شود و از پایان شمارش معکوس، به کار عادی شمارش از ۰ تا ۹ بازگردد.



شکل ۵ : مدار قسمت سوم آزمایش

پرسش :

۱ ـ برنامه ای بنویسید که یک سیگنال مربعی را دریافت کند و فرکانس آن را محاسبه و بر روی یک *LCD* نمایش دهد.

راهنمایی : در این پرسش، بایستی از تایمر به عنوان شمارنده (کانتر) استفاده شود. عملکرد این مدار فرکانس متر به این صورت است که توسط تایمر صفر، زمانی به مدت یک ثانیه تولید می شود و همزمان با آن، تایمر یک نیز شروع به شمارش تعداد پالس های اعمالی بر روی پایه *T1* می کند (در صورت استفاده از *ATmega16* منظور از پایه *T1* همان پایه *PB1* است). پس از طی زمان یک ثانیه، مقدار شمارش شده در رجیستر تایمر یک، معادل فرکانس موج روی پایه *T1* است و می توان آن را بر روی *LCD* نمایش داد. می توان از هر لبه پایین رونده مانند یک وقفه خارجی استفاده کرد.

آزمایشگاه ریزپردازنده ها

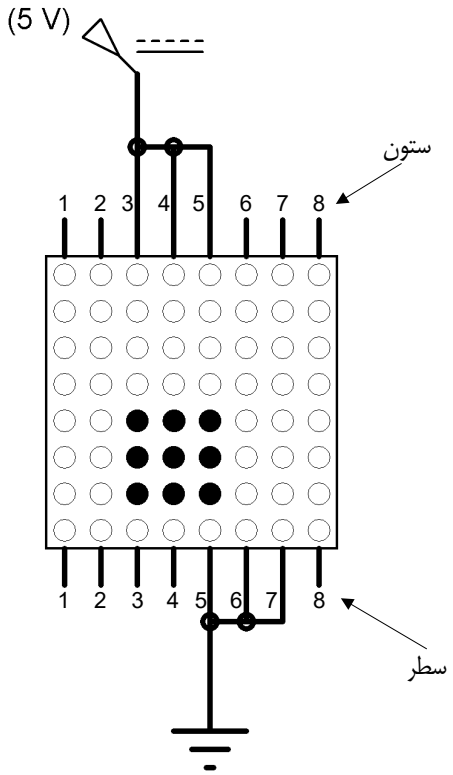
آزمایش شماره ۸

آشنایی با اصول اسکن *Dot Matrix* و نمایش اطلاعات بر روی آن

(تابلو روان)

مقدمه

از *Dot Matrix* برای ساخت تابلو روان استفاده می شود. برای نمایش تصویر بر روی *Dot Matrix*ها از اصول اسکن سطری پیروی می شود؛ به این ترتیب که هر *LED* را که بخواهیم روشن کنیم باید ستونی که آن *LED* در آن قرار دارد را یکِ منطقی (۵ ولت) و سطری که آن *LED* در آن قرار دارد را صفر (زمین) نماییم. یک نمونه از چنین ساختاری در شکل ۱ نشان داده شده است. می توان دید که با یک کردن ستون های ۳، ۴ و ۵ و صفر کردن سطرهای ۵، ۶ و ۷ یک دسته از *LED*ها در سطر و ستون های متناظر روشن شده اند.



شکل ۱ : نحوه اسکن شدن سطرها و ستون ها در *Dot Matrix*

شرح آزمایش

ـ قسمت اول :

می خواهیم برنامه ای را بنویسیم که توسط آن، حرف *A* بر روی یک *Dot Matrix* نمایش داده شود. این برنامه در ادامه آورده شده است.

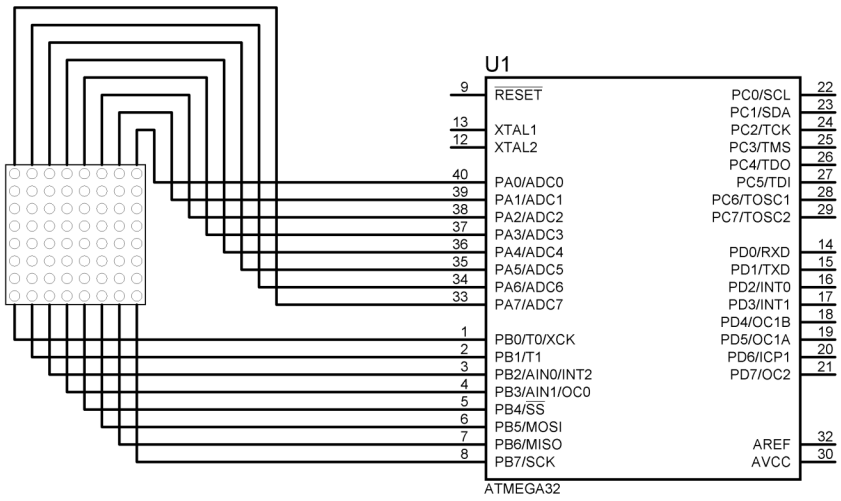
```
$regfile = "M32def.dat ″
$crystal = 8000000

Ddra = &HFF
Ddrb = &HFF
Dim K As Byte , A As Byte
A = &B01111111

Do
  For K = 0 To 7
        Porta = Lookup(k , Dta)
        Rotate A , Left , 1
        Portb = A
        Waitus 100
        Portb = &HFF
  Next
Loop
End

Dta:
Data &H18 , &H3C , &H66 , &H66 , &H7E , &H66 , &H66 , &H00
```

این برنامه را به دقت بررسی نمایید و با بستن مدار نشان داده شده در شکل ۲ طرز کار آن را مشاهده و درستی برنامه را تحقیق نمایید.



شکل ۲ : مدار قسمت اول آزمایش

ـ قسمت دوم :

با توجه به قسمت اول آزمایش، برنامه را به گونه ای تغییر دهید که بر روی *Dot Matrix* حروف *K* و *Z* و اعداد *3* و *6* و حروف فارسی آ و پ نمایش داده شود.

ـ قسمت سوم :

برنامه ای بنویسید که اعداد صفر تا ۹ (فارسی) را با یک فاصله زمانی دلخواه، بر روی یک *Dot Matrix* نمایش دهد.

پرسش :

۱ ـ به کمک چندین *Dot Matrix* در کنار یکدیگر، یک تابلو روان طراحی کنید که هر رشته دلخواه مانند نام خودتان را نمایش دهد. برای سادگی، رشته ها را انگلیسی در نظر بگیرید و برنامه آن را به گونه ای بنویسید که رشته ها از چپ به راست حرکت کنند.

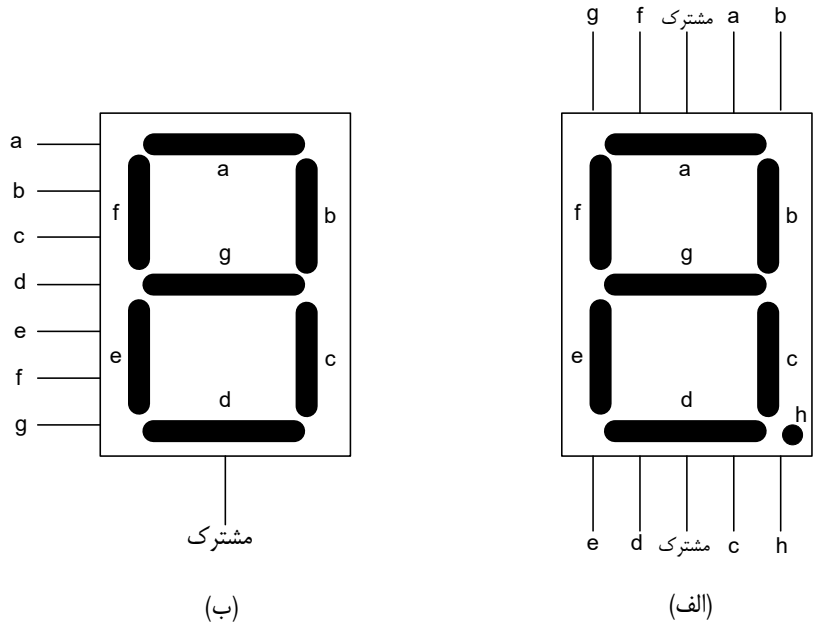
آزمایشگاه ریزپردازنده ها

آزمایش شماره ۹

پیاده سازی چراغ راهنمایی به کمک میکروکنترلر *AVR*

مقدمه

در این آزمایش، هدف، پیاده سازی یک چراغ راهنمایی به صورت ساده می باشد. برای نمایش اعداد از *7-segment* کاتد مشترک استفاده شده است. در این نوع *7-segment* پایه مشترک را به زمین متصل کرده و برای روشن کردن هر یک از *LED*های درون *7-segment* پایه متناظر با آن *LED* را به یکِ منطقی (۵ ولت) متصل می نماییم. ارتباط بین پایه ها و *LED* ها در شکل ۱ نشان داده شده است.



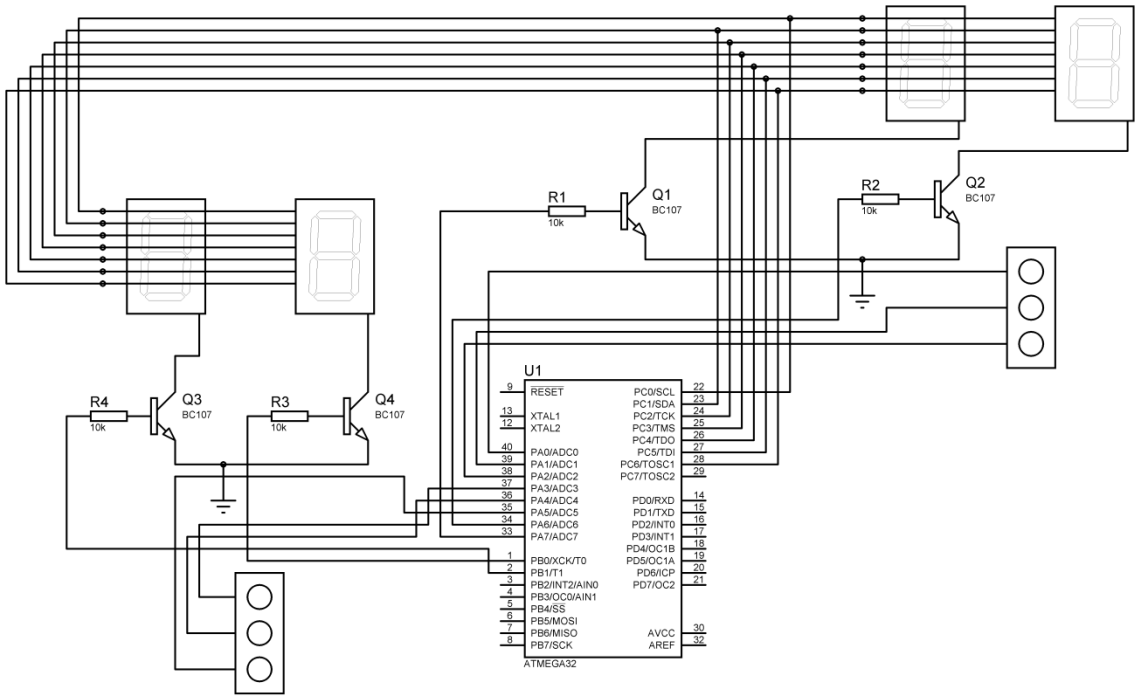
شکل ۱ : ارتباط بین پایه ها و *LED* ها در یک *7-segment*

شرح آزمایش

برای مدار نشان داده شده در شکل ۲ برنامه ای بنویسید که یک چراغ راهنمایی را پیاده سازی نماید؛ به این صورت که هر گاه *7-segment* با رنگ سبز از ۹ به صفر شمارش می کند، چراغ راهنمایی مربوط به آن، سبز باشد و *7-segment* با رنگ قرمز در کنار آن، خاموش باشد، همچنین در این حالت بایستی چراغ راهنمایی دیگر قرمز باشد و *7-segment* قرمز رنگ مربوط به آن چراغ راهنمایی از ۹ به صفر شمارش نموده و *7-segment* سبز خاموش باشد. پس از پایان شمارش بایستی جای آنها با یکدیگر عوض شود. البته برنامه را به گونه ای بنویسید که

با پایان یافتن شمارش (از ۹ به صفر)، برای یک مدت زمانی کوتاه (مثلاً ۲ ثانیه) *7-segment* های در حال شمارش در عدد صفر بمانند و چراغ راهنمایی که می خواهد به رنگ قرمز درآید، رنگ زرد را نشان دهد.

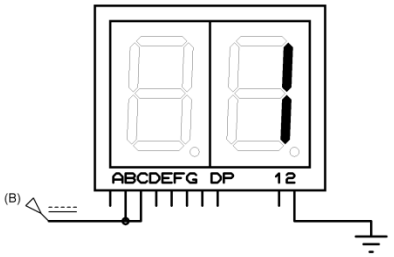
توجه : این مدار به گونه ای طراحی شده است که اگر *Base* هر یک از ترانزیستورهای *BC107* به منطق یک (۵ ولت) متصل شود، *7-segment* متصل به آن فعال می شود.



شکل ۲ : مدار به کار رفته برای پیاده سازی چراغ راهنمایی

پرسش :

۱ ـ برنامه این آزمایش را برای حالتی بنویسید که از *7-segment* های نشان داده شده در شکل ۳ استفاده می شود. بدیهی است که در این حالت می توان شمارش را برای اعداد بزرگتر از ۹ نیز انجام داد. به عنوان نمونه شمارش را از ۳۰ به صفر انجام دهید. این نوع *7-segment* نیز مشابه آنچه در این آزمایش به کار رفت، می باشد. برای فعال کردن *7-segment* سمت چپ باید پایه یک، *7-segment* سمت راست باید پایه ۲ و برای فعال کردن هر دو باید هر دو پایه ۱ و ۲ را به **زمین** متصل نمود. پایه های دیگر همانند یک *7-segment* کاتد مشترک معمولی است. البته نوع آند مشترک آن نیز وجود دارد که اتصال پایه های آن به ۵ ولت و زمین برعکس حالتی است که در بالا شرح داده شد.



شکل ۳ : *7-segment* به کار رفته برای پرسش