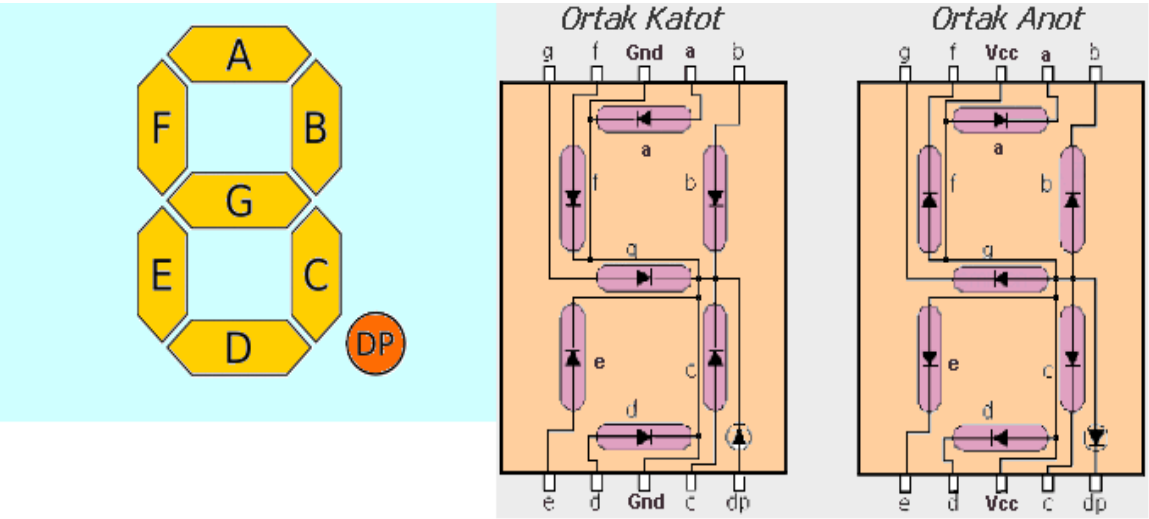
Modül 5: MSP430 Denetleyiciler ile 7 Parça Gösterge Kullanımı

Giriş

7 parça göstergeler elektronik sistemlerde oldukça kullanılan görüntüleme elemanlarıdır. Karakter LCDler gibi harf ve rakamları görüntüleyebilir yalnız daha kısıtlıdır. Şöyle ki adı üstünde içerisinde 7 adet çubuk segment şeklinde led bulunmaktadır. Bu ledler uygun düzende ışıtılarak istenen değer görüntülenir. 7 segment göstergeler, 7 parçalı gösterge, 7 segment display gibi isimlerle ilede anılabilir. Ucuz olması büyük boyutlarda gösterge tipleri olması, basit yapısı ve kullanımı gibi nedenlerden dolayı tercih edilmektedir. İç yapısında ledler bulunduğunu söylemiştik. İsterseniz uygulamanıza özel segment gösterge tasarlayabilirsiniz.



Şekil 1: 7 Parçalı Gösterge ve İç Yapısı

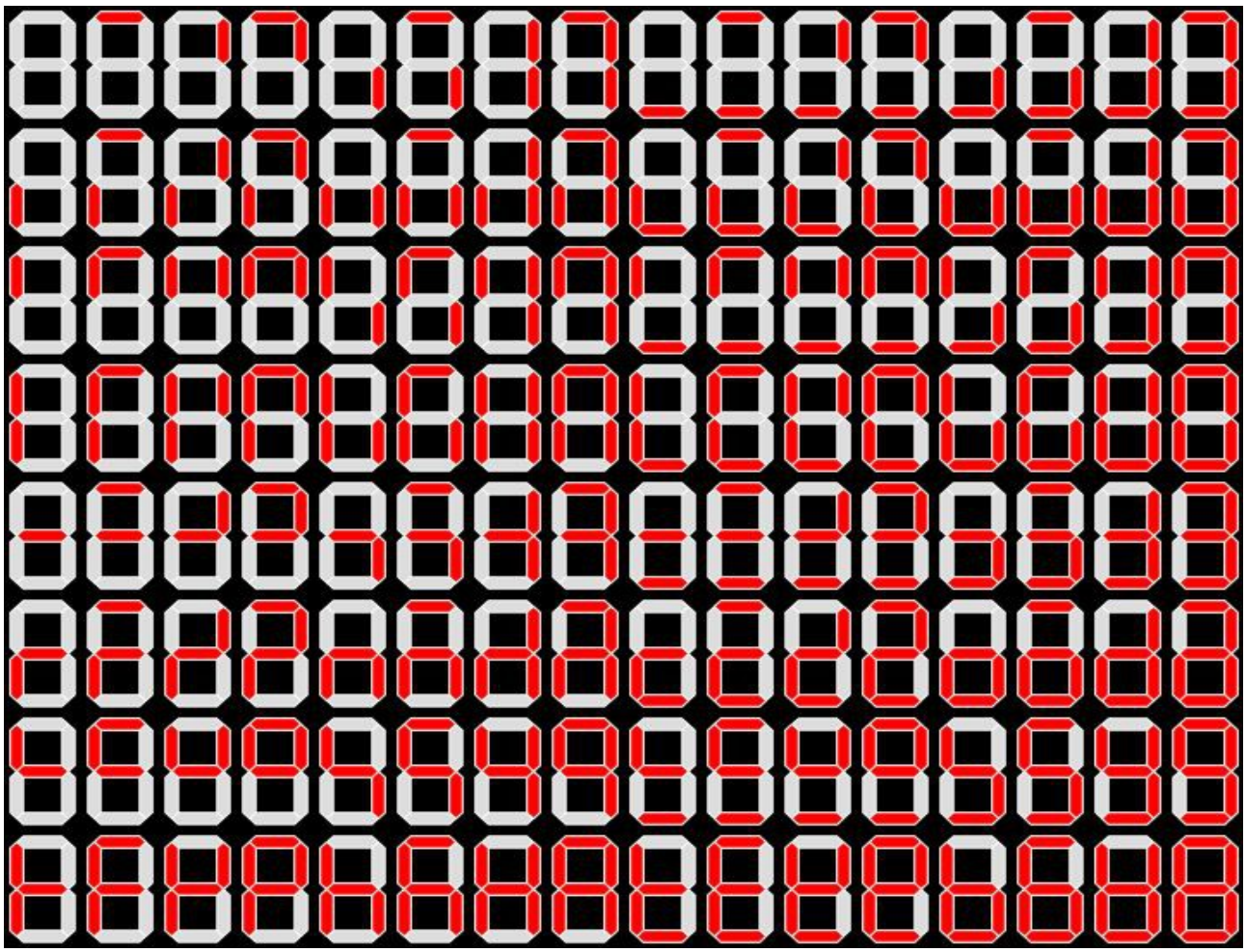
Şekil 1'de 7 segment göstergenin iç yapısı daha iyi anlaşılabilir. 7 segment yanında birde dp(Dot Point) denilen nokta segmenti vardır. Bu segment harici olarak nokta işareti oluşturmak için kullanılır.

7 parçalı göstergler iç yapısına göre ortak anot ve ortak katot diye ikiye ayrılır. Ortak katot düzende ledlerin anotları ayrı ayrı göstergeden dışarı çıkarılır. Katotları ise birleştirilerek ortak halde göstergeden dışarı çıkarılır. Ortak anot düzende ise tersi olarak ledlerin katotları ayrı ayrı göstergeden dışarı çıkarılır. Anotları ise birleştirilerek ortak halde gösterden dışarı çıkarılır. Görsel olarak bir fark bulunmamasına karşın kullanırken uygulanan işaretler bir birinden farklıdır yani tersidir.

Önceleri sadece sink(içe akıtan) olarak akım akıtan yada open drain port yapısına sahip denetleyiciler olduğu için ortak anot gösterge kullanmak yada harici bir tampon entegre kullanmak gerekliydi. Göstergeler bu ve benzeri nedenlerden dolayı ortak katot ve ortak anot şeklinde üretilmektedir. Günümüzde mikrodenetleyicilerin çoğu iki tip göstergeyi de sürebilecek yeteneğe sahiptirler.

Görüldüğü gibi nokta parçasını saymaz isek göstergeler adı üstünde 7 parça ışık yayan elemana(LED) sahiptir. Bundan dolayı 7 parça göstergeler grafik yada karakter LCDlere göre daha kısıtlı görüntüleme imkanına sahiptir.

Şekil 2'de 7 parça gösterge ile oluşturabileceğimiz görüntüleri görebilirsiniz.



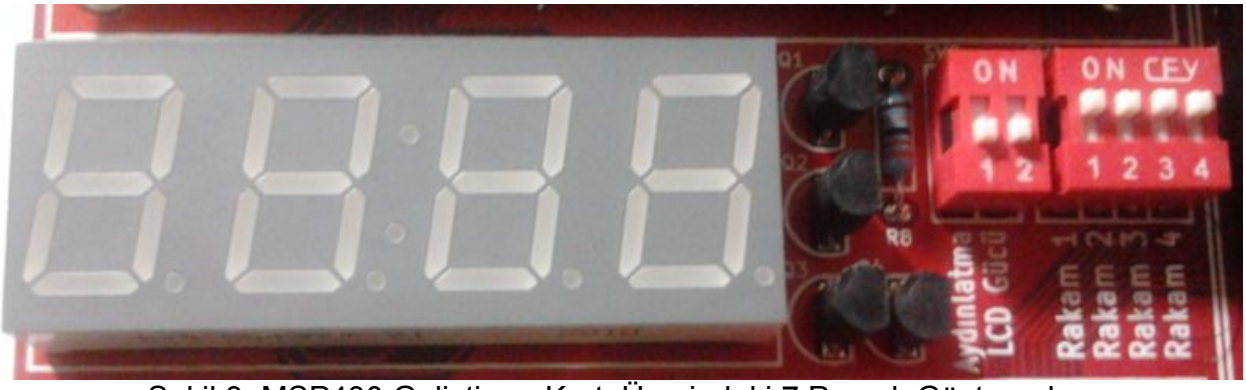
Şekil 2: 7 Parça Gösterge Görüntü Şekilleri

Şekil 2'de görüldüğü üzere 7 parça gösterge ile elde edilebilecek 2^7=128 görüntü vardır. Bu görüntülerin çoğu anlamsızdır. Uygulamalarda genelde 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F karakterlerinin yanı sıra bazı semboller ve işaretler kullanılır.

Uygulamalarda daha fazla karakter ihtiyacı için 7 parçalı göstergeler yanyana getirilip istenildiği sayıda kullanılabilir.

MSP430 7 Parçalı Gösterge Kullanımı

7 Parçalı göstergeler oldukça kullanılan bir gösterge çeşiti olduğu için MSP430 geliştirme kartımız üzerinde tek paket halinde içerisinde 4 adet 7 segment ortak katot gösterge bulunan ekranımız bulunmaktadır.



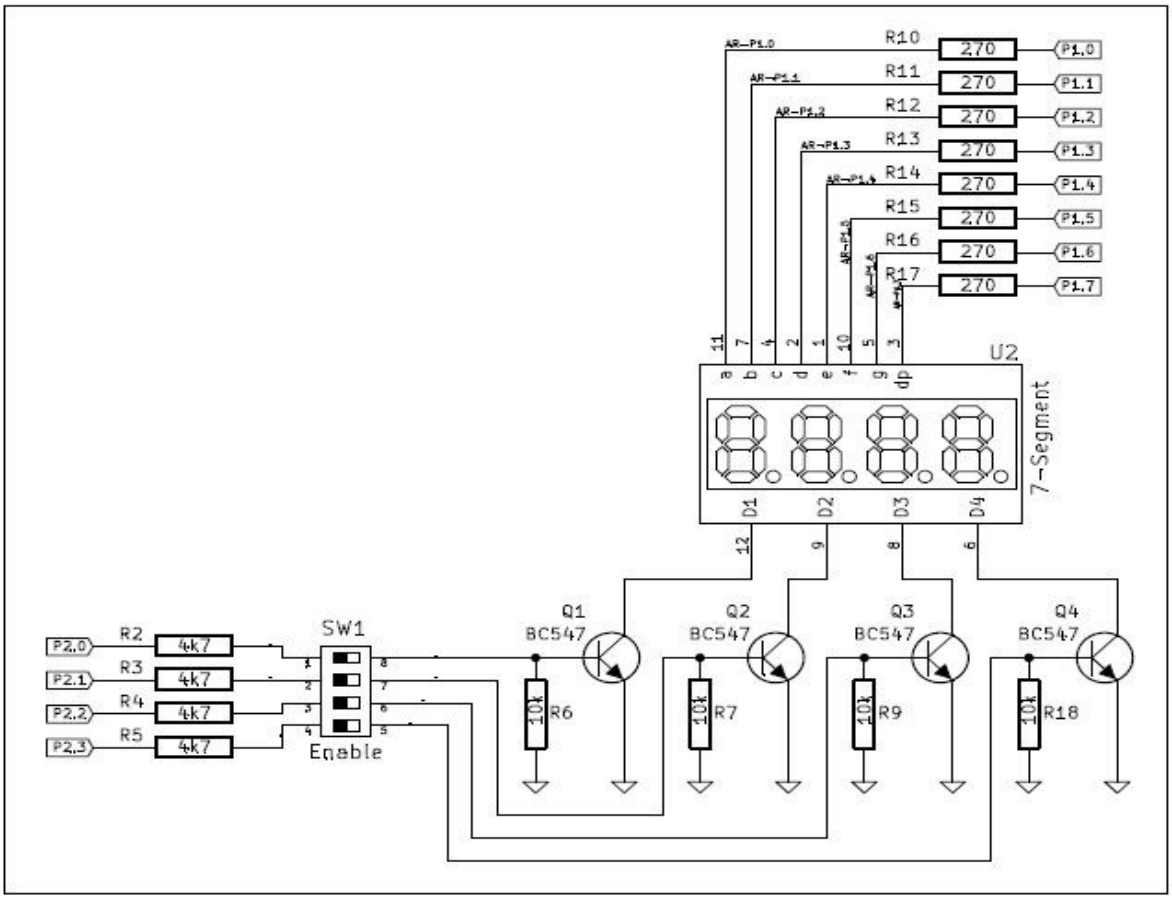
Şekil 3: MSP430 Geliştirme Kartı Üzerindeki 7 Parçalı Göstergeler

Şekil 3'te geliştirme kartımız üzerinde bulunan 7 parçalı göstergeler görülebilir. Sw1 anahtarlarından anlaşıldığı üzere uygulamaya göre kullanılmayan göstergelerin anahtarları kapatılarak güç tüketmesi önlenir.

7 parçalı göstergeleri mikrodenetleyiciler ile kullanabilmek için her parçaya bağlı ledin ayrı ayrı kontrol edilebilmesi gereklidir. Bu işlem için direk olarak denetleyici portlarını kullanırsak 1 tane 7 parça gösterge için nokta göstergesi dahil 8 bit port girişine ihtiyaç olacaktır. Ayrıca gösterge tipine göre ortak anot ise anot uygun gerilime, ortak katot ise katot gndye bağlanmalıdır.

Bu şekilde örneğin 4 adet gösterge için 32 adet port pini gereklidir. Tabi bu kadar port pini kullanımı fazladır. Uygulamalarda pin sayısını azaltmak için genelde zaman aralıklı tarama yöntemi kullanılır. Bu yöntem ile N adet 7 parça gösterge kullanılan bir uygulamada nokta gösterge dahil 8+N tane port pini yeterlidir. Örneğin 4 adet gösterge için sadece 12 port pini yeterlidir. 12 port pini 32 port pinine göre daha makuldur.

Bu yöntemle kullanılan gösterge sayısı her ne kadar N ile ifade edilsede gösterge sayısını asgari tutmakta fayda var. Zira bu yöntem ile port pininde tasarruf edilmesine karşın uygulama yazılımının yükü artmaktadır.



Şekil 4: MSP430 Geliştirme Kartı 7 Parça Gösterge Devre Şeması

Şekil 4'te geliştirme kartımız üzerinde bulunan 7 parça göstergelerin bağlantı şeması görülmektedir. Dikkat edilirse mikrodenetleyici bağlantısı zaman aralıklı taramaya uygun şekilde yapılmıştır. Port2.0-3 bitilerine bağlı transistörler ile istenilen göstergenin katodu gndye bağlanıp aktif edilebilir. Göstergelerin segmentleri ise birleştirilip Port1'e bağlanmıştır. Böylece port1'e yazılan bilgi tüm göstergelerin segmentlerine uygulanır. Hangi gösterge aktif edilmiş ise yazılan değer ilgili göstergede görüntülenir.

Bu yöntem ile yazılım içerisinde belirli zaman süreleri boyunca(genelde 5ms) sadece tek bir gösterge aktif edilir ve portlara o gösterge ile ilgili değerler yazılır. Yani göstergeler sürekli olarak 5'er ms aktif yapılır. Örneğin ilk anda sadece gösterge 1, 5ms aktif edilir, ilgili değer port1'e yazılır, göstergede görüntülenir. 5 ms sonra sadece gösterge 2, 5ms boyunca aktif edilir ve port1'e ilgili değer yazılır. Bu sayede tüm göstergeler sürekli aktif olur. Buradaki püf nokta göstergelerde bulunan ledlerin enerjileri kesilse bile içerilerinde bulunan fosfor yapı hemen sönmediği için bir daha ki taramaya kadar yanık kalırlar. Yani böylece göstergelerde sürekl bir görüntü sağlanır. Tarama süresi göstergelerin yapısına ve gösterge sayısına bağlı olarak değişebilir. Uygun gösterge zamanını deneyerek bulmanız gerekebilir. Tarama zamanı uygun seçilmez ise çalışma anında göstergelerde titreme gibi sorunlar yaşanabilir.

Not: Geliştirme kartımız üzerinde bulunan göstergeler küçük boyutta ve düşük akım tükettikleri için direk olarak mikrodenetleyiciye bağlanabilir. Daha büyük gösterge kullanımı için gösterge ile denetleyici arasına uygun tampon devreyi bağlamanız gerekmektedir.

Uygulama 5.1 7 Parça Gösterge Kullanımı

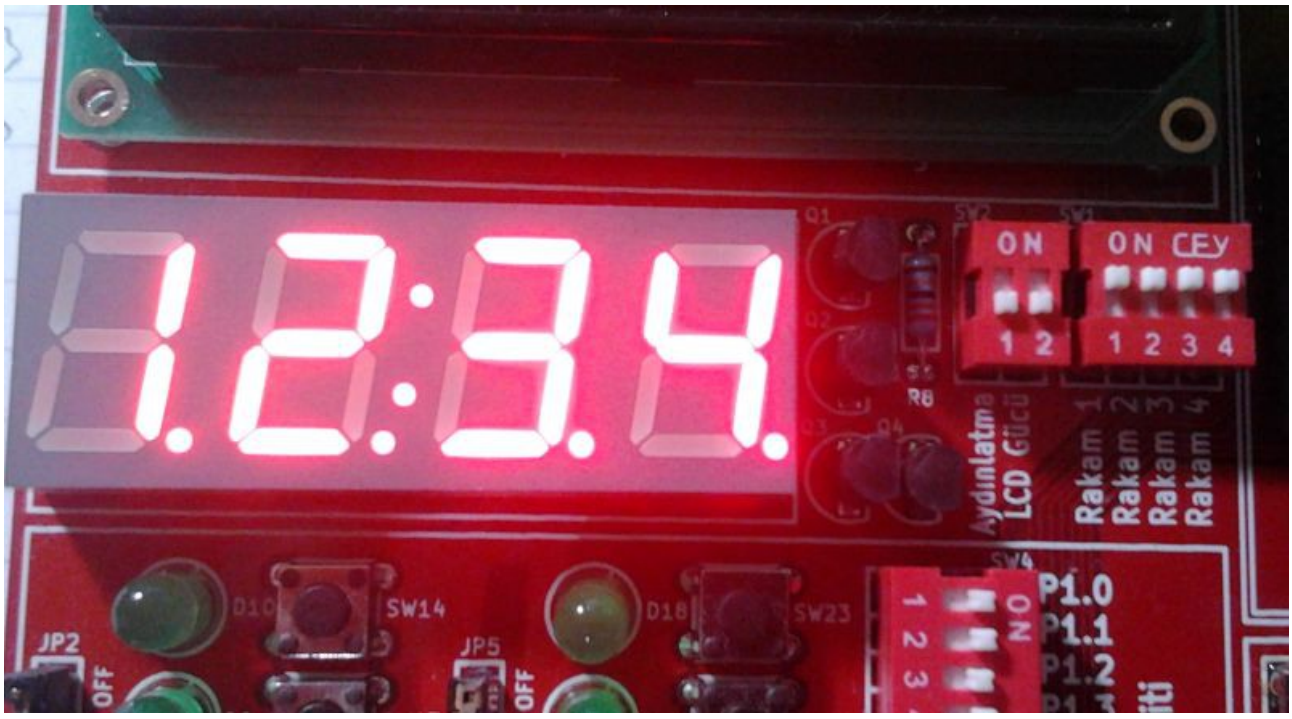
Bu uygulamamızda geliştirme kartımızda bulunan 7 parçalı göstergeleri çalıştırıp her birine sabit bir sayı yazdıracağız. Uygulamanın kodları aşağıdaki gibidir.

```
#include <msp430.h>                                    // MSP430 başlık dosyası


#define   GOSTERGE1            BIT0                    // P2.0 Gösterge1 yetki pini
#define   GOSTERGE2                  BIT1              // P2.1 Gösterge2 yetki pini
#define   GOSTERGE3            BIT2                    // P2.2 Gösterge3 yetki pini
#define   GOSTERGE4            BIT3                    // P2.3 Gösterge4 yetki pini
#define   NOKTA_SEGMENET       BIT7                    // Nokta segmenti
#define   SEGMENT_DATA_PORT          P1OUT             // Segment veri portu
#define   SEGMENT_DATA_PORT_DIR  P1DIR                 // Segment veri portu yönlendirme
kaydedicisi
#define   SEGMENT_DATA_PORT_SEL1      P1SEL            // Segment veri portu seçme1
kaydedicisi
#define   SEGMENT_DATA_PORT_SEL2      P1SEL2           // Segment veri portu seçme2
kaydedicisi
#define   GOSTERGE_PORT              P2OUT             // Gösterge seçme portu
#define   GOSTERGE_PORT_DIR          P2DIR             // Gösterge seçmme portu
yönlendirme kaydedicisi
#define   GOSTERGE_PORT_SEL1         P2SEL             // Gösterge seçme portu seçme1
kaydedicisi
#define   GOSTERGE_PORT_SEL2         P2SEL2            // Gösterge seçme portu seçme2
kaydedicisi
unsigned char bGostergeSayac=1;      // Gösterge sayaç değişkeni
unsigned char bGostergeData[4];      // Göstergelere yazdıralan değerleri tutan dizi
unsigned char bNokta=0;              // Göstergelerin nokta bilgisini tutan değişken
// 7 Parça gösterge porta yazılan karakter değerleri 0-9, A-F karakterleri
const unsigned char
SegmentDataTablo[16]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X77,
0X7C,0X39,0X5E,0X79,0X71};
```

```
void main(void) {
   WDTCTL  = WDTPW | WDTHOLD;            // Watchdog timeri durdur.
   BCSCTL1 = CALBC1_1MHZ;                // Dahili osilatörü 1MHz'e ayarla.
   DCOCTL = CALDCO_1MHZ;                 // Dahili osilatörü 1MHz'e ayarla.
   SEGMENT_DATA_PORT_DIR  = 0xFF;  // Segment portu çıkış
   SEGMENT_DATA_PORT          = 0x00; // Segment portunu başlangıçta sıfırla
   SEGMENT_DATA_PORT_SEL1 = 0x00; // Segment portu port işlemlerinde kullanılacak
   SEGMENT_DATA_PORT_SEL2 = 0x00; // Segment portu port işlemlerinde kullanılacak
   GOSTERGE_PORT_DIR  |=   GOSTERGE1 + GOSTERGE2 +
   GOSTERGE3+GOSTERGE4; // Gösterge pinlerini çıkış yap
   GOSTERGE_PORT        &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
   GOSTERGE4);   // Göstergeler başlangıçta pasif
   GOSTERGE_PORT_SEL1 &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
   GOSTERGE4);   // Gösterge portu port işlemlerinde kullanıalacak
   GOSTERGE_PORT_SEL2 &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
   GOSTERGE4); // Gösterge portu port işlemlerinde kullanıalacak
   TA0CCTL0 = CCIE;                      // Timer0 CCR0 ayarları
   TA0CCR0  = 5000-1;                    // Timer0 kesme periyodu 5ms
   TA0CTL   = TASSEL_2 + MC_2;           // Timer0 ayarları
   bGostergeData[0] = 1;                 // Gösterge 1'e 1 yaz
   bGostergeData[1] = 2;                 // Gösterge 2'ye 2 yaz
   bGostergeData[2] = 3;                 // Gösterge 3'e 3 yaz
   bGostergeData[3] = 4;                 // Gösterge 4'e 4 yaz
   bNokta=BIT0 + BIT1 + BIT2 + BIT3;     // Gösterge noktalarını yak
   _BIS_SR(LPM0_bits + GIE);             // Kesmeleri aç uykuya gir
}

// Sırayla göstergeleri aktif eden ve ilgili verileri segment portuna yazan CCR0 kesme rutini
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
        switch(bGostergeSayac){         // Sırada hangi gösterge var?
        case 1:                 // Gösterge 1 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[0]] ;
        if(bNokta & BIT0) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE1;
        GOSTERGE_PORT &= ~(GOSTERGE2 + GOSTERGE3 + GOSTERGE4);
        bGostergeSayac=2;               // 2. göstergeye geç
        break;
        case 2:                 // Gösterge 2 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[1]] ;
        if(bNokta & BIT1) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE2;
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE3 + GOSTERGE4);
        bGostergeSayac=3;               // 3. göstergeye geç
        break;
        case 3:                 // Gösterge 3 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[2]] ;
        if(bNokta & BIT2) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE3;
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE4);
        bGostergeSayac=4;               // 4. göstergeye geç
```

```
        break;
        case 4:                 // Gösterge 4 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[3]] ;
        if(bNokta & BIT3) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE4;
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3);
        bGostergeSayac=1;                   // 1. göstergeye geç
        break;
        }
        TA0CCR0 += 5000;          // Timeri yeniden kur.
}
```

Kodları incelersek; Görüldüğü gibi başlangıçta temel işlemlerin yanı sıra define komutlarıyla ilgili portlara akılda kalıcı isimler ve değerler verilerek uygulamanın anlaşılırlığı arttırılmıştır. Bu tarz tanımlamalar kodlarınızın anlaşılırlığı ve kolaylığı açısından önemlidir.
Uygulamanın çalışması basittir. SegmentDataTablo adlı dizide 0-9, A-F karakterlerine karşılık gelen segment verileri bulunmaktadır. Göstergelerde gösterilmek istenen veriler bGostergeData adlı değişkene yazdırılır. Gösterilmek istenen veri 0-15(0-F) aralığında olmalıdır. Sonrasında zamanlayıcı 5ms de bir kesme oluşturacak şekilde ayarlanarak kesmelere izin verilir işlemci uykuya sokulur.
Kesme programı çalıştığında yani 5ms sonra, her seferde sırasıyla bir gösterge aktif edilir diğerleri pasif edilir. Sonrasında ilgili bGostergeData dizisinde ki gösterge değerine göre SegmentDataTablosunda ki verilerden segment verisi seçilerek segment portuna yazdırılır. Yazdırılıktan sonra sıradaki diğer gösterge seçilir ve timer yeniden kurularak kesme rutininden çıkılır. Kesme programı her çalışmasında aynı işlem devam eder. Böylece göstergelere yazılan değerler görüntülenir.



Şekil 5: Uygulama 5.1 Çalışma Görüntüsü

Şekil 5'te uygulamanın çalışmasına ait görüntüyü görebilirsiniz.
Kodlarımızı derleyip debug işlemi ile kartımıza yüklemeden önce kartımız üzerinde bulunan sw1 anahtarlarını on konumuna getirerek göstergelerimizi aktif hale getirmeliyiz. Ayrıca kullanılmayan diğer donanımların anahtarlarını kapatmakta fayda var.

Kodlarımızı derleyip sorunsuz bir şekilde kartımıza yükledikten sonra kart üzerinde ki göstegelerde sırasıyla 1,2,3,4 karakterlerinin oluştuğunu ve nokta segmentlerinin ışıdığını görebilirsiniz. Şekil 5'te görüldüğü gibi birde çift nokta segmenti bulunmaktadır. Bu segment genellikle saat, tarih gösteren uygulamalarda ayraç olarak kullanılmak amacıyla düşünülmüş ve dörtlü göstergemizde harici bir pin ile sürülmektedir. Fakat geliştirme kartımızda bu pin 2. göstergenin nokta segmentiyle birlikte bağlandığı içinde beraber ışımaktadır.

Uygulama 5.2 7 Parça Gösterge ile Sayıcı

Bu uygulamamızda bir önceki uygulamadan biraz farklı olarak sabit görüntü yerine basit bir sayıcı uygulaması yapacağız. Uygulama kodları aşağıdaki gibidir.

```
#include <msp430.h>                                    // MSP430 başlık dosyası

#define   GOSTERGE1             BIT0                   // P2.0 Gösterge1 yetki pini
#define   GOSTERGE2             BIT1                   // P2.1 Gösterge2 yetki pini
#define   GOSTERGE3             BIT2                   // P2.2 Gösterge3 yetki pini
#define   GOSTERGE4             BIT3                   // P2.3 Gösterge4 yetki pini
#define   NOKTA_SEGMENET        BIT7                   // Nokta segmenti
#define   SEGMENT_DATA_PORT                 P1OUT      // Segment veri portu
#define   SEGMENT_DATA_PORT_DIR             P1DIR      // Segment veri portu
yönlendirme kaydedicisi
#define   SEGMENT_DATA_PORT_SEL1            P1SEL      // Segment veri portu seçme1
kaydedicisi
#define  SEGMENT_DATA_PORT_SEL2             P1SEL2     // Segment veri portu seçme2
kaydedicisi
#define   GOSTERGE_PORT             P2OUT              // Gösterge seçme portu
#define   GOSTERGE_PORT_DIR         P2DIR              // Gösterge seçmme portu
yönlendirme kaydedicisi
#define   GOSTERGE_PORT_SEL1        P2SEL              // Gösterge seçme portu seçme1
kaydedicisi
#define   GOSTERGE_PORT_SEL2        P2SEL2             // Gösterge seçme portu seçme2
kaydedicisi
void BCDCevir(unsigned short);          // Fonksiyon prototipi.
unsigned char bGostergeSayac=1;         // Gösterge sayaç değişkeni
unsigned char bGostergeData[4];         // Göstergelere yazdıralan değerleri tutan dizi
unsigned char bNokta=0;                 // Göstergelerin nokta bilgisini tutan değişken
unsigned char bGecikmeSayac=0;          // Gecikme sayacı
unsigned short wSayici=0;               // Sayıcı değişkeni
// 7 Parça gösterge porta yazılan karakter değerleri 0-9, A-F karakterleri
const unsigned char
SegmentDataTablo[16]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X77,
0X7C,0X39,0X5E,0X79,0X71};
```

```
void main(void) {
   WDTCTL  = WDTPW | WDTHOLD;            // Watchdog timeri durdur.
   BCSCTL1 = CALBC1_1MHZ;                // Dahili osilatörü 1MHz'e ayarla.
   DCOCTL = CALDCO_1MHZ;                 // Dahili osilatörü 1MHz'e ayarla.
   SEGMENT_DATA_PORT_DIR  = 0xFF;        // Segment portu çıkış
   SEGMENT_DATA_PORT      = 0x00;        // Segment portunu başlangıçta sıfırla
   SEGMENT_DATA_PORT_SEL1 = 0x00;        // Segment portu port işlemlerinde
kullanılacak
   SEGMENT_DATA_PORT_SEL2 = 0x00;        // Segment portu port işlemlerinde
kullanılacak
   GOSTERGE_PORT_DIR  |=   GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
GOSTERGE4;       // Gösterge pinlerini çıkış yap
   GOSTERGE_PORT        &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
GOSTERGE4);      // Göstergeler başlangıçta pasif
   GOSTERGE_PORT_SEL1 &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
GOSTERGE4);      // Gösterge portu port işlemlerinde kullanıalacak
   GOSTERGE_PORT_SEL2 &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3 +
GOSTERGE4); // Gösterge portu port işlemlerinde kullanıalacak
   TA0CCTL0 = CCIE;                      // Timer0 CCR0 ayarları
   TA0CCR0  = 5000-1;                    // Timer0 kesme periyodu 5ms
   TA0CTL   = TASSEL_2 + MC_2;           // Timer0 ayarları
   bNokta=0;                             // Gösterge noktalarını söndür
    _BIS_SR(GIE);                        // Kesmeleri aç
   while(1){                             // Sonsuz döngü
   BCDCevir(wSayici);      // wSayici değerini BCD'ye çevir gösterge değişkenlerine yaz.
   _BIS_SR(LPM0_bits);                   // Uykuya gir
}}

// 0-9999 Arası girilen sayıyı BCD'ye çeviren fonksiyon
void BCDCevir(unsigned short Sayi){
        unsigned char bSayac;
        for(bSayac=0;bSayac<4;bSayac++){
        bGostergeData[3-bSayac]=Sayi%10;
        Sayi /= 10;
        }
}
// Sırayla göstergeleri aktif eden ve ilgili verileri segment portuna yazan CCR0 kesme rutini
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
        switch(bGostergeSayac){          // Sırada hangi gösterge var?
        case 1:                 // Gösterge 1 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[0]] ;
        if(bNokta & BIT0) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE1;
        GOSTERGE_PORT &= ~(GOSTERGE2 + GOSTERGE3 + GOSTERGE4);
        bGostergeSayac=2;                // 2. göstergeye geç
        break;
        case 2:                 // Gösterge 2 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[1]] ;
        if(bNokta & BIT1) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE2;
```

```
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE3 + GOSTERGE4);
        bGostergeSayac=3;                   // 3. göstergeye geç
        break;
        case 3:                 // Gösterge 3 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[2]] ;
        if(bNokta & BIT2) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE3;
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE4);
        bGostergeSayac=4;                   // 4. göstergeye geç
        break;
        case 4:                 // Gösterge 4 ise aktif et veriyi segment portuna yaz.
        SEGMENT_DATA_PORT = SegmentDataTablo[bGostergeData[3]] ;
        if(bNokta & BIT3) SEGMENT_DATA_PORT |= NOKTA_SEGMENET;
        GOSTERGE_PORT |= GOSTERGE4;
        GOSTERGE_PORT &= ~(GOSTERGE1 + GOSTERGE2 + GOSTERGE3);
        bGostergeSayac=1;                   // 1. göstergeye geç
        break;
        }
        if(++bGecikmeSayac>=100){               // 500 ms oldu mu?
        bGecikmeSayac=0;                        // Zaman sayacını sıfırla
        if(++wSayici>9999)wSayici=0;            // Sayacı 1 arttır 9999'da büyükse sıfırla
        __bic_SR_register_on_exit(CPUOFF);      // İşlemciyi uykudan uyandır
        }
        TA0CCR0 += 5000;                        // Timeri yeniden kur.
}
```

Kodlardan alaşıldığı gibi wSayici isimli bir değişken 0-9999 arası değerleri tutması için kullanılmıştır. Timer 5ms ye kurulu olduğu için her 100 kesmede bir yaklaşık 500 ms zaman geçer. Her 500 ms de sayacın değeri 1 arttırılır. Sonrasında bu değer BCDCevir fonksiyonu ile 4 adet rakama dönüştürülerek gösterge değişkenlerine kayıt edilir. Kayıt edilen rakamlar süre doldukça sıra ile göstergelerde görüntülenir. İşlem yapılmadığı sürelerde işlemci uyku moduna sokularak güç tasarrufu sağlanır.

Şekil 6: Uygulama 5.2 Çalışma Görüntüleri

Şekil 6'da uygulamanın çalışma görüntüleri görülmektedir. Geliştirme kartımızın ayarlarının bir önceki uygulama ile aynı kalması yeterlidir. Uygulama kodlarını sorunsuz bir şekilde derleyip kartımız üzerindeki denetleyicimize yüklersek uygulama çalışacaktır.

Uygulama çalışır çalışmaz 0000dan başlayarak 9999'a kadar yarım saniye aralıklar ile saymaya devam eder.

Bu modül kapsamında 7 parçalı göstergelerin anlatımı yapılmış MSP430 (G2553) denetleyiciler ile uygulaması yapılmıştır. Gösterge veya ekran kullanmanız gereken uygulamalarda, maliyet, boyut, görsellik, çözünürlük gibi konuları dikkate alarak Parçalı gösterge, Karakter LCD yada başka çeşit görüntüleme elemanları kullanabilirsiniz.