

12. Bölüm

TUŞ TAKIMI (KEYPAD) UYGULAMALARI

- ↪ Tuş Takımı (Keypad) Hakkında Bilgi
- ↪ Tuş Takımı Uygulaması-1
- ↪ 74C922 Tuş Takımı Encoder Entegresi
- ↪ Tuş Takımı Uygulaması-2 (74C922 İle)

Bu bölümde tuş takımı diğer ifade ile keypad-klavye hakkında bilgiler verilmiştir. Bu bilgiler uygulamalar ile pratiğe dökülmüştür. Bölümde ilk başta program komutları ile bir tuş takımı tarama işlemleri ve uygulamaları incelenmiştir. Ayrıca 74C922 tuş takımı encoder entegresi tanıtılmış ve bu entegre ile bir uygulama gerçekleştirilmiştir.

12.1. TUŞ TAKIMI (KEYPAD) HAKKINDA BİLGİ

Kontrol sistemlerinde dış dünyadan insanlar tarafından veri girişleri genellikle tuş takımı (keypad-klavye) ile yapılır. Tuş takımı butonlarla gerçekleştirilebileceği gibi çeşitli hazır tuş takımları piyasada bulunmaktadır. Tuş takımı isimlendirmelerinde ilk sayı sütun, ikinci sayı ise satır sayısını belirtir. Örneğin 4x3'lük bir keypad, 4 sütun ve 3 satırdır.

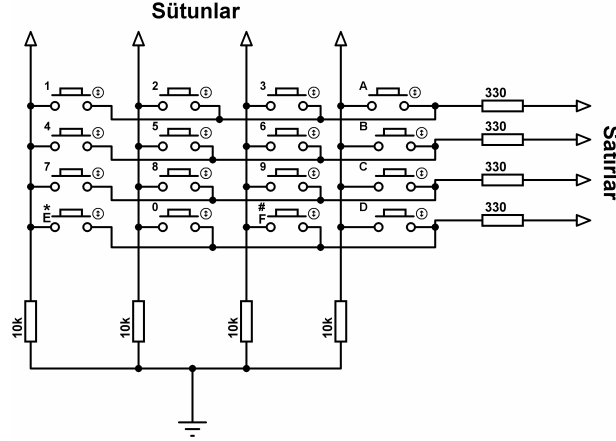


Şekil-12.1. 4x4 Tuş takımı.



Şekil-12.2. 4x3 Tuş takımı.

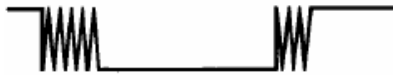
Tuş takımında hangi tuşa basıldığını bulmak için çeşitli yöntemler kullanılabilir. Bu yöntemlerden biri tarama yöntemidir. Şekil-12.3.'de butonlarla yapılmış 4x4 tuş takımı görülmektedir. Butonların bir ucu satır kısmına, bir ucu da sütun kısmına bağlıdır. Denetleyici ile tarama yapılırken **satırlar çıkış, sütunlar ise giriş olarak tanımlanır.**



Şekil-12.3. Butonlarla yapılmış 4x4 tuş takımı.

Sütunlarda hep lojik-0 (GND-şase) vardır. Hangi tuşa basıldığını anlamak için önce satırlardan biri lojik-1 değerleri lojik-0 yapılır. Sonra sütunlar okunur, hangi giriş lojik-1 ise o satıra ait sütundaki tuşa basılmış demektir. İstenen tuşa hangi değer verileceği programcıya aittir. Şekil-12.3'teki bağlantıda 1. satır lojik-1 diğer satırlar lojik-0 iken, 1.satırda 3 nolu tuşa basıldığında 3.sütunda lojik-1 bilgisi okunur. Böylece basılan tuş bulunabilir. Tabi ki 1.satırda değil de diğer satırlardaki tuşlara basılmışsa algılama yapılamaz. Bu nedenle 1.satırdan sonra sıra ile bir satır lojik-1, diğer satırlar lojik-0 yapılarak tüm sütunlar okunur. Bahsedilen tarama işlemi bu şekilde yapılmaktadır. Tablo-12.1'de, Şekil-12.3'teki tuş takımına göre uyarlanmış tarama bilgileri verilmiştir.

Butona basıldığında ve bırakıldığında bir ark (parazit) meydana gelir. Buna tuş sıçraması da (key debounce) denilir. Şekil-12.4'te örnek bir tuş sıçraması görülmektedir. Bu sıçramayı önlemek için programda gerekli önlemler alınmalıdır. Tedbir olarak butona basıldıktan sonra 15-20 msn gecikme verilmesi gerekir veya butondan el çekilene kadar içinden çıkılmayacak bir döngü kurulmalıdır. Ayrıca tuş takımında aynı anda iki tuşa birden basılabilir. Bu gibi durumlarda hangi tuşun geçerli olacağı programla belirtilerek istenmeyen durumlar önlenmelidir.



Şekil-12.4. Tuş basılıp bırakılmasında oluşan parazit.

Satırlar				Sütunlar				Basılan Tuş
4	3	2	1	4	3	2	1	
0	0	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0	2
0	0	0	1	0	1	0	0	3
0	0	0	1	1	0	0	0	A
0	0	1	0	0	0	0	1	4
0	0	1	0	0	0	1	0	5
0	0	1	0	0	1	0	0	6
0	0	1	0	1	0	0	0	B
0	1	0	0	0	0	0	1	7
0	1	0	0	0	0	1	0	8
0	1	0	0	0	1	0	0	9
0	1	0	0	1	0	0	0	C
1	0	0	0	0	0	0	1	*, E
1	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	#, F
1	0	0	0	1	0	0	0	D

Tablo-12.1. Tuş takımı tarama bilgileri.

12.2. TUŞ TAKIMI UYGULAMASI-1

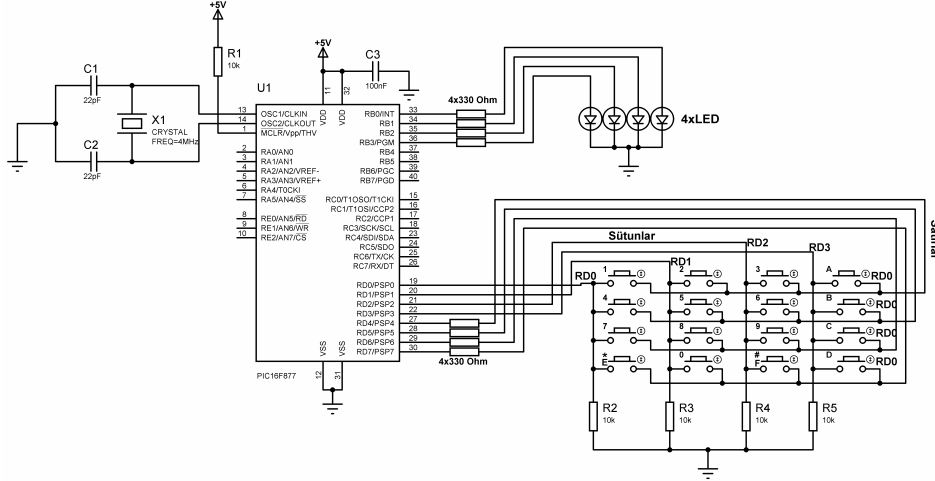
Benzetim: PROTEUS-ISIS ☺ Deneme Kartı: PIC PROG/DEKA 16F87x ☺

Bu uygulamada butonlarla yapılmış 4x4 bir tuş takımı uygulaması yapılmıştır. Tuş takımı D port'una bağlanmıştır. B port'unda RB0, RB1, RB2 ve RB3 pin'lerine LED bağlanmıştır. Tuş takımından basılan tuşların ikili karşılıkları LED'lerde gösterilmektedir. Tuş takımı için programda bir fonksiyon yazılmıştır. Programda tuş takımında bir tuşa basıldıktan sonra tuşun değeri B port'una bağlı LED'lerde görünecektir.

Programda her satır sırayla lojik-1 yapılmış ve ardından sütun girişleri kontrol edilmiştir. Hangi sütun lojik-1 ise o sütundaki butona basılmış demektir. Örneğin 2. satır lojik-1 yapıldığında, 2. sütun girişi lojik-1 ise demek ki 5 tuşuna basılmış demektir.

Programda yapılan fonksiyonda tarama sonucu basılan tuş değeri "tus" değişkenine aktarılmıştır. "tus" değişkeni kullanılarak elde edilen sonuç istenen şekilde kullanılır. Tuş takımında birden fazla tuşa basılması için gerekli önlem alınmamıştır. İstenirse bu gibi durumlarda basılan en son tuşun geçerli olması veya ilk basılan tuştan el

kalkana kadar, diğer basılan tuşların geçerli olmaması sağlanabilir. İstenen amaca göre program değiştirilebilir.



Şekil-12.5. Tuş takımı uygulaması-1 devre şeması.

Devre programı;

```

/*****
PIC16F877 ile Tuş Takımı Uygulaması-1
PIC PROG/DEKA : Port B jumper'ı LED konumunda olmalı
*****/
#include <16f877.h> // Kullanılacak denetleyicinin başlık
// dosyası tanıtılıyor.
// Denetleyici konfigürasyon ayarları
#fuses XT,NOVDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD

#use delay (clock=4000000) // Gecikme fonksiyonu için kullanılacak
// osilatör frekansı belirtiliyor.
#use fast_io(b) //Port yönlendirme komutları B port'u için geçerli
#use fast_io(d) //Port yönlendirme komutları C port'u için geçerli

#byte portb=0x06 // B port'u "portb" kelimesine eşitleniyor.

#define sut1 pin_d0 // sut1 ifadesi pin_d0 ifadesine eşitleniyor
#define sut2 pin_d1 // sut2 ifadesi pin_d1 ifadesine eşitleniyor
#define sut3 pin_d2 // sut3 ifadesi pin_d2 ifadesine eşitleniyor
#define sut4 pin_d3 // sut3 ifadesi pin_d2 ifadesine eşitleniyor

#define sat1 pin_d4 // sat1 ifadesi pin_d4 ifadesine eşitleniyor
#define sat2 pin_d5 // sat2 ifadesi pin_d5 ifadesine eşitleniyor
#define sat3 pin_d6 // sat3 ifadesi pin_d6 ifadesine eşitleniyor
#define sat4 pin_d7 // sat4 ifadesi pin_d7 ifadesine eşitleniyor

char tus=0; // karakter tipinde değişken tanımlanıyor

```

```
//***** Keypad Tarama Fonksiyonu *****  
char keypad_oku() // Fonksiyon ismi  
{  
    output_d(0x00); // D port'u çıkışı sıfırlanıyor  
  
    output_high(sat1); // 1. satır lojik-1 yapılıyor  
    if (input(sut1)) // 1. sütun okunuyor  
        { delay_ms(20); tus=1; }  
    if (input(sut2)) // 2. sütun okunuyor  
        { delay_ms(20); tus=2; }  
    if (input(sut3)) // 3. sütun okunuyor  
        { delay_ms(20); tus=3; }  
    if (input(sut4)) // 4. sütun okunuyor  
        { delay_ms(20); tus=0xA; }  
    output_low(sat1); // 1. satır lojik-0 yapılıyor  
  
    output_high(sat2); // 2. satır lojik-1 yapılıyor  
    if (input(sut1)) // 1. sütun okunuyor  
        { delay_ms(20); tus=4; }  
    if (input(sut2)) // 2. sütun okunuyor  
        { delay_ms(20); tus=5; }  
    if (input(sut3)) // 3. sütun okunuyor  
        { delay_ms(20); tus=6; }  
    if (input(sut4)) // 4. sütun okunuyor  
        { delay_ms(20); tus=0xB; }  
    output_low(sat2); // 2. satır lojik-0 yapılıyor  
  
    output_high(sat3); // 3. satır lojik-1 yapılıyor  
    if (input(sut1)) // 1. sütun okunuyor  
        { delay_ms(20); tus=7; }  
    if (input(sut2)) // 2. sütun okunuyor  
        { delay_ms(20); tus=8; }  
    if (input(sut3)) // 3. sütun okunuyor  
        { delay_ms(20); tus=9; }  
    if (input(sut4)) // 4. sütun okunuyor  
        { delay_ms(20); tus=0x0C; }  
    output_low(sat3); // 3. satır lojik-0 yapılıyor  
  
    output_high(sat4); // 3. satır lojik-1 yapılıyor  
    if (input(sut1)) // 1. sütun okunuyor  
        { delay_ms(20); tus=0xE; }  
    if (input(sut2)) // 2. sütun okunuyor  
        { delay_ms(20); tus=0; }  
    if (input(sut3)) // 3. sütun okunuyor  
        { delay_ms(20); tus=0xF; }  
    if (input(sut4)) // 4. sütun okunuyor  
        { delay_ms(20); tus=0xD; }  
    output_low(sat4); // 3. satır lojik-0 yapılıyor  
  
    return tus; // Fonksiyon "tus" değeri ile geri döner  
}
```

```

/***** ANA PROGRAM FONKSİYONU*****/
void main ( )
{
    setup_psp(PSP_DISABLED); // PSP birimi devre dışı
    setup_spi(SPI_SS_DISABLED); // SPI birimi devre dışı
    setup_timer_1(T1_DISABLED); // T1 zamanlayıcısı devre dışı
    setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
    setup_adc_ports(NO_ANALOGS); // ANALOG giriş yok
    setup_adc(ADC_OFF); // ADC birimi devre dışı
    setup CCP1(CCP_OFF); // CCP1 birimi devre dışı
    setup CCP2(CCP_OFF); // CCP2 birimi devre dışı

    set_tris_b(0x00); // B port'u komple çıkış
    set_tris_d(0x0F); // D portunun Yüksek değerlikli 4 bit çıkış,
    // düşük değerlikli 4 bit giriş

    output_b(0x00); // İlk anda B port'u çıkışı sıfırlanıyor

    while(1) // Sonsuz döngü
    {
        portb=keypad_oku(); // Basılan tuş değerini B port'una aktar
    }
}
/*****

```

Programda `#byte portb=0x06` komutu ile B port'unun adresi (0x06) "portb" ifadesine eşitlenmiştir. Yani bu komuttan sonra artık "portb" ifadesi direkt olarak B port'u değerini temsil etmektedir. Programda `#define sut1 pin_d0` komutu ile "sut1" ifadesi pin_d0'a yani RD0 pin'ine eşitlenmiştir. Diğer sütun ve satır ifadeleri de D port'unun ilgili pin'lerine aynı şekilde eşitlenmiştir.

`char tus=0;` komutu ile karakter tipinde "tus" isminde değişken tanımlanmış ve ilk değer olarak sıfır (0) verilmiştir. Tuş takımı tarama fonksiyonu olarak `char keypad_oku()` komutu ile `keypad_oku()` isminde fonksiyon tanımlanmıştır. Fonksiyonun başındaki "char" ifadesi, **fonksiyonun geri dönüş değeri olduğunu** ve bu değer karakter tipinde olduğunu belirtir. Fonksiyonda ilk başta `output_d(0x00);` komutu ile D port'una bağlı sütunların hepsi lojik-0 yapılmıştır. Daha sonra `output_high(sat1);` komutuyla sırayla her satır lojik-1 yapılmış ve `if (input(sut1))` komutu ile de sütun girişleri okunmuştur. Eğer sütun girişi "1" olursa if fonksiyonu döngüsü işlem görecektir. if fonksiyonu içinde, `delay_ms(20);` komutu ile tuş hatalarını önlemek için bir gecikme konulmuş ve ardından `tus=1;` komutu ile de basılan tuş değeri "tus" değişkenine aktarılmıştır. Burada `tus=` ifadesine istenen değer yazılarak hangi tuşun neye karşılık geleceği istenen şekilde değiştirilir. İlk satırda 4 sütunun taraması aynı mantıkla yapıldıktan sonra `output_low(sat1);` komutu ile 1. satır lojik-0 yapılmıştır. Daha sonra aynı işlemler diğer satırlar için yapılmıştır. Fonksiyonda en son olarak da `return tus;` komutu ile fonksiyonun "tus" değişkeni değeri ile geri dönmesi sağlanmıştır.

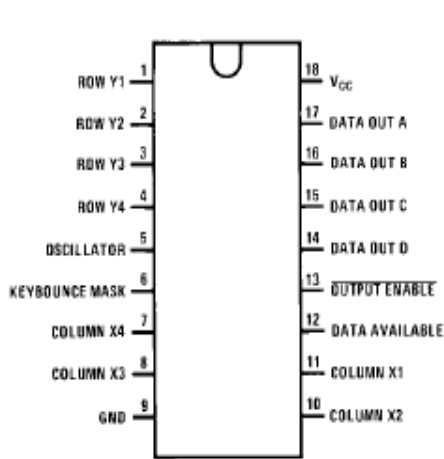
Dikkat edilmesi gereken bir nokta da, fonksiyon bir geri dönüş değeri içerdiğinden fonksiyon tanımlarken `char` `sutun_tara()` komutunda başta "char" kısmı ile fonksiyonun geri dönüş değeri olduğunu ve türünün karakter olduğunu belirtmemizdir. Fonksiyonun geri dönüş değeri olmasaydı başına "void" koymamız gerekirdi.

Ana fonksiyondaki sonsuz döngüde `portb=keypad_oku();` komutu ile tuş takımı tarama fonksiyonu çağrılmış ve bu fonksiyonda tuş değerinin tutulduğu "tus" değişkeni değeri B port'una gönderilmiştir.

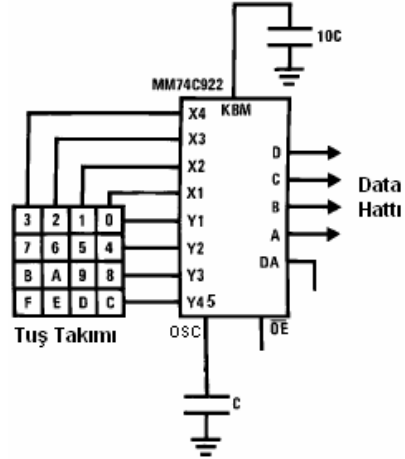
🔗 Tuş takımı tarama fonksiyonunu kendi amaçlarınıza ve ihtiyaçlarınıza göre değiştirebilirsiniz.

12.3. 74C922 KEY ENKODER ENTEGRESİ

Tuş tarama işlemi bir önceki uygulamada yapıldığı gibi kullanıldığında denetleyiciyi her zaman meşgul eder. Yani tuş tarama işleminin sürekli olarak yapılması gerekmektedir. Aynı anda başka işlemlerin de yapılacağı bir uygulamada bu yöntem sorunlara yol açabilir. Aynı zamanda önceki uygulamada tuş tarama işlemi için 8 adet pin kullanılmaktaydı, bu da port ihtiyacının fazla olduğu uygulamalarda sıkıntıya yol açabilir. Bu nedenle tuş takımı tarama işlemlerinde 74C922 16 Key Encoder entegresi kullanılabilir. Bu entegre sayesinde tarama işlemi için denetleyici tümüyle meşgul edilmeyecek ve aynı zamanda fazla pin de kullanılmayacaktır.



Şekil-12.7. 74C922 Key encoder entegresi pin diyagramı.



Şekil-12.8. 74C922 Entegresi genel bağlantısı.

74C922 entegresi 3V ile 15V çalışma gerilim aralığında çalışabilir. Genel besleme gerilimleri 5V, 10V ve 15V'tur. Klavye satır hatları Y girişlerine (ROW), sütun hatları