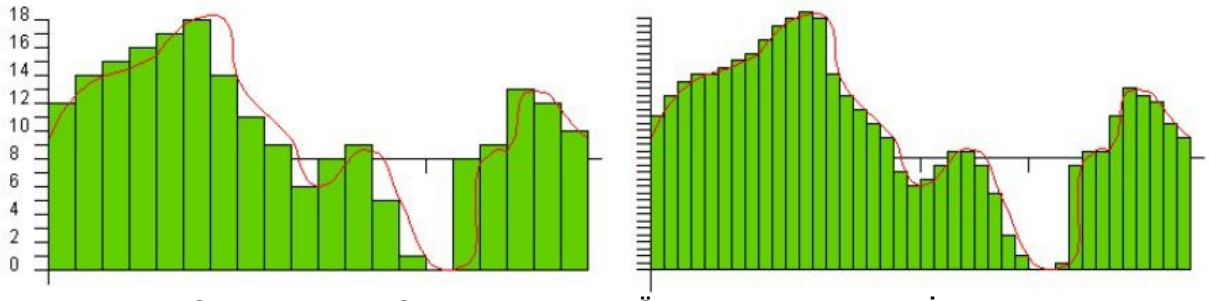
Modül 6: MSP430 ADC10 Analog Sayısal Dönüştürücü Birimi ve Uygulamaları

Giriş

Analog sayısal dönüştürücüler mikro işlemci, mikrodenetleyici gibi sayısal sistemler ile analog sistemler arasında köprü görevi görür dersek yanlış olmaz. Yalnız bu köprü tek yönlü çalışır. Yani analog sayısal dönüştürücüler analog bir işaretin mikrodenetleyici gibi sayısal sistemler tarafından alınması için kullanılır. Mikrodenetleyici benzeri sayısal sistemlerden analog sistemlere analog bilgi göndermek için ise analog sayısal dönüştürücülerin tersi olarak çalışan sayısal analog dönüştürücüler kullanılır.

Analog sayısal dönüştürücüler, ADC, Analog to Digital Converter, Analog Dijital çevirici gibi isimlerle de anılabilir. Günümüzde çoğu mikrodenetleyicide ADC birimi bulunmaktadır. Aynı şekilde harici entegre olarak üretilen çeşitli özelliklere sahip ADC entegreleri de bulunmaktadır.

ADC'nin çalışma mantığını anlamak için 0 ve 1 sayılarını ele alalım. Örneğin 0-1V arasında değişen bir işaretimiz olsun. Bu işaretimiz kaç değer alabilir? Elbette sonsuz reel değer alabilir. Sayısal sistemler sonlu sayıda değer alabilen değişkenler ile çalıştığı için sonsuz değerler alabilen değişkenleri işleyemezler. Bunun için ADC sayesinde örneğin 0-1V arası N miktar aralıklar ile örneklenerek 0 ile 1 arasında 0, 0+N, 0+2N, 0+3N....1 gibi sonlu sayıda değere sahip bir sayısal işaret oluşur. N=0.1 alırsak İşaretimiz 0, 0.1, 0.2, 0.3 .... 1 şeklinde sonlu sayıda değere sahip olacaktır. Burada N sayısı ADC çeviricinin adım aralığını belirtir. Adım aralığı ne kadar küçük seçilirse ADCmizin çözünürlüğü o kadar artar ve örneklenen analog işaret gerçek analog işarete o kadar yakınlaşır.



Şekil 1: Farklı Çözünürlüklerde Örneklenmiş Analog İşaret

Şekil 1'de AD çevrim çözünürlüğünün etkisi açıkca görülmektedir. Çözünürlüğün artması işaretin orijinale yakın olmasını sağlar fakat aynı zamanda ADC çeviricinin maliyeti artar ve elde edilen işareti işlemeyi zorlaştırır. Bu gibi tezat oluşturan durumlara Trade-Off denir. Bu durumlarda sistemin gereksinimlerini iyi belirleyip uygun özellikte AD çevirici seçmek gerekir. Piyasada 8 bitten 24 bite kadar analog sayısal dönüştürücü ürünleri bulunmaktadır.

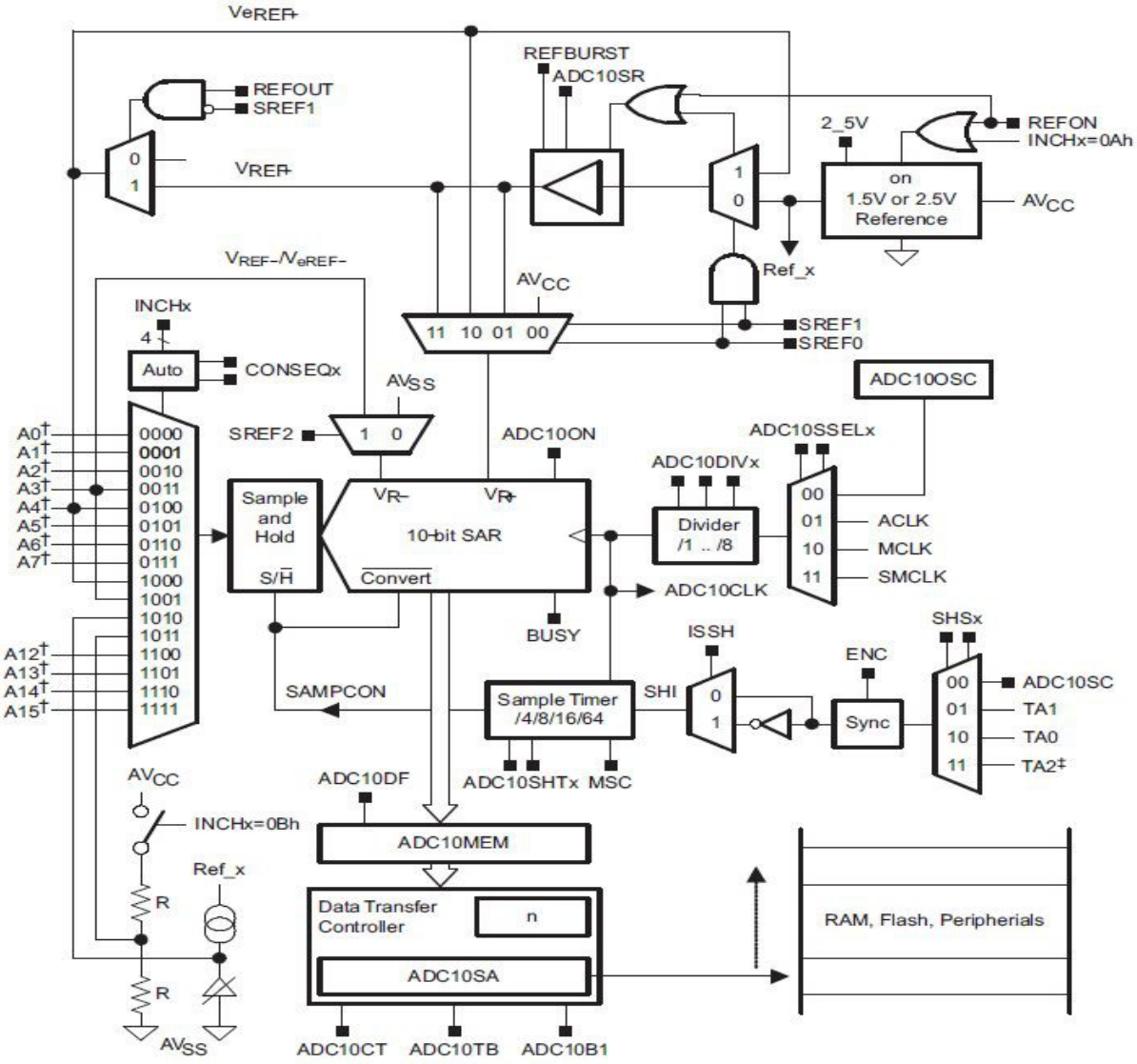
Analog sayısal dönüştürücünün diğer bir önemli özelliği ise AD çevrim süresidir. Genelde ksps, Msps gibi saniyede kaç bin yada kaç milyon çevrim yapabileceklerini belirten değerleri vardır. Aynı şekilde piyasada 200 ksps'den (Saniyede 200bin çevrim) Gsps'lere (Saniyede milyarlarca çevrim) kadar çevrim hızı olan ADCler bulunabilmektedir.

Biz bu modül kapsamında MSP430G2553 denetleyicimiz içerisinde bulunan ADC10 Analog sayısal dönüştürücüsünden bahsedip uygulamalarına değineceğiz.

MSP430 ADC10 Birimi

ADC10, MSP430 denetleyicilerde bulunan 10 bit çözünürlüğe sahip gelişmiş özellikleri bulunan Analog Sayısal Dönüştürücü birimidir. Temel özellikleri aşağıda ki gibidir.

- 200 ksps'ten fazla çevrim hızı
- 10 Bit çözünürlük
- Timer_A tarafından çevrim başlatılabilme
- 1.5V veya 2.5V dahili referans gerilim üreteci
- Yazılımsal seçilebilir dahili, harici referans girişi
- Harici 8 kanal girişi, (MSP430F22xx için 12 kanal)
- Dahili sıcaklık sensörü, VCC, ve harici referans ölçme girişleri
- Seçilebilir saat kaynağı
- Tek kanal, tekrarlı tek kanal, ardışık ve ardışık tekrarlı çevrim seçenekleri
- Güç tasarruf özellikleri
- DTC birimi ile ölçüm sonuçlarını depolayabilme



Şekil 2: ADC10 Blok Diyagramı

Şekil 2'de ADC10 blok diyagramı görülmektedir. Görüldüğü üzere birimin ortasında 10 bit SAR(Successive Approximation Register/Ardışıl Yaklaşım Kaydedici) ADC bulunmaktadır. Onun dışında adcnin giriş kanalları ve referans girişleri görülebilir.

Genel olarak ADC10 birimi harici veya dahili girişlerine uygulanan analog işareti referans gerilimlerini temel alarak 10 bit çözünürlükte sayısal değere dönüştürür. Referans gerilimleri denetleyicinin çalışma gerilim değerlerinin altında yada üstünde olmamalıdır.

10 bit değişken 0-1023 arasında değişen bir değişkeni ifade eder. Örneğin Vref-=0V ve Vref+=2.5V olarak seçersek, 2.5/1023=2.44mV aralıklar ile analog işaretimizi örnekleriz. Sonuç olarak 0-2.5V gerilim değerini 0-1023 arası sayısal değere dönüştürürüz. Bu sayede analog değerimizi işlemcimizin anlayıp işleyebileceği sayısal değere çevirimiş oluruz.

ADC Saat Kaynağı Seçimi: Saat kaynağı seçimi ADC biriminin çevrim hızını belirler. İstediğimiz çalışma hızına göre saat kaynağımızı belirleyebiliriz. ADC10 saat kaynağı olarak ACLK, MCLK, SMCLK ve kendi dahili saat kaynağını kullanabilmektedir. Kendi dahili saat kaynağının yaklaşık 5 MHz çalışma frekansı vardır. Diğer saat kanalları ise Önceki modül 2'de anlatıldığı gibi uygulama ihtiyaçlarına göre belirlenebilir.

Voltaj Referans Üreteci: ADC10 birimi içerisinde 2.5V ve 1.5V hassas çıkış üreten referans üreteçleri bulunmaktadır. ADC birimi işlemlerini referans gerilimlerini temel alarak yaptığı için referans gerilminin doğruluğu çok önemlidir. Örneğin referans olarak denetleyici besleme gerilimi(VCC, ~3.3V) kullanılırsa, devrenin çalışma anında yaşanabilecek gerilim düşmeleri yada yükselmeleri ADC işlemini doğrudan etkiler ve yanlış sonuç almamıza neden olur. Hassas ölçüm gerektirmeyen uygulamalarda bu durum sorun olmasada hassas ölçüm gerektiren uygulamalarda sabit referans kullanmak gereklidir.

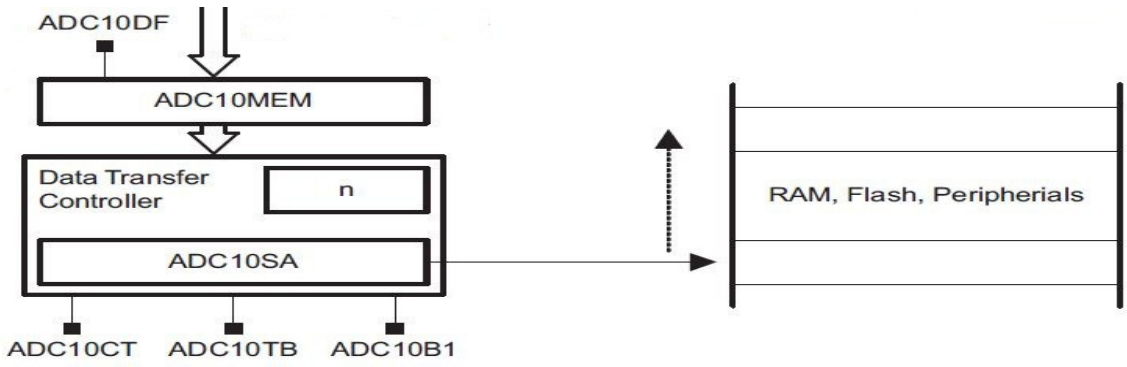
Aynı şekilde harici entegre olarak referans gerilimi üreten entegrelerde bulunmaktadır. ADC10 birimimiz içersinde dahili referans üreteci bulunduğu için fazladan alıp maliyet arttırmaya gerek yoktur.

ADC10 Çevrim Modları: ADC10 birimi uygulamalarınıza esneklik kazandırması açısından farklı çalışma modlarına sahiptir. Bu modlar aşağıdaki gibidir.

1. Tek kanal, tek çevrim: Tek kanalden bir kez çevrim alınır.
2. Ardışıl kanallar tek kanal çevrim: Ardışıl kanallardan bir kez çevrim alınır.
3. Tek kanal sürekli çevrim: Tek kanaldan sürekli olarak çevrim alınır.
4. Ardışıl kanallar sürekli çevrim: Ardışıl kanallardan sürekli olarak çevrim alınır.

Görüldüğü gibi çeşitli çevrim modları bulunmaktadır. Uygulamanıza göre istediğiniz çevrim modunu kullanabilirsiniz.

ADC10 DTC Birimi



Şekil 3: ADC10 DTC Birimi Blok Diyagramı

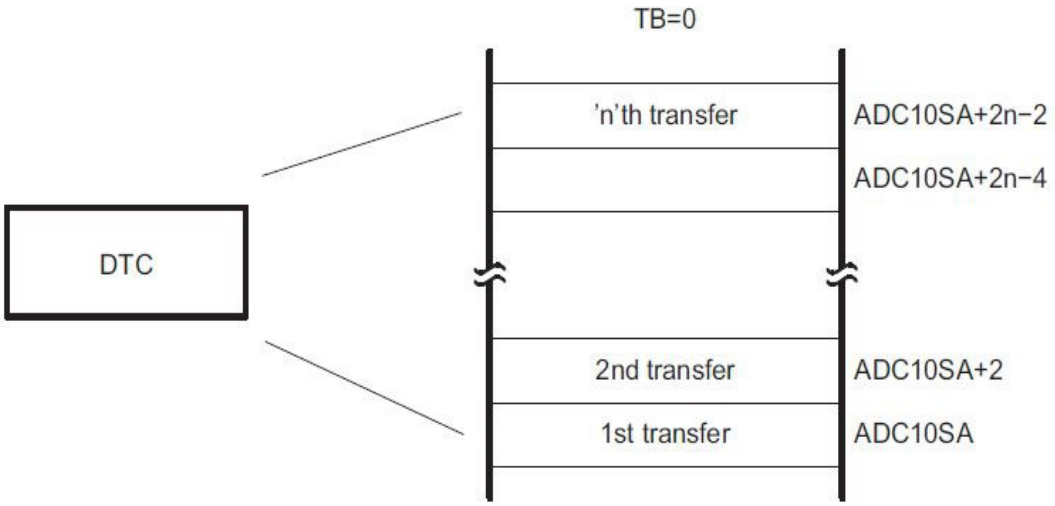
ADC10 biriminin gelişmiş bir özelliğide içerisinde DTC (Data Transfer Controller/ Veri Transfer Denetleyicisi) biriminin olmasıdır. Bu birim sayesinde yaptığımız tekli veya çoklu ölçüm sonuçlarını denetleyicimizin hafızasında saklayabiliriz.

Basitçe çalışmasını analtacak olursak; ADC10 birimi uygun şekilde ölçümler yapmak üzere ayarlanır. Aynı şekilde DTC birimininde istenilen şekilde kaç adet çevrim sonucunu hangi hazfıza bölgesine kayıt edileceği gibi ayarları yapılır. Sonrasında ADC10 biriminin her çevrim işlemini bitirmesi durumunda bu durum DTC birimine bildirilir. DTC birimi elde edilen bu çevrim sonucunu önceden ayarlandığı şekilde istenilen hafıza bölgesinde saklar.

DTC birimi ile tek kanaldan yada ardışık çoklu kanaldan tekli yada sürekli ölçümler yaparak sonuçları denetleyici hafızasında saklayabilirsiniz. Bu sayede denetleyiciyi bu gibi işlemler ile meşgul etmeden hızlı bir şekilde işleminizi gerçekleştirebilirsiniz. Ayrıca DTC birimi trasnfer işlemi bittiğinde ilgili bayrakları set ederek kesme oluşturabilir.

DTC birimi tek blok transfer ve çift blok transfer olmak üzere iki farklı çalışma yapısına sahiptir.

Tek Blok Transfer Modu: ADC10DTC0 kaydedicisinde bulunan ADC10TB biti reset edilerek bu moda geçilir. Bu modda DTC birimi ADC10DTC1 kaydedicisine yazılan değer adedince çevrim sonucunu ADC10SA kaydedicisinde bulunan hafıza adres değerinden itibaren kayıt eder. MSP430 adres haritası byte(8bit) temelli olduğu için 10bit çevrim sonuçları 16 bit olarak saklanır. Bu durumda her çevrim işlemi sonrasında hafıza adres değeri ikişer arttırılır. ADC10DTC1 kaydedicisine 0-255 arası değer yazılabilir. Sıfır yazılması durumunda DTC birimi devre dışı bırakılır.



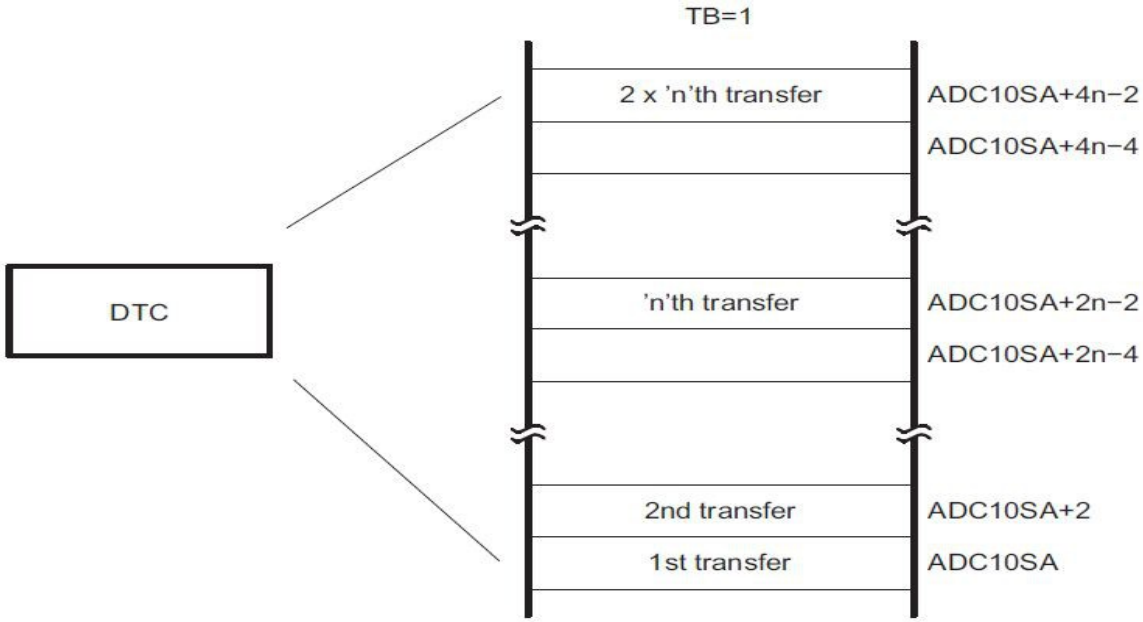
Şekil 4: DTC Birimi Tek Blok Transfer Modu

Şekil 4'te görüldüğü gibi ADC10DTC1 kaydedicisi ile belirtilen n sayıda çevrim sonucu ADC10SA kaydedicisi ile belirtilen adres değerinden ADC10SA+2n-2 değerine kadar belirtilen hafıza bölgesinde saklanır.

Trasnfer işlemi tamamlandığında ADC10IFG bayrağı set edilir. İstenirse kesme oluşturulabilir. Böylece her çevrim işlemi için işlemciyi uğraştırmadan tüm çevrimlerin bitmesini bekleyip, kesme oluşturarak hızlı bir şekilde işlemlerinizi yapabilirsiniz.

Çift Blok Transfer Modu: ADC10DTC0 kaydedicisinde bulunan ADC10TB biti set edilerek bu moda geçilir. Tek blok trasnfer moduna benzer olarak çevrim saklama işlemi 2 blok halinde yapılır. ADC10DTC1 kaydedicisine yazılan n değeri adedince çevrim sonucu ADC10SA değişkeninde belitilen adres değerinden ADC10SA+2n-2 adres değerine kadar olan hafızada saklanır. Birinci blok dolduktan sonra ADC10DTC0 kaydedicisinde bulunan ADC10B1 biti set edilir. Aynı zamanda ADC10IFG bayrağı set edilir. İstenirse kesme oluşturulur.

Birinci blok dolduktan sonra çevrim sonuçları 2.bloğa kaydedilmeye başlanır. Aynı şekilde n adet çevrim sonucu ADC10SA+2n adresiden başlayarak ADC10SA4n-2 adresine kadar kayıt edilir. Aynı şekilde n syaısı 0-255 arasında olmalıdır. İkinci blok dolduktan sonra ADC10DTC0 kaydedicisinde bulunan ADC10B1 biti resetlenir. Aynı zamanda ADC10IFG bayrağı set edilir. İstenirse kesme oluşturulur. İkinci bloğunda dolmasıyla transfer işlemi tamamlanır.



Şekil 5: DTC Birimi Çift Blok Transfer Modu

Şekil 5'te çitf blok transfer işleminin yapısı görülmektedir. Görüldüğü gibi 2n adet ADC sonucunu işlemciyi uğraştırmadan istenilen hafıza bölgesine kayıt edebilirsiniz.

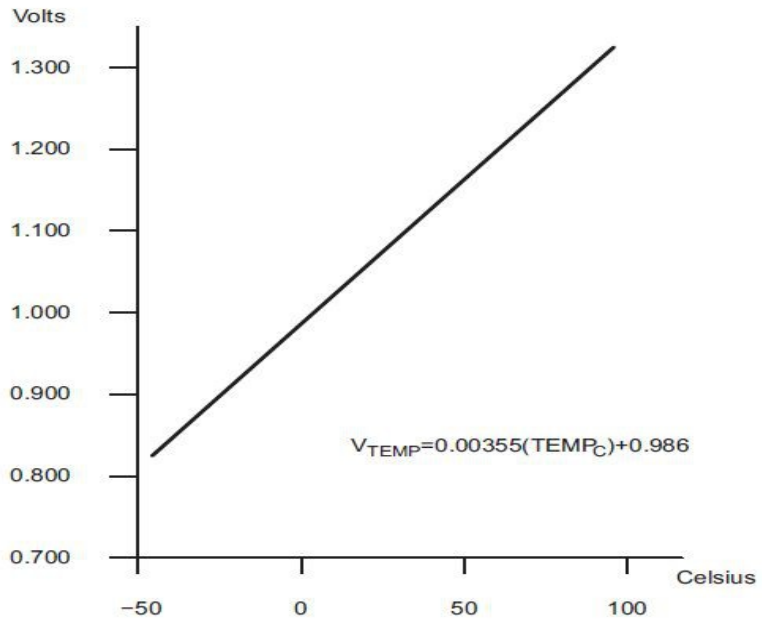
Birinci blok ölçümleri elde edildikten sonra bu ölçümleri işlemek zaman alabilir. Bu durum sonrasında yapılan diğer çevrim sonuçlarını kaybetmemek için yapılan çevrimler 2. bloğa kayıt edilebilir. Bu sayede sürekli ölçüm yapıp işlememiz gereken durumlarda çift blok transfer modu bize kolaylık sağlar.

Sürekli Transfer Modu: ADC10DTC0 kaydedicisinde ki ADC10CT biti set edilerek bu moda geçilir. Tek blok ve çift blok transfer işlemleri bir kez yapıldıktan sonra işlem tamamlanır. DTC biriminin çoklu trasnfer modu ile istenirse sürekli olarak tranfer işlemleri tekrarlanabilir. Yani örneğin tek blok transfer için n adet çevrim yapıldı ve değerler ADC10SA adresinden ADC10SA+2n-2 adresine kadar kayıt edildi. Bu aşamadan sonra eğer sürekli transfer modu aktif ise DTC birimi çalışmaya devam eder ve alınan ölçümleri aynı şekilde tekrar ADC10SA adresinden başlayarak daha önce bulunan ölçüm değerlerinin üstüne yazar. Benzer işlem çift blok transfer işlemi içinde geçerlidir.

Sürekli olarak ölçmeniz gereken giriş işaretleriniz olduğunda bu moddan yararlanılabilir. Böylece sürekli olarak girişe gelen işaretler adc ile sayısal değere dönüştürülüp işlemci uğraştırılmadan istenilen hafıza bölgesine sürekli olarak kayıt edilir. Uygulamanızın çalışmasına göre bu işaretler zamanında işlenerek gereken işlemler yaptırılabilir.

ADC10 Dahili Sıcaklık Sensörü

ADC10 biriminin bir diğer özelliğide içerisinde dahili sıcaklık sensörü bulunmasıdır. Bu sensör denetleyicinin içerisinde yer almaktadır. Normal çalışmada ADC10 ile A10 kanalı okunarak sıcaklık değeri elde edilebilir. Dahili sıcaklık sensörü kullanıldığında otomatik olarak dahili referans üreteci aktif edilir. Sıcaklık sensörüne ait sıcaklık-gerilim grafiği şekil 6'da görülmektedir.



Şekil 6: Dahili Sıcaklık Sensörü Tipik Sıcaklık-Gerilim Grafiği

Şekil 6'da görüldüğü gibi dahili sıcaklık sensörü algıladığı sıcaklık değerine göre değişken bir analog gerilim üretmektedir. Sıcaklık sensörünün grafiği doğrusal olduğu için sıcaklık değerine göre elde edilen gerilim değeri V_{temp}=0.00355($TEMP_C$)+0.986 bağıntısıyla bulunabilir. Bu gerilim değeri ADC birimi ile sayısal değerlere çevrilip uygulamalarda kullanılabilir.
Dahili sıcaklık sensörü yeterince hassas olmadığından hassas sıcaklık ölçümü gerektiren yerlerde kullanımı uygun değildir. Basit ortam sıcaklıklarının yada işlemci sıcaklığının ölçülmesi gibi temel işlemlerde kullanılabilir.
ADC10 biriminin tüm ayarlamaları v.s. ADC10 ile ilgili kaydediciler ile yapılır. Bu kaydediciler ile ilgili detaylı bilgi referans kılavuzu okunarak elde edilebilir. Uygulamalarda yeri geldikçe gerekli kaydedici ayarları yapılarak analog sayısal çevrim işlemleri yapılacaktır.
Bu kadar teorik anlatımdan sonra ADC10 birimi ile ilgili uygulamalarımıza geçelim.

Uygulama 6.1 ADC10 Birimi ile Ölçüm Uygulaması

Bu uygulamada ADC10 biriminin temel çalışmasına örnek olması için A0 kanalından ölçüm yapıp ölçülen değeri, sayısal değer ve gerilim olarak ifade edip LCD ekrana yazdıracağız. Uygulamanın kodları aşağıdaki gibidir.

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

#include "LCD.h"                       // LCD başlık dosyası
void BCDCevir(unsigned short);         // Fonksiyon prototipi.
unsigned char bBCDDeger[4];            // BCD değerleri tutan dizi
unsigned short wADCHam, wADCVolt;      // Ham ve Volt değerlerini tutan değişkenler
```

```
void main(void) {
   WDTCTL = WDTPW | WDTHOLD;              // Watchdog timeri durdur.
   LCD_Ayarla();                          // Başlangıçta LCD ayarlarını yap.
   ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE; // ADC ayarları, kesme aktif
   ADC10CTL1 = INCH_0;                    // A0 girişini seç
   ADC10AE0 |= BIT0;                      // PA.0 ADC özelliğini aktif et
   _BIS_SR(GIE);                          // Kesmeleri aç
   while(1){                              // Sonsuz döngü
   ADC10CTL0 |= ENC + ADC10SC;            // AD çevrimi başlat
   _BIS_SR(LPM0_bits);                    // Uykuya gir
   wADCHam = ADC10MEM;                    // Çevrim sonucunu kaydet.
   wADCVolt = (wADCHam*10)/31;            // Çevrim sonucunu volta çevir.
   LCD_Git_XY(1,1);                       // İmleci 1.satır 1.sütuna götür.
   LCD_Yazi_Yaz("ADC Ham=");              // Ekrana ADC Ham= yazdır.
   BCDCevir(wADCHam);                     // Ham değeri BCD'ye çevir
   LCD_Karakter_Yaz(bBCDDeger[3]);        // Binler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[2]);        // Yüzler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[1]);        // Onlar basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[0]);        // Birler basamağını yazdır.
   LCD_Git_XY(2,1);                       // İmleci 2.satır 1.sütuna götür.
   LCD_Yazi_Yaz("Volt=");                 // Ekranda Volt= yazdır.
   BCDCevir(wADCVolt);                    // Volt değerini BCD'ye çevir
   LCD_Karakter_Yaz(bBCDDeger[2]);        // Birler basamağını yazdır.
   LCD_Karakter_Yaz('.');                 // Nokta karakteri yazdır.
   LCD_Karakter_Yaz(bBCDDeger[1]);        // Onda birler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[0]);        // Yüzde birler basamağını yazdır.
   }
}
// 0-9999 Arası girilen sayıyı BCD'ye çevirip dizide saklayan fonksiyon.
void BCDCevir(unsigned short Sayi){
        unsigned char bSayac;             // Sayaç değişkeni.
        for(bSayac=0;bSayac<4;bSayac++){
        bBCDDeger[bSayac] = Sayi%10 + 0x30;
        Sayi /= 10;
        }
}

// ADC10 kesme vektörü
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);      // İşlemciyi uykudan uyandır.
}
```

Kodların çalışmasından bahsedelim. Başlangıçta ilk ayarlar ve lcd ayarları yapılır. LCD ayarlarından sonra ADC10 birimi tek kanaldan(A0) her tetiklemede tek çevrim yapmak üzere ayarlanır ve ADC10 kesmesi aktif edilir. Gerilim referansı olarak denentleyici beslemesi(3.3V) seçilir ve ADC dahili osilatörü ile çalışır. Sonrasında işlemci sonsuz döngüye girer. Sonsuz döngüde ise AD çevrim başlatılarak işlemci uyku moduna sokulur ve genel kesmelere izin verilir. AD çevrim bittiğinde ADC kesmesi oluşur. Kesme rutininde işlemci kesmede uyandırılır ve sonsuz döngüde kaldığı yerden devam eder.

AD çevrimi bitip işlemci uyandıktan sonra elde edilen çevrim değeri ADCHam ve ADCVolt olmak üzere 2 değişkende saklanır. ADC ham direk olarak okunan değerdir. ADC volt değeri ise okunan değerin 3.1'e bölünmüş halidir. 3.1 sayısı 0-1023 olan adc çevrim değerini 0-3.30V (0-330)gerilim değerine dönüştürmek için gerekli kat sayıdır. Uygulamada float sayılarla işlem yapmak yerine sayı önce 10 ile çarpılarak 31'e bölünmüştür. Böylece gerilim değeri elde edilir. Float sayılarla işlem hem fazla kod boyutu, hemde işlem hızı gerektirdiği için basit uygulamalarınızda bu şekilde bir yöntem uygulayabilirsiniz.

Ham ve Volt cinsinden değerler elde edildikten sonra bu değeler LCD ekranda görüntülenerek kullanıcıya gösterilir. Sonrasında işlemci tekrar AD çevrimi başlatarak uyku moduna girer. Sonsuz döngü böyle devam eder.

Şekil 7: Uygulama 6.1 Çalışma Görüntüleri

Şekil 7'de uygulamaya ait çalışma görüntüleri görülmektedir. Uygulama çalıştırmadan önce geliştirme kartımızın ayarlarını uygulamaya göre yapalım. LCD kullanımı için sw4, sw15 anahtarlarını off sw2 anahtarlarını ise on konumuna getirelim. Kart üzerinden bulunan RV1 trimpotu ile A0 girişimize ayarları gerilim verebilmek için jp8 atlamasını P1.0 konumuna jp7 atlamasını ise P1.0 olmayan herhangi bir konuma getirelim. Ayrıca kullanılmayan diğer birimlerin anahtalarının off yapılmasında fayda var.

Gerekli ayarları yaptıktan sonra debug işlemi ile kodlarımızı kartımıza yükleyelim. Sorunsuz bir şekilde yükledikten sonra ekranda o anki okunan adc ham ve volt değerinin görünmesi lazım. İnce uçlu bir düz tornavida ile VR1 trimpot direncini çevirerek ham ADC ve gerilim değerlerininin değiştiğini görebilirsiniz.

ADC ölçümü hızlı bir şekilde sürekli olarak yapıldığı için besleme kartında oluşan küçük değişimler çevrim sonucunu etkileyebilir. Bu nedenle trimpot değeri sabit olsa bile okunan ham ve dolayısıyla volt değerinin en düşük değerli basamağında hızlı değişimler görebilirsiniz.

Uygulamada diğer bir dikkat edilmesi gereken diğer bir husus ise referans voltajının Vcc yani yaklaşık 3.3V kabul edilmesidir. Besleme gerilimi zaman içerisinde 3V yada 3.6V gibi değerler arasında olabilir. Biz besleme gerilimini 3.3V kabul edip işlemlerimizi yaptığımızdan bu durumun farkında olup ona göre hata payını kabul edip işlemlerimizi yapmalıyız. Hassas ölçüm gereken uygulamalarda ise tabiki dahili referans üretecini kullanmakta fayda var.

Uygulama 6.2 ADC10 DTC Modu Kullanımı

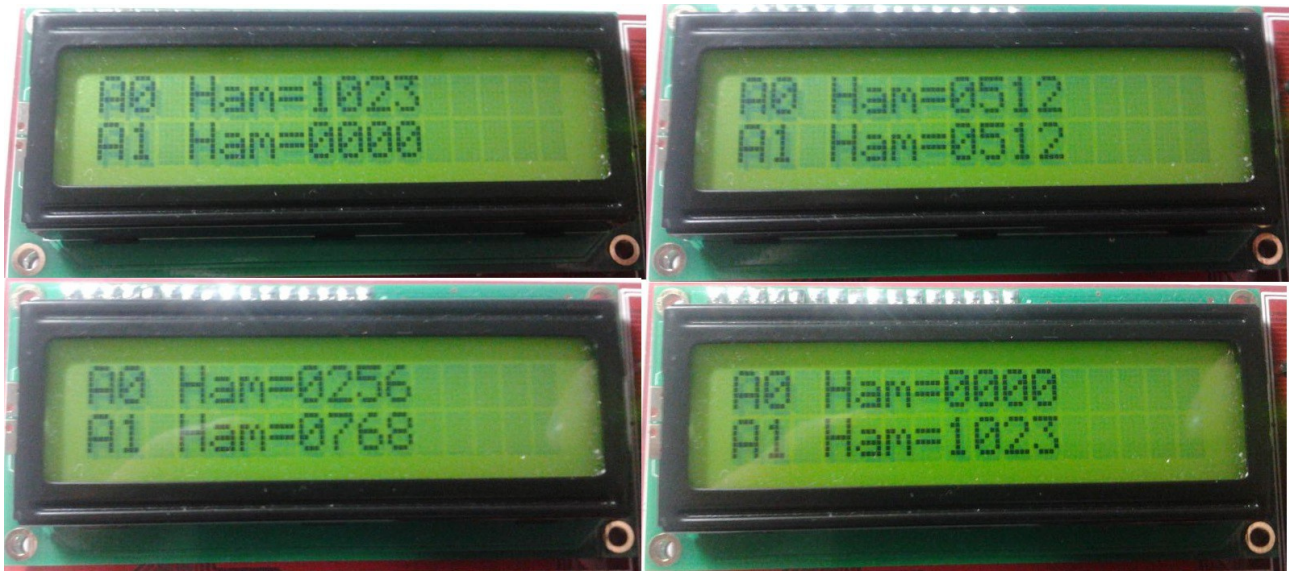
Bu uygulamamızda ADC10 birimi ile DTC birimini birlikte kullanıp A0 ve A1 kanallarını ardışıl okuyarak çoklu okuma yapacağız. Okunan bu değerler ham olarak LCD ekranda görüntülenecetir. Uygulama kodları aşağıda ki gibidir.

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

void BCDCevir(unsigned short);              // Fonksiyon prototipi.
unsigned char bBCDDeger[4];          // BCD değerleri tutan dizi
unsigned short wADCHam[2];
void main(void) {
   WDTCTL = WDTPW | WDTHOLD;     // Watchdog timeri durdur.
   LCD_Ayarla();                 // Başlangıçta LCD ayarlarını yap.
   ADC10CTL1 = INCH_1 + CONSEQ_3;        // A1-A0, tekrarlı çoklu kanal
   ADC10CTL0 = ADC10SHT_2 + MSC + ADC10ON + ADC10IE; // Kesme aktif, çoklu
çevrim aktif
   ADC10AE0 = BIT0 + BIT1;       // A0, A1 ADC özelliğini aktif et
   ADC10DTC1 = 0x02;             // Toplamda her seferinde 2 çevrim yapılacak
   _BIS_SR(GIE);                 // Kesmeleri aç
   while(1){                     // Sonsuz döngü
   ADC10CTL0 &= ~ENC;            // ADC kapalı
   while (ADC10CTL1 & BUSY);     // ADC10 meşgul iken bekle
   ADC10SA = (unsigned int)&wADCHam;     // Çevrim kayıt başlangıç adresi
   ADC10CTL0 |= ENC + ADC10SC;   // ADC10'u aç çevrimi başlat
   _BIS_SR(LPM0_bits);           // Uykuya gir
   LCD_Git_XY(1,1);              // İmleci 1.satır 1.sütuna götür.
   LCD_Yazi_Yaz("A0 Ham=");      // Ekrana A0 Ham= yazdır.
   BCDCevir(wADCHam[1]);         // A0 Ham değerini BCD'ye çevir
   LCD_Karakter_Yaz(bBCDDeger[3]);  // Binler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[2]);  // Yüzler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[1]);  // Onlar basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[0]);  // Birler basamağını yazdır.
   LCD_Git_XY(2,1);                      // İmleci 2.satır 1.sütuna götür.
   LCD_Yazi_Yaz("A1 Ham=");      // Ekrana A1 Ham= yazdır.
   BCDCevir(wADCHam[0]);                 // A1 Ham değerini BCD'ye çevir
   LCD_Karakter_Yaz(bBCDDeger[3]);  // Binler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[2]);  // Yüzler basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[1]);  // Onlar basamağını yazdır.
   LCD_Karakter_Yaz(bBCDDeger[0]);  // Birler basamağını yazdır.
   }
}

// 0-9999 Arası girilen sayıyı BCD'ye çevirip dizide saklayan fonksiyon.
void BCDCevir(unsigned short Sayi){
        unsigned char bSayac;                // Sayaç değişkeni.
        for(bSayac=0;bSayac<4;bSayac++){
        bBCDDeger[bSayac] = Sayi%10 + 0x30;
        Sayi /= 10;
        }
}
```

```
// ADC10 kesme vektörü
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // İşlemciyi uykudan uyandır.
}
```

Kodların çalışmasını açıklayalım. Başlangıçta ilk ayarlar ve lcd ayarları yapılır. Sonrasında ADC10 biriminin A0 ve A1 girişi DTC birimi ile tekrarlı okunmak üzere ayarlanır. Ayarlama işleminden sonra sonsuz döngüye girilir.

Sonsuz döngüde AD çevrim başlatılarak kesmeler açılır uykuya girilir. Kesme işlemi gerçekleştiğinde çevrim işlemi tamamlanmış ve işlemci uykudan uyanmış olur. İşlemci uykudan uyanır ve sonsuz döngüde kaldığı yerden devam eder. Devamında işlemci sırasıyla okunan çevrimleri BCD sayıya çevirerek LCD ekranda yazdırır.



Şekil 8: Uygulama 6.2 Çalışma Görüntüleri

Şekil 8'de uygulamaya ait çalışma görüntüleri görülmektedir. Geliştirme kartı ayarlarını bir önceki uygulama 6.1 ile aynı yapalım. Ayrıca RV2 trimpotunu A1 kanalına bağlamak için jp7 atlamasını P1.1 konumuna getirelim.
Uygulama kodlarını sorunsuz bir şekilde derleyip debug işlemi ile kartımıza yükledikten sonra ölçüm sonuçlarını ekranda görüntülenecektir.
VRV1 trimpotu ile A0 kanalında ki gerilimi değiştirebiliriz. Aynı şekilde VR2 trimpotu ile de A1 kanalında ki gerilimi değiştirebiliriz. Bu sayede girişlerde ki gerilimleri trimpotlar ile değiştirip, değişimi LCD ekranda gözlemleyebiliriz.
DTC modülü ile çoklu ADC işlemlerini daha az işlem gücü ile yapıp daha verimli uygulamalar yapabilirsiniz.

Uygulama 6.3 ADC10 ile Besleme Gerilimi Ölçümü

Bu uygulamamızda ADC10 birimi ile denetleyicimizin besleme gerilimini izleyeceğiz. İzlediğimiz sonuçları 7 parça göstergelerde göstereceğiz.
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
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 short wAVCC=0;               // AVCC gerilim değerini tutan değişken
unsigned char bGecikmeSayac=0;        // Gecikme sayacı
// 7 Parça gösterge porta yazılan karakter değerleri 0-9, A-F karakterleri ve U harfi
const unsigned char
SegmentDataTablo[17]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X77,
0X7C,0X39,0X5E,0X79,0X71,0x3E};
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
   ADC10CTL1 = INCH_11;              // AVcc/2 dahili kanalını seç
   // ADC10 ayarları, dahili 2.5V referans, kesmeleri aç
   ADC10CTL0=SREF_1 + ADC10SHT_2 + REFON + REF2_5V + ADC10ON + ADC10IE;
```

```
    bNokta = BIT0;                              // Gösterge 1 noktasını yak
    bGostergeData[3] = 16;                      // Gösterge 4'te U(Volt) değerini göster
    ADC10CTL0 |= ENC + ADC10SC;                 // AD çevrimi başlat
    _BIS_SR(GIE);                                       // Kesmeleri aç
    while(1){                                           // Sonsuz döngü
    _BIS_SR(LPM0_bits);                                 // Uykuya gir
    wAVCC = (ADC10MEM*20)/41;                   // Okunan değeri gerilime çevir
    BCDCevir(wAVCC);       // Gerilim değerini BCD'ye çevir gösterge değişkenlerine yaz.
}}
// 0-999 Arası girilen sayıyı BCD'ye çeviren fonksiyon
void BCDCevir(unsigned short Sayi){
        unsigned char bSayac;
        for(bSayac=0;bSayac<3;bSayac++){
        bGostergeData[2-bSayac]= Sayi%10;
        Sayi /= 10;
}}
// ADC10 kesme vektörü
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // İşlemciyi uykudan uyandır.
}
// 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
        ADC10CTL0 |= ENC + ADC10SC;         // AD çevrimi başlat
        }
        TA0CCR0 += 5000;                    // Timeri yeniden kur.
}
```

Kodların çalışmasını inceleyelim. Uygulamada 7 parça gösterge kullanıldığı için gösterge kısmı uygulama 5.2 ile benzerdir. Başlangıçta ilk ayarlar ve gösterge ayarları yapılır. Sonrasında ADC10 birimi dahili olarak AVCC/2 kanalını okumak için ayarlanır. Dahili 2.5V referans gerilimi kullanılır ve ADC10 kesmesine izin verilir. Sonrasında gösterge 1'in nokta segment değerini tutan değişken set edilir ve Gösterge 4 değişkenine U karakteri yazılır. Sonrasında işlemci AD çevrimini başlatarak sonsuz döngüye girer.
Sonsuz döngü içerisinde işlemci uyku moduna girer. Bu sırada her 5 ms de bir göstergelerin yenilenmesi yapılır. Aynı zamanda her 500ms de bir AD çevrim tekrar başlatılır. AD çevrim bitince işlemci uykudan uyandırılır. Elde edilen çevrim değeri gerilim değerine çevrilerek göstergelerde görüntülenir.
Dikkat edilmesi gereken önemli bir nokta, çevrim değeri 20/41 kat sayısı ile çarpılarak 0-499.5 yani yaklaşık 0-500 arası çevrim değerleri elde edilir. Bu değer 1. basamaktan itibaren nokta kayarak yazıldığında 0.00-5.00 sayısı elde edilir. Yani referans geriliminin 2 katına kadar olan gerilim değerlerini ölçüp göstergede gösterebiliriz. Bu durum uygulamanın kart üzerinde çalıtırılması ile daha iyi anlaşılabilir.

Şekil 9: Uygulama 6.3 Çalışma Görüntüleri

Şekil 9'da uygulama 6.3'e ait çalışma görüntüleri görülmektedir. Uygulamayı kartımıza yüklemeden önce sw1 anahtarlarını on, sw2, sw4, sw15 anahtarlarını off, konumuna getirmeniz gerekmektedir. Ayrıca port pinlerini etkilememesi için jp7, jp8 atlamalarını boşta bırakmamız gerekmektedir.

Kodlarımızı sorunsuz bir şekilde derleyip kartımıza yükledikten sonra o anki denetleyici besleme gerilim değerinin göstergelerde görüntülenmesi gerekmektedir.

Şekil 9'da görüldüğü gibi besleme gerilimimiz 3.51V, 3.52V değerleri arasında değişmektedir. Yani besleme gerilimimiz tam olarak 3.3V değildir. Eğer besleme gerilimin 3.3V kabul edip uygulamalarımızda kullansaydık 200mv kadar bir hatamız olacaktı.

Göstergelerin sınırlı sayıda karakterleri gösterebildiğinden bahsetmiştik. Şekil 9'da görüldüğü gibi 4. gösterge de Volt birimini simgelemek için V harfi oluşturamadığımız için onun yerina U harfi kullanılmıştır. Fakat aynı zamanda 7 parça gösterge ile karakter lcdye göre daha büyük ve parlak bir görünüm elde edilmiştir.

ADC10 biriminin besleme gerilimi ölçme özelliği, pil veya batarya ile çalışan devrelerde besleme gerilimini kontrol edip gerilim düşmeleri, pil bitmesi gibi durumlarda uyarı veya hata vermek amacı ile kullanılabilir.

Uygulama 6.4 ADC10 Dahili Sıcaklık Sensörü Ölçümü

ADC10 birimi ile ilgili son olarak dahili sıcaklık sensörü ölçüm uygulamasını yapacağız.

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
unsigned char bGecikmeSayac=0;                       // Gecikme sayacı
unsigned long      wSicaklik;         // Sıcaklık değerini tutan değişken
// 7 Parça gösterge porta yazılan karakter değerleri 0-9,A-F karakterleri ve derece sembolü
const unsigned char
SegmentDataTablo[17]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X77,
0X7C,0X39,0X5E,0X79,0X71,0x63};

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
    ADC10CTL1 = INCH_10 + ADC10DIV_3;   // Sıcaklık sensörü seçilir. ADC10CLK/4
    // ADC10 ayarları, dahili 1.5V referans, kesmeleri aç
    ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE;
    bNokta = 0;                                 // Gösterge noktalarını söndür.
    bGostergeData[2] = 16;                      // Gösterge 3'te derece sembolünü göster.
    bGostergeData[3]= 0x0C;                     // Gösterge 4'te C harfini göster.
    ADC10CTL0 |= ENC + ADC10SC;                 // AD çevrimi başlat
    _BIS_SR(GIE);                                       // Kesmeleri aç
    while(1){                                           // Sonsuz döngü
    _BIS_SR(LPM0_bits);                                 // Uykuya gir
    wSicaklik = ADC10MEM;                       // Çevrim sonucunu kaydet
    wSicaklik = ((wSicaklik - 673) * 423) / 1024; // Çevrim sonucunu santigrat dereceye
çevir
    wSicaklik %=100;                            // Sonucun 0-99 arasında olmasını sağla
    bGostergeData[0]= wSicaklik/10;             // Onlar basamağını gösterge 1'de göster.
    bGostergeData[1]= wSicaklik%10;             // Birler basamağını gösterge 2'de göster.
}}

// ADC10 kesme vektörü
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // İşlemciyi uykudan uyandır.
}
// 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
        ADC10CTL0 |= ENC + ADC10SC;             // AD çevrimi başlat
        }
        TA0CCR0 += 5000;                        // Timeri yeniden kur.
}
```

Uygulamanın kodları bir önceki uygulama 6.3 ile benzerdir. Başlangıçta aynı şekilde ilk ayarlar ve ADC10 ayarları yapılır. Bu sefer AD çevrim için A10 dahili sıcaklık sensörü giriş olarak seçilir. Sonrasında sonsuz döngüye girilir.
Uygulama 6.3 ile aynı şekilde 500 ms de bir ölçüm yapılır. Ölçülen değer sıcaklık değerine dönüştürülür. Dönüştürülen değer BCD olarak basamaklarına ayrılarak gösterge 1 ve gösterge 2'de görüntülenir. Ayrıca gösterge 3'te santigrat işareti gösterge 4'te ise C karakteri gösterir. Böylece göstergede gösterilen değerin santigrat cinsinden sıcaklık bilgisi olduğu anlaşılır.

Şekil 10: Uygulama 6.4 Çalışma Görüntüleri

Şekil 10'da uygulama 6.4'e ait çalışma görüntüleri görülebilir. Geliştirime kartımızı bir önceki uygulamada yaptığımız şekilde ayarlamamız gerekmektedir.
Uygulama kodlarını derleyip sorunsuz bir şekilde derledikten sonra göstergelerde o anki ölçülen sıcaklığın görüntülenmesi gerekmektedir.
Görüldüğü gibi basit bir termometre yaptık. Göstergelerimiz kısıtlı olduğu için termometremiz sadece 0-99 santigrat derece arasında pozitif sıcaklıkları gösterebilir.

Bu modül kapsamında MSP430G2553 denetleyicisi içerisinde bulunan ADC10 biriminin anlatımı ve uygulamaları yapılmıştır. Görüldüğü üzere ADC10 birimi bir çok özelliğie sahip yetenekli bir ADC birimidir. Bu özellikleri ve örnek uygulamalardan faydalanılarak çok çeşitli ADC uygulamaları yapılabilir.