

نام و نام خانوادگی: عرفان راستی

شماره ی دانشجویی : 9823034

توضیحات مربوط به پروژه:

در این پروژه قرار است با گرفتن وضعیت rising و falling کلید ها اعداد روی 7 segment را تغییر دهیم.

ابتدا با تنظیم حالت های زیر برای تعیین وضعیت های پین های interrupt کلید نخست را به پایه ی INT0 یا به عبارتی PORTD.2 متصل می کنیم. همچنین کلید دوم را به INT1 یا PORTD.3 متصل می کنیم.

این نکته حائز اهمیت است که در چیپ های atmega16/atmega32 دارای 3 پایه ی interrupt هستیم که اسامی آن ها به صورت زیر می باشد:

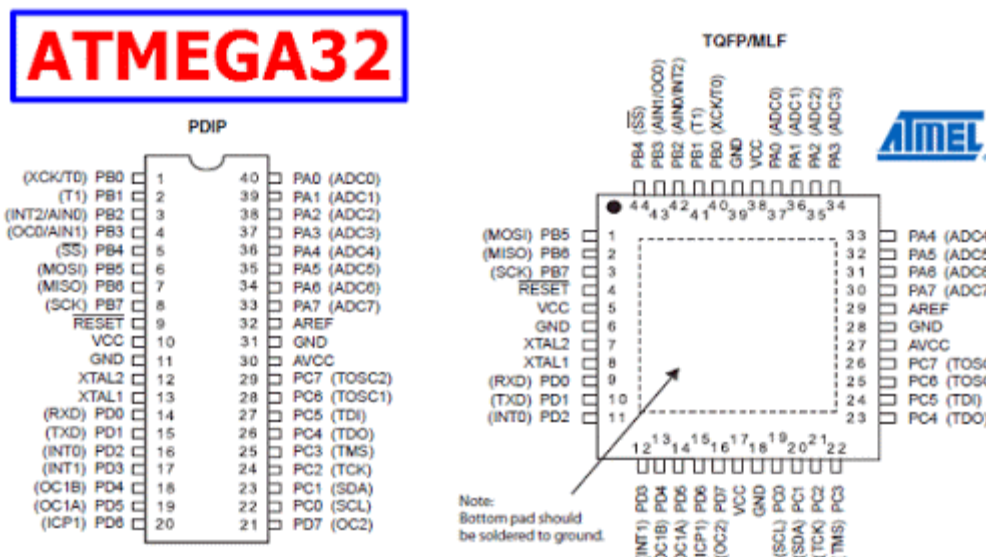
PD2(INT0)

PD3(INT1)

PB1(INT2)

این موضوع از datasheet قابل تشخیص است. نقشه ی سمت چپ مربوط به تایپ SMD می باشد.

SMD stands for surface-mount device.



تنظیمات انجام شده در کدویژن به صورت زیر می باشد:



External Interrupts Settings

☒ INT0 Enabled Mode: Rising Edge ▾

☒ INT1 Enabled Mode: Falling Edge ▾

☐ INT2 Enabled

دستور ها در بلوک interrupt در اولویت اجرا قرار می گیرند و در صورتی که دستور interrupt در هر بخش از کد در مرحله ی اجرا قرار گیرد، اجرا می شود. وضعیت و تعریف falling و rising به صورت زیر می باشد:

ISC2		Description
0		The falling edge on the INT2 pin generates an interrupt request.
1		The rising edge on the INT2 pin generates an interrupt request.

در نهایت برنامه ی نوشته شده به صورت مرحله به مرحله نشان داده می شود.

تعریف مشخصات اولیه، سند کد و لایبرری های مربوطه:

```
1  /**
2   * We wanna increase 7segment number with falling and rising state of keys.
3   * @file KeysBounces.c
4   * @author Erfan Rasti
5   * @version 1.0.0
6   */
7
8  // Setting CPU Frequency
9  #ifndef F_CPU
10 #define F_CPU 8000000UL //clock speed is 8MHz
11 #endif
12
13 // Importing Libraries
14 #include <mega32.h>
15 #include <delay.h>
```

تعریف متغیر های global و دستور های مورد نیاز برای تغییر وضعیت اعداد seven segment در بلوک های interrupt:

```
16
17 // Defining Variables
18 unsigned char digit[10] = {0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe, 0xf6};
19 int i = 0;
20
21 interrupt[EXT_INT0] void ext_int0_isr(void)
22 {
23     // Place your code here
24     i++;
25     delay_ms(20);
26 } // end interrupt 0
27
28 // External Interrupt 1 service routine
29 interrupt[EXT_INT1] void ext_int1_isr(void)
30 {
31     // Place your code here
32     i--;
33     delay_ms(20);
34 } // end interrupt 1
35
```

دستورات و تعاریف از پیش نوشته شده توسط code wizard برای تعیین وضعیت data direct register و initial state:

```
35
36 void main(void)
37 {
38
39     // Input/Output Ports initialization
40     // Port A initialization
41     // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
42     DDRA = (0 << DDA7) | (0 << DDA6) | (0 << DDA5) | (0 << DDA4) | (0 << DDA3) | (0 << DDA2) | (0 << DDA1) | (0 << DDA0);
43     // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
44     PORTA = (0 << PORTA7) | (0 << PORTA6) | (0 << PORTA5) | (0 << PORTA4) | (0 << PORTA3) | (0 << PORTA2) | (0 << PORTA1) | (0 << PORTA0)
45
46     // Port B initialization
47     // Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
48     DDRB = (1 << DDB7) | (1 << DDB6) | (1 << DDB5) | (1 << DDB4) | (1 << DDB3) | (1 << DDB2) | (1 << DDB1) | (1 << DDB0);
49     // State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
50     PORTB = (0 << PORTB7) | (0 << PORTB6) | (0 << PORTB5) | (0 << PORTB4) | (0 << PORTB3) | (0 << PORTB2) | (0 << PORTB1) | (0 << PORTB0)
51
52     // Port C initialization
53     // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=Out Bit0=Out
54     DDRC = (0 << DDC7) | (0 << DDC6) | (0 << DDC5) | (0 << DDC4) | (0 << DDC3) | (0 << DDC2) | (1 << DDC1) | (1 << DDC0);
55     // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=0 Bit0=0
56     PORTC = (0 << PORTC7) | (0 << PORTC6) | (0 << PORTC5) | (0 << PORTC4) | (0 << PORTC3) | (0 << PORTC2) | (0 << PORTC1) | (0 << PORTC0)
57
58     // Port D initialization
59     // Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
60     DDRD = (0 << DDD7) | (0 << DDD6) | (0 << DDD5) | (0 << DDD4) | (0 << DDD3) | (0 << DDD2) | (0 << DDD1) | (0 << DDD0);
61     // State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
62     PORTD = (0 << PORTD7) | (0 << PORTD6) | (0 << PORTD5) | (0 << PORTD4) | (0 << PORTD3) | (0 << PORTD2) | (0 << PORTD1) | (0 << PORTD0)
```

دستور های نوشته شده توسط code wizard جهت راه اندازی interrupt توسط متغیر داخلی GICR و تنظیم وضعیت trigger برای حالات falling و rising:

```
63
64     // External Interrupt(s) initialization
65     // INT0: On
66     // INT0 Mode: Rising Edge
67     // INT1: On
68     // INT1 Mode: Falling Edge
69     // INT2: Off
70     GICR |= (1 << INT1) | (1 << INT0) | (0 << INT2);
71     MCUCR = (1 << ISC11) | (0 << ISC10) | (1 << ISC01) | (1 << ISC00);
72     MCUCSR = (0 << ISC2);
73     GIFR = (1 << INTF1) | (1 << INTF0) | (0 << INTF2);
74
75     // Global enable interrupts
76     #asm("sei")
77
```

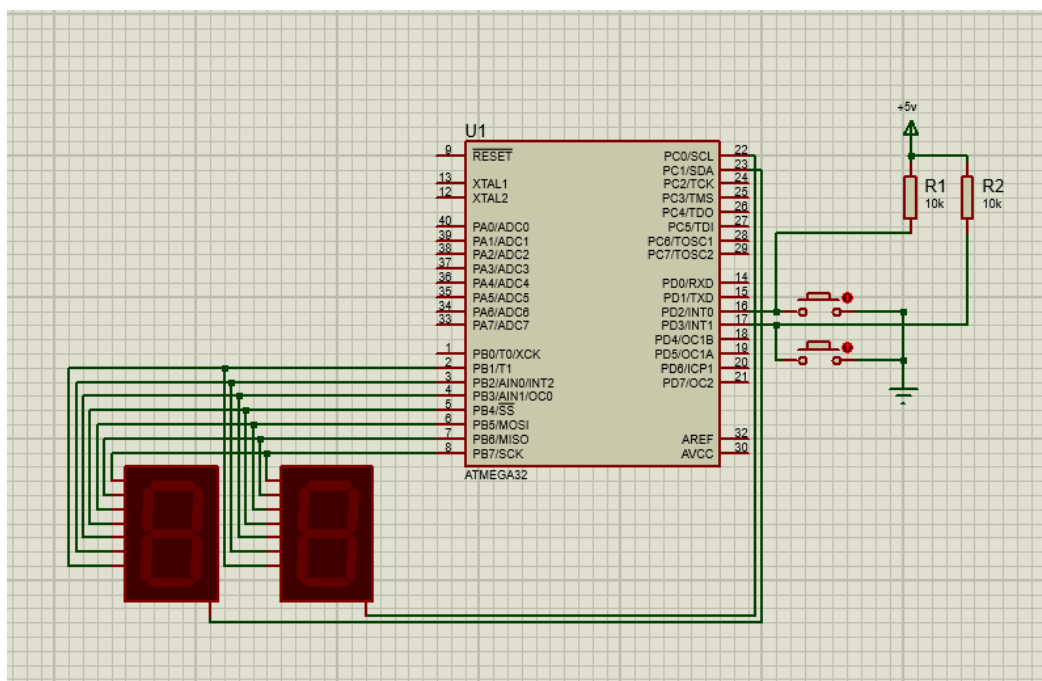
کد های مربوط به while برای نوسان بین دو 7 segment و reset شدن اعداد در صورت برون ریزی اعداد و همچنین پایان main:

```

77
78  while (1)
79  {
80
81      // Restart Digits
82      if ((i > 99) || (i < 0))
83          i = 0;
84
85      PORTC = 0x02;
86      PORTB = digit[i % 10];
87      delay_ms(10);
88
89      PORTC = 0x01;
90      PORTB = digit[i / 10];
91      delay_ms(10);
92
93  } // end while
94 } // end main

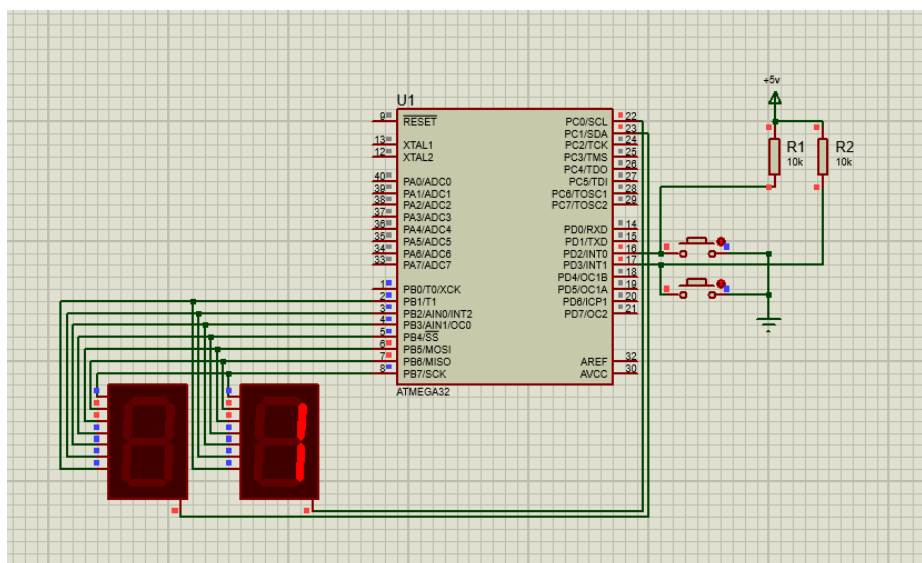
```

برنامه ی بالا توسط compiler به کد hex تبدیل شد و به proteus منتقل شد. اتصالات پروتئوس به صورت زیر می باشد:

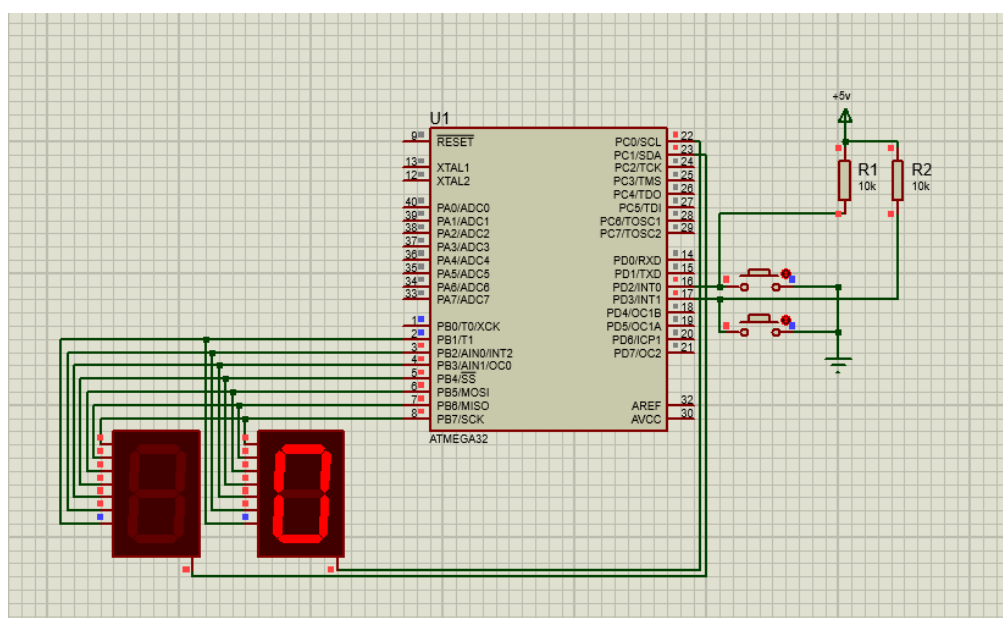


کلید های ورودی به صورت pull up متصل شده اند و 7 segment ها جهت صرفه جویی در پورت ها موازی شده اند.

در صورت فشردن کلید بالایی، پس از رها کردن کلید در مرحله ی rising عدد روی سون سگمنت یک واحد اضافه می شود:



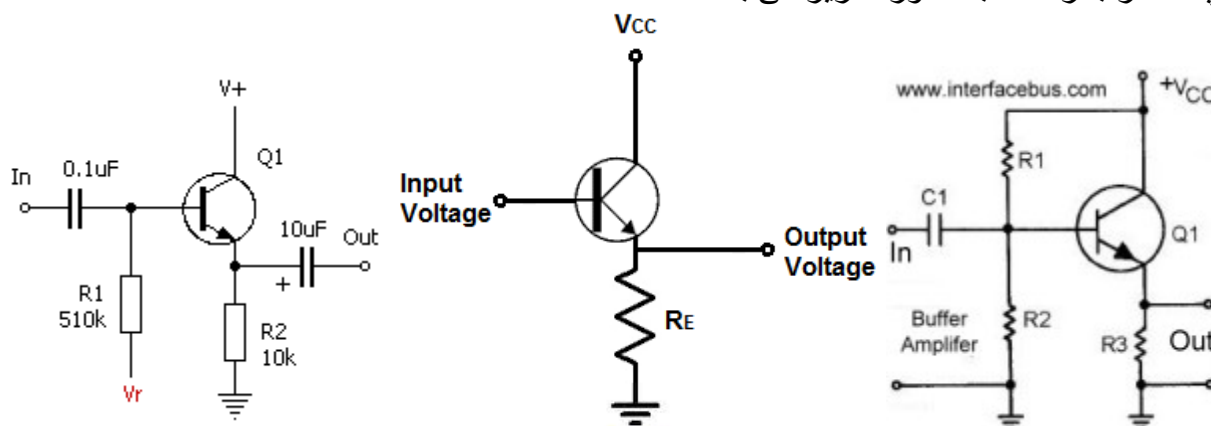
در صورت فشردن کلید پایینی، به محض وصل شدن مدار در حالت falling، عدد یک 7 segment می شود:



Bounce داخلی به وسیله ی دستور interrupt و delay در داخل بلوک interrupt برطرف شد.

در یک مدار واقعی به صورت عملی نیازمند یک مدار بافر برای تنظیم جریان پایه ی common cathode در سون سگمنت ها هستیم تا به چیپ آسیب نرسد.

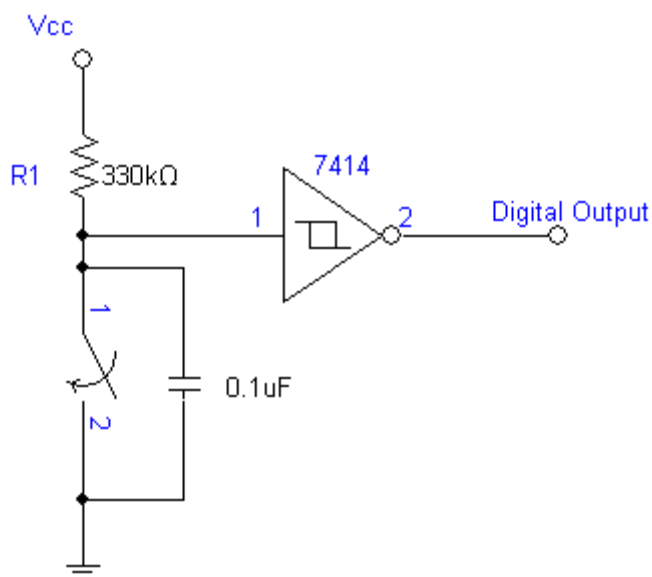
چند مدار بافر ساده به صورت زیر می باشند:



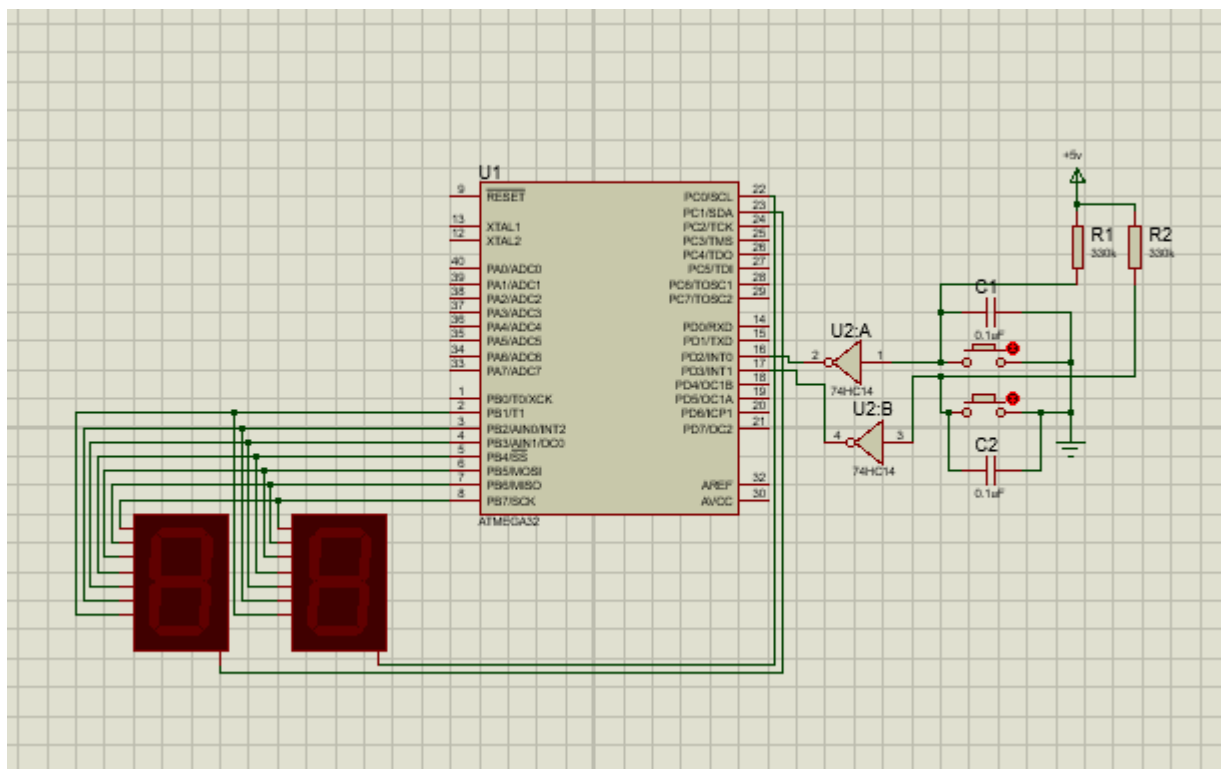
برای بستن مدار با bounce خارجی از مدار اشmitt تريگر استفاده می کنیم. برای این منظور از چیپ 74HC14 استفاده می کنیم.

در صورت debounce خارجی نیازی به تنظیم و ایجاد تاخیر به استفاده از دستور delay نداریم.

مدار مورد نظر به صورت زیر می باشد:



مدار external debounce به صورت زیر می باشد:



تفاوت برنامه نویسی تنها در عدم وجود delay در بلوک های interrupt می باشد.