دانشکده برق خواجه نصیر

پروژه درس سیستم‌های دیجیتال ۲ (ریزپردازنده)

استاد: سرکار خانم دکتر رودکی

اعضای گروه:

زهرا ایران‌پور مبارکه ۹۸۱۹۸۹۳

امیرحسین شرفی ۹۷۲۷۵۸۳

موضوع پروژه: ساختن یک کرنومتر، به صورتی که از ۰ تا ۹۹ ثانیه را بشمارد و بر روی دو عدد سون سگمنت نشان دهد و هر موقع دکمه این کرنومتر را یکبار فشار دهیم، بر روی سون سگمنت‌ها خروجی Hi چاپ شود و بعد از مکث کوتاهی شمارنده دوباره شروع به کار کند و از عدد قبلی بشمارد. اگر شمارنده به عدد ۹۹ رسید ریست شده و دوباره از صفر شروع به شمردن کند. دکمه باید به عنوان interrupt خارجی عمل کند.

کد پروژه به همراه توضیحات

مشخص کردن نوع میکروکنترلر (ATmega64):

.Include "M64DEF.INC"

مشخص کردن آدرس برنامه اصلی که از صفر شروع می‌شود:

.ORG 0X0000

JMP main

مشخص کردن آدرس برنامه وقفه۵ که با توجه به دیتاشیت و طبق جدول کتاب که در زیر آمده است، باید به آدرس 0X000c برود:

وقفهٔ خارجی ۵	INT5	0x000CH	7

.ORG 0X000C

JMP INT5_ISR

مشخص کردن آدرس برنامه تایمر صفر که با توجه به دیتاشیت و طبق جدول کتاب که در زیر آمده است، باید به آدرس 0X0020 برود:

سرریز شمارندهٔ زمان‌سنج ۰	Timer0 OVF	0x0020H	17

.ORG 0X0020

JMP TIMER0_ISR

شروع برنامه را از آدرس ۵۰ مینویسیم چراکه از آدرس 0X0000 تا 0X0045 برای وقفه‌ها و تایمرها رزرو شده است. بنابراین برنامه را از آدرسی خارج این محدوده شروع به نوشتن می‌کنیم:

.ORG 0X0050

برنامه اصلی:

main:

تنظیم اشاره‌گر در پشته: باید آخرین خانه‌های رم را در بر بگیرد. از آنجایی که SP یا همان اشاره‌گر به پشته باید حاوی آدرس سلولی از حافظه رم سیستم باشد، یک رجیستر ۱۶ بیتی بوده و پشته بخشی از حافظه ۸ بیتی پردازنده است، این رجیستر در فضای SFR، با اتصال دو رجیستر ۸ بیتی به نام‌های SPL برای بیت‌های پایین و SPH برای بیت‌های بالا تعریف می‌شود. در میکرو کنترلر ATmega64 پشته از آدرس زیاد به طرف آدرس کم پر می‌شود (با هربار نوشتن، SP کاهش می‌یابد):

LDI R16, high(RAMEND)

```
OUT SPH, R16
LDI R16, low(RAMEND)
OUT SPL,R16
```

از آنجایی که می‌خواهیم دو پورت A و B را به عنوان خروجی داشته باشیم، با دادن 0XFF به آن‌ها، آن‌ها را یک کرده و به عنوان خروجی تعریف می‌کنیم:

```
LDI R16,0Xff
OUT DDRB,R16      ;Port B -> output
OUT DDRA,R16      ;PORT A -> Output
```

سپس مقدار ابتدایی خروجی‌ها را صفر قرار می‌دهیم (هر یک از این خروجی‌ها به یک سون سگمنت متصل می‌شوند):

```
LDI R16,0X00
OUT PORTB,R16
OUT PORTA,R16
```

تنظیمات وقفه: از آنجایی که گفته شده با فشردن کلید وقفه ایجاد شود، پس ما لبه پایین رونده اینتراپت یا همان وقفه را ملاک عمل قرار می‌دهیم که با پایین رفتن و فشرده شدن کلید، وقفه ایجاد شود. برای انتخاب حالت پایین رونده، طبق دیتاشیت و جدول‌های موجود در کتاب عمل می‌کنیم:

7	6	5	4	3	2	1	0
ISC7-1	ISC7-0	ISC6-1	ISC6-0	ISC5-1	ISC5-0	ISC4-1	ISC4-0

شکل (۱۶.۶) رجیستر EICRB

ISCn-1	ISCn-0	شرط تولید وقفه
0	0	سطح پائین INTn
0	1	هر دو لبه
1	0	لبه پائین رونده INTn
1	1	لبه بالا رونده INTn

```
LDI R16,0X08
OUT EICRB,R16      ;sencitive falling edge int5
```

سپس وقفه پنجم را فعال می‌کنیم:

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

شکل (۱۲.۶) رجیستر EIMSK

```
LDI R16,0X20
OUT EIMSK,R16      ;enable int5
```

تنظیمات تایمر: پیش تقسیم کننده را برابر ۱۰۲۴ قرار می‌دهیم. تا زمان تایمر در حدود یک ثانیه شود.

CS02	CS01	CS00	فرکانس پالس
0	0	0	بدون پالس
0	0	1	CLK
0	1	0	CLK/8
0	1	1	CLK/32
1	0	0	CLK/64
1	0	1	CLK/128
1	1	0	CLK/256
1	1	1	CLK/1024

$$\frac{1M}{1024} = 976.5625 \rightarrow \frac{1}{976.5625} = 1.024ms$$

```
LDI R16,0X07
OUT TCCR0,R16      ;PRESCALER = 1024
```

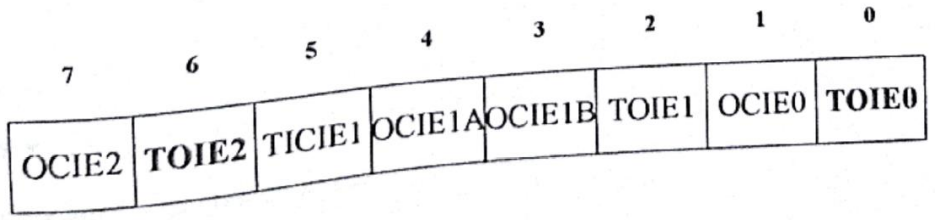
شروع شمارش تایمر(این عدد با استفاده از نرم‌افزار codevision بدست آمد):

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

```
LDI R16,0X9E
OUT TCNT0,R16      ; START TIMER FROM = 256-158=98
```

درواقع کارکرد به این صورت است که هر ۱.۰۲۴ میلی ثانیه، ۹۸ تا تغییر داریم و درون خود تایمر هم یک ضریب ده قرار میدهیم تا در نهایت هر ۱ ثانیه تغییر در سون سگمنت یکان داشته باشیم.

فعال کردن تایمر: TOIE0 را برابر یک قرار می‌دهیم تا در صورت سرریز شمارنده صفر می‌تواند وقفه تولید کند.




```
LDI R16,0X01
OUT TIMSK,R16      ; ENABLE INT'S TIMER0
```

R17 و R23 را بعدا مورد استفاده قرار می‌دهیم پس مقدار اولیه‌شان را صفر قرار می‌دهیم:

```
LDI R17,0
LDI R23,0
```

برای فعال کردن وقفه‌ها:

```
SEI
```

ساخت یک لوپ برای اینکه میکرو به کار خود مداوم ادامه دهد:

```
END:
JMP END
```

برنامه وقفه پنجم:

```
INT5_ISR:
```

شروع شمارش (یکان (PORTA) ۱ و دهگان (PORTB) ۰ باشد):

```
LDI R22,0B00110000
OUT PORTA,R22
LDI R22,0B01110110
OUT PORTB,R22
```

R29 و R31 و R30 را بعدا مورد استفاده قرار می‌دهیم پس مقدار اولیه‌شان را صفر قرار می‌دهیم.

```
LDI R30,0
LDI R31,0
LDI R29,0
```

۴ لوپ ساده در کد قرار می‌دهیم که در روتین وقفه است برای موندن Hi مقداری دیلی در وقفه ایجاد می‌کند:

```
LOOP1:
INC R30
CPI R30,250
BRNE LOOP1
LOOP2:
INC R31
CPI R31,250
BRNE LOOP2
LOOP3:
INC R29
CPI R29,250
BRNE LOOP3
LDI R29,0X00
LOOP4:
INC R29
CPI R29,250
BRNE LOOP4
```

در پایان برنامه وقفه باید مجددا امکان فعال شدن وقفه‌های بعدی را فعال کنیم:

```
RETI
;-------------------------------------
```

برنامه تایمر صفر:

```
TIMER0_ISR:
```

تنظیمات تایمر (به همان صورتی که پیشتر توضیح داده شد. مجددا آورده می‌شود تا در صورت ریست شدن مشکلی پیش نیاید):

```
LDI R16,0X9E
OUT TCNT0,R16
```

رجیستر R17 را با ۱۰ مقایسه می‌کنیم تا اگر با ۱۰ برابر شد، دیگه به یکان اضافه نشود و به دهگان اضافه شود.

```
INC R17
```

```
CPI R17,10
BREQ NUM
```

برای خارج شدن از وقفه و return شدن و I برای فعال کردن وقفه‌های بعدی:

```
RETI
```

برنامه‌ای که اگر R17 با ۱۰ برابر بود، به آن وارد می‌شویم:

```
NUM:
```

R17 را صفر کرده و یکی به R23 اضافه می‌کنیم.

```
LDI R17,0
INC R23
```

R23 را با ۱-۱۰ مقایسه می‌کنیم و اگر هرکدام بود، به زیربرنامه مخصوص به همان عدد می‌رود:

```
CPI R23,10
BREQ DAH
CPI R23,9
BREQ NOH
CPI R23,8
BREQ HASHT
CPI R23,7
BREQ HAFT
CPI R23,6
BREQ SHISH
CPI R23,5
BREQ PANJ
CPI R23,4
BREQ CHAR
CPI R23,3
BREQ SEE
CPI R23,2
BREQ DO
CPI R23,1
```

BREQ YEK

;CPI R23,0

;BREQ SEFR

برای خارج شدن از وقفه و return شدن و I برای فعال کردن وقفه‌های بعدی:

TT:RETI

این قسمت از کد که کامنت شده اضافه بوده و نیازی نبود. چرا که ر صورتی که یکان ۱۰ بشه، خود به خود صفر شده و یکی به دهگان اضافه می‌شود:

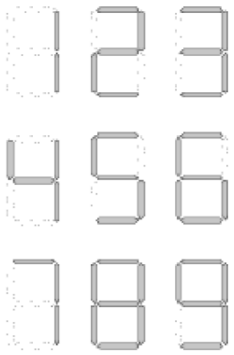
;SEFR:LDI R20,0B01111110

;OUT PORTA,R20

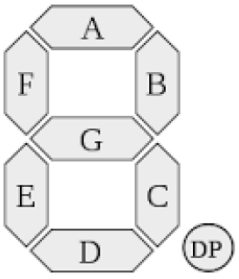
;OUT PORTB,R20

;JMP TT

برنامه گفته شده برای اعداد یک تا ۱۰ که برابر هریک بود، ledهای مربوط به همان عدد طبق دیتاشیت بر روی سون سگمنت روشن می‌شود (در اینجا R20 یکان و R21 دهگان است):



Digit Shown	Illuminated Segment (1 = illumination) a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	0	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1



YEK:LDI R20,0B00110000

OUT PORTA,R20

OUT PORTB,R21

JMP TT

DO:LDI R20,0B01101101

OUT PORTA,R20

OUT PORTB,R21

JMP TT

SEE:LDI R20,0B01111001

```
OUT PORTA,R20
OUT PORTB,R21
JMP TT
CHAR:LDI R20,0X72
OUT PORTA,R20
OUT PORTB,R21
JMP TT
PANJ:LDI R20,0B01011011
OUT PORTA,R20
OUT PORTB,R21
JMP TT
SHISH:LDI R20,0B01011111
OUT PORTA,R20
OUT PORTB,R21
JMP TT
HAFT:LDI R20,0X31
OUT PORTA,R20
OUT PORTB,R21
JMP TT
HASHT:LDI R20,0B01111111
OUT PORTA,R20
OUT PORTB,R21
JMP TT
NOH:LDI R20,0B01111011
OUT PORTA,R20
JMP TT
```

میدانیم که یکان نباید ده شود پس اگر برابر ده شد، یکان را روی سون سگمنت صفر نشان میدهیم و تنظیمات قبلی مان دهگان را یکی زیاد میکند:

```
DAH:LDI R20,0B00111111
OUT PORTA,R20
```

به همان روش قبلی، این بار برای دهگان رجیستر مربوطه به آن را با اعداد ۹ تا ۱ مقایسه کرده و با هرکدام برابر بود، به برنامه مربوط به همان عدد میرود و سپس در مرحله بعد در برنامه آن عدد ledهای مربوط به آن در سون سگمنت دوم مربوط به دهگان روشن میشود (درصورتی که دهگان هم برابر ۱۰ شد، برنامه ریست شده و از اول شروع به شمردن میکند):

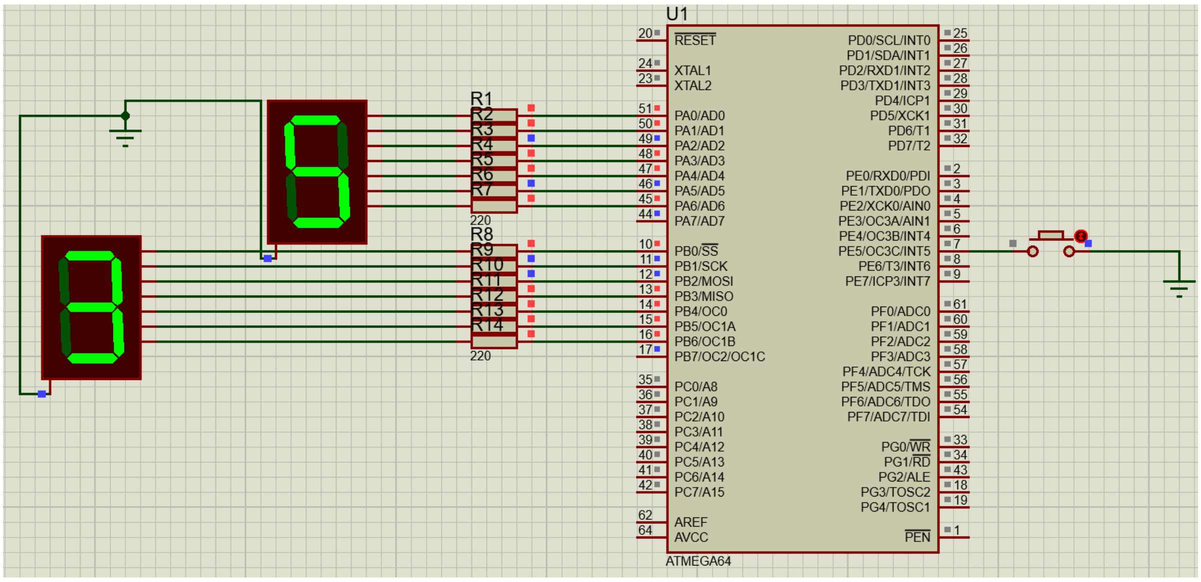
```
LDI R23,0X00
INC R24
CPI R24,9
BREQ NOH2
CPI R24,8
BREQ HASHT2
CPI R24,7
BREQ HAFT2
CPI R24,6
BREQ SHISH2
CPI R24,5
BREQ PANJ2
CPI R24,4
BREQ CHAR2
CPI R24,3
BREQ SE2
CPI R24,2
BREQ DO2
CPI R24,1
BREQ YEK2
```

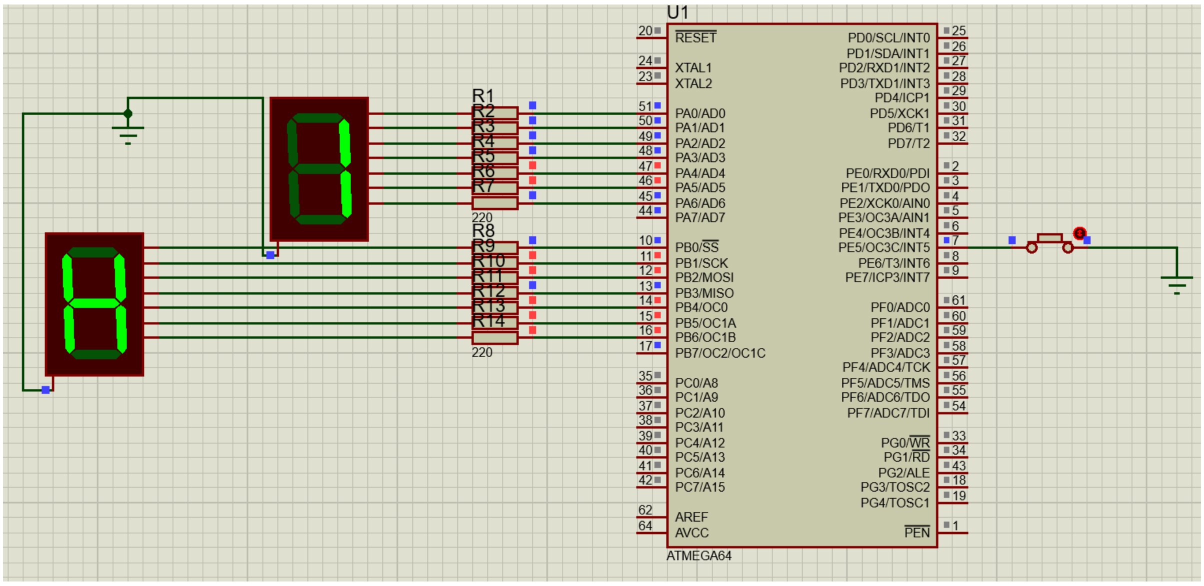
روشن کردن ledها:

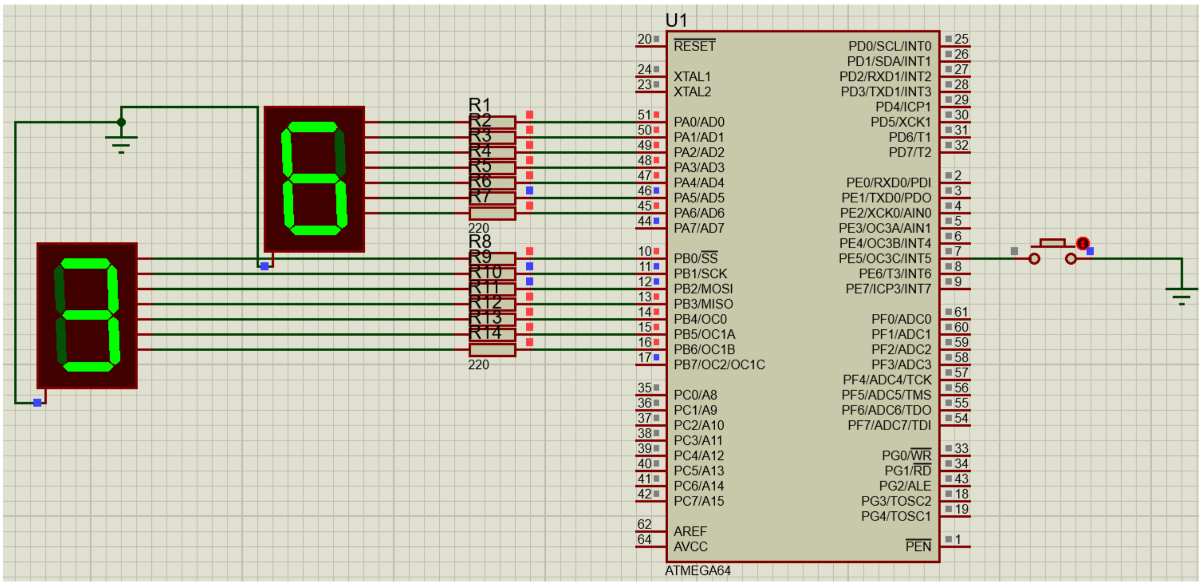
```
YEK2:LDI R21,0B00110000
OUT PORTB,R21
JMP TT
DO2:LDI R21,0B01101101
OUT PORTB,R21
```

```
JMP TT
SE2:LDI R21,0B01111001
OUT PORTB,R21
JMP TT
CHAR2:LDI R21,0X72
OUT PORTB,R21
JMP TT
PANJ2:LDI R21,0B01011011
OUT PORTB,R21
JMP TT
SHISH2:LDI R21,0B01011111
OUT PORTB,R21
JMP TT
HAFT2:LDI R21,0X31
OUT PORTB,R21
JMP TT
HASHT2:LDI R21,0B01111111
OUT PORTB,R21
JMP TT
NOH2:LDI R21,0B01111011
OUT PORTB,R21
JMP TT
```

تصاویر خروجی در proteus







ساخت عملی پروژه

برای ساخت عملی از کد ATmega32 استفاده کردیم. چون میکرو این مدل را پیدا کردیم. برای تغییر کد 64 به 32 قسمت اصلی کد تغییری نمیکند ولی در ابتدای کار مقداری از توابع که در این دو میکرو متفاوند تغییر داده شد. در فایل زیپ که پیوست می‌شود، هر دو کد ۶۴ و ۳۲ به همراه شبیه ساز پروتئوس آن‌ها قرار دارند.