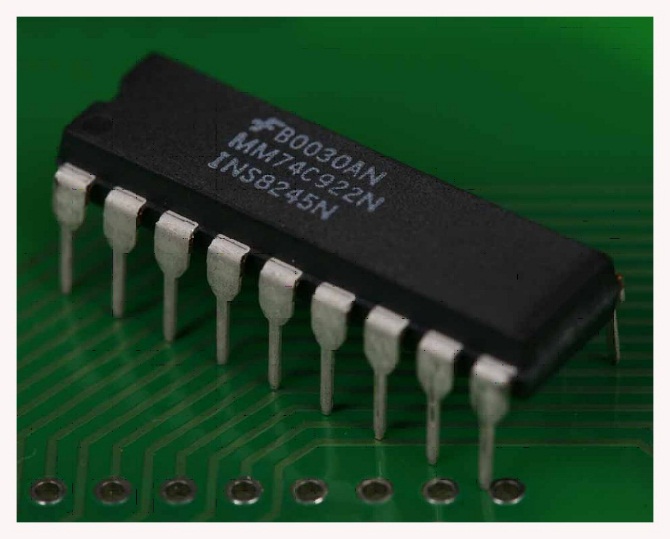
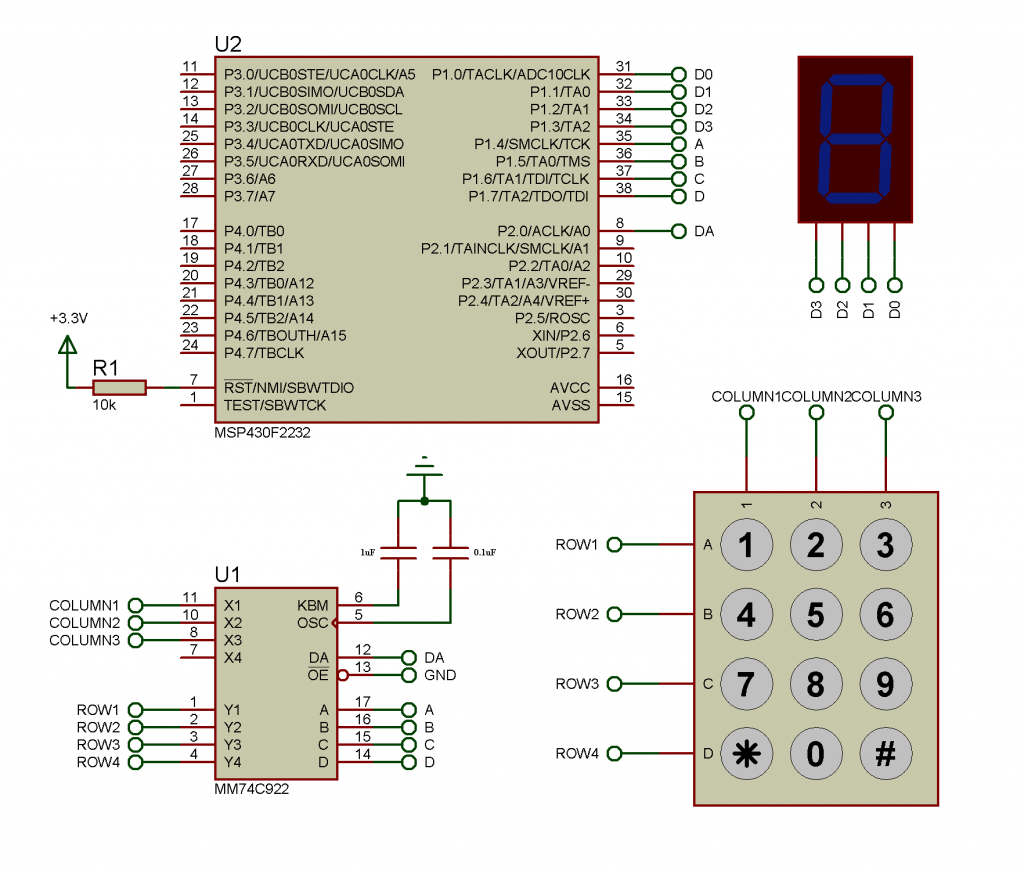
**MSP430 Uyg.24 – Keypad Uygulaması\_2 (MM74C922 ile)**

Ferudun GÖKCEGÖZ, 10 Ağustos 2011, Çarşamba

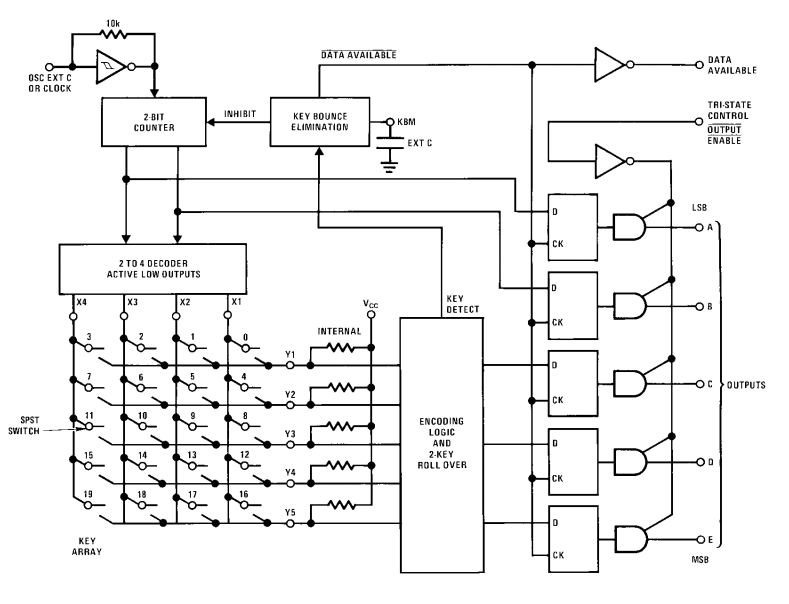


Merhaba arkadaşlar. Yeni bir dersimizde yine sizlerle birlikteyiz. Geçenki dersimizde söylediğimiz gibi bu dersimizde MM74C22 entegresi ile bir tuş takımı uygulaması yapacağız. Bu uygulamamızda geçenki gibi tarama yöntemi ile değil, 74C922 entegresi ile tuş takımından hangi tuşa basıldığını kesme ile algılayacağız. Kullanacağımız  entegrenin datasheet ini [**buradan**](http://fgokcegoz.files.wordpress.com/2011/08/mm74c922.pdf)indirebilirsiniz. Anlayacağınız üzere kullanacağımız entegre KeyPad in satır ve sütunlarına direkt olarak bağlanmaktadır. Herhangi bir tuşa basılma durumunda ise “Data Available” pinini lojik1 yapmaktadır. Yani tuşa basıldığı zaman aslında bize haber vermektedir. Bizde bu durumdan faydalanarak ve kesme kullanarak hangi tuşa basıldığını yazılımda ekstra bir kod gerektirmeden okuyarak, display üzerinde göstereceğiz.

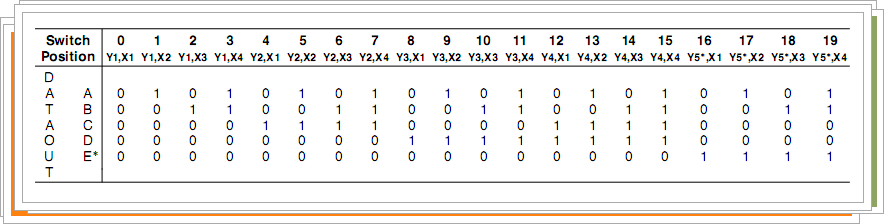
**İşte devre şemamız…**

****

Devre şemasında görüldüğü gibi tuş takımının satır ve sütunları, MM74C922 entegresinin direkt olarak X1/2/3 ve Y1/2/3/4 pinlerine bağlanmıştır. ABCD pinleri ise basılan tuş bilgisini aldığımız pinlerdir. OE(Output Enable) pini ise elde edilen tuş bilgisinin çıkışa aktarılması için lojik0 çekilmesi gereklidir. Biz direkt olarak GND a çekmişiz. Bu demek oluyorki, tuş takımından herhangi bir tuşa basılırsa tuş bilgisi ABCD pinlerine aktarılacaktır. Peki yazının başında bahsettiğimiz kesme kullanarak tuş bilgisini okuyacağımız pin hangisidir. Oda DA(Data Available) pinidir. Tuş a basıldığında tuş bilgisi entegre içerisinde okunduktan sonra DA pini lojik1 e çekilir. Bizde bu sayede sürekli yazılımı meşgul etmek yerine sadece DA pini lojik0 dan lojik1 çekildiğinde yani DA pininde yükselen kenarlı bir sinyal oluştuğunda mikrodenetleyici kesme üretecek ve tuş bilgisi güncellenecektir. Bu arada KBM ve OSC pinlerinden de bahsedecek olursak, OSC pinine entegrenin çalışması için gerekli bir adet kondansatör bağlanması gereklidir. KBM pini ise tuşa basıldığında oluşacak parazitleri engellemek amacıyla, bu pine bir kondansatör bağlamak suretiyle kullanılır. KBM pinine bağlanan kondansatörün değeri OSC pinine bağlanan kondansatörün 10 katı olmalıdır. Aşağıda entegrenin blok diyagramını inceleyebiliriz.

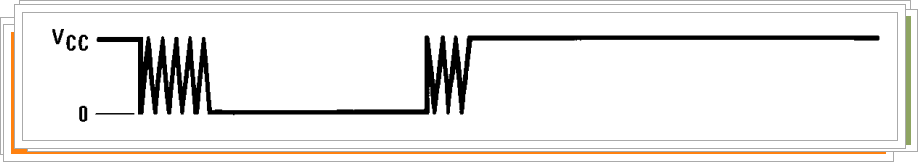


Aşağıda da tuş takımının pinlerine basılması durumunda ABCD pinlerindeki değişiklere ilişkin doğruluk tablosunu inceleyebiliriz.



Bu arada şundanda bahsedelim. Yukarıdaki doğruluk tablosunda 0 dan 19 a kadar toplam 20 adet tuş bilgisi bulunmaktadır. MM74C922 entegresinde 0-15 arası yani 4\*4 lük tuş takımı kullanılabilir. 15 den sonrasıni bizim entegremiz için geçerli değildir.

Daha önce tuşlara basılma durumunda mekanik anahtarlama yapıldığından dolayı bir miktar parazit oluşmaktadır. KBM pinine bağlanan kondansatör ile bu parazitler engellenmeye çalışılır. Aşağıda tuşa basıldığında anahtarlamadan dolayı oluşan parazitlere bir örnek grafik görmekteyiz…



Entegremiz ile alakalı bu kadar bilgi verdikten sonra uygulamaya ait yazılımıda verip üzerinde konuşmaya başlayalım.

|  |  |
| --- | --- |
| **[main.c](http://www.mcu-turkey.com/msp430-uyg-24/" \l "codesyntax_1" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-24/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-24/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#include "io430.h"

#include "in430.h"

const unsigned char Keys[] = { 1,2,3,0,

4,5,6,0,

7,8,9,0,

0,0,0,0};

void main(void)

{

WDTCTL = WDTPW + WDTHOLD;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

P1OUT = 0x00;

P1DIR = 0x0F;

P2OUT = 0x00;

P2DIR &= ~BIT0;

P2IE = BIT0;

P2IFG = 0x00;

\_BIS\_SR(GIE);

for(;;);

}

#pragma vector=PORT2\_VECTOR

\_\_interrupt void P2\_ISR(void)

{

if(P2IFG\_bit.P0)

P1OUT = Keys[P1IN >> 4];

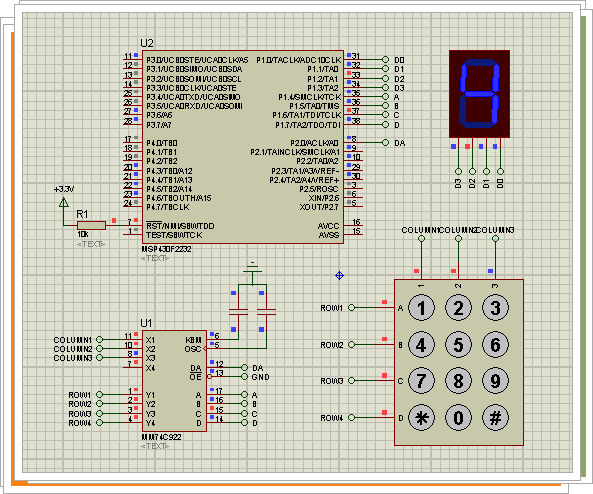
}

Yazılımı baştan itibaren incelemeye başlarsak, ilk olarak tuş bilgisini display üzerinde göstermek için kullanılmak üzere Keys isimli bir dizi tanımlanmıştır. Daha sonra main fonksiyonunda WDT ve Clock ayarları yapılmış ardından da port ayarlarına geçilmiştir. Port ayarlarında ise ilk olarak P1 portu temizlenmiş ve P1 portunun ilk dört biti giriş çıkış olarak, son dört biti ise giriş olarak şartlanmıştır. Daha sonra ise P2 portu ile ilgili olarak “Data Available” pininin bağlı olduğu P2.0 pini giriş olarak şartlanmış, bu pine ait kesme aktif edilmiş ve kesme bayrağıda temizlenmiştir. Daha sonra ise tüm kesmelere izin verilerek sonsuz dögüye girilmiştir.

Bu kısımdan sonra kesme alt programından bahsedecek olursak, P2.0 pinine yükselen kenar sinyal geldiğinde mikrodenetleyici ilgili kesme vektörüne dallanacaktır. Kesme vektöründe ise ilk olarak kesmenin P2.0 pininden mi geldiği kontrol edilmiştir. Eğer kesme P2.0 dan geliyorsa ABCD pinlerinden okunan değer alınıp, 4 bit sağa kaydırılıp daha sonrada Keys isimli dizideki karşılık gelen eleman displaye gönderilmiştir.

Bu şekilde bir mantıkla, mikrodenetleyici yazılımda sürekli tuş takımını okumak mecburiyetinde kalmamaktadır.  Zaten bir tuşa basılırsa, kesme vektörüne dallanılacak ve tuş bilgisi okunacaktır. Buda bize gerçekten büyük kolaylık sağlamıştır.

Aşağıda tuş takımı üzerinde 4 tuşuna basılması durumundaki ekran görüntüsü verilmiştir.

****

Bu konu hakkında benim söyleyeceklerim bu kadar. İnşallah faydalı olabilmişimdir. Bir sonraki dersimizde görüşmek üzere. Şimdilik Hoşçakalın…

***Ferudun GÖKCEGÖZ***

***fgokcegoz@yahoo.com***