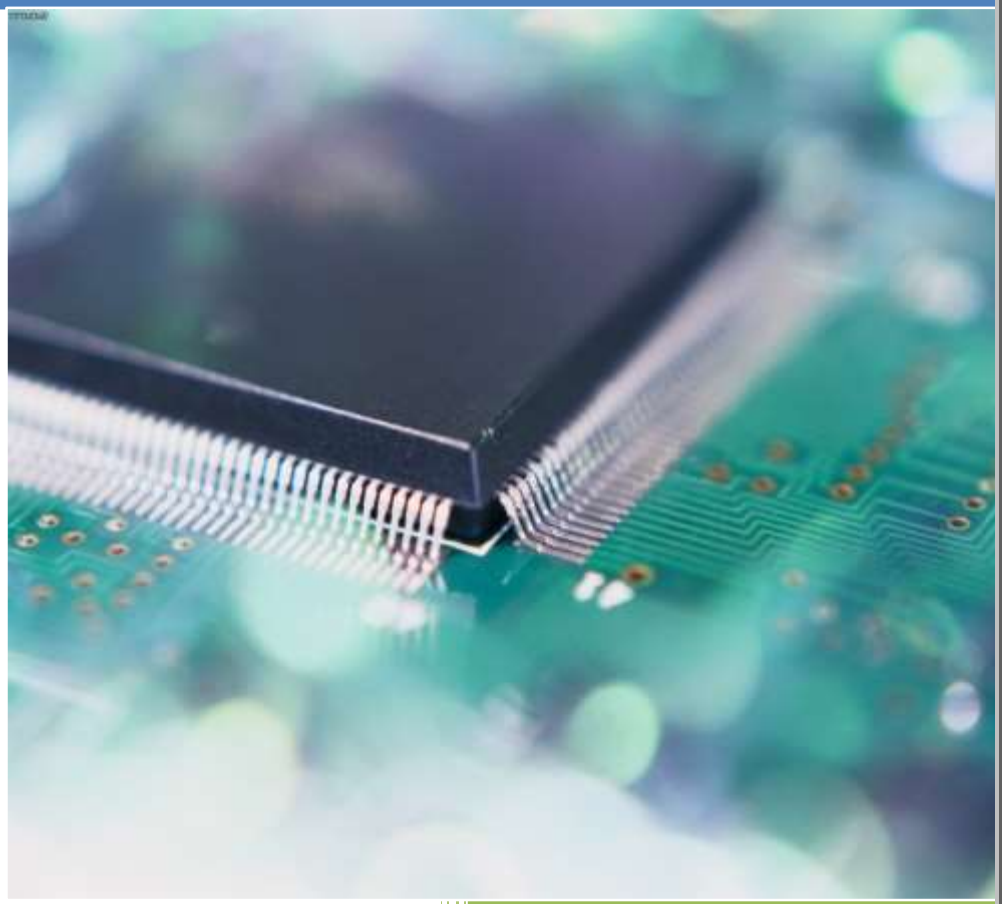
بسمه تعالي

دانشگاه کاشان

آزمایشگاه ریزپردازنده

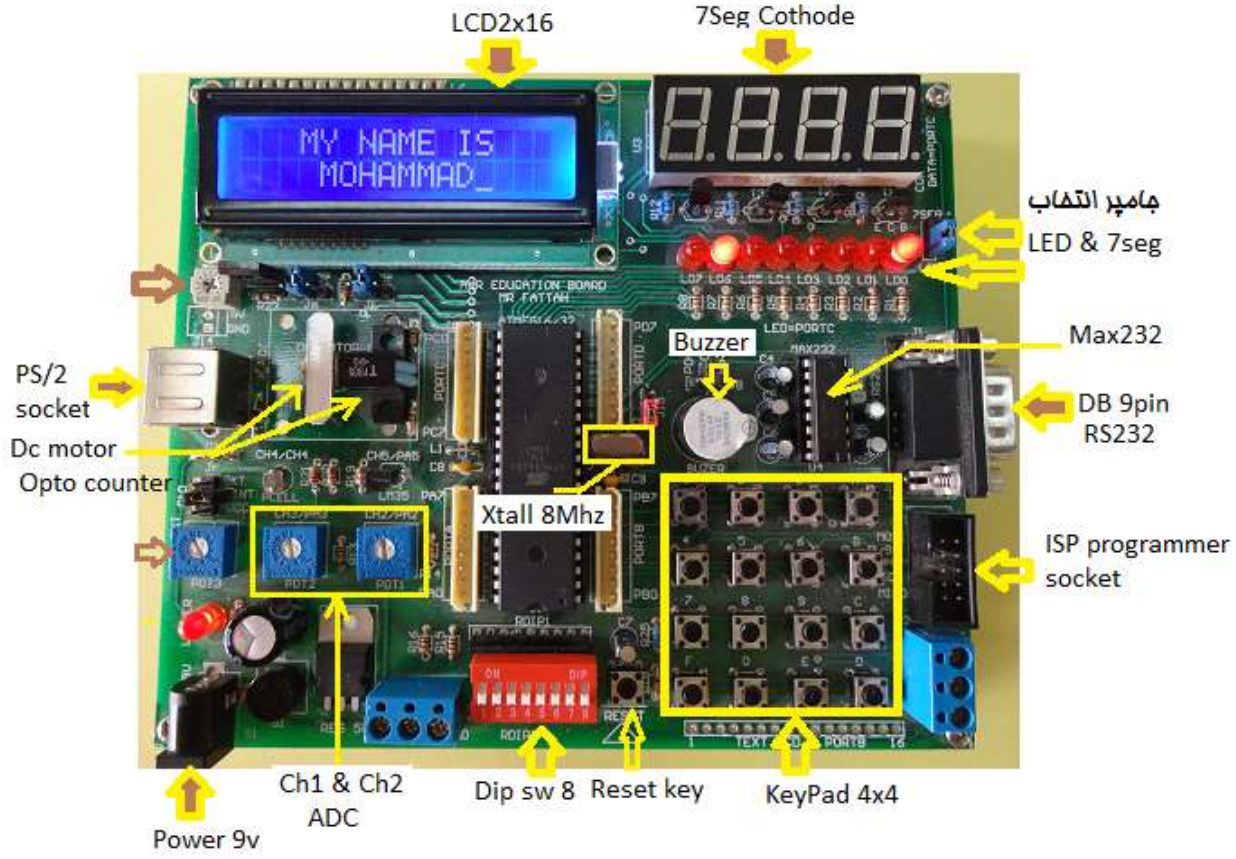


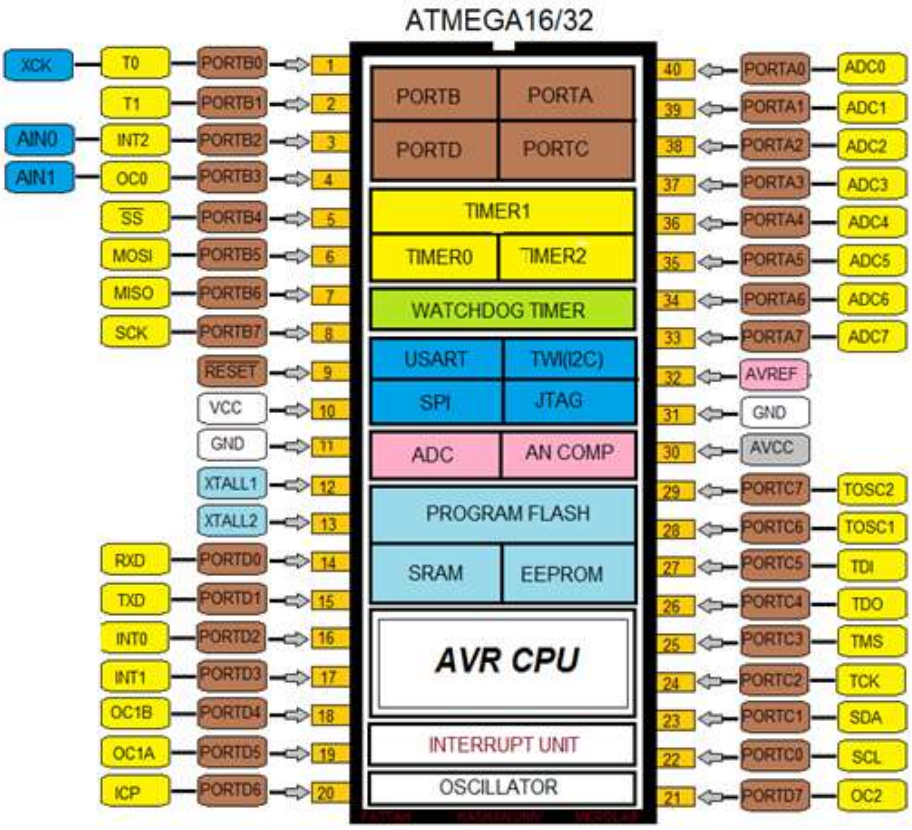
تهیه کننده

محمدرضا فتاح

دانشکدهٔ برق و کامپیوتر

بسمه تعالی


LCD2x16
7Seg Cothode
MY NAME IS
MOHAMMAD_
جامپر انتخاب
LED & 7seg
Max232
DB 9pin
RS232
Buzzer
PS/2
socket
Dc motor
Opto counter
Xtall 8Mhz
ISP programmer
socket
Ch1 & Ch2
ADC
Power 9v
Dip sw 8
Reset key
KeyPad 4x4


ATMEGA16/32
XCK
T0
PORTB0
1
T1
PORTB1
2
AIN0
INT2
PORTB2
3
AIN1
OC0
PORTB3
4
SS
PORTB4
5
MOSI
PORTB5
6
MISO
PORTB6
7
SCK
PORTB7
8
RESET
9
VCC
10
GND
11
XTALL1
12
XTALL2
13
RXD
PORTD0
14
TXD
PORTD1
15
INT0
PORTD2
16
INT1
PORTD3
17
OC1B
PORTD4
18
OC1A
PORTD5
19
ICP
PORTD6
20
PORTB
PORTA
PORTD
PORTC
TIMER1
TIMER0
TIMER2
WATCHDOG TIMER
USART
TWI(I2C)
SPI
JTAG
ADC
AN COMP
PROGRAM FLASH
SRAM
EEPROM
AVR CPU
INTERRUPT UNIT
OSCILLATOR
40
PORTA0
ADC0
39
PORTA1
ADC1
38
PORTA2
ADC2
37
PORTA3
ADC3
36
PORTA4
ADC4
35
PORTA5
ADC5
34
PORTA6
ADC6
33
PORTA7
ADC7
32
AVREF
31
GND
30
AVCC
29
PORTC7
TOSC2
28
PORTC6
TOSC1
27
PORTC5
TDI
26
PORTC4
TDO
25
PORTC3
TMS
24
PORTC2
TCK
23
PORTC1
SDA
22
PORTC0
SCL
21
PORTD7
OC2

آزمایش اول

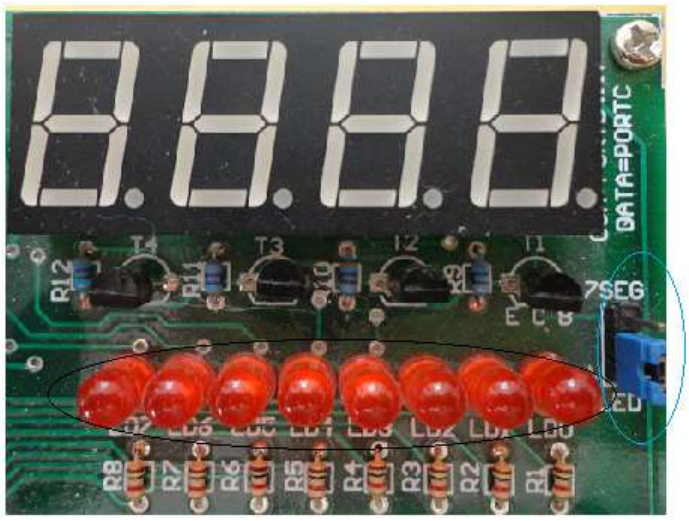
آشنایی با پورتهای ورودی و خروجی

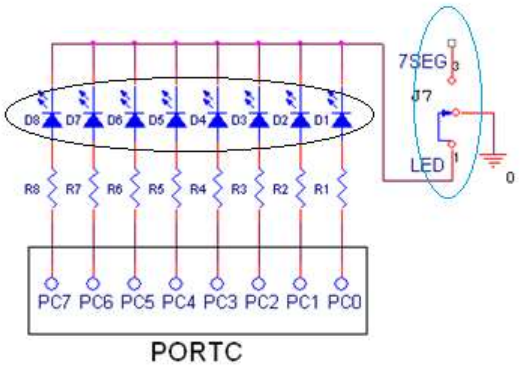
پیش نیاز: مطالعهٔ ضمیمهٔ آزمایش

آزمایش ۱:شمارندهٔ باینری با نمایش روی LED

برنامهٔ شمارندهٔ باینری هشت بیتی با تأخیر نیم ثانیه را در محیط CodeVision نوشته و سپس آنرا کامپایل کنید. در صورت نداشتن خطا ، فایل HEX آنرا روی میکروکنترلر بورد آموزشی پروگرام کنید و اجرای آنرا ببینید .

توجه : برای فعال شدن LED های متصل به پورت C جامپر J7 باید در وضعیت LED قرار گیرد .



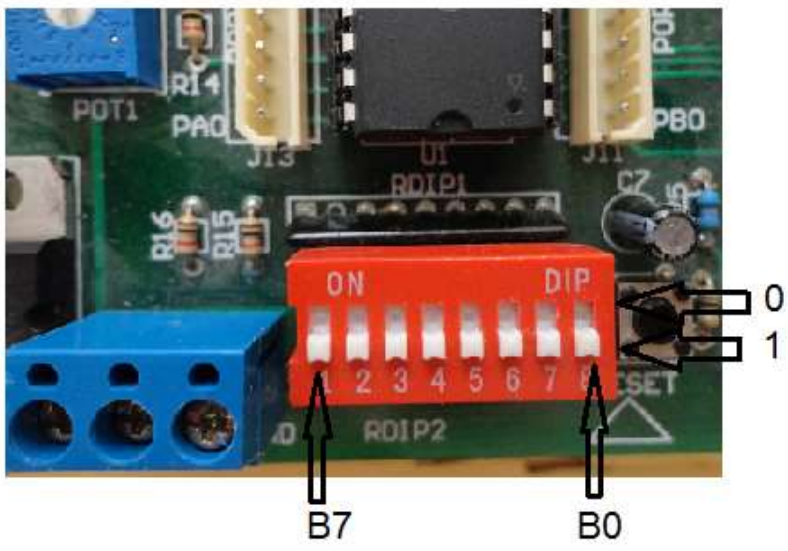


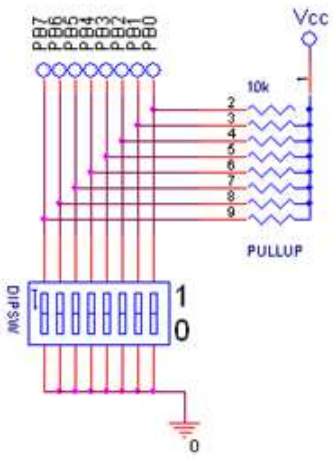
آزمایش ۲ : شمارندهٔ جانسون

برنامه ای بنویسید که شمارندهٔ جانسون هشت بیتی را روی LEDها نمایش دهد.بطوریکه LEDها از سمت راست یک به یک روشن شده تا اینکه هر هشت LED روشن شود و سپس از سمت راست به همان ترتیب خاموش گردد.

آزمایش۳ : تبدیل باینری به دهدهی

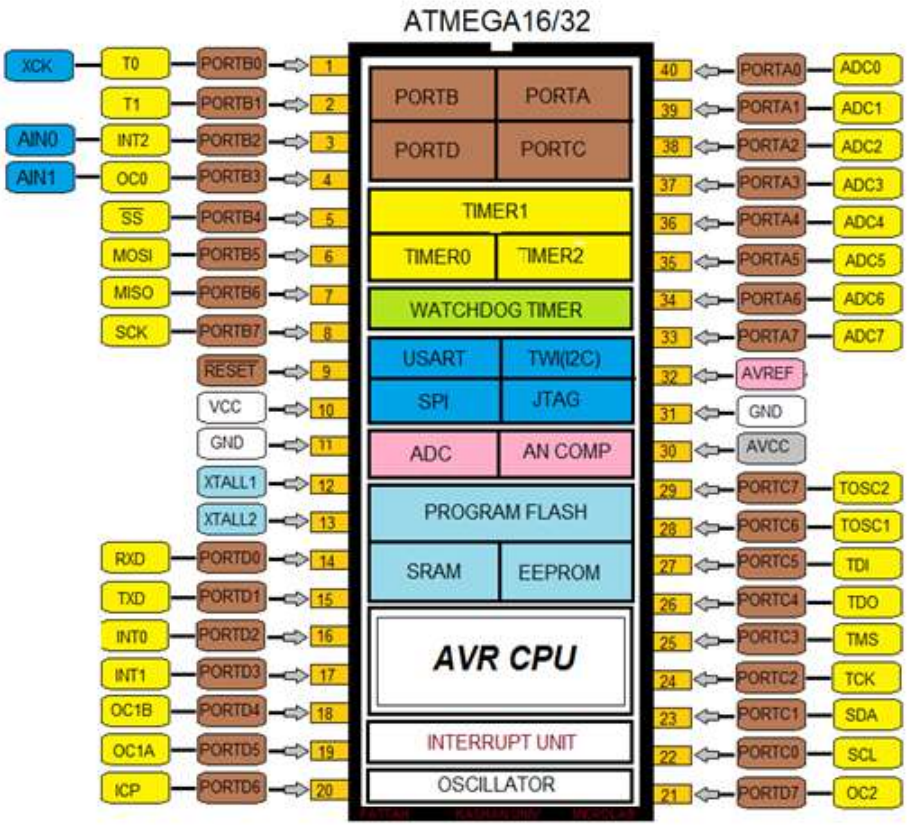
برنامه بنویسید که مقدار باینری سوئیچ هشت تایی را بخواند و مقدار دهدهی آن را روی LED ها نمایش دهد. بطوریکه رقم یکان و دهگان بصورت همزمان بترتیب روی چهار LED کم ارزش و پرارزش در مدت یک ثانیه قرار گیرد و سپس رقم صدگان





نیز بمدت یک ثانیه روی چهار LED کم ارزش جایگزین شود. برای بدست آوردن هر یک از ارقام عدد از عملیات تقسیم (/) و باقیمانده (%) استفاده کنید.

ضمیمهٔ آزمایش اول :



الف – پورتهای میکروکنترلر ATmega16/32 **:**

همانطور که در شکل بالا می بینید این میکروکنترلر دارای چهار پورت هشت بیتی است بنامهای A ، B ، C و D که هر بیت از این پورتها می تواند به صورت مجزا بعنوان ورودی و یا خروجی مورد استفاده قرار گیرد . پس می توان گفت این تراشه دارای ۳۲ دریچهٔ I/O مستقل است .

در این تراشه هر پورت هشت بیتی دارای دو ثبات و یک بافر برای تعیین وضعیت پورت و انتقال اطلاعات است که در زیر وظیفهٔ هر کدام توضیح داده شده است :

ثبات DDRx : این ثبات ورودی و یا خروجی بودن پورت را مشخص می‌کند. اگر در هر بیت این ثبات مقدار یک نوشته شود پایهٔ معادل این بیت به عنوان خروجی و اگر مقدار صفر نوشته شود پایه به عنوان ورودی تعریف می‌گردد .

مثال :

```
DDRA = 0x0F
```

این دستور مشخص می‌کند که چهار بیت کم آرزش پورت A بعنوان خروجی و چهار بیت پرارزش بعنوان ورودی تعریف شود.

ثبات PORTx : مقدار نوشته شده در این ثبات روی پایه های پورت قرار می گیرد به شرط اینکه پورت بعنوان خروجی تعریف شده باشد . اگر پایه در حالت ورودی تعریف شده باشد مقدار نوشته شده در این ثبات مشخص کنندهٔ وضعیت فعال و یا غیر فعال مقاومت Pullup است .

مثال :

```
PORTB = 0xF5;
```

این دستور مقدار باینری (11110101) را روی پایه های پورت B قرار می‌دهد . توجه داشته باشید که اگر پورت در حالت خروجی تعریف نشده باشد مقدار نوشته شده در این ثبات روی پایه های پورت قرار نخواهد گرفت .

بافر PINx : اگر پورت را بعنوان ورودی تعریف کرده باشیم مقداری که توسط عنصر خارجی روی پایه های پورت قرار داده شده را آز طریق این بافر دریافت می کنیم . به دستور زیر توجه کنید :

```
A=PINB;
```

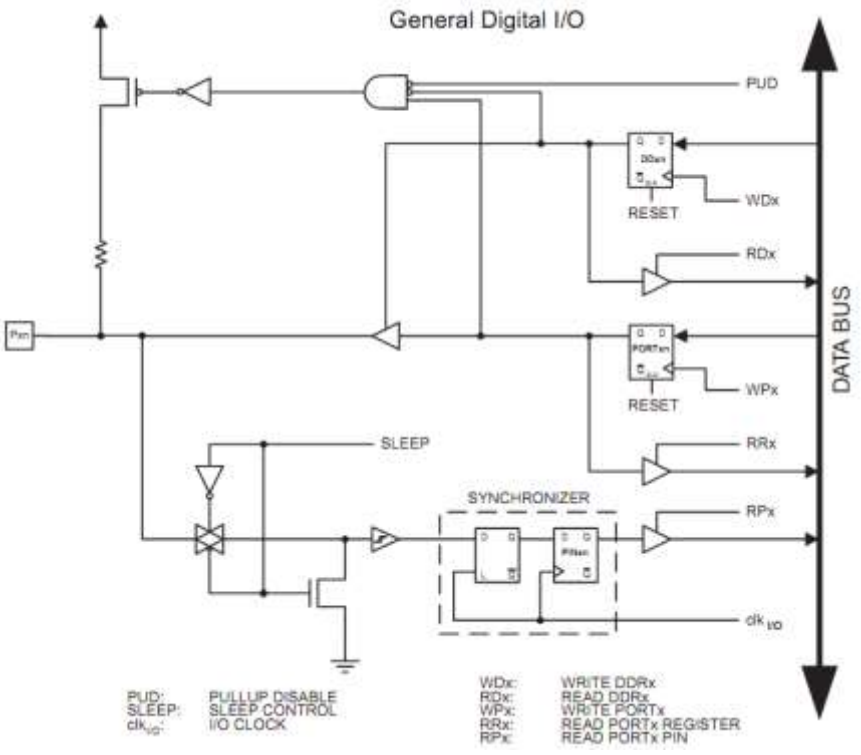
این دستور مقدار قرار گرفته روی پایه های پورت B را در متغیر A قرار می دهد .

البته به این نکته نیز باید توجه داشته باشید که اگر پورت را بعنوان ورودی تعریف می‌کنید باید مقاومت داخلی Pullup آنرا فعال کنید و یا اینکه یک مقاومت خارجی بین هر کدام از پایه های پورت و ولتاژ Vcc قرار دهید . مقدار این مقاومت میتواند از 1KΩ تا 100KΩ باشد . به مثال زیر توجه کنید :

```
DDRA= 0x0f;

PORTA= 0xf0 ;
```

دستور اول ، چهار پایهٔ کم ارزش پورت A را بعنوان خروجی و چهار بیت بالا را بعنوان ورودی تعریف می‌کند . دستور بعدی با نوشتن مقدار یک در چهار بیت بالای ثبات PORTA باعث فعال شدن مقاومت Pullup داخلی چهار پایهٔ بالای پورت A می‌شود . و همینطور مقدار (0000) را روی چهار پایهٔ کم ارزش پورت که بعنوان خروجی تعریف شده‌است قرار می‌دهد .



مدار داخلی هر یک از پایه های پورت

ب – فرکانس کار میکروکنترلر:

در میکروکنترلها ، مرجع تولید فرکانس می تواند بصورت داخلی و یا خارجی تعیین گردد . این کار توسط فیوزبیتهای داخلی میکروکنترلر و در هنگام برنامه ریزی انجام می گیرد. در حالت پیش فرض و در هنگام ساخت مقدار این فرکانس یک مگاهرتز تعیین می شود. چهار فیوز بیت CKSEL0 تا CKSEL3 برای این کار در نظر گرفته شده است. در بورد آموزشی آزمایشگاه یک کریستال 8MHz خارجی روی دو پایهٔ XTAL1 و XTAL2 قرار داده شده است بنابراین در آزمایشها برای تعیین فرکانس از حالت کریستال خارجی استفاده می شود پس در اینصورت فرکانس کاری میکروکنترلر مقدار هشت مگاهرتز خواهد بود. برای

انتخاب این حالت در هنگام برنامه ریزی هیچ یک از فیوزبیتهای مذکور نباید انتخاب شود. در میکروکنترلرهای AVR اکثر دستورات در یک سیکل و بعضی از آنها در دو و یا سه سیکل اجرا می شوند . زمان یک سیکل برابر با عکس فرکانس کاری میکروکنترلر است : Tcycel=1/8 µsec

ج – جدول آدرس پورتهای AVR

I\O PORT:

Port	Register	Function	Port-Address
A	PORTA	Data Register	0x1B
	DDRA	Data Direction Register	0x1A
	PINA	Input Pins Address	0x19
B	PORTB	Data Register	0x18
	DDRB	Data Direction Register	0x17
	PINB	Input Pins Address	0x16
C	PORTC	Data Register	0x15
	DDRC	Data Direction Register	0x14
	PINC	Input Pins Address	0x13
D	PORTD	Data Register	0x12
	DDRD	Data Direction Register	0x11
	PIND	Input Pins Address	0x10

Port Pin Configurations:

DDxn	PORTxn	PUD (inSFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

د- ثبات وضعیت پردازندهٔ AVR

Status-Register, Accumulator flags

Port	Function	Port-Address	RAM-Address
SREG	Status Register Accumulator	0x3F	0x5F

7	6	5	4	3	2	1	0
I	T	H	S	V	N	Z	C

آزمایش دوم

خواندن پورت ورودی با روش سرزدن و وقفه

پیش نیاز: با مطالعهٔ ضمیمهٔ آزمایش به سوالات زیر پاسخ دهید:

۱- میکروکنترلر Atmega16 دارای چند وقفهٔ خارجیست و هر یک روی کدام پایه های میکروکنترلر قرار دارد؟

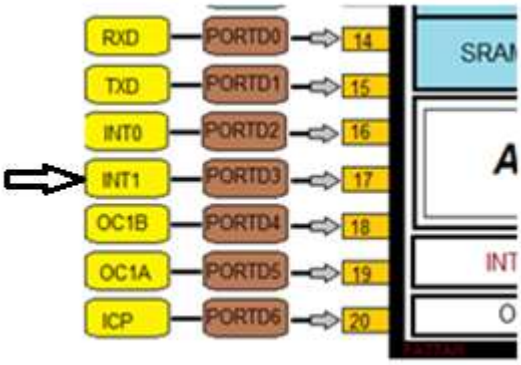
۲- کدام یک از وقفه های خارجی دارای اولویت بالاتری نسبت به دیگری است؟

۳- برای فعال کردن وقفهٔ خارجی چه ثباتهایی باید مقدار دهی شوند؟

آزمایش ۱ : شمارش تغییرات روی پایهٔ پورت

برنامه ای بنویسید که تعداد تغییر از حالت صفر به یک روی پایهٔ PD3 را شمارش نماید. مقدار شمارش روی LED ها نمایش داده شود. برای ایجاد تغییرات روی این پایه با استفاده از سیم مناسب و بصورت متناوب این پایه را به ولتاژ ۵ ولت و زمین متصل کنید. (با اضافه نمودن دستورات مناسب در برنامه ، اثر لرزش دست را خنثی نمایید)

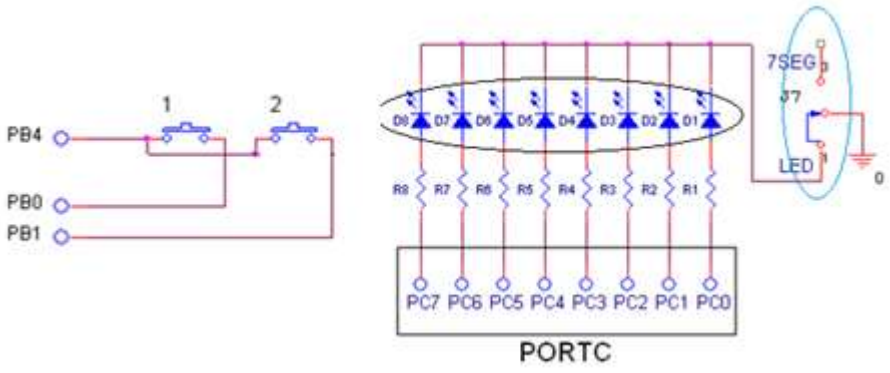
آزمایش ۲ : شمارش تغییرات روی پایهٔ پورت با استفاده از وقفه



برنامه ای بنویسید که با استفاده از وقفه خارجی یک (INT1) که روی پایهٔ PD3 قرار دارد شمارش تغییرات روی این پایه بصورت دهدهی و روی LED ها انجام شود. برای این حالت چهار LED اول مقدار یکان و چهار LED دوم مقدار دهگان را نمایش دهد. وقفه را در حالت حساس به لبهٔ پایین رونده تنظیم نمایید.

آزمایش ۳ : خواندن کلید فشاری (Push Button)

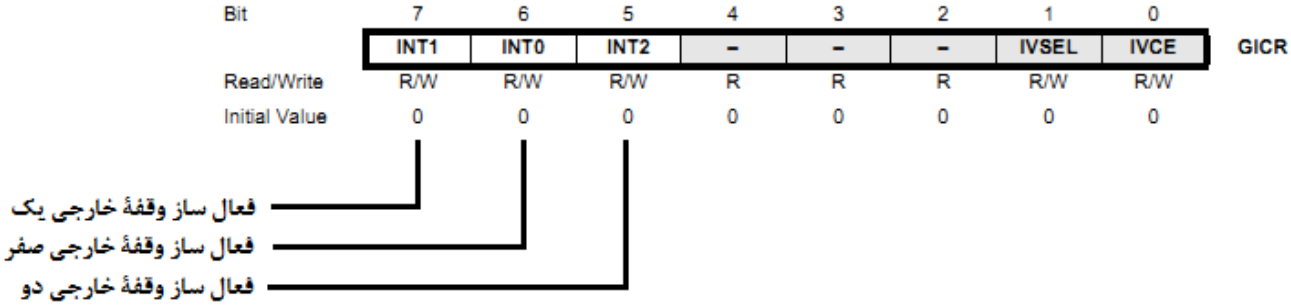
یک شمارندهٔ دهدهی روی LED ها بسازید بطوریکه با هر بار فشار دادن کلید '1' از صفحه کلید ماتریسی مقدار آن افزایش و با هر بار فشار دادن کلید '2' مقدار آن کاهش یابد.(فرض کنید فقط همین دو کلید روی بورد وجود دارد)



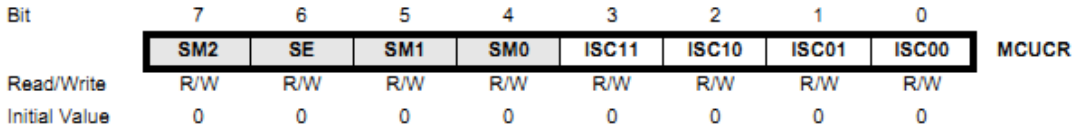
توجه : برای خواندن کلید فشاری یک پایهٔ آن باید به زمین متصل و پایهٔ دیگر به یک پورت ورودی متصل شود و سپس وضعیت آن خوانده شود. روی بورد آموزشی یکی از پایه های کلید '1' به پورت PB4 و پایهٔ دیگر آن به پورت PB0 متصل است بنابراین پایهٔ PB4 را در حالت خروجی قرار داده و مقدار صفر را در آن بنویسید و PB0 را در حالت ورودی قرار داده و وضعیت کلید را از طریق آن بخوانید.

ضمیمهٔ آزمایش دوم : وقفه های خارجی

میکروکنترلر Atmega16 دارای سه وقفهٔ خارجی INT0 ، INT1 و INT2 بترتیب روی پایه های PD2 ، PD3 و PB2 است. برای فعال کردن هر یک از این وقفه ها ابتدا باید پورت متناظر را در حالت ورودی قرار داد و مقاومت PullUp داخلی را نیز فعال کرد. در مرحلهٔ دوم برای فعال کردن هر یک از وقفه های مذکور بیت فعال ساز متناظر هر یک را در ثبات GICR باید در حالت یک قرار داد.



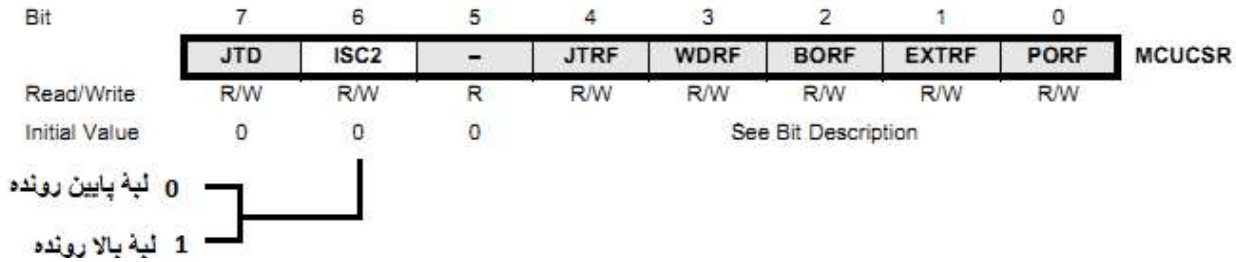
در مرحلهٔ سوم برای هر یک از وقفه های INT0 و INT1 باید نوع حساسیت وقفه را در ثبات MCUCR مطابق شکل و جداول زیر مشخص کرد.



ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

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.

برای وقفهٔ خارجی دو این کار در ثبات MCUCSR انجام می گیرد.



در مرحلهٔ آخر بیت وقفهٔ عمومی (I) در ثبات وضعیت پردازنده (SREG) باید در حالت یک قرار گیرد.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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 کامپایلر Codevision این کار با دستور (**#asm**("**sei**") انجام می گیرد. به مثال زیر توجه کنید.

مثال: در این مثال با هر بار تحریک وقفهٔ خارجی INT1 در لبهٔ بالارونده به مقدار پورت C یک واحد اضافه می شود.

```
#include <mega16a.h>
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
  PORTC++;
}

void main(void)
{
        PORTC=0x00;
        DDRC=0xff ;
        DDRD=0x00;
        PORTD= 0x08;
        // External Interrupt(s) initialization
        GICR |= (1<<INT1) | (0<<INT0) | (0<<INT2);          // INT1: On
        MCUCR = (1<<ISC11) | (1<<ISC10) | (0<<ISC01) | (0<<ISC00); // INT1 Mode: Rising Edge
        GIFR = (1<<INTF1) | (0<<INTF0) | (0<<INTF2); //Reset INT1 flag
        #asm("sei")    // Global enable interrupts
        while (1) {  }
}
```

جدول بردارهای وقفه :

Table 18. Reset and Interrupt Vectors

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

آزمایش سوم

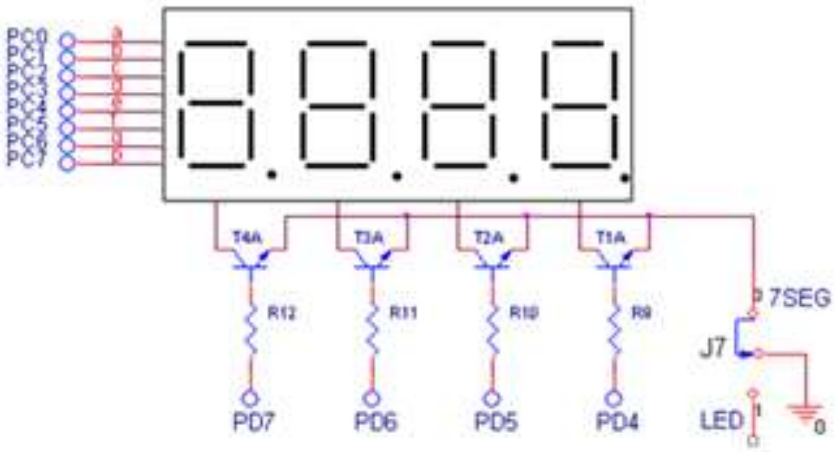
نمایش اعداد توسط نمایشگر سون سگمنت چهار تایی

پیش گزارش: با مطالعهٔ ضمیمهٔ آزمایش به سوالات زیر پاسخ دهید.

۱- با توجه به توضیحات ضمیمه، تابع نمایش سون سگمنت با نام SevenSegDisplay را تکمیل کنید.

۲- آرایهٔ کدهای سون سگمنت برای اعداد 0 تا F را کامل کنید.

۳- در تابع نمایش، مقدار حداکثر تاخیر ((Delay_ms(x) برای هر سون سگمنت چقدر می تواند باشد؟

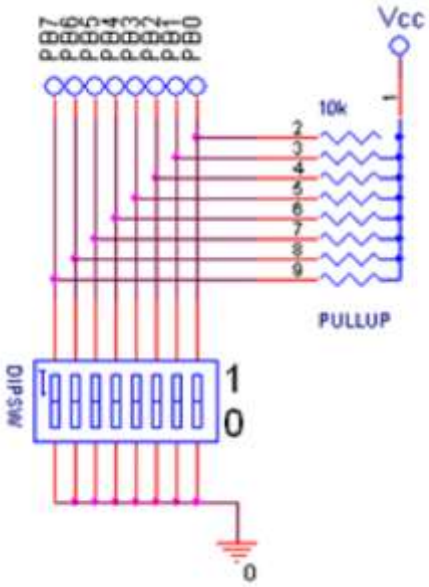


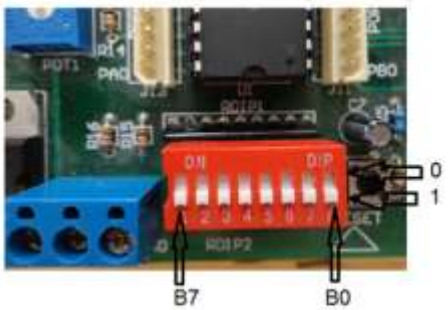
آزمایش ۱ : نمایش اعداد بصورت چرخشی

با استفاده از تابع نمایش سون سگمنت برنامه‌ای بنویسید که اعداد 0 تا 9 را به صورت چرخشی روی سون‌سگمنت چهارتایی نمایش دهد به طوریکه اعداد از سمت راست وارد و از سمت چپ خارج شوند.

آزمایش ۲ : نمایش اعداد چند رقمی

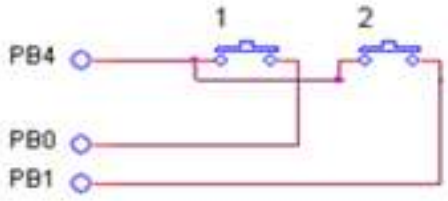
برنامه ای بنویسید که مقدار چهار بیت پایین سوئیچ هشت تایی در مقدار چهار بیت بالای آن ضرب شده و حاصل ضرب توسط سون سگمنت نشان داده شود. (سوئیچ هشت تایی (دیپ سوئیچ) به پورت B متصل است)





آزمایش ۳ : شمارش صعودی و نزولی روی سون سگمنت

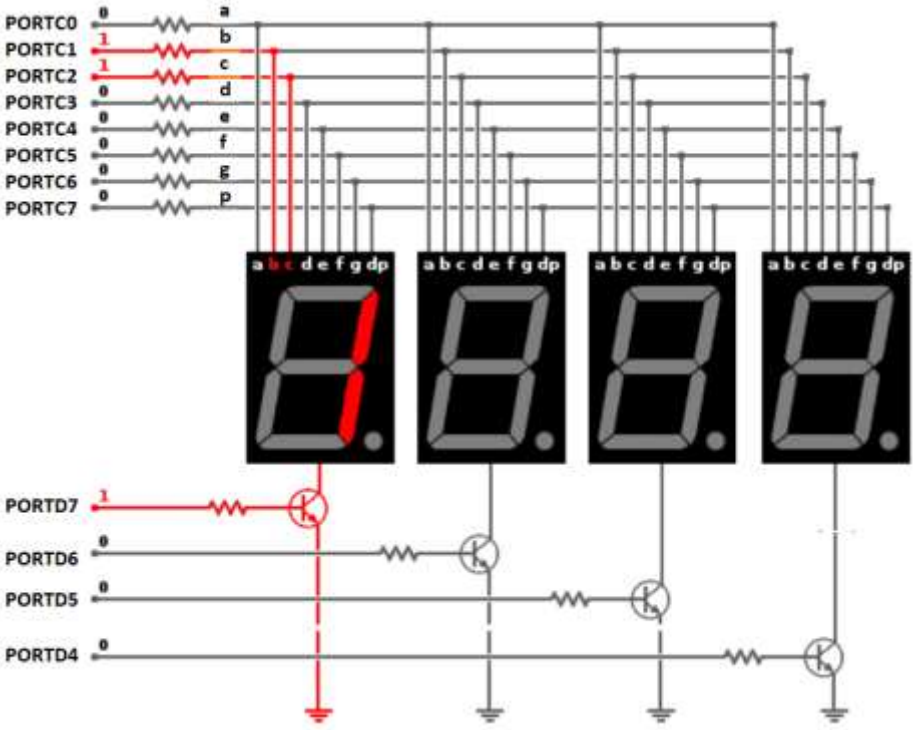
یک شمارندهٔ دهدهی چهاررقمی روی سون سگمنت بسازید بطوریکه با هر بار فشار دادن کلید ’1‘ از صفحه کلید ماتریسی مقدار آن افزایش و با هر بار فشار دادن کلید ’2‘ مقدار آن کاهش یابد.(فرض کنید فقط همین دو کلید روی بورد وجود دارد)



ضمیمهٔ آزمایش سوم :

از سون سگمنت برای نمایش اعداد دسیمال (0 تا 9) و یا هگزا دسیمال (0 تا F) استفاده می شود. با چگونگی کار با این اِلمان در آزمایشگاه مدار منطقی آشنا شده اید. برای یادآوری لازم بذکر است که برای نمایش یک عدد توسط سون سگمنت ابتدا باید پایهٔ مشترک آنرا فعال کرد. برای سون سگمنت کاتد مشترک این پایه به زمین و برای آند مشترک به ولتاژ بالا متصل می شود. سپس کد سون سگمنت عدد مورد نظر باید روی پایه های دیتای سون سگمنت قرار گیرد. مثلاً برای نمایش عدد 1 مقدار کد معادل آن یعنی 0b00000110= 0x06 روی خطوط دیتا قرار داده می شود.

مثال بالا در شکل زیر برای سون سگمنت سمت چپ نمایش داده شده است. در اینجا با یک کردن PD7 ترانزیستور مربوطه روشن شده در نتیجه کاتد سون سگمنت به زمین متصل می شود و همچنین کد سون سگمنت عدد 1 (0b0000 0110) نیز توسط پورت C روی خط دیتای سون سگمنت قرار داده شده است.



حال اگر بخواهیم یک عدد چهار رقمی را توسط سون سگمنت چهارتایی نمایش دهیم چگونه باید عمل کنیم؟

در این نوع سون سگمنت در هر لحظه فقط می توان یکی از سون سگمنتها را فعال کرد در غیر این صورت هر چهار سون سگمنت یک عدد را نمایش خواهند داد. برای حل این مشکل از خطای چشم در تشخیص روشن و خاموش شدن سریع یک

لامپ استفاده می شود. اگر یک لامپ در بازۀ ۲۰ میلی ثانیه بصورت پی در پی روشن و خاموش شود چشم انسان آنرا همیشه روش می بیند.

بطور مثال اگر سون سگمنت اول از سمت راست را بمدت ۵ میلی ثانیه فعال کنیم و عدد مثلاً 4 را روی آن نمایش دهیم سپس آنرا خاموش کرده و این کار را برای سون سگمنت دوم و برای عدد 3 تکرار کنیم و همچنین برای دو سون سگمنت بعدی این کار را برای اعداد 2 و 1 انجام دهیم و این پروسه را بصورت پی در پی تکرار کنیم عدد چهار رقمی 1234 را روی نمایشگر خواهیم دید.

مقدار تأخیر برای هر سون سگمنت طوری انتخاب می شود که کل زمان نمایش برای چهار سون سگمنت از ۲۰ میلی ثانیه بیشتر نشود. بازۀ مناسب می تواند از ۳ تا ۵ میلی ثانیه انتخاب شود. هر چه این زمان کمتر باشد روشنایی سون سگمنت کمتر خواهد بود.

زیر برنامۀ زیر با نام SevenSegDisplay یک عدد چهار رقمی را از طریق آرایۀ چهارتایی digit دریافت و هر یک از ارقام یکان تا هزارگان را بترتیب روی سون سگمنت چهارتایی برای یک بار نمایش می دهد و سپس از زیر برنامه خارج می شود. حال اگر این تابع بصورت پیوسته فراخوانی شود عدد چهار رقمی مذکور روی سون سگمنت نمایش داده خواهد شد.

توجه : با توجه به توضیحات بالا این زیر برنامه و آرایۀ کدهای سون سگمنت را تکمیل نمایید. آرایۀ []SevenSegCode ، شامل کد اعداد از 0 تا F در مبنای هگزادسیمال است. بطور مثال عدد 0x3f کد سون سگمنت عدد صفر است.

```
flash char SevenSegCode[16]={0x3f , 0x06 , ….}; // آرایه شامل کدهای سون سگمنت اعداد
void    SevenSegDisplay ( char digit[4] ){
        DDRD |= 0xf0 ;       // تعریف چهار بیت بالای پورت بعنوان خروجی (خطوط کنترل نمایشگر)
        DDRC = 0xff ;        // تعریف تمام پورت بعنوان خروجی(خط دیتای نمایشگر)
        PORTD.4=1;           // فعال کردن سون سگمنت اول سمت راست (رقم یکان)
        PORTC = SevenSegCode[ digit[0] ] ;   // گذاشتن کد سون سگمنت رقم یکان روی خط دیتا
        Delay_ms(3);      //ایجاد تأخیر مناسب
        PORTD.4 = 0;         // غیر فعال کردن سون سگمنت اول
        .
        .
        .
}
```

مثال : با استفاده از زیر برنامۀ بالا و تابع اصلی main که در زیر آمده سون سگمنت عدد چهار رقمی 1234 را نمایش خواهد داد. در برنامۀ زیر و در حلقۀ بی نهایت while زیر برنامه نمایش بصورت پیوسته فراخوانی می شود.

```
int     main ( void ){
        char digit[4]={4 , 3 , 2 , 1};
                while(1){
                   SevenSegCode[ digit] ;
                }
        }
```

آزمایش چهارم

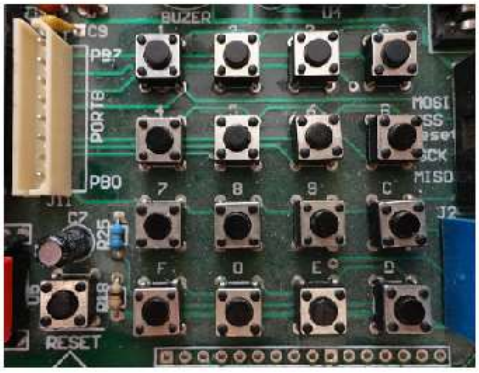
خواندن صفحه کلید ماتریسی و نمایش LCD

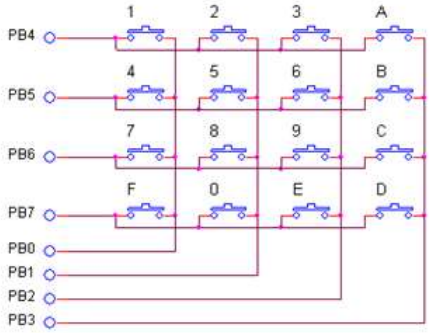
پیش گزارش:

با توجه به شکل صفحه کلید ماتریسی بورد برنامه ای با قالب زیر بنویسید که صفحه کلید را یکبار بخواند و در صورت فشار داده شدن کلید کد آنرا بازگرداند(15-0) و در غیر این صورت کد اختصاصی 0x80 را ارسال کند.

```
Char  GetKeyCode( void ){

}
```



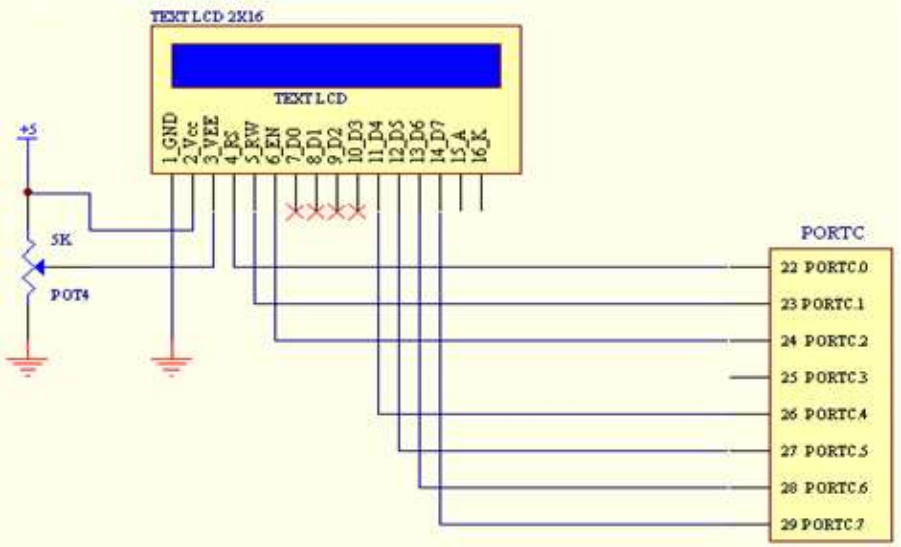


آزمایش ۱ : نمایش کد کلید توسط سون سگمنت

با استفاده از زیر برنامهٔ صفحه کلید برنامه‌ای بنویسید که با فشار دادن هر کلید از صفحه کلید شانزده تایی کد آنرا در مبنای شانزده روی بلوک سون سگمنت نمایش دهد بصورتیکه با هر بار فشار دادن یک کلید ، کد کلید قبلی روی نمایشگر به سمت چپ منتقل شده و کد کلید جدید روی سون سگمنت اول جایگزین گردد .

آزمایش ۲ : نمایش کد کلید روی LCD

برنامه ای بنویسید که با فشار دادن هر کلید از صفحه کلید ماتریسی ، کد کلید زوج روی خط اول LCD و در موقعیت متناسب با عدد کلید و کد کلید فرد روی خط دوم ودر موقعیت متناسب با عدد کلید نمایش داده شود . مثلاً با فشار دادن کلید 8 ، عدد 8 در موقعیت هشتم از خط اول قرار گیرد . البته محتویات صفحه LCD نباید در نمایش هر کاراکتر پاک شود.



ضمیمهٔ آزمایش سوم:

کامپایلر CodeVision دارای کتابخانه LCD کاراکتری و گرافیکی است. در زیر چند تابع مهم این کتابخانه برای کار با LCD های کاراکتری آورده شده است:

- lcd_init(x)

این تابع برای آماده سازی اولیهٔ LCD مورد استفاده قرار می گیرد. مقدار x می تواند ۸ ، ۱۶ و یا ۲۰ باشد. این مقدار تعداد کاراکتر قابل نمایش روی یک خط LCD را مشخص می کند.

- lcd_putchar(ch)

این تابع برای نمایش یک کاراکتر از نوع Ascii روی LCD بکار می رود. مانند :

```
lcd_putchar( 'A' );                    نمایش حرف A
char  ch= '1';
lcd_putchar(ch);                       نمایش مقدار متغیر ch
```

- lcd_puts(char *str)

این تابع برای نمایش یک رشته از نوع Ascii که در حافظهٔ RAM قرار دارد روی LCD استفاده می شود.

- lcd_putsf(flash char *str)

این تابع برای نمایش یک رشته از نوع Ascii که در حافظهٔ flash قرار دارد روی LCD استفاده می شود. مانند:

```
lcd_putsf ( "Atmega16" );
```

- lcd_clear()

این تابع برای پاک کردن مقادیر نمایش داده شده روی LCD استفاده می شود.

- lcd_gotoxy(x , y)

این تابع برای انتقال کرسر (مکان نما) به موقعیت خاص روی LCD استفاده می شود. متغیر y خط و متغیر x موقعیت روی خط را مشخص می کند. مثال:

```
lcd_gotoxy ( 0 , 1 );
lcd_putchar('A');
```

در این مثال کاراکتر A در ابتدای (x=0) خط دوم (y=1) چاپ می شود.

برای تبدیل مقادیر عددی به کدهای Ascii می توان از تابع sprint() استفاده کرد. این تابع از کتابخانهٔ stdio.h است. قالب کلی این تابع به شکل زیر است :

```
int  sprintf(char *str, char flash *fmtstr [ , arg1, arg2, ...]) ;
```

مثال ۱: نمایش اعداد صحیح

```
int  temp=34;
char *str ;
sprint( str , "temp=  %d  " , temp );
lcd_puts ( str );
```

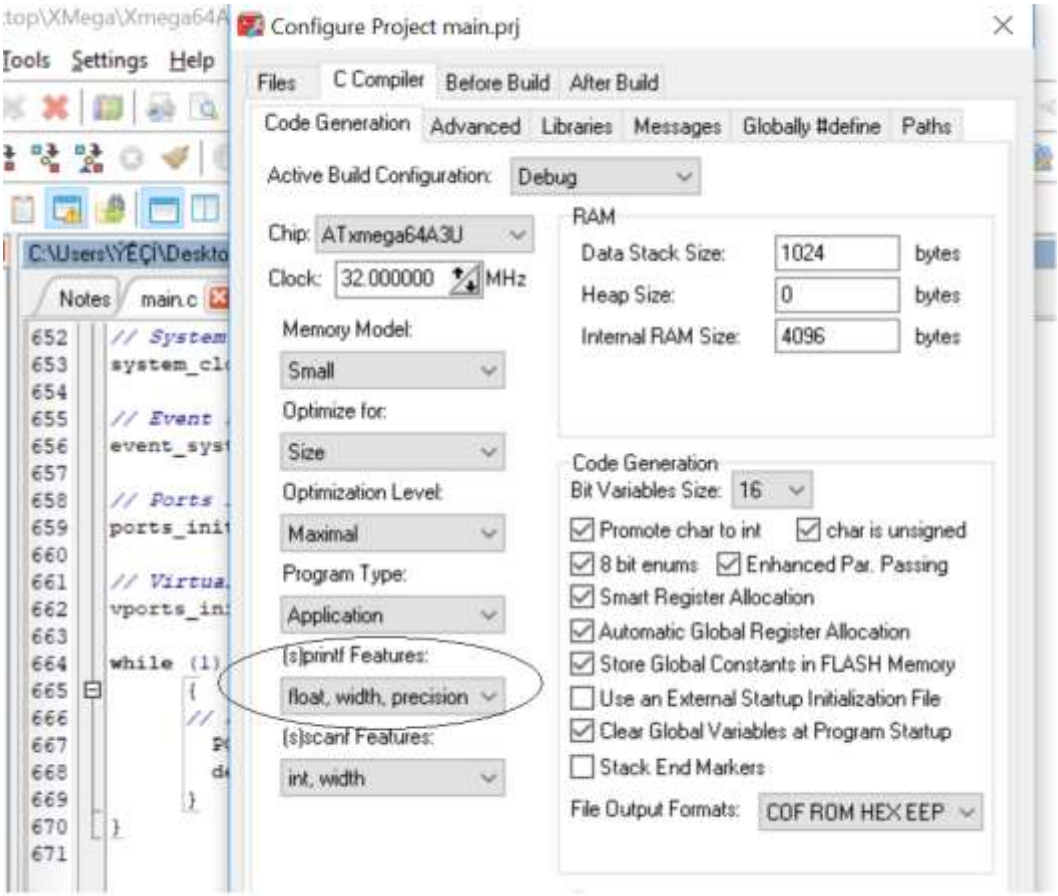
مثال ۲: نمایش عدد اعشاری

```
float  temp=34.5;
char *str ;
sprint( str , "temp=  %4.1f  " , temp );
lcd_puts ( str );
```

عبارت %4.1f در تابع sprint به این معنی است که می خواهیم یک عدد اعشاری با دقت یک دهم اعشار و دو عدد صحیح که با نقطهٔ اعشار در مجموع دارای چهار کاراکتر است را چاپ کنیم.

برای اینکه تابع sprint برای اعداد اعشاری به درستی عمل کند باید مانند شکل زیر این تابع برای اعداد اعشاری تنظیم گردد.

Project \ Configure \ C compiler \ Generation


Configure Project main.prj
Files
C Compiler
Before Build
After Build
Code Generation
Advanced
Libraries
Messages
Globally #define
Paths
Active Build Configuration:
Debug
Chip:
ATxmega64A3U
Clock:
32.000000
MHz
Memory Model:
Small
Optimize for:
Size
Optimization Level:
Maximal
Program Type:
Application
(s)printf Features:
float, width, precision
(s)scanf Features:
int, width
RAM
Data Stack Size:
1024
bytes
Heap Size:
0
bytes
Internal RAM Size:
4096
bytes
Code Generation
Bit Variables Size:
16
Promote char to int
char is unsigned
8 bit enums
Enhanced Par. Passing
Smart Register Allocation
Automatic Global Register Allocation
Store Global Constants in FLASH Memory
Use an External Startup Initialization File
Clear Global Variables at Program Startup
Stack End Markers
File Output Formats:
COF ROM HEX EEP

آزمایش پنجم

ارتباط سریال توسط بخش USART

پیش نیاز: با مطالعهٔ ضمیمهٔ آزمایش به سوالات زیر پاسخ دهید.

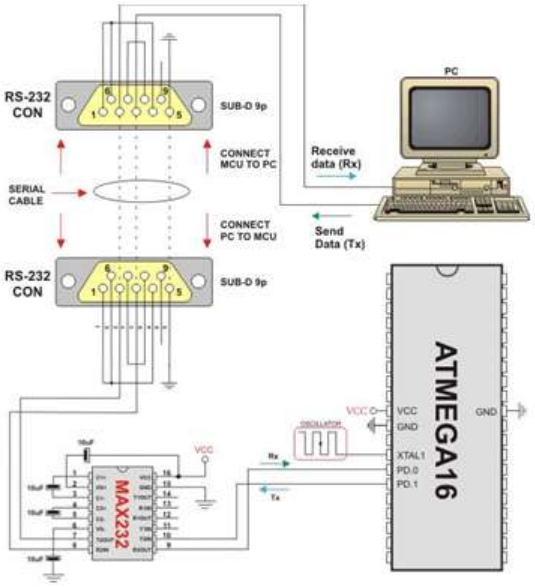
۱- در مود ارتباطی ناهمگام قالب اطلاعات به چه صورت است؟

۲- تفاوت پرچمهای TXC و UDRE در ثبات UCSRA چیست؟

آزمایش ۱ : ارسال و دریافت همزمان اطلاعات از طریق پورت سریال

برنامه‌ای بنویسید که اطلاعات دریافتی توسط بخش UART میکروکنترلر را روی LCD نمایش دهد و همچنین کد کلید فشارداده شده از صفحه کلید ماتریسی بورد آموزشی توسط بخش UART ارسال شود. برای سهولت کار خواندن اطلاعات از طریق وقفهٔ دریافت بخش UART استفاده شود. (Baud Rate = 9600)

توجه : در این آزمایش مطابق شکل زیر با استفاده از کابل موجود در آزمایشگاه ، سوکت ۹ پین بورد آموزشی را به پورت COM کامپیوتر متصل کنید سپس با استفاده از نرم‌افزار Hyper Terminal ویندوز و یا Terminal نرم‌افزار کدویژن اطلاعات را به میکروکنترلر ارسال و یا اطلاعات ارسالی از میکروکنترلر را روی نمایشگر کامپیوتر مشاهده نمایید.

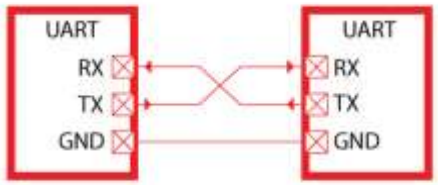


آزمایش ۲ : کنترل عملیات میکروکنترلر توسط کامپیوتر

برنامه‌ای بنویسید که ابتدا عددی حداکثر چهار رقمی از کامپیوتر دریافت کرده و آنرا توسط سون سگمنت نمایش دهد. سپس با زدن کلید U بصورت صعودی و یا با زدن کلید D بصورت نزولی شمارش نماید (فاصله بین هر شمارش ۵۰۰ میلی ثانیه باشد). در هر زمان از اجرای برنامه با زدن کلیدهای مذکور در صورت لزوم جهت شمارش عوض شود.

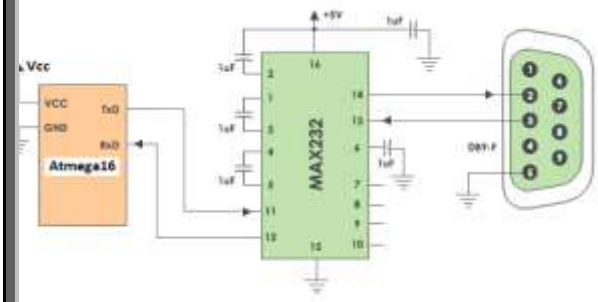
ضمیمه : راهنمای تنظیم بخش USART

بخش USART دارای دو مود ارتباطی همگام(سنکرون) و ناهمگام(آسنکرون) است. در این آزمایش از حالت آسنکرون برای ارتباط بین میکروکنترلر و کامپیوتر استفاده می کنیم. بنابراین بصورت مختصر به شرح این نوع ارتباط از نظر سخت افزاری و نرم افزاری می پردازیم.



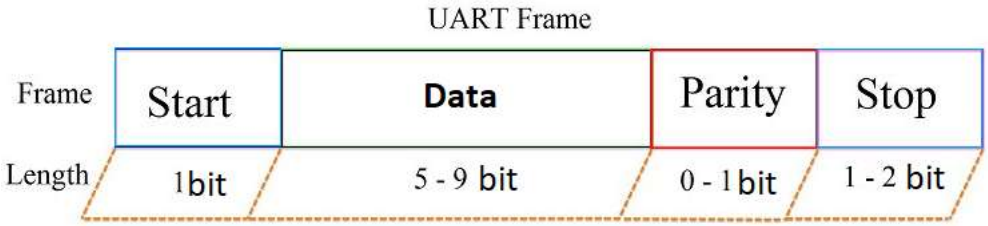
بخش UART برای ارسال و دریافت اطلاعات بصورت سریال دارای یک پایه به نام Tx برای ارسال اطلاعات و یک پایه به نام Rx برای دریافت اطلاعات است. برای اتصال دو میکروکنترلر این پایه ها بصورت ضربدری به یکدیگر متصل می شوند.

برای اتصال میکروکنترلر به کامپیوتر چون سطوح ولتاژ این دو وسیله با هم متفاوت است نمی توان ایندو را مستقیماً بهم متصل نمود. پایه های میکروکنترلر دارای سطح ولتاژ TTL هستند در حالیکه پورت سریال کامپیوتر دارای سطح ولتاژ استاندارد RS232 است. بنابراین برای اتصال این دو باید از یک مدار واسط برای تبدیل ولتاژ استفاده کرد. در شکل زیر این مدار نشان داده شده است. تراشهٔ



MAX232 عمل تبدیل این دو سطح ولتاژ را انجام میدهد.

همانطور که قبلاً گفته شد عمل انتقال اطلاعات بصورت بیت به بیت (سریال) انجام می شود. قالب اطلاعات ارسالی معمولاً بصورت زیر است.



در بیشتر مواقع قالب اطلاعات دارای هشت بیت دیتا و یک بیت پایان (Stop bit) و بدون بیت توازن است.

ثباتهای بخش USART و چگونگی راه اندازی آن در این ادامه توضیح داده می شود:

- ثبات دریافت و ارسال : UDR

بخش USART دارای ثبات RXB برای دریافت و TXB برای ارسال اطلاعات است. البته این دو ثبات از نظر کاربر دارای یک آدرس و به نام UDR شناخته می شود. پس در دو حالت نوشتن و یا خواندن کاربر از نام UDR استفاده می کند.

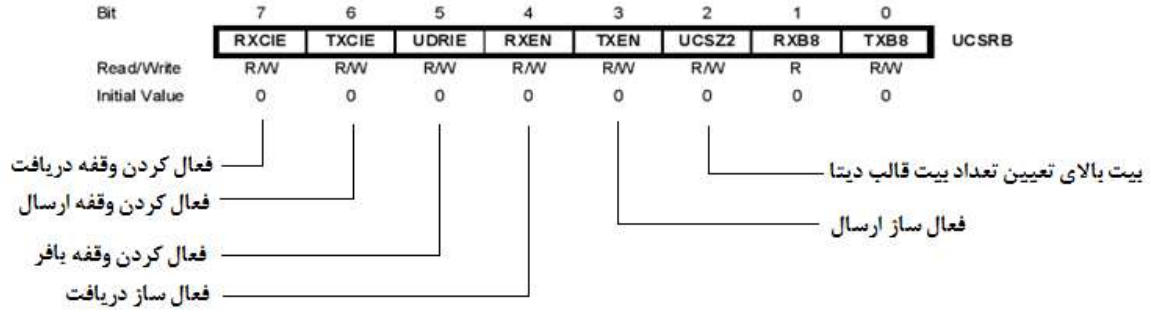
Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- ثبات پرچمهای USART : UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

با قرار گرفتن اطلاعات دریافتی در ثبات UDR پرچم RXC در ثبات UCSRA یک می شود تا نشان دهد که اطلاعات بصورت کامل دریافت شده است. همچنین با اتمام ارسال اطلاعات قرار داده شده در ثبات UDR توسط کاربر بیت TXC در حالت یک قرار میگیرد. با ارسال کامل اطلاعات و خالی شدن ثبات UDR بیت UDRE نیز یک می شود.

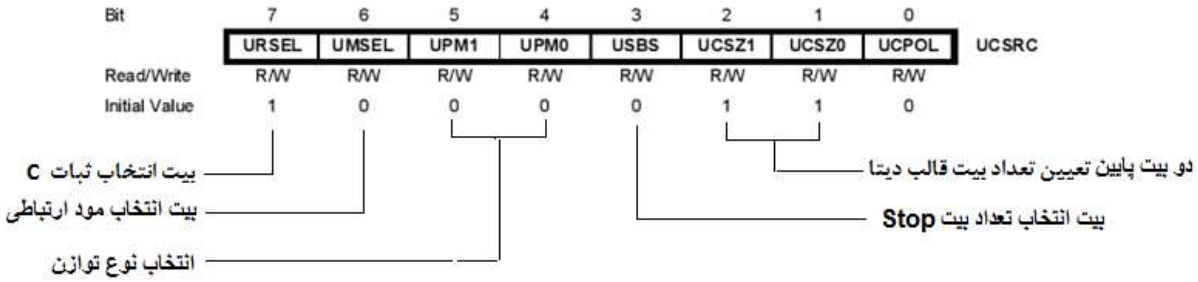
- ثبات کنترلی UCSRB



بیتهای RXEN و TXEN برای فعال کردن عمل دریافت و ارسال اطلاعات مورد استفاده قرار می گیرد. برای این منظور باید هر یک از این دو بیت در حالت یک قرار گیرد. برای فعال کردن وقفه‌های دریافت ، ارسال و خالی بودن بافر بترتیب باید بیتهای RXCIE ، TXCIE و UDRIE باید یک شوند. بطور مثال برای آزمایش اول هم دریافت و هم ارسال باید فعال شود همچنین می خواهیم دریافت اطلاعات از طریق وقفهٔ دریافت انجام گیرد . بنابراین در ابتدای برنامه دستور زیر قرار می گیرد:

```
UCSRB = 0x98 ;  //0b10011000
```

- ثبات کنترلی UCSRC



برای نوشتن یک مقدار خاص در این ثبات بیت آخر این مقدار باید '1' باشد. با توجه به جداول زیر و توضیحات شکل ثبات مقدار 0x86 را در این ثبات می نویسیم:

```
UCSRC = 0x86 ; // 0b10000110
```

با نوشتن این مقدار مود ارتباطی آسنکرون ، حالت بدون توازن ، تعداد یک Stop بیت و قالب دیتای هشت بیتی انتخاب میشود.

UMSEL Bit Settings

UMSEL	Mode	
0	Asynchronous Operation	مود نا همگام
1	Synchronous Operation	مود همگام

UPM Bits Settings — بیت های تنظیم توازن

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

USBS Bit Settings — بیت تنظیم تعداد Stop بیت

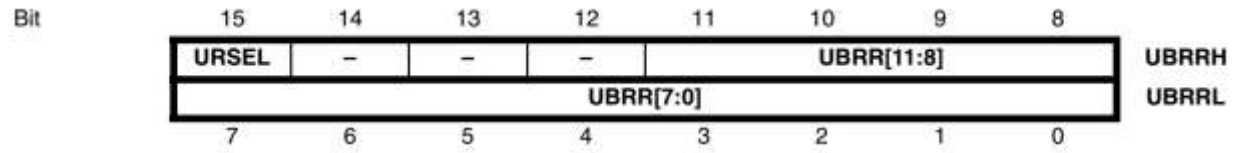
USBS	Stop Bit(s)
0	1-bit
1	2-bit

UCSZ Bits Settings — بیتهای تعیین اندازهٔ قالب اطلاعات

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- تنظیم نرخ انتقال اطلاعات (Baud Rate) در ثبات UBRR

سرعت انتقال دیتا با واحد بیت در ثانیه توسط ثبات شانزده بیتی UBRR مشخص می شود. مقدار این ثبات با استفاده از روابطی که در جدول زیر آمده و یا از جداولی که مقدار محاسبه شده در آن قرار گرفته مشخص می شود. مثلاً اگر بخواهیم با نرخ انتقال ۹۶۰۰ کار کنیم مطابق جدول صفحه بعد و با توجه به فرکانس کاری میکروکنترلر (fcpu=8Mhz) مقدار ۵۱ دهدهی باید داخل این ثبات قرار گیرد : UBRR = 51; // 0b0000000000110011



Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$

Table 70. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

- سرویس وقفهٔ دریافت و ارسال

سرویس وقفه ، زیر برنامه ای است که در صورت تحریک وقفهٔ متناظر ، پردازنده برنامهٔ جاری خود را رها و آنرا اجرا می کند. هر سرویس وقفه نام مختص به خود را دارد. در کامپایلر Codevision نامهای سرویس وقفهٔ ارسال و دریافت به صورت زیر تعریف شده است:

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void  usart_rx_isr(void)
{
}
// USART Transmitter interrupt service routine
interrupt [USART_TXC] void  usart_tx_isr(void)
{
}
```

برای فعال کردن وقفه علاوه بر بیت اختصاصی هر وقفه ، بیت وقفهٔ عمومی را نیز باید فعال کرد. در این کامپایلر با دستور زیر این کار انجام می شود:

```
// Global enable interrupts
#asm("sei")
```

با اضافه نمودن فایل کتابخانه ای Stdio.h به ابتدای برنامه می توانید از توابع putchar ، getchar بترتیب برای ارسال و دریافت یک کاراکتر استفاده کنید. همچنین برای ارسال یک جمله می توانید از تابع printf و یا puts بهره بگیرید.

مثال: برنامهٔ زیر یک جملهٔ مشخص را از میکروکنترلر ارسال می کند و کارکترهای دریافتی توسط میکروکنترلر را روی LCD چاپ می کند.

```
#include <io.h>
#include <alcd.h>
#include <delay.h>
// Standard Input/Output functions
#include <stdio.h>

// USART Receiver interrupt service routine
```

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
   lcd_putchar(UDR);
}


void main(void)
{
        DDRA=0x00;
        DDRB=0x00;
        DDRC=0x00;
        DDRD=0x00;
        // USART initialization
        // Communication Parameters: 8 Data, 1 Stop, No Parity
        // USART Receiver: On
        // USART Transmitter: On
        // USART Mode: Asynchronous
        // USART Baud Rate: 9600
        UCSRA=0x00;
        UCSRB=0x98;
        UCSRC=0x86;
        UBRRH=0x00;
        UBRRL=0x33;
        lcd_init(16);
        // Global enable interrupts
        #asm("sei")
        while (1)
            {
                printf("Atmega16\n\r");
                delay_ms(1000);
            }
}
```

آزمایش ششم

مبدل آنالوگ به دیجیتال

پیش گزارش: با مطالعهٔ ضمیمهٔ آزمایش به سوالات زیر پاسخ دهید.

۱- مبدل آنالوگ به دیجیتال در میکروکنترلر Atmega16 چند بیتی و دارای چند کانال ورودیست؟

۲- میکروکنترلر AVR دارای چند مرجع ولتاژ تبیل است؟ و چگونه انتخاب می شود؟

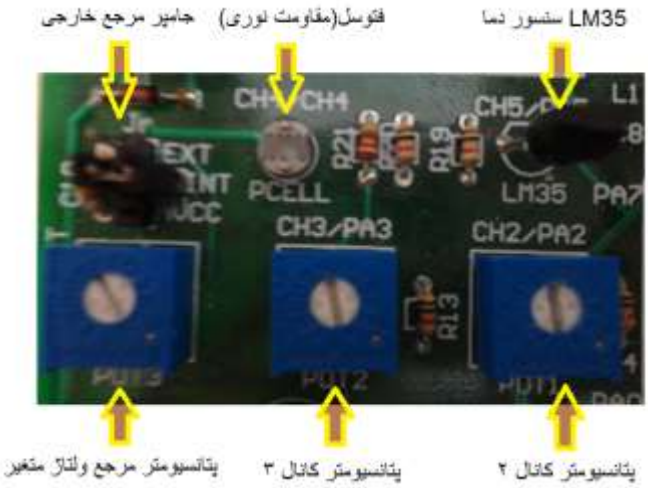
۳- رابطهٔ بین مقدار آنالوگ ولتاژ ورودی و مقدار دیجیتال نتیجهٔ تبدیل چیست؟

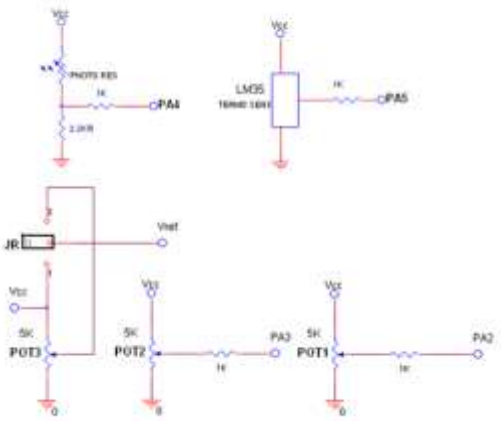
آزمایش ۱ : ساخت ولتمتر با نمایشگر LCD

برنامه‌ای بنویسید که مقدار ولتاژ پتانسیومتر (POT1) متصل به کانال ۲ (PA.2) مبدل آنالوگ به دیجیتال را با دقت دو رقم اعشار روی خط اول نمایشگر LCD با قالب V= x.yy volt نمایش دهد. به نکات گفته شده در زیر توجه کنید.(برنامهٔ مثال ضمیمه را در فایل برنامهٔ خود کپی کنید و تغییرات لازم را درآن اِعمال کنید)

توجه:

در این آزمایش از حالت ۱۰ بیتی مبدل استفاده کنید و مرجع ولتاژ را AVCC انتخاب کنید. همچنین فرکانس ورودی مبدل را روی ۱۲۵ کیلوهرتز تنظیم نمایید.





آزمایش ۲ : نمایش خروجی سنسورهای دما ، نور و پتانسیومتر روی نمایشگرسون سگمنت

می‌خواهیم مقادیر چهار کانال ۲ ، ۳ ، ۴ و ۵ را بصورت انتخابی روی نمایشگر سون سگمنت نشان دهیم. کانالهای دو و سه (POT1,POT2) در واحد ولتاژ با دو رقم اعشار نمایش داده شود. کانال چهار (فتوسل) از مقدار عددی صفر تا صد و کانال پنج (سنسوردما LM35) مقدار دما بر حسب درجهٔ سانتی‌گراد با دقت یک رقم اعشار نمایش داده شود. انتخاب کانالها با فشار دادن کلید F از صفحه کلید ماتریسی انجام می‌شود. برای انجام این آزمایش به نکات زیر توجه شود.

توجه :

۱. تابع نمایش سون سگمنت را برای نمایش نقطهٔ اعشار شناور بصورت زیر بازنویسی نمایید:

void SevenSegDisplay (*char* digit[4] , *char* position)

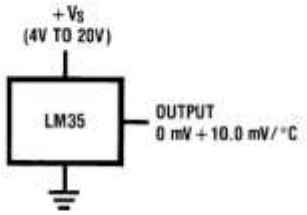
متغیر position تعداد ارقام اعشار روی سون سگمنت را مشخص می‌کند.

۲. ولتاژ ورودی هر چهار کانال را بر حسب میلی ولت محاسبه کنید(مثل رابطه زیر) سپس تبدیلات لازم را انجام دهید. در محاسبات خود از تعریف متغیرهای اعشاری خودداری کنید.

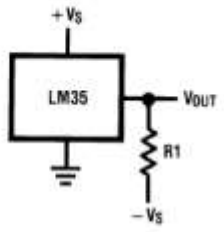
$$Volt = ADC \times \frac{5000.0}{1023} \qquad V_{ref} = 5v \qquad n = 10$$

۳. برای نمایش دما و برای داشتن بیشترین دقت از مرجع داخلی 2.56v و برای بقیهٔ کانالها از مرجع AVCC استفاده نمایید. فرکانس ورودی را مانند قبل ۱۲۵ کیلوهرتز انتخاب کنید.
۴. برای خواندن کلید فرض کنید فقط کلید F روی بورد وجود دارد بنابراین از تابع صفحه کلید ماتریسی استفاده نکنید.
۵. ولتاژ خروجی سنسور LM35 به ازاء هر یک درجه سانتی‌گراد تغییر دما ' به اندازهٔ ۱۰ میلی‌ولت تغییر می‌کند. بطور مثال اگر دما صفر درجه سانتی‌گراد باشد خروجی سنسور صفر ولت و اگر دما صد درجه باشد خروجی آن یک ولت (1000mv) خواهد بود. تغییرات ولتاژ نسبت به دما در این سنسور خطی است.

Typical Applications



Basic Centigrade Temperature Sensor
(+2°C to +150°C)



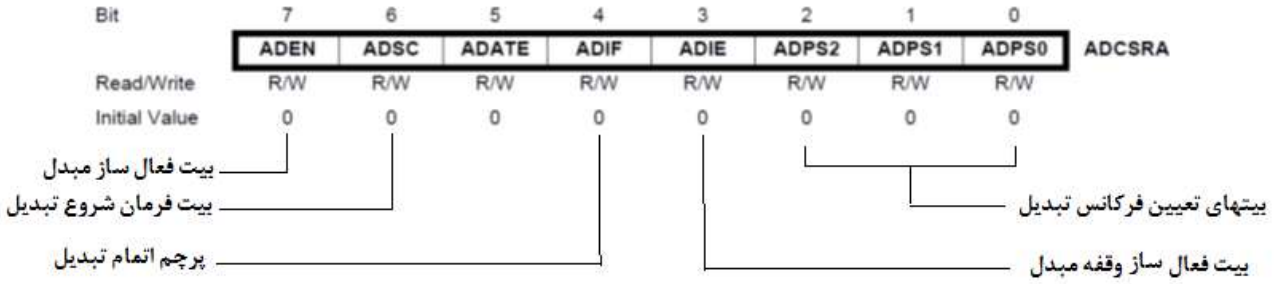
Choose $R_1 = -V_S/50\ \mu A$
V_{OUT}=+1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

Full-Range Centigrade Temperature Sensor

ضمیمهٔ آزمایش ششم : راه اندازی و ثباتهای مبدل آنالوگ به دیجیتال

- **ثبات پرچمها و کنترلی** ADCSRA

برای فعال کردن بخش مبدل آنالوگ به دیجیتال میکروکنترلر بیت ADEN در این ثبات باید در حالت '1' قرار گیرد. در این حالت پورت A از میکروکنترلر می تواند به عنوان ورودی آنالوگ عمل کند.



برای شروع تبدیل برای یک کانال مشخص باید بیت ADSC در حالت '1' قرار گیرد. بعد از شروع تبدیل مدت زمان مشخصی طول می کشد تا مقدار تبدیل محاسبه شده و در ثبات تبدیل قرار گیرد. اتمام تبدیل با یک شدن بیت ADIF به کاربر اعلان می گردد. کاربر باید بعد از خواندن ثبات مقدار تبدیل، این بیت را صفر کند(با نوشتن مقدار '1' در این بیت عمل صفر شدن انجام می شود). لازم به ذکر است در حالت معمول (ADAT=0) برای هر بار تبدیل باید بیت ADSC در حالت '1' قرار گیرد. سه بیت ADPS برای تعیین ضریب مقسم فرکانس ورودی مبدل مطابق جدول زیر بکار می رود. معمولا با بالا رفتن سرعت تبدیل ، دقت آن کاهش می یابد و بالعکس. در میکروکنترلهای AVR بهترین فرکانس برای تبدیل مقدار ۱۲۵ کیلوهرتز است بنابراین با توجه به فرکانس هشت مگاهرتز سیستم ، با انتخاب مقدار ضریب 1/64 (ADPS=110) برای مقسم ، فرکانس ۱۲۵ کیلوهرتز به مبدل اعمال خواهد شد.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

با توجه به این شرایط مقدار اولیه برای ثبات ADCSRA می تواند مطابق زیر باشد:

ADCSRA = 0xc6 ; // 0b11000110

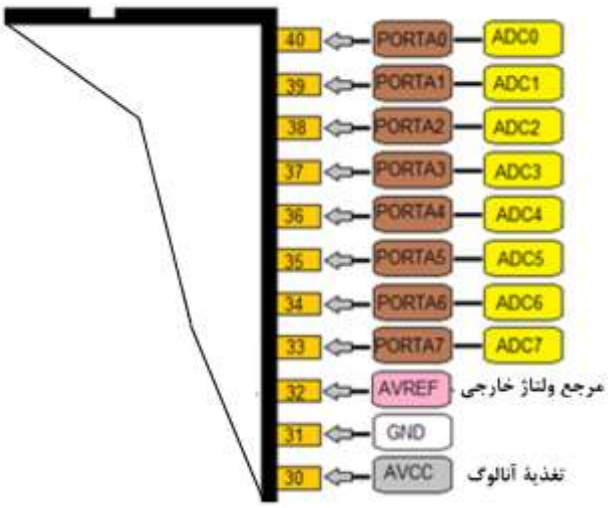
- ثبات ADMUX

هشت کانال ورودی آنالوگ در میکروکنترلر Atmega16 روی پورت A قرار دارد. برای انتخاب هر یک از این هشت کانال میتوان از سه بیت اول ثبات ADMUX مطابق جدول زیر استفاده کرد. کانال صفر (ADC0) متصل به PA0 و به همین ترتیب کانال هفت (ADC7) متصل به PA7 است.

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

انتخاب مرجع ولتاژ — انتخاب کانال ورودی آنالوگ

MUX4..0	Single Ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7



دو بیت آخر این ثبات (REFSx) مربوط به انتخاب مرجع ولتاژ تبدیل است که مطابق جدول زیر انتخاب می شود. میکروکنترلر AVR دارای سه مرجع ولتاژ است که عبارت است از مرجع ولتاژ AVCC که همان ولتاژ تغذیهٔ آنالوگ میکروکنترلر است که معمولا ۵ ولت خواهد بود ، مرجع ولتاژ داخلی Vref=2.56v و همچنین ولتاژی که از طریق پایهٔ اختصاصی AVREF به میکروکنترلر اعمال می شود.

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

مرجع ولتاژ حداکثر دامنهٔ ولتاژ ورودی را مشخص می کند مثلا اگر مرجع 2.56 را انتخاب کنیم حداکثر مجاز ورودی از مقدار 2.56 نباید بیشتر شود و همچنین در این حالت بازهٔ آنالوگ 0 تا 2.56 ولت به حوزهٔ عددی 0 تا 1023 تبدیل خواهد شد. رابطهٔ بین مقدار ولتاژ ورودی آنالوگ و مقدار دیجیتال حاصل تبدیل بصورت زیر است.

$$Volt = ADCW \times \frac{V_{ref}}{2^n - 1} \qquad n = 8 \; or \; 10$$

در این رابطه ADCW مقدار موجود در ثبات تبدیل و Vref مرجع ولتاژ انتخاب شده است.

بیت ADLAR در ثبات ADMUX برای انتخاب نوع چینش مقدار ده بیتی حاصل تبدیل در ثبات شانزده بیتی مبدل است. در شکل زیر چگونگی این کار آمده است.

- **Bit 5 – ADLAR: ADC Left Adiust Result**

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

حالت ADLAR=1 معمولا وقتی انتخاب می شود که بخواهیم از حالت هشت بیتی استفاده کنیم. در این صورت فقط مقدار ثبات ADCH خوانده می شود و در محاسبات منظور می گردد.

با توجه به توضیحات قبل مقدار اولیهٔ ثبات ADMUX برای انتخاب کانال دو و ولتاژ مرجع AVCC و حالت راست چین ۱۰ بیتی بصورت زیر خواهد بود:

```
ADMUX = 0x42 ; // 0b0100 0010
```

مثال: در برنامهٔ زیر مقدار ثبات مبدل خوانده شده و بصورت عددی روی LCD چاپ می شود.

```
#include <io.h>
#include <alcd.h>
#include <delay.h>
#include <stdio.h>


void  main(void)
{
lcd_init(16);
ADMUX=0x42;      // Voltage Reference: AVCC pin ,Channe: 2
ADCSRA= 0xc6;    // frequency: 125.000 kHz , ADC Started
while (1)
   {
      // Start the AD conversion
      ADCSRA|=(1<<ADSC);
      // Wait for the AD conversion to complete
      while ((ADCSRA & (1<<ADIF))==0);
      ADCSRA|=(1<<ADIF); // Clear flag
      sprint( str ," adc=%d  ", ADCW);
      lcd_gotoxy(0,0);
      lcd_puts(str);
      delay_ms(1000);
   }
}
```

- ثبات SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:5 – ADTS2:0: ADC Auto Trigger Source**

ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

آزمایش هفتم

تایمر– کانترها

پیش نیاز : با مطالعهٔ ضمیمهٔ آزمایش به پرسشهای زیر پاسخ دهید.

۱- مودهای مختلف تایمر یک را نام ببرید.

۲- مود CTC در تایمر یک چگونه کار می کند؟

۳- اگر بخواهیم با استفاده از مود CTC یک پالس مربعی با فرکانس ۱۰۰ هرتز تولید کنیم مقدار ثبات OCR1A چقدر باید باشد ؟ با فرض اینکه فرکانس تایمر یک مگاهرتز باشد.

آزمایش ۱ : تولید پالس مربعی با فرکانس متغیر با استفاده از تایمرِ یک

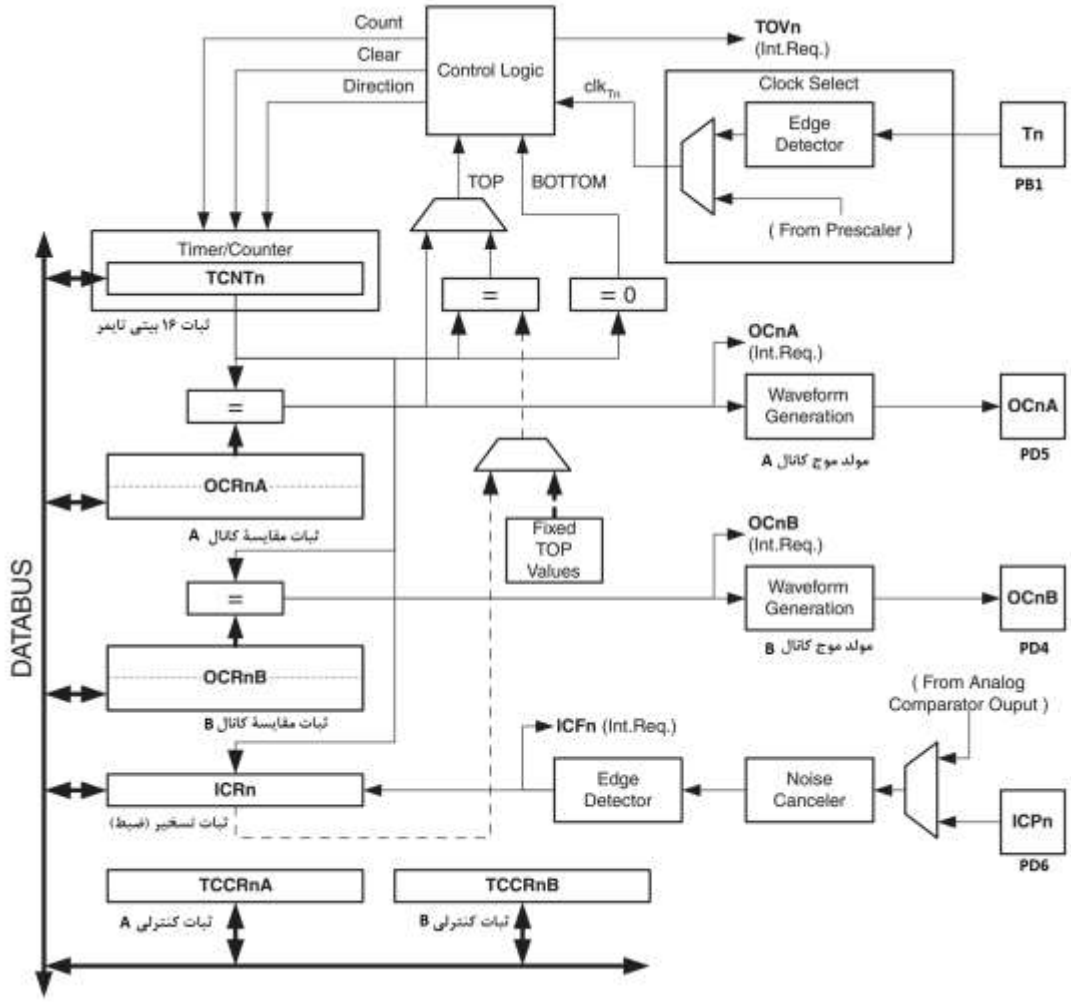
با استفاده از مود CTC تایمر یک برنامه‌ای بنویسید که یک پالس مربعی با فرکانس قابل تغییر روی پایهٔ PD.5 تولید کند. فرکانس از ۱۰۰ هرتز شروع و در هر چهار ثانیه مقدار آن صد هرتز و با عبور از ۱ کیلوهرتز مقدار آن هزار هرتز افزایش یابد و در نهایت روی فرکانس ۱۰ کیلوهرتز متوقف گردد. فرکانس خروجی را توسط حالت فرکانس‌متر(Hz) مالتی‌متر اندازه‌گیری کنید.

آزمایش ۲ : ساخت کرنومتر

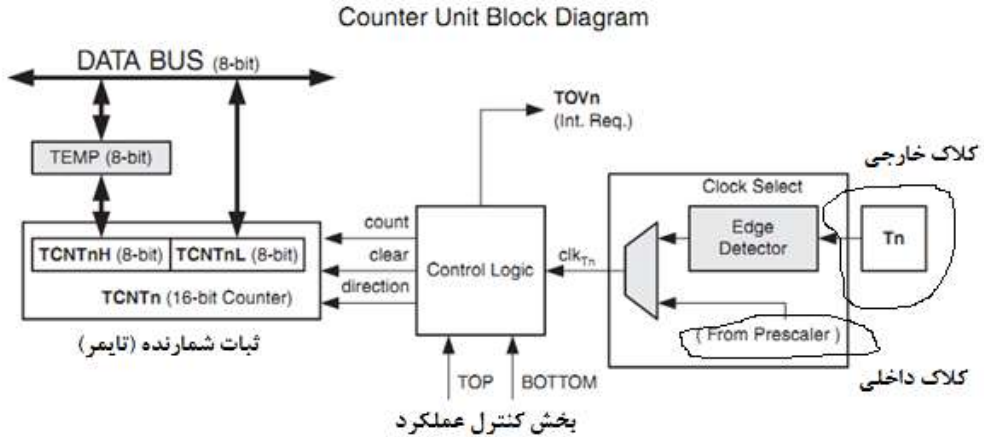
با استفاده از زیر برنامه‌های آزمایش مربوط به 7_seg و با استفاده از مود CTC و وقفهٔ مقایسه، یک کرنومتر با دقت صدم ثانیه بسازید. این کرنومتر با فشار دادن کلید 1 از صفحه کلید ماتریسی زمانگیری را شروع می‌کند و با فشار دادن همین کلید در بار دوم متوقف می شود . مقدار ماکزیمم شمارش 99.99 ثانیه خواهد بود .

ضمیمهٔ آزمایش هفتم : معرفی تایمر/کانتر یک

نمای کلی تایمر/کانتر یک در شکل زیر نشان داده شده است. در ادامه بخشهای مهم این تایمر بصورت مختصر توضیح داده می شود.



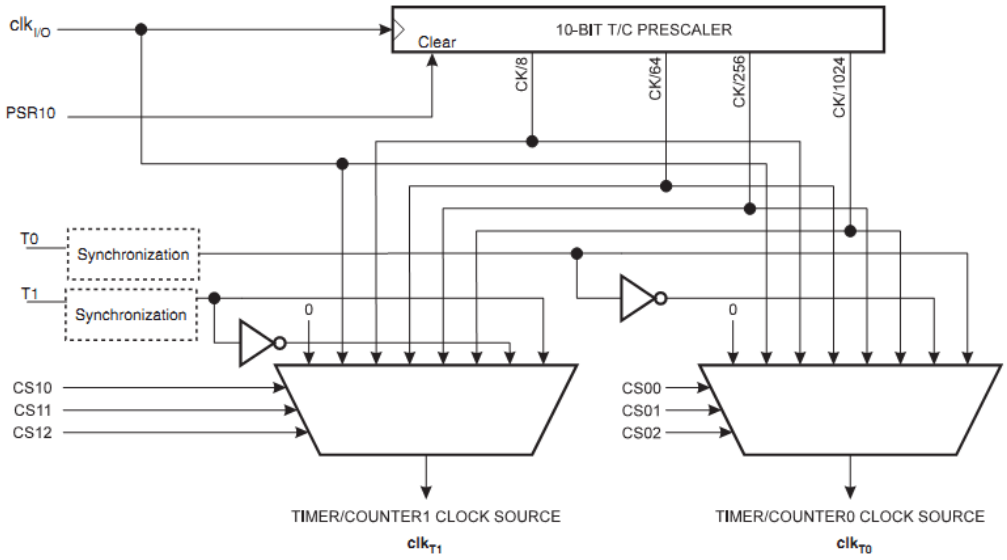
۱- ثبات شمارندهٔ تایمر/کانتر (TCNT1) : این ثبات یک شمارندهٔ شانزده بیتی است که از مقدار صفر تا ۶۵۵۳۵ را میتواند شمارش کند. این شمارنده کلاک ورودی خود را از پایهٔ خارجی (Tn) و یا خروجی مقسم کلاک تایمر دریافت میکند. حالت اول را اصطلاحاً مود کانتر و حالت دوم را مود تایمر می نامیم. در شکلهای زیر ثبات شمارنده و مقسم کلاک تایمر آمده است.



ثبات شمارنده با توجه به فرامین بخش کنترل (Control Logic) می تواند بصورت صعودی و یا نزولی شمارش نماید و یا مقدار تایمر را صفر کند.

شکل زیر بخش مقسم کلاک تایمر نشان داده شده است . این بخش ، کلاک خود را از کلاک سیستم دریافت کرده و چهار خروجی $\frac{1}{8}$ ، $\frac{1}{64}$ ، $\frac{1}{256}$ و $\frac{1}{1024}$ کلاک را تولید کرده و به ورودیهای مالتی پلکسر ۸ به ۱ اعمال می کند. با تغییر خطوط انتخاب مالتی پلکسر مطابق جدول می توان هشت حالت متفاوت برای کلاک تایمر انتخاب نمود.

Prescaler for Timer/Counter0 and Timer/Counter1



Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

در ردیف اول جدول، تایمر در حالت Stop قرار دارد و در دو انتخاب آخر ، کلاک از پایهٔ خارجی (مود کانتر) دریافت میگردد. و در بقیهٔ حالات کلاک تایمر از طریق مقسم و کلاک سیستم تامین می شود.

۲- ثباتهای TCCR1A و TCCR1B

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

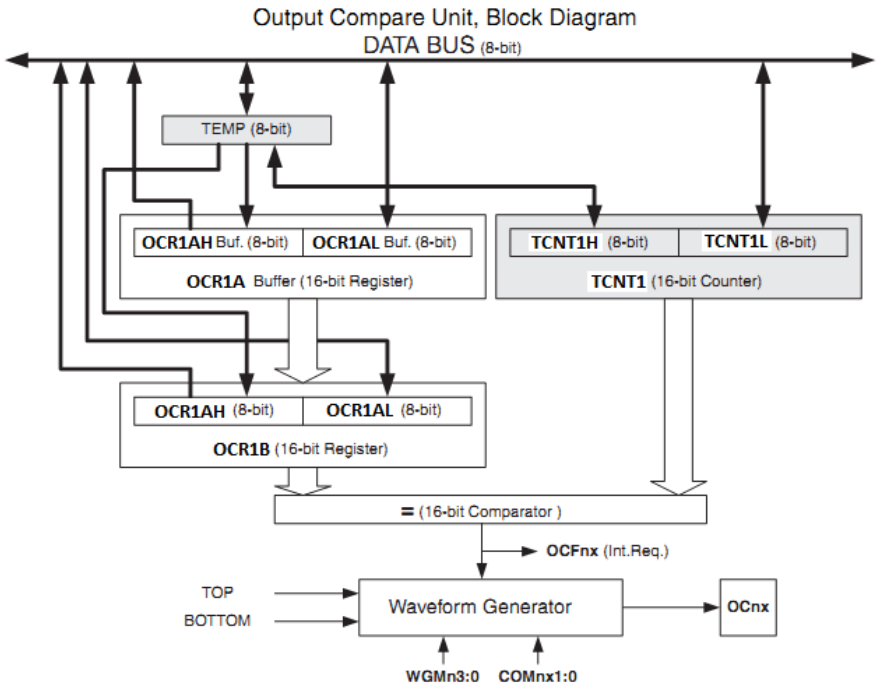
Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیتهای CS10 تا CS12 در ثبات TCCR1B همان خطوط انتخاب مالتی پلکسر بخش مقسم فرکانس است که در جدول بالا مقادیر انتخابی آن آمده است.

بیتهای WGM10 و WGM11 در ثبات TCCR1A و بیتهای WGM12 و WGM13 که در ثبات TCCR1B قرار دارند برای انتخاب نوع عملکرد تایمر مانند مودهای نرمال ، CTC ، PWM و ضبط (Capture) بکار می رود. در جدول زیر مقادیر متفاوت برای این چهار بیت و نوع عملکرد مربوطه آورده شده است.

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

شکل زیر بلوک دیاگرام بخش مقایسهٔ تایمر یک را نشان می دهد.



زوج بیتهای COM1Ax و COM1Bx در ثبات TCCR1A عملکرد پایه های OC1A (PD5) و OC1B (PD4) در هنگام تساوی مقدار تایمر با مقادیر ثباتهای مقایسهٔ OCR1A و OCR1B و در مودهای CTC و PWM را مشخص می کند. لازم بذکر است در همهٔ مودها در هنگام شمارش تایمر، مقدار آن با مقادیر ثباتهای مقایسهٔ دو کانال (OCR1A و OCR1B) مقایسه می شود و در هنگام تطابق این دو مقدار ، پرچمهای تساوی دو کانال (OCF1A و OCF1B) در ثبات TIFR یک میشود. با یک

شدن هر یک از این دو پرچم ، مدار مولد موج هر کانال با توجه به مقادیر بیتهای COM1Ax و COM1Bx واکنش متفاوت و مطابق جداول زیر را از خود نشان می دهد.

Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

Compare Output Mode, Non-PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level)
1	1	Set OC1A/OC1B on Compare Match (Set output to high level)

Compare Output Mode, Fast PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM, (inverting mode)

ثبات پرچمهای تایمر : TIFR

بیت TOV1 پرچم سرریز ، OCF1A و OCF1B پرچمهای مقایسه و ICF1 پرچم حالت ضبط تایمر یک هستند. برای صفر کردن هر یک از این پرچمها بصورت نرم افزاری باید مقدار '1' در آنها نوشته شود.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

۳- ثبات تنظیم وقفهٔ تایمرها : TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت TOIE1 برای فعال کردن وقفهٔ سرریز ، بیتهای OCIE1A و OCIE1B برای وقفهٔ مقایسه و بیت TICIE1 برای وقفهٔ ضبط بکار می روند.

۴- ثبات مقایسهٔ کانال A (OCR1A) و کانال B (OCR1B)

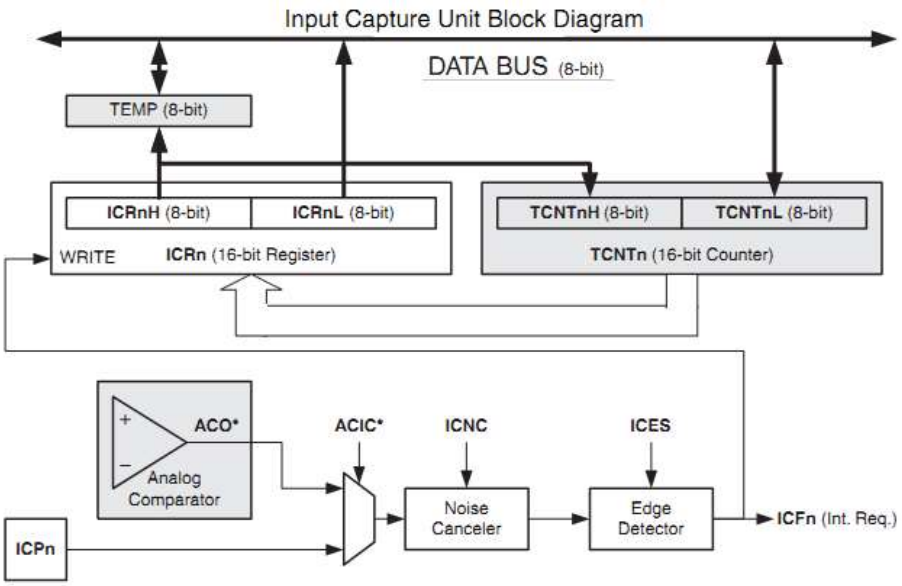
Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

لازم بذکر است ثباتهای تایمر و مقایسه در زبان C به بصورت شانزده بیتی با نامهای TCNT1 ، OCR1A و OCR1B قابل دسترسی هستند.

۵- ثبات ضبط ICR (Capture)

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
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	

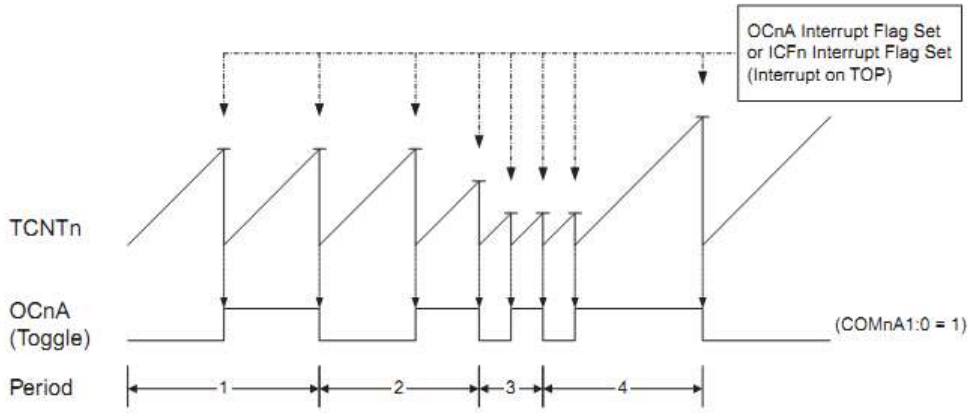
هنگامی که تایمر در حال شمارش است اگر لبهٔ پالس به پایهٔ خارجی ICPn (PD6) اِعمال شود مقدار فعلی تایمر در ثبات ICR ذخیره می گردد. با استفاده از حالت ضبط می توان پریود و یا پهنای پالس را اندازه گیری نمود.



در CodeVision این ثبات بصورت شانزده بیتی در دسترس نیست بلکه باید مقادیر بایت بالا ICRH و بایت پایین ICRL را بصورت جداگانه استفاده کرد.

مثال : تولید پالس با استفاده از مود CTC در تایمر یک

در مود CTC تایمر از مقدار صفر شروع به شمارش می کند و در هنگام رسیدن به مقدار ثبات مقایسهٔ OCR1A مقدار تایمر صفر می شود و این روند ادامه پیدا می کند. حال اگر بخواهیم در هر بار تساوی، خروجی کانال A (PD5) تغییر حالت دهد باید بیتهای OC1A0 و OC1A1 را مطابق جدول ابتدایی صفحهٔ ۲۸ مقدار دهی مناسب انجام دهیم. (انتخاب حالت Toggle) همچنین برای انتخاب حالت CTC مطابق جدول عملکرد تایمر در صفحهٔ ۲۶ (ردیف پنجم) مقدار بیتهای WGM10 تا WGM13 را مشخص می کنیم. برای انتخاب یک فرکانس خاص مثلاً 1KHz برای خروجی، باید ثبات مقایسهٔ OCR1A را مطابق رابطهٔ زیر مقدار دهی مناسب کنیم.



$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

N ضریب مقسم کلاک

```
 #include <io.h>
void main(void)
{
        DDRD= 0x20; //0010 0000 //PD5  output
        PORTD= 0x00;
        // Timer/Counter 1 initialization
        TCCR1A = 0x40 ;  // OC1A output: Toggle on compare match
        TCCR1B = 0x0A;  // Mode: CTC top=OCR1A  & Clock value: 1000.000 kHz
        TCNT1=0x0000;
        OCR1A=500;   // OCR1A = T/2
        while (1)
        {
        }

}
```

آزمایش هشتم

پیش نیاز : با مطالعهٔ ضمیمهٔ آزمایش هفتم و هشتم به سوالات زیر پاسخ دهید.

۱- در چه صورت تایمر یک و صفر در مود کانتر عمل می کنند؟ برای این دو تایمر ورودی کلاک در حالت کانتر از کدام یک از پایه های میکروکنترلر تأمین می شود؟

۲- برای تولید موج PWM دو شیب با استفاده از تایمر صفر برای فرکانس خروجی ۱۰ کیلوهرتز و فرکانس کلاک تایمر یک مگاهرتز مقدار ثبات کنترلی را مشخص کنید.

آزمایش ۱ : شمارش تعداد پالس با استفاده از حالت کانتر تایمرِ یک

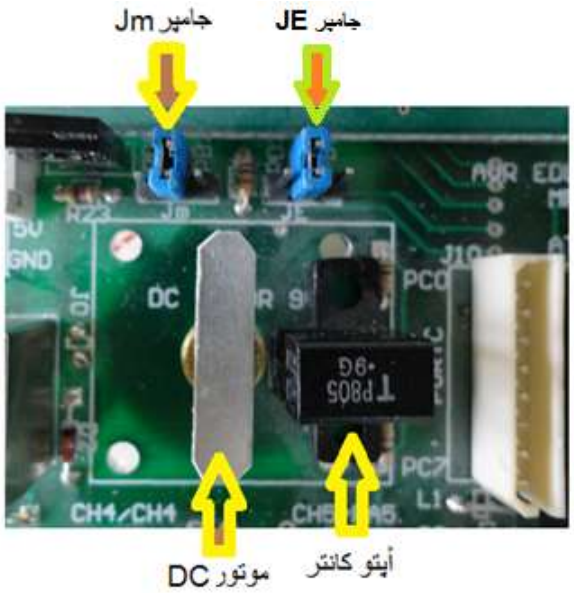
با استفاده از حالت کانترِ تایمرِ یک ، تعداد پالسهای تولید شده توسط اُپتوکانتر را شمارش کرده و توسط سون سگمنت در دو حالت زیر نمایش دهید:

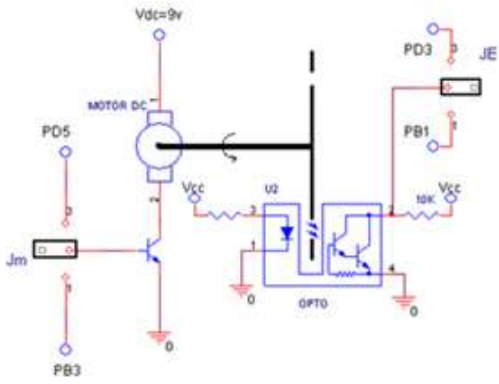
۱. با چرخاندن پروانهٔ موتور DC با دست و عبور لبه‌های آن از دهانهٔ اُپتوکانتر

خروجی اُپتوکانتر موجود روی بورد توسط جامپر JE می‌تواند به پایه‌های PD3 (ورودی INT1) و یا PB1 (ورودی کانتر یک) متصل شود. برای این آزمیش جامپر را در وضعیت PB1 قرار دهید.

۲. تولید پالس PWM توسط تایمر صفر با duty ۳۰ درصد و فرکانس 10Khz و اِعمال آن به ورودی موتور DC .

توجه داشته باشید که خروجی تایمر صفر روی پایهٔ PB3 قرار دارد بنابراین با تغییر وضعیت جامپر Jm بصورت مناسب موتور را راه‌اندازی کنید.





آزمایش۲ : اندازه گیری دور موتور بر حسب RPM

با استفاده از حالت کانتر تایمر یک دور موتور dc را بر حسب RPM (تعداد دور در یک دقیقه) محاسبه کرده و روی نمایشگر سون سگمنت نمایش دهید. برای تحریک موتور مانند آزمایش قبل از مود PWM تایمر صفر استفاده کنید.

توجه: برای اندازه گیری rpm تعداد دور موتور در یک ثانیه را اندازه گیری کرده سپس این مقدار را در ۶۰ ضرب کنید و نمایش دهید.

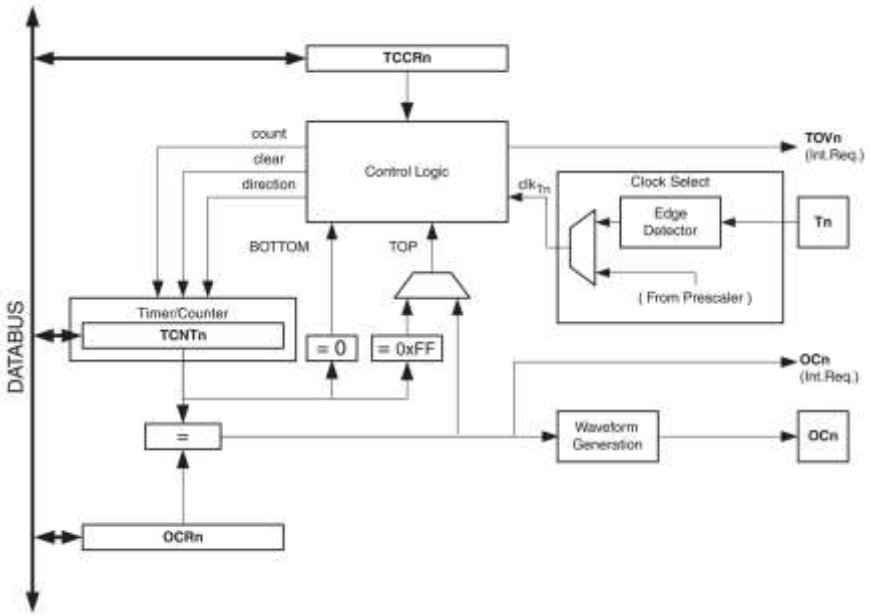
آزمایش۳ : تغییر دور موتور و اندازه گیری دور موتور

در این مرحله می خواهیم با تغییر یکی از پتانسیومترهای روی بورد (POT2 و یا POT3) دور موتور را تغییر دهیم . تغییر دور موتور با تغییر Duty cycle شکل موج خروجی تایمر صفر امکان پذیر است. مقدار Duty cycle نیز وابسته به مقدار ثبات

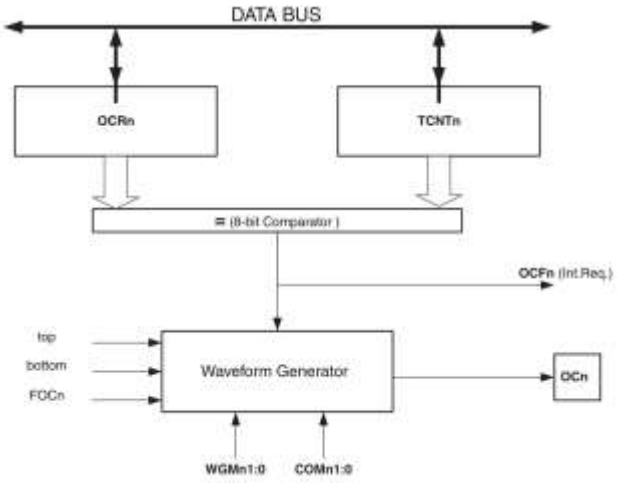
OCR0 تایمر صفر است. بنابراین با راه اندازی مبدل آنالوگ به دیجیتال و با توجه به شرایط زیر برنامهٔ مناسب را بنویسید.
توجه : مبدل ADC را در حالت هشت بیتی قرار دهید. مقدار ثبات نتیجهٔ مبدل باید به شکلی در ثبات OCR0 قرارگیرد که حداقل پهنای پالس ۱۰ درصد و حداکثر آن ۹۰ درصد باشد.

ضمیمهٔ آزمایش هشتم : معرفی تایمر/کانتر صفر

نمای کلی تایمر/کانتر صفر در شکل زیر نشان داده شده است. در ادامه بخشهای مهم این تایمر بصورت مختصر توضیح داده می شود.



- نمای بخش مقایسه و مولد موج



- ثباتهای تایمر صفر

۱- ثبات کنترلی TCCR0

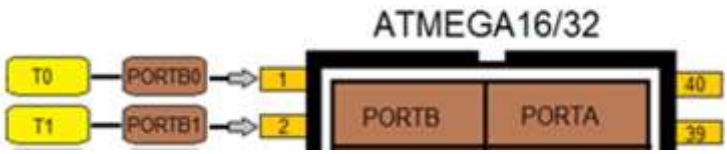
Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- CS02:CS01:CS00

این سه بیت برای انتخاب منبع کلاک تایمر صفر مطابق جدول زیر مورد استفاده قرار می گیرد.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

در دو حالت آخر جدول، کلاک تایمر از پایهٔ T0 که در میکروکنترلر Atmega16 روی پایهٔ PORTB0 قرار دارد تأمین میشود.



- WGM00:WGM01

این دو بیت برای انتخاب مود عملکرد تایمر صفر مطابق جدول زیر تنظیم می شود.

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

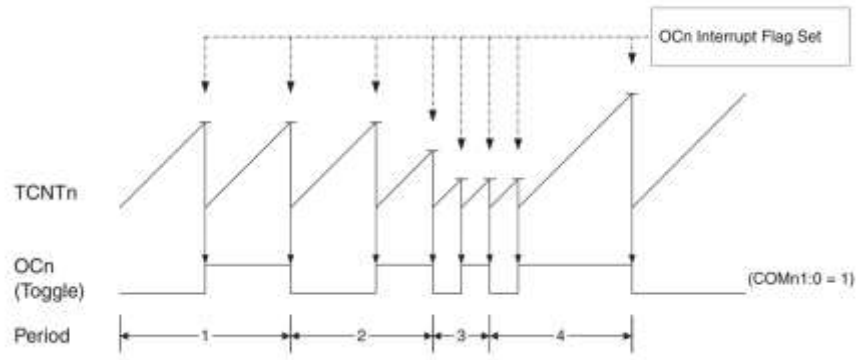
- مود نرمال

در این مود تایمر در هر پالس کلاک ساعت از صفر تا مقدار ماکزیمم خود (۲۵۵) شمارش می کند و در کلاک بعدی سرریز اتفاق می افتد و مقدار تایمر صفر می شود. در زمان سرریز پرچم TOV0 در ثبات TIFR یک می شود.

- مود CTC

در این حالت تایمر تا مقدار موجود در ثبات مقایسه OCR0 شمارش می کند و در کلاک بعدی مقدار تایمر صفر می شود و پرچم OCF0 در ثبات TIFR یک می شود.

Figure 31. CTC Mode, Timing Diagram



در مود CTC می توان بیتهای COM00 و COM01 در ثبات TCCR0 را مطابق جدول زیر تنظیم کرد تا حالت پایهٔ خروجی OC0 تغییر کند.این خروجی در میکروکنترلر Atmega16 روی پایهٔ PORTB3 قرار دارد.

Table 39. Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

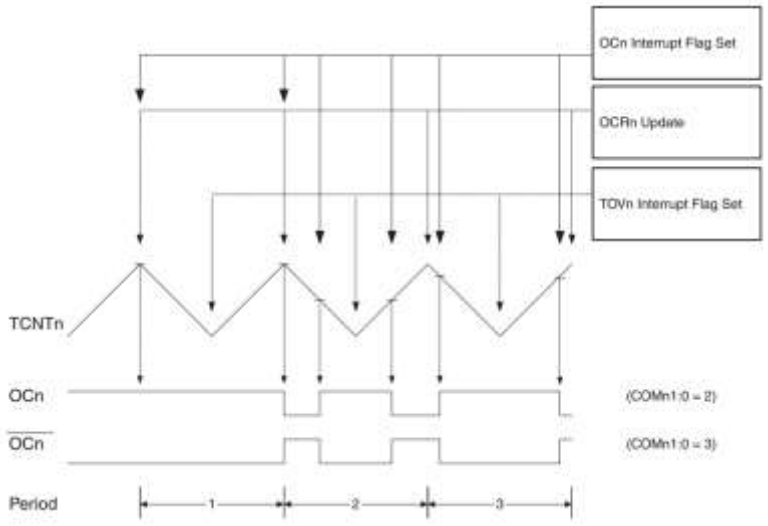
اگر مطابق جدول بالا حالت Toggle انتخاب شود فرکانس شکل موج خروجی مطابق رابطهٔ زیر خواهد بود:

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

در این رابطه N دارای مقادیر 1 ، 8 ، 64 ، 256 و یا 1024 می تواند باشد. همچنین OCRn مقدار قرار گرفته در ثبات OCR0 است.

- مود PWM دو شیب (PWM Phase corect)

این مود برای تولید شکل موج با Duty cycle متغیر استفاده می شود. در این حالت تایمر تا رسیدن به مقدار بیشینهٔ خود یعنی مقدار ۲۵۵ بصورت صعودی و پس از آن تا رسیدن به مقدار صفر بصورت نزولی شمارش می کند. همچنین در هر بار تساوی مقدار ثبات تایمر(TCNT0) با مقدار ثبات مقایسه (OCR0) خروجی پایهٔ OC0 تغیر حالت می دهد. این تغییر حالت وابسته به مقادیر تنظیم شده برای بیتهای COM00:COM01 مطابق جدول زیر خواهد بود.



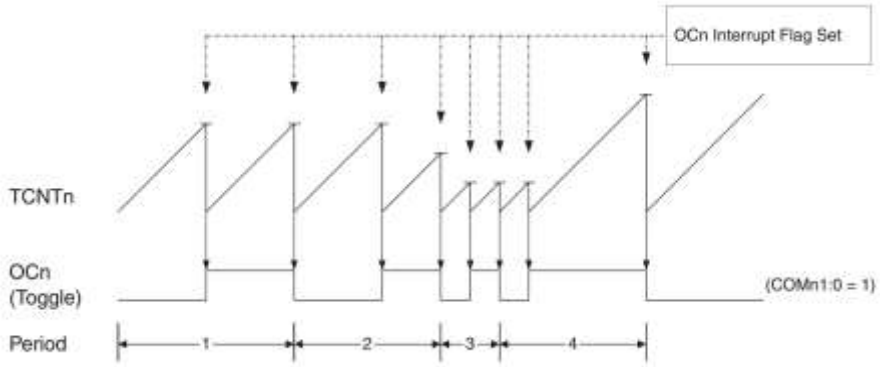
$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Table 41. Compare Output Mode, Phase Correct PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

- مود PWM سریع (Fast PWM)

در این مود تایمر یصورت متناوب از مقدار صفر بصورت صعودی شروع به شمارش می کند و با رسیدن به مقدار بیشینه صفر می شود. در این پروسه در زمان تساوی مقدار ثبات تایمر با مقدار ثبات مقایسه خروجی OC0 با توجه به وضعیت بیتهای COM00:COM01 مطابق جدول زیر تغییر حالت می دهد.



$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

Table 40. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP
1	1	Set OC0 on compare match, clear OC0 at TOP

۲- ثبات شمارندهٔ تایمر TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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- ثبات مقایسه OCR0

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

۴- ثبات پرچمهای تایمر TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

این ثبات برای همهٔ تایمرهای میکروکنترلر مشترک است. بیتهای TOV0 (بیت سرریز) و OCF0 (بیت مقایسه) برای تایمر صفر است.