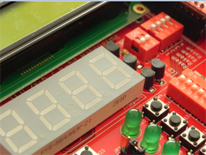
**4-7-Segment Uygulaması**

Muhammed Fatih İNANÇ, 08 Aralık 2012, Cumartesi



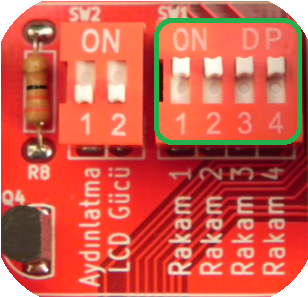
Bu uygulama notunda MSP430 Geliştirme Kiti üzerinde 7-Segment Uygulamasının nasıl yapılacağını inceleyeceğiz.

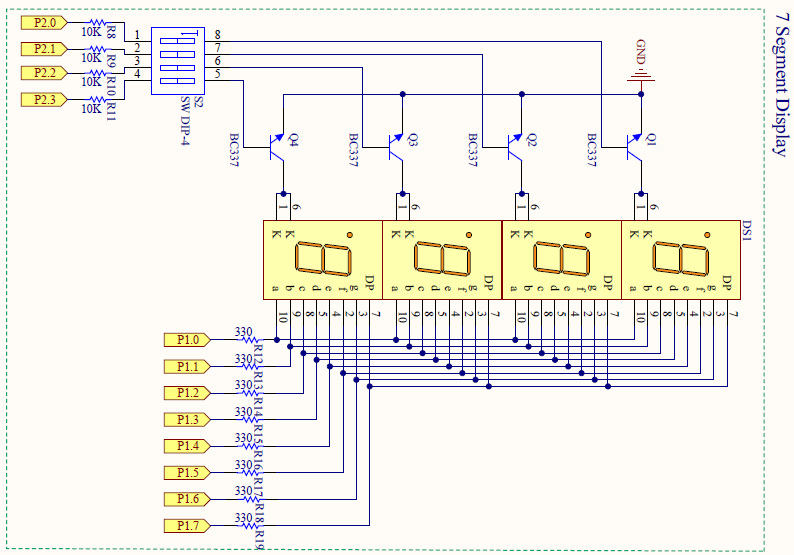
Bu uygulamada gerçekleştirilmesi amaçlanan işlem, kit üzerinde bulunan “7-Segment” kısmındaki 4 adet 7 parçalı göstergeyi “sütun tarama sistemi” ile kontrol etmektir.

**Portların Ayarlanması**

Ayarlanması gereken switchler;

* SW1 tümü **açık** konumda
* Diğer tüm switchler **kapalı** konumda
* LaunchPad üzerindeki J3 konnektöründen TXD ve RXD jumperlerı **çıkarılmalıdır.**

****

****

**Uygulamanın Çalışması**

Bu uygulamada mikrodenetleyicinin pinlerine bağlanmış 4 lü 7 segment display kontrol edilmektedir. 4 lü 7 Segment display’in 8 adet *segment,*4 adet ise *satır*seçme pinleri bulunmaktadır. 8 adet *segment*pinlerinin 7 si segmentler, 1 i de nokta segmenti içinidir. Ve bu pinlerin tamamı 4 adet farklı 7 segment displayde paralel bağlanmış şekildedir.

Aksi takdirde tüm segmentlerin kontrolü için 4\*8 = 32 adet pin kullanılmak zorunda kalınacaktır. Bu sayede mikrodenetleyici pinlerinden tasarruf edilerek sadece görüntülemek istenilen segment seçildiğinde o segmentin LED leri yanmaktadır.

Fakat bu yöntemde her seferinde yanlızca 1 segment seçilebildiği için o an seçilen segment yanmakta ve diğerleri sönük durumda kalmaktadır. Bunu aşabilmek için çok hızlı bir şekilde segment seçimlerini güncellemek gerekmektedir. Eğer çok hızlı bir şekilde segmentler güncellenirse bu frekans insan gözünün algılayabileceğinden daha yüksek olduğu için tüm segmentleri yanıyomuş gibi görürüz. Bu yönteme ***satır tarama***yöntemi denilmektedir.

Program içerisinde toplam 2 farklı zaman rutini bulunmaktadır. Bunlardan biri 5mS lik zaman periyodunda çalışmakta, diğeri ise 50mS lik zaman periyodunda çalışmaktadır. 5mS lik zaman periyodu satır tarama işlemini gerçekleştirirken 50mS lik zaman periyodu ise sayma hızını belirlemektedir.

Programın tamamında gecikme kullanımından kaçınılmıştır. Bu sayede mikrodenetleyici yanlıca gerekli zamanlarda uykudan uyanarak gerekli işlemleri gerçekleştirmektedir.

**Uygulama Kodu**

//

//LaunchPad Üzerindeki TXD ve RXD Jumplerlarını Çıkarınız

#include "msp430.h"

#define SEGMENT0 BIT0 // P2.0

#define SEGMENT1 BIT1 // P2.1

#define SEGMENT2 BIT2 // P2.2

#define SEGMENT3 BIT3 // P2.3

const unsigned char segment[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

unsigned int sayi=0,i=0;

void main( void )

{

WDTCTL = WDTPW + WDTHOLD;

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

\_\_delay\_cycles(1000000);

P1DIR = 0xFF; // P1 Tum Pinler Çikis

P2DIR = BIT0 + BIT1 + BIT2 + BIT3; // Segment Seçme Pinleri

P1OUT = 0x00;

P2OUT = 0x00;

TA0CTL = MC\_1 + TASSEL\_2;

TA0CCR0 = 5000 - 1;

TA0CCTL0 = CCIE;

TA1CTL = MC\_1 + TASSEL\_2;

TA1CCR0 = 50000 - 1;

TA1CCTL0 = CCIE;

\_\_bis\_SR\_register(LPM1\_bits + GIE);

}

#pragma vector=TIMER1\_A0\_VECTOR

\_\_interrupt void TA1\_A0\_ISR(void)

{

sayi++;

if(sayi > 9999)

sayi=0;

}

#pragma vector=TIMER0\_A0\_VECTOR

\_\_interrupt void TA0\_A0\_ISR(void)

{

i++;

switch(i)

{

case 1 :

{

P2OUT = 0;

P1OUT = segment[sayi/1000];

P2OUT = SEGMENT0;

break;

}

case 2 :

{

P2OUT = 0;

P1OUT = segment[(sayi/100)%10];

P2OUT = SEGMENT1;

break;

}

case 3 :

{

P2OUT = 0;

P1OUT = segment[(sayi/10)%10];

P2OUT = SEGMENT2;

break;

}

case 4 :

{

P2OUT = 0;

P1OUT = segment[(sayi%10)];

P2OUT = SEGMENT3;

i = 0;

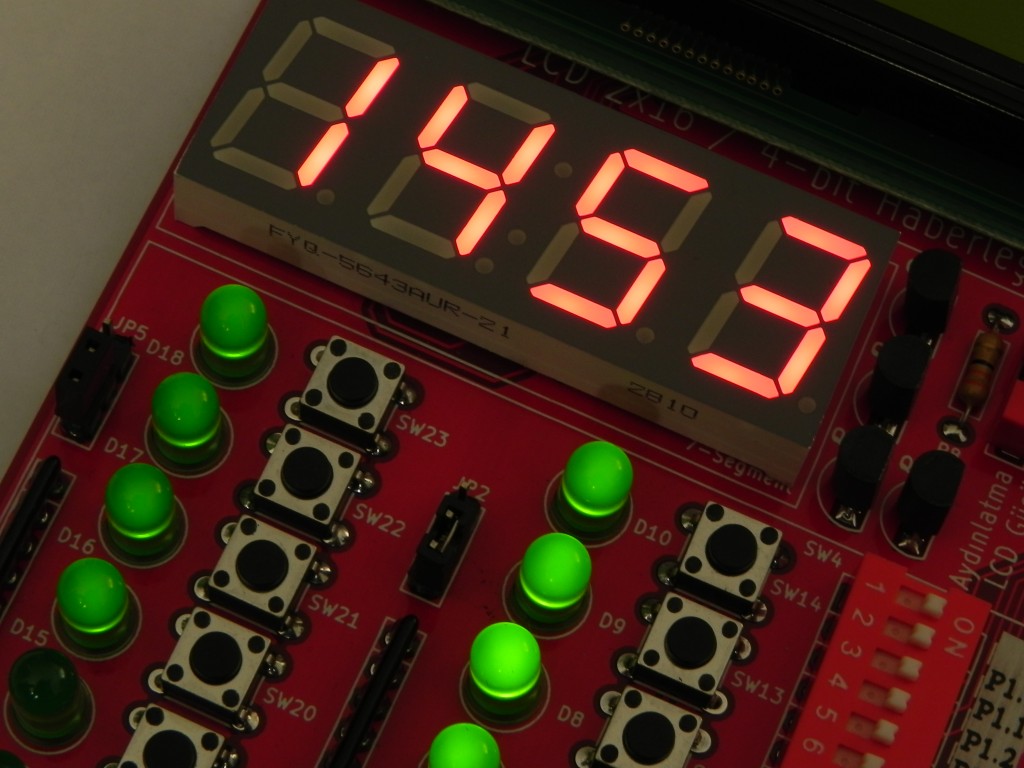
break;

}

}

}

* 22.Satırda P1 portunun tüm pinleri çıkış olarak ayarlanıyor.
* 23.Satırda P2 portunun yanlızca ilk 4 pini(P2.0-P2.3) çıkış olarak ayarlanıyor.
* 25 ve 26.Satırda P1 ve P2 portlarının ikisi de sıfırlanıyor.
* 28.Satırda **TA0**(Timer\_A0) **5mS** de bir kez kesme vektörüne dallanacak şekilde ayarlanıyor.
* 32.Satırda **TA1**(Timer\_A1) **50mS** de bir kez kesme vektörüne dallanacak şekilde ayarlanıyor.
* 36.Satırda MCU, LPM1 düşük güç moduna alınarak timer kesmelerinin oluşması bekleniyor.
* 40.Satırdaki *TA1\_A0\_ISR*kesme vektörü **50mS** lik kesme vektörüdür. Her 50mS lik zaman diliminde *sayi*isimli değişken 1 arttırılıyor. Eğer *sayi*değişkeni 9999 a ulaşmışsa sıfırlanıyor.
* 49.Satırdaki *TA0\_A0\_ISR*kesme vektörü **5mS** lik kesme vektörüdür. Bu vektör içerisinde *i*değişkeninin durumuna göre hangi segmentin seçileceğine karar veriliyor.
* 53.Satırda *i*değişkeninin durumuna göre kontrol sorgusu *switch*yapısı ile başlatılıyor. Ve *sayi*değişkeninin o anki değerine göre basamaklara ayırma işlemi yapılarak her 5mS de bir basamak segmentlerde gösteriliyor. 5mS yeteri kadar az bir süre olduğu için gözümüz tüm segmentleri yanıyormuş gibi görüyor.

****