Modül 8: MSP430 ile DS18B20 Sıcaklık Sensörü Kullanımı ve Uygulamaları

Giriş

Sıcaklık sensörleri gerek elektronik sistemlerde gerekse genel olarak uygulamalarda sıkça kullanılan sensörlerdir. Sıcaklık sensörleri fiyatı, kullanım alanı ve sağlaması gereken koşullara göre çok çeşitte olabilmektedir. Sıcaklık sensörleri çıkışına göre analog çıkışlı ve dijital çıkışlı olmak üzere ikiye ayrılmaktadır. Aynı zamanda piyasada ölçüm aralığı, çözünürlüğü, kararlılığı ve tepki süresi gibi parametrelerine göre bir çok firmanın çok çeşitte sıcaklık sensörleri bulunmaktadır.

Temel sıcaklık sensörleri termistörlerdir.(NTC, PTC) Bilindiği gibi termistörlerin direnci üzerine düşen sıcaklığa göre değişmektedir. Bu sıcaklık değişimi sayesinde termistörler ile basit devreler kullanılarak ADC birimi ile sıcaklık ölçümü yapılabilir. Hatırlayacak olursanız MSP430 denetleyicilerimiz içerisinde bulunan dahili sıcaklık sensörüde ADC ile birlikte kullanılarak sıcaklık ölçümü yapılmaktadır.

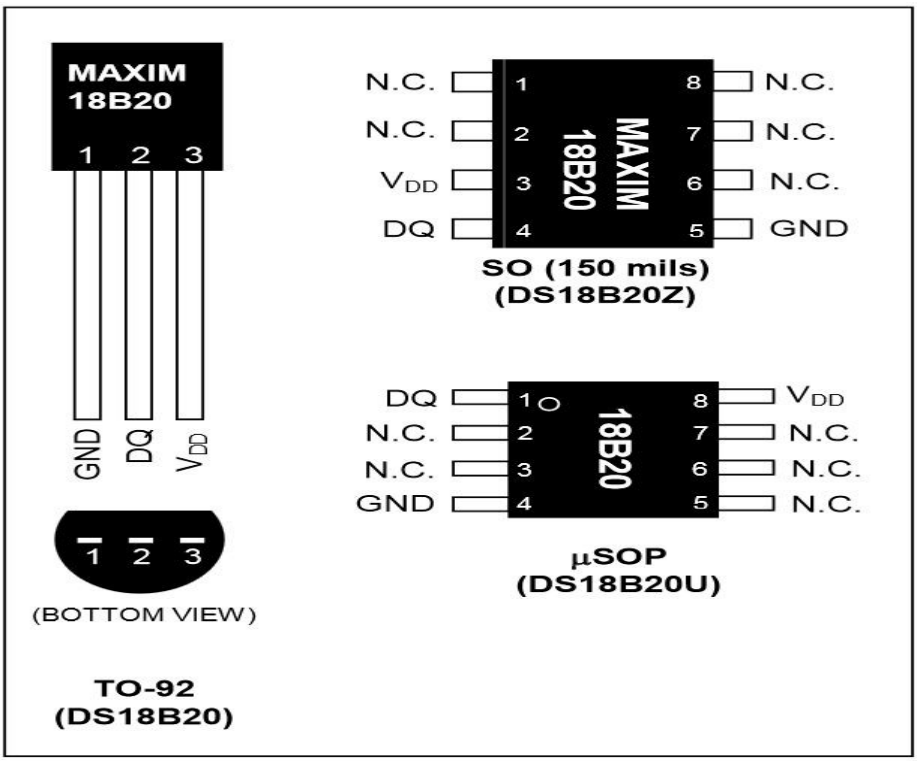
Sıcaklık ölçümü gereken uygulamalar yaparken sistemin gereksinimleri göz önünde bulundurularak uygun sensörün seçilmesi gerekir. Sensör kalitesinin artması maliyetide arttıracağından uygulamanın gereksinimlerini ne eksik ne fazla tam olarak sağlayan sensörleri seçmek yerinde olacaktır. Örneğin basit bir oda termometresi için MSP430 denetleyicimiz içerisinde bulunan sıcaklık sensörü kullanılabilir. Fakat hassas ölçüm gerektiren örneğin medikal bir sistem için daha kaliteli sensörler kullanmak gereklidir.

Biz bu modül kapsamında geliştirme kartımız üzerinde bulunan DS18B20 dijital sıcaklık sensörünü inceleyip uygulamalarını yapacağız. Böylece birden fazla sıcaklık sensörleri ile uygulama yapmış, farklı sensörler arasında ki farklılıkları kavramış olacağız.

DS18B20 Sıcaklık Sensörü

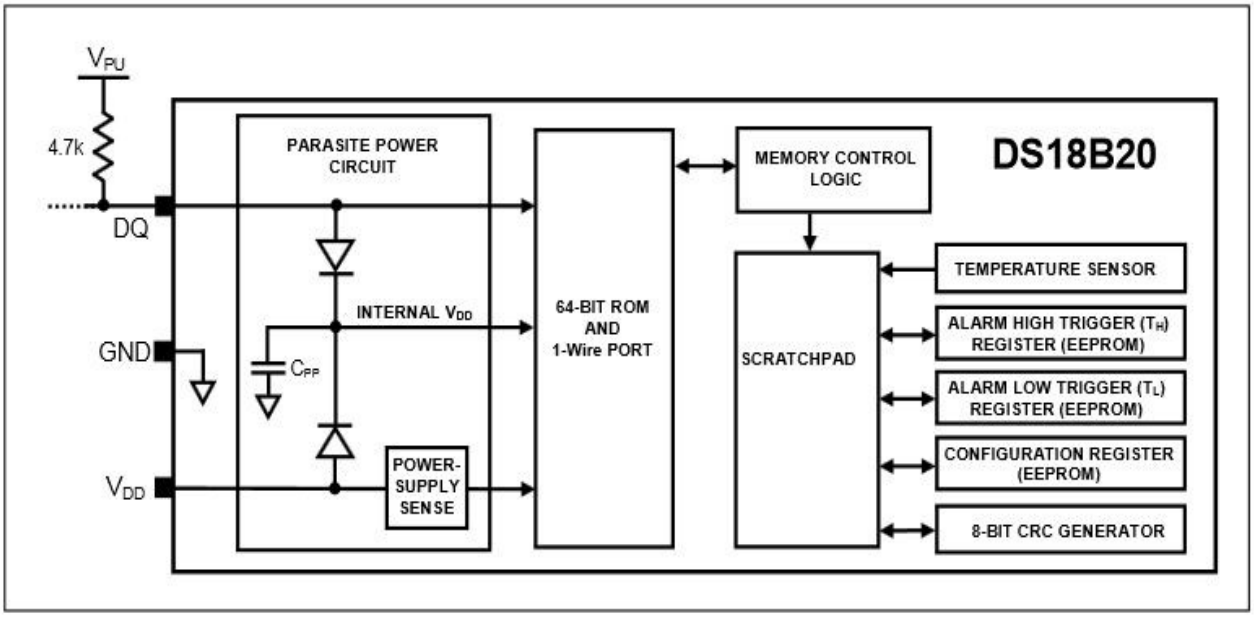
DS18B20 maxim integrated firmasının ürettiği programlanabilir çözünürlükte dijital çıkışlı sıcaklık sensörüdür. DS18B20 -10°C +85°C arasında ±0.5°C tölerans ile sıcaklık ölçümü yapabilir. DS18B20'nin özellikleri aşağdaki gibidir.

- Benzersiz 1-Wire® arayüzü ile haberleşme için sadece bir port pini gereklidir.
- Her sensör kendine ait benzersiz 64 bit seri numarasına sahiptir.
- Harici eleman gerektirmez.
- Data hattı üzerinden beslenebilir.
- Ölçüm aralığı -55°C to +125°C (-67°F to +257°F)
- -10°C +85°C arasında ±0.5°C tölerans
- Kullanıcı tarafından programlanabilir 9-12 bit ölçüm çözünürlüğü
- Azami 750 ms çevrim süresi(12 bit)
- Kullanıcı tanımlı silinmez alarm ayarları
- 8-Pin SO (150 mils), 8-Pin µSOP, 3-Pin TO-92 paket desteği
- DS1822 ile yazılım uyumluluğu



Şekil 1: DS18B20 Paket Yapıları ve Görünüşleri

Görüldüğü gibi DS18B20 bir çok özelliğe sahip gelişmiş bir sıcaklık sensörüdür. Hassas ölçüm gerektiren uygulamalarda, endüstriyel sistemlerde DS18B20 sensörü kullanılabilir. Şekil 1'de DS18B20'nin paket yapıları ve görünüşleri görülmektedir. Geliştirme kartımız üzerinde U4 referanslı entegre olarak TO-92 kılıflı DS18B20 bulunmaktadır.

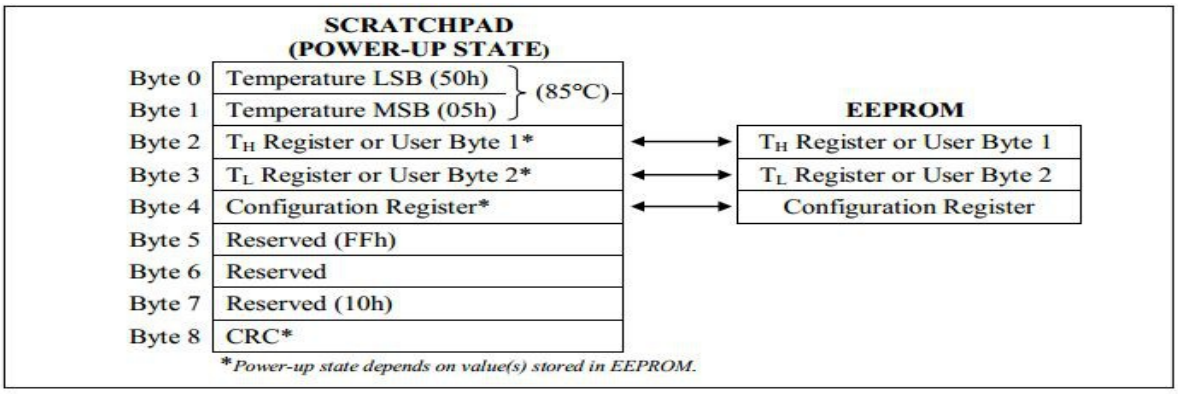


Şekil 2: DS18B20 İç Yapısı

Şekil 2'de DS18B20 sensörünün iç yapısı görülmektedir. DS18B20 sensörününün diğer bir özelliği ise maxim integrated firmasına ait 1-Wire haberleşme protokolü ile haberleşmesidir. Bu protokol maxim integrated firmasının ürünlerinde kullanılmakta olan asenkron bir haberleşme protokolüdür. 1-wire iletişimde haberleşme için adı üstünde tek port pini yeterlidir. Bu özellik ile port pini sınırlı olan uygulamalarda avantaj sağlanabilir. DS18B20 ile uygulamalar yapmadan önce 1-Wire iletişmini incelemekte fayda var.

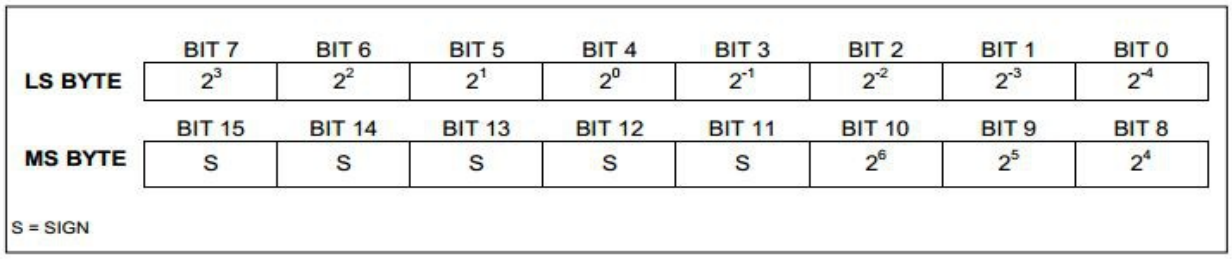
DS18B20 Sıcaklık Bilgisi Kaydedicileri ve Veri Formatı

DS18B120 sensöründen sıcaklık bilgisi 2byte halinde dijital çıkış olarak 1-wire haberleşmesi ile iletilmektedir. Ölçülen sıcaklık değerleri DS18B20'nin içerisinde bulunan hafıazanın (Scratchpad) ilk 2 bytelık bölgesinde yer alır.



Şekil 3: DS18B20 Hafıza(Scratchpad) Yapısı

Şekil 3'te görüldüğü gibi sıcaklık bilgisinin düşük değerli bytei 0. yüksek değerli bytei ise 1. adreste bulunmaktadır. DS18B20 ayrıca alarm ve konfigürasyon değerlerini saklamak için 3 byte eeprom hafızaya sahiptir.



Şekil 4: DS18B20 Sıcaklık Veri Formatı

Şekil 4'te DSB18B20'nin sıcaklık veri formatı görümektedir. Görüldüğü gibi 16 bit olarak saklanan sıkcalık değerinin en yüksek değerlikli 5 biti işaret bitleridir. 4-10 Bitleri sıcaklık değerinin tam sayı kısmını 0-3 bitleri ise ondalıklı kısmını tutarlar.

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

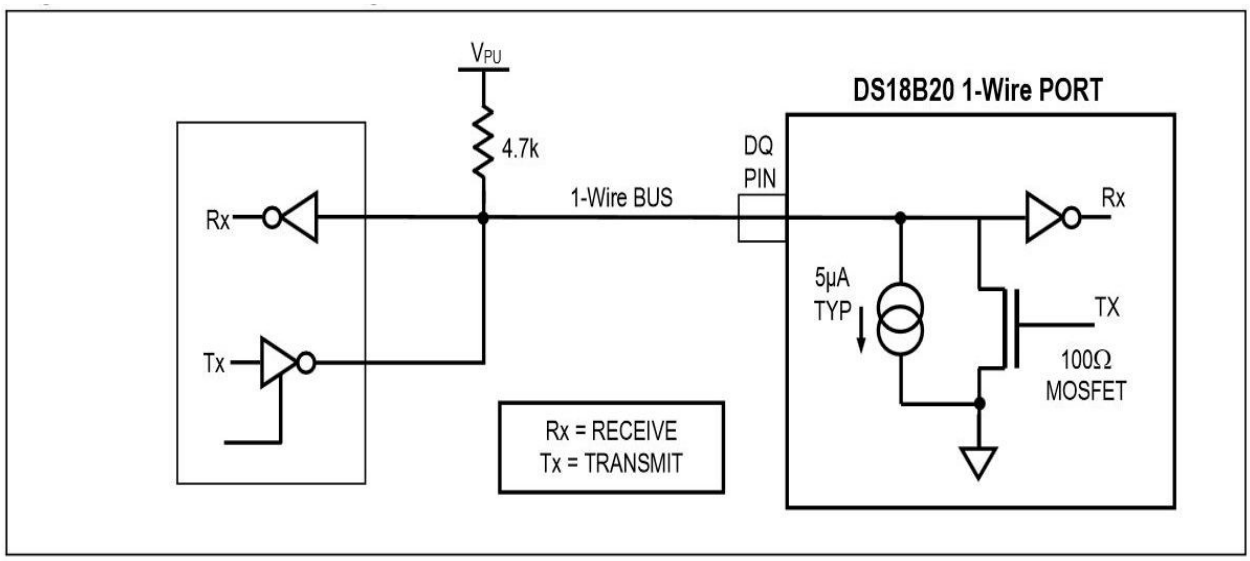
Şekil 5: DS18B20 Örnek Sıcaklık Değerleri

Şekil 5'te örnek sıcaklık değerleri görülmektedir. Görüldüğü gibi negatif değerlerde 11-15 bitleri 1 pozitif değerlerde ise 0 olmaktadır. Sıcaklık değeri 2'ye tümleyen olarak saklanmaktadır. DS18B20'nin power-on reset anındaki varsayılan değeri +85°C dir.

1-Wire Haberleşme Protokolü

1-Wire bir master tarafından kontrol edilen bir veya daha fazla slave cihazın bağlanabildiği haberleşme protokolüdür. DS18B20, DS1822 gibi sensörler slave olarak hatta bağlanırlar. 1-wire haberleşmede master çoğu zaman mikrodenetleyicidir. Hatta tek slave cihaz bağlı ise bu sistem single-drop olarak adlandırılır. Eğer hatta birden çok slave cihaz var ise sistem multi-drop olarak adlandırılır.

1-wire haberleşmede tüm komutlar ve veriler LSB ilk olarak hat üzerinden gönderilir.



Şekil 6: 1-Wire İletişim Donanım Yapısı

Şekil 6'de 1-Wire iletişim için gerekli donanım yapısı görülmektedir. 1-Wire iletişim için master ve slave cihazların port yapıları open-drain yada 3-durumlu yapıda olmadır. Böylece hattı kullanmayan cihazlar yüksek empedans konumuna geçerek hattı boş bırakabiliriler. Şekil 3'te görüldüğü gibi DS18B20 sensörünün DQ haberleşme pini open-drain yapıdadır. 1-Wire haberleşmede hattın boşta olma durumu bir olma(yüksek) durumudur. Hattı bire(yüksek) çekmek için ortalama 5K bir direnç kullanılmalıdır. 1-Wire haberleşmede eğer haberleşme hattı 480us'de fazla düşük seviyede tutulursa hattaki tüm cihazlar kendini resetler. 1-Wire ile DS18B20 haberleşmesi için işlem sırası aşağıda ki gibidir.

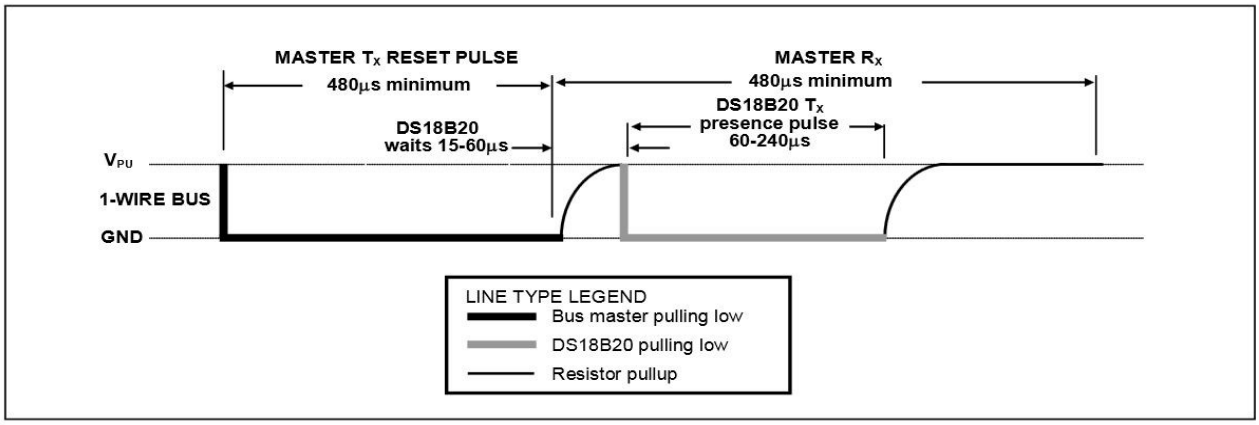
İşlem Sırası

1. Başlatma
2. ROM Komutu
3. DS18B20 Fonksiyon Komutu

DS18B20 için yukarıdaki işlem sırası çok önemlidir. İşlem sırasında herhangi bir aksama yada eksiklik durumunda DS18B20 ile iletişim kurulamaz.

Başlatma

1-Wire haberleşmede tüm işlemler başlatma rutini ile başlar. Başlatma rutininde master cihaz hat üzerinden reset işareti gönderir. Reset işareti ardından hatta bulunan cihazlar hatta bulunduklarını belirten var olma işareti gönderirler. Var olma işareti master cihaza hat üzerindeki slave cihazın hat üzerinde olduğunu ve işlem için hazır olduğunu bildirir. Reset ve var olma işaretleri zamanlamaları şekil 7'de görülebilir.



Şekil 7: Reset ve Var Olma İşaretleri Zamanlamaları

Şekil 7'de başlatma işleminde gönderilen reset ve var olma işaretlerinin zamanlamaları görülmektedir. Görüldüğü gibi ilk olarak master cihaz hattı 15-60us kadar düşük seviyeye çekerek reset işareti yollar sonra hattı boş bırakır. Bu işareti alan slave cihazda aynı şekilde hattı 60-240us kadar düşük seviyeye çekip sonra hattı boş bırakır. Böylece slave cihaz hat üzerinde olduğunu ve işleme hazır olduğunu bildirir.

ROM Komutları

Rom komutları hatta birden fazla slave cihaz bulunması durumunda kullanılır. Rom komutları slave cihazların 64-bit benzersiz seri numaraları ile birlikte çalışır. Rom komutları ile hat üzerinde kaç tane ve hangi tip slave cihaz var öğrenilebilir. Aynı zamanda slave cihazlarda ki alarm durumları rom komutları ile tespit edilebilir. Toplamda her biri 8-bit uzunlukta 5 adet rom komutu vardır. Tek slave kullanılan uygulamalarda 64-bit seri numarası kullanmaya gerek yoktur. Bu durumda Skip ROM(ROM Atla) komutu ile seri numarası gönderme işlemleri ile uğraşmadan fonksiyon komutları gönderme işlemine geçilebilir. Biz uygulamalarımızda tek slave cihaz kullanacağımız için rom komutları ile ilgili bu kadar bilgi yeterlidir. İstenirse DS18B20 sensörünün kullanma kılavuzundan detaylı bilgi edinilebilir.

DS18B20 Fonksiyon Komutları

DS18B20 ile haberleşirken ROM komutlarından sonra bu komutların gönderilmesi gerekir. Bu komutlar ile sıcaklık ölçümü başlatabilir, hafızaya okuma/yazma yapabiliriz. 6 adet fonksiyon komutu vardır.

Convert T (44h): Sıcaklık ölçme işlemini başlatan komuttur. Tek ölçüm yapılır bit çözünürlüğüne göre azami 750 ms içinde ölçüm tamamlanır.

Write Scratchpad (4Eh): Bu komut ile DS18B20'nin hazfızasına yazmak için izin alınır. DS18B20 içerisinde 3 byte'a yazma yapılır bunlar T_H, T_L ve Konfigürasyon kaydedicileridir. Veriler LSB ilk gönderilmelidir.

Read Scratchpad (BEh): Bu komut ile DS18B20 hafızasından okuma için izin alınır. Bu komut gönderildikten sonra DSB18B20 8byte(Hafızası) + CRC(Hata denetim verisi) olmak üzere toplam 9 byte veri okunur.

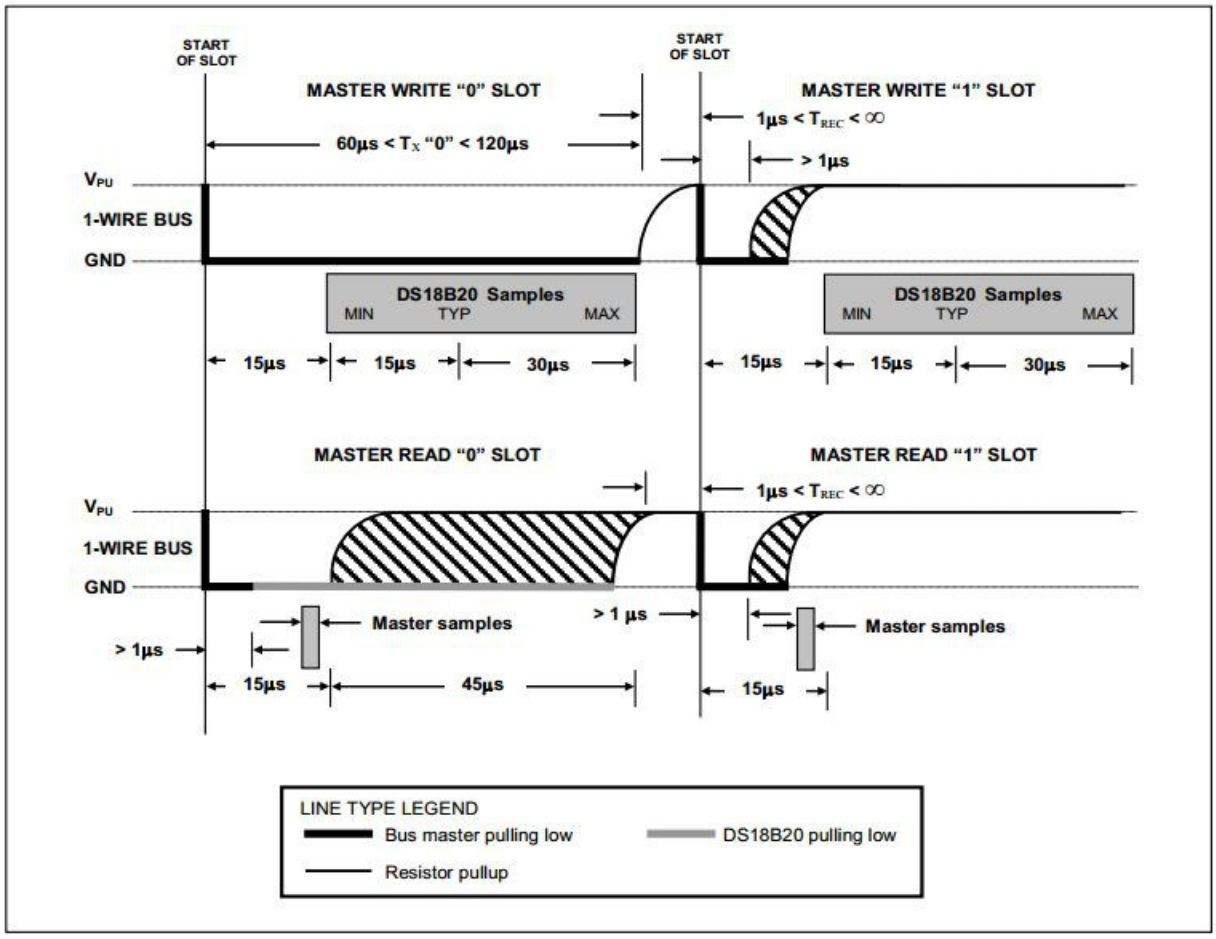
Copy Scratchpad (48h): Bu komut ile geçici hafızada bulunan T_H, T_L ve Konfigürasyon kaydedicileri kalıcı olan eeprom hafızaya yazılır.

Recall E^2 (B8h): Bu komut ile eeprom hafızada saklı olan T_H, T_L ve Konfigürasyon kaydedicileri tekrardan geçici hafızadaki yerlerine yazılır.

Read Power Suply (B4h): Bu komut ile master cihaz hattaki DS18B20'lerin hat üzerinden mi yoksa harici olarak mı beslendiklerini kontrol eder.

1-Wire Okuma Yazma Zamanları

DS18B20'ye veri okuma yazmak için gerekli zamanlamaları sağlayarak işaretlerimizi uygun şekilde hat üzerinden göndermemiz gerekir. DS18B20 ile haberleşmek için gerekli zamanlamalar şekil 5'te ki gibidir.



Şekil 8: 1-Wire Okuma Yazma Zamanlamaları

Şekil 8'te kalın siyah renkli çizgiler hattın master tarafından düşük seviyeye çekildiğini, ince siyah renkli çizgiler hattın pull-up direnci tarafından yüksek seviyeye çekildiğini ve gri renkli çizgiler ise hattın DS18B20 tarafından düşük seviyeye çekildiğini gösterir.

Şekil 5'in sol üstünden yani master tarafından 1 yazma ile başayalım. Hatta 1 yazma işlemi için master cihazın en az 60us hattı düşük seviyeye çekmesi gerekir. Yazma işlemleri arasında en az 1us kurtarma zamanı bırakılmalıdır. Benzer şekilde 0 yazma işlemi için master cihazın hattı 15us kadar düşük seviyede tutması gerekir. Bu şekilde master cihaz ard arda bit yazma işlemlerini yaparak gerekli verileri iletebilir.

Okuma işlemi için ise master cihazın hattı en az 1us düşük seviyeye çekmesi gerekir. Ardından şekil 8'te görüldüğü gibi en az 15us bekledikten sonra hattı okuyup slave cihazdan gönderilen bit değerini okuyabilir. Eğer veri 0 ise slave cihaz hattı düşük seviyeye çeker. Bit değeri 1 ise slave cihaz tepki vermez. Hat pull-up direncinden dolayı yüksek seviyede olur. Bu şekilde ard arda okuma işareti gönderek slave cihazdan veriler okunabilir.

1-Wire haberleşme için okuma yazma ve reset zamanlamaları bu şekildedir. Sistemimizde master cihaz mikrodenetleyici olduğu için 1-Wire cihazlar ile haberleşecek uygun yazılımların yazılması hatta taşınabilir olması açısından kütüphane şeklinde hazırlanması gereklidir.

1-Wire Haberleşme Kütüphanesi

1-Wire ile haberleşmesi yapabilmek için MSP430 mikrodenetleyicimizin gerekli işaretleri gönderebilmesi gerekir. 1-Wire haberleşme işlemlerini fonksiyonlar halinde düşünürsek temel olarak başlangıç, reset işlemi, bit okuma, bit yazma, byte okuma ve byte yazma gibi fonksiyonların olması gerekmektedir. Bu fonksiyonları kullanarak 1-Wire haberleşme kullanan cihazlar ile haberleşebiliriz. Tüm bu fonksiyonları tek bir kaynak ve başlık dosyası halinde tanımlayıp 1-Wire kütüphanemizi oluşturabiliriz. MSP430 denetleyicimiz için gerekli 1-Wire kütüphanesi aşağıdaki gibidir. Kütüphanemiz onewire.h başlık dosyası ve onewire.c kaynak dosyasından oluşmaktadır.

```
#ifndef __ONEWIRE_H
#define __ONEWIRE_H

#define CONVERT_TMP     0x44            // Sıcaklık ölçüm komutu
#define SKIPROM_CMD     0xCC            // Rom atlama komutu
#define READ_SCRATCHPAD 0xBE            // Scratchpad okuma komutu
#define OW_PORT_OUT         P2OUT       // One wire bit portu
#define OW_PORT_DIR         P2DIR       // One wire biti yönlendirme kaydedicisi
#define OW_PORT_IN          P2IN        // One wire biti port okuma kaydedicisi
#define OW_PORT_BIT         BIT4        // One wire iletişim port biti
#define OW_OUT_HIGH         (OW_PORT_OUT |=  OW_PORT_BIT)
#define OW_OUT_LOW          (OW_PORT_OUT &= ~OW_PORT_BIT)
#define OW_DIR_HIGH         (OW_PORT_DIR |=  OW_PORT_BIT)
#define OW_DIR_LOW          (OW_PORT_DIR &= ~OW_PORT_BIT)
#define OW_INPUT_READ       (OW_PORT_IN & OW_PORT_BIT)
// Fonksiyon prototipleri
void TekHatAyarla ();
unsigned char TekHatReset();
void TekHatBitYaz (unsigned char bit);
unsigned char TekHatBitOku ();
void TekHatByteYaz(unsigned char byte);
unsigned char TekHatByteOku();

#endif
```

Yukarıda onewire.h başlık dosyamız görülmektedir. Görüldüğü gibi başlık dosyamız içerisinde sabit değer tanımlamaları, 1-Wire pin ayarları ve ilgili makrolar ile kaynak dosyamızda kullanılan fonksiyonların prototipleri bulunmaktadır.
Benzer şekilde onewire.c kaynak dosyamız aşağıdaki gibidir.

```
#include "msp430.h"
#include "onewire.h"
// One wire iletişim için hattı hazırlar.
void TekHatAyarla (){
        OW_DIR_LOW;
        OW_OUT_HIGH;
}

// One wire iletişim için hatta reset işareti gönderir.
unsigned char TekHatReset(){
        OW_OUT_LOW;
```

```
        OW_DIR_HIGH;
        __delay_cycles(500);        // 500us
        OW_DIR_LOW;
        __delay_cycles(100);        // 100us
        if(OW_INPUT_READ){
        __delay_cycles(400);        // 400us
        return 1;}
        else{
        __delay_cycles(400);        // 400us
        return 0;}
}

// One wire iletişim için hatta 1 bit veri yazar.
void TekHatBitYaz (unsigned char bit){
        __delay_cycles(2);          // 2us
        OW_DIR_HIGH;
        OW_OUT_LOW;
        if(bit)
        __delay_cycles(4);          // 4us
        else
        __delay_cycles(64);         // 64us
        OW_DIR_LOW;
        if(bit)
        __delay_cycles(64);         // 64us
        else
        __delay_cycles(4);          // 4us
}

// One wire iletişim için hattan 1 bit veri okur.
unsigned char TekHatBitOku (){
        unsigned char bit;
        __delay_cycles(2);          // 2us
        OW_OUT_LOW;
        OW_DIR_HIGH;
        __delay_cycles(2);          // 2us
        OW_DIR_LOW;
        __delay_cycles(15);         // 15us
        bit=OW_INPUT_READ;
        __delay_cycles(60);         // 60us
        return bit;
}

// One wire iletişimde hatta 1 byte veri yazar.
void TekHatByteYaz(unsigned char veri){
        unsigned char sayac;
        for(sayac = 0; sayac < 8; sayac++)
        {
   TekHatBitYaz(veri & 0x01);
   veri >>= 1;
        }
}
```

```
// One wire iletişimde hattan bir byte veri okur.
unsigned char TekHatByteOku(){
        unsigned char sayac;
        unsigned char veri;
   for(sayac = 0; sayac < 8; sayac++)
   {
   veri >>= 1;
   if (TekHatBitOku()) veri |= 0x80;
        else
        veri &= 0x7F;
   }
   return veri;
}
```

onewire.c kaynak dosyamız görüldüğü gibi sırasıyla TekHatAyarla, TekHatReset, TekHatBitYaz, TekHatBitOku, TekHatByteYaz ve TekHatByteOku fonksiyonlarından oluşmaktadır. Kütphanemiz 1-wire iletişim için tüm fonksiyonları içermektedir.

1-Wire Kütüphanemiz genel bir kütüphanedir. Yani 1-Wire ile haberleşen tüm cihazlar ile haberleşmede kullanılabilir. DS18B20 sensörümüzde 1-wire ile haberleştiğinden 1-wire kütüphanesini kullanarak DS18B20 sensörümüz için özel bir kütüphane oluşturmamız daha faydalı olacaktır.

DS18B20 Kütüphanesi

DS18B20 kütüphanesi DS18B20 sensörünü uygulamalarımızda kullanabilmek için gerekli fonksiyonları içerir. DSB18B20 sensörü 1-wire ile haberleştiğinden kütüphanemiz arka planda onewire kütüphanesini kullanmaktadır.

DS18B20 kütüphanemiz aşağıda ki gibidir.

```
#ifndef __DS18B20_H
#define __DS18B20_H

// Fonksiyon prototipleri
void DS18B20_Ayarla();
unsigned int DS18B20_Oku();
char* DS18B20_Yaziya_Cevir(unsigned int);

#endif
```

Yukarıda DS18B20.h başlık dosyamız görülmektedir. Görüldüğü gibi dosyamızda sadece kaynak dosyamızda kullanılan fonksiyon prototipleri bulunmaktadır.
DS18B20.c kaynak dosyamız aşağıda ki gibidir.

```
#include "msp430.h"
#include "DS18B20.h"
#include "onewire.h"
char yazi[9];

void DS18B20_Ayarla(){
        TekHatAyarla();
}
```

```
// DS18B20'de ham olarak sıcaklık verisini okuyan fonksiyon
unsigned int DS18B20_Oku(){
        unsigned char high_byte,low_byte;
        while(TekHatReset());
        TekHatByteYaz(SKIPROM_CMD);
        TekHatByteYaz(CONVERT_TMP);
        while(TekHatReset());
        TekHatByteYaz(SKIPROM_CMD);
        TekHatByteYaz(READ_SCRATCHPAD);
        __delay_cycles(120);
        low_byte  = TekHatByteOku();
        high_byte = TekHatByteOku();
        return (high_byte<<8 & 0xff00) + low_byte;
}

// DS18B20_Oku fonkisyonu ile okunan değeri işaret,
// ondalık ve tam sayı kısımlarına ayırıp asciiye çeviren fonksiyon
char* DS18B20_Yaziya_Cevir(unsigned int sayi){
        unsigned char gecici;
        if(sayi & 0xF000)
        yazi[0] = '-';
        else
        yazi[0] = '+';
        gecici = sayi>>4 & 0xff;
        yazi[1] = gecici/100 + 0x30;
        gecici %= 100;
        yazi[2] = gecici/10 + 0x30;
        yazi[3] = gecici%10 + 0x30;
        yazi[4] = '.';
        gecici = sayi & 0x000f;
        sayi = gecici * 625;
        yazi[5] = sayi/1000 + 0x30;
        sayi = sayi % 1000;
        yazi[6] = sayi/100 + 0x30;
        sayi = sayi % 100;
        yazi[7] = sayi / 10 + 0x30;
        yazi[8] = sayi % 10 + 0x30;
        return yazi;
}
```

Görüldüğü gibi kaynak dosyamızda DS18B20_Ayarla, DS18B20_Oku ve DS18B20_Yaziya_Cevir fonksiyonları bulunmaktadır. DS18B20_Ayarla başlangıçta çalıştırılır ve DS18B20 için gerekli ilk ayarlamaları yapar. DS18B20_Oku fonksiyonu ise DS18B20 sensöründen 16 bit olarak ham halde sıcaklık verisini okur. DS18B20_Yaziya_Cevir fonksiyonu ise okunan ham sıcaklık değerinin işaret, tam sayı, ve ondalık kısımlarını ayırarak karaktere çevirerek yazi[9] karakter dizisine kayıt eder. Böylece okunan sıcaklık değeri karakterlerine çevirilerek LCD ekranda görüntülenebilir yada UART üzerinden gönderilebilir.

DS18B20 ve onewire kütüphanelerimiz sayesinde MSP430 dentleyicilerimiz ile DS18B20 sensörünü uygulamalarımızda kullanabiliriz. DS18B20 ile ilgili bu kadar teorik bilgiden sonra uygulamalarımıza geçelim.

Uygulama 8.1 DS18B20 Kullanımı

Bu uygulamamızda DS18B20 sensörünü varsayılan çözünürülük olan 12 bit(azami çözünürlük) çözünürlükte çalıştırıp hassas bir şekilde sıcaklık ölçme işlemini yapacağız. Ölçtüğümüz sıcaklık bilgisini işaret, tam sayı ve ondalık kısımlarıyla birlikte LCD ekranda görüntüleyeceğiz.

Uygulamanın kodları aşağıda ki gibidir.

```
#include <msp430.h>                          // MSP430 başlık dosyası
#include "DS18B20.h"                         // DS18B20 başlık dosyası
#include "LCD.h"                             // LCD başlık dosyası

unsigned int wISIHam;                        // Ham sıcaklık değerini tutan değişken
void main(void) {                            // Ana fonksiyon başlangıcı
        WDTCTL  = WDTPW | WDTHOLD;           // Watchdog timeri durdur.
        BCSCTL1 = CALBC1_1MHZ;               // Dahili osilatörü 1MHz'e ayarla
        DCOCTL = CALDCO_1MHZ;                // Dahili osilatörü 1MHz'e ayarla
        DS18B20_Ayarla();                    // DS18B20 ayarla
        LCD_Ayarla();                        // LCD ayarla
        while(1){                            // Sonsuz döngü
        wISIHam=DS18B20_Oku();               // DS18B20'den sıcaklığı oku
        LCD_Git_XY(1,1);                     // LCD kursörü 1. satır 1. sütuna götür.
        LCD_Yazi_Yaz("DS18B20 DENEMESi"); // Ekrana DS18B20 DENEMESi yazdır.
        LCD_Git_XY(2,1);                     // LCD kursörü 2. satır 1. sütuna götür.
        LCD_Yazi_Yaz("ISI= ");               // Ekrana ISI yazdır.
        LCD_Yazi_Yaz(DS18B20_Yaziya_Cevir(wISIHam));  // Sıcaklık değerini ayırıp
ekrana yazdır
        LCD_Karakter_Yaz(0xDF);              // Santigrat işaretini yazdır
        LCD_Karakter_Yaz('C');               // C karakterini yazdır.
        }
}
```

Kodları inceleyecek olursak, başlangıçta osilatör ve ilk ayarlar yapılır. Sonrasında DS18B20 ve LCD ayarları yapılarak program sonsuz döngüye girer. Sonsuz döngüde program sürekli olarak sıcaklık bilgisini okuyarak LCD ekranda görüntüler. LCD_Yazi_Yaz(DS18B20_Yaziya_Cevir(wISIHam)) kod satırında iki foksiyon iç içe kullanılmıştır. Yani önce DS18B20_Yaziya_Cevir(wISIHam) fonksiyonu ölçülen sıcaklık değerini yazıya çevirir. Sonrasında LCD_Yazi_Yaz() fonskiyonu çevirilen sıcaklık değerini LCD ekrana yazdırır.

Kodları denetleyiciye atmadan önce geliştirme kartı ayarlarını yapmamız gerekmektedir. Uygulamayı çalıştırabilmek için sw2 anahtarlarını ve jp3 atlmasını on konumuna getirmek gerekir. Ayrıca sorun yaşamamak için diğer birimlerini atlama ve anahtarlarını off konumuna getirmemiz gerekir. MSP430 denetleyicimizin pinleri paylaşımlı olarak kullanıldığı için uygulamada kullanılmadığı halde açık kalan birimler uygulamanın çalışmasını etkileyebilir.

Geliştirma kartı ayarlarını yaptıktan sonra kodlarımızı derleyip MSP430G2553 denetleyicimize yükleyelim. Sorunsuz bir şekilde derleyip kodlarımızı yükledikten sonra uygulamamız çalışacaktır.

Şekil 9: Uygulama 8.1 Çalışma Görüntüleri

Şekil 9'da uygulamamızın çalışma görüntüleri görülmektedir. Görüldüğü üzere DS18B20 sensörü virgülden sonra 4 basamak 0.0625 °C çözünürlükle sıcaklık ölçümü yapabilmektedir. Üst sıcaklık seviyelerini test etmek için bir çakmak ile sensörü ısıtıp yüksek sıcaklık değerlerini ölçebilirsiniz.

Uygulama 8.2 DS18B20 ile Dahili Sıcaklık Sensörünün Karşılaştırılması

Son olarak bu uyuglamamızda DS18B20 sensörü ile MSP430 denetleyicimiz içerisinde bulunan dahili sıcaklık sensörünün karşılaştırmasını yapacağız.

Uygulamanın kodları aşağıda ki gibidir.

```
#include <msp430.h>                    // MSP430 başlık dosyası
#include "DS18B20.h"                   // DS18B20 başlık dosyası
#include "LCD.h"                       // LCD başlık dosyası

void BCDCevir(unsigned short);         // BCDCevir fonksiyon prototipi

unsigned char bBCDSayi[4];             // BCD değerleri tutan dizi
unsigned char bGecikmeSayac=0;         // Gecikme sayacı
unsigned int  wISIHam;                 // Ham sıcaklık değerini tutan değişken
unsigned long wSicaklik;               // Sıcaklık değerini tutan değişken

void main(void) {                      // Ana fonksiyon başlangıcı
    WDTCTL  = WDTPW | WDTHOLD;         // Watchdog timeri durdur.
    BCSCTL1 = CALBC1_1MHZ;             // Dahili osilatörü 1MHz'e ayarla
    DCOCTL = CALDCO_1MHZ;              // Dahili osilatörü 1MHz'e ayarla
    DS18B20_Ayarla();                  // DS18B20 ayarla
    LCD_Ayarla();                      // LCD ayarla
    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;
```

```
        _BIS_SR(GIE);                   // Kesmeleri aç
        while(1){                       // Sonsuz döngü
        _BIS_SR(LPM0_bits);             // Uykuya gir
        wSicaklik = ADC10MEM;           // Çevrim sonucunu kaydet
        wSicaklik = ((wSicaklik - 673) * 42300) / 1024; // Çevrim sonucunu santigrat
dereceye çevir
        BCDCevir(wSicaklik);            // Sıcaklık değerini rakamlarına ayır.
        LCD_Git_XY(1,1);                // LCD kursörü 1.satır 1.sütüna götür.
        LCD_Yazi_Yaz("Dahili=");        // Ekrana "Dahili=" yazdır.
        LCD_Karakter_Yaz(bBCDSayi[0]+0x30);     // Onlar basamağını yazdır
        LCD_Karakter_Yaz(bBCDSayi[1]+0x30);     // Birler basamağını yazdır.
        LCD_Karakter_Yaz('.');                  // Noktayazdır
        LCD_Karakter_Yaz(bBCDSayi[2]+0x30);     // Onda birler basamağını yazdır.
        LCD_Karakter_Yaz(bBCDSayi[3]+0x30);     // Yüzde birler basamağını yazdır.
        LCD_Karakter_Yaz(0xDF);         // Santigrat işaretini yazdır
        LCD_Yazi_Yaz("C  ");            // "C  " yazısını yazdır.
        wISIHam=DS18B20_Oku();          // DS18B20'den sıcaklığı oku
        LCD_Git_XY(2,1);                // LCD kursörü 2. satır 1. sütuna götür.
        LCD_Yazi_Yaz("Harici=");        // Ekrana "Harici=" yazdır.
        LCD_Yazi_Yaz(DS18B20_Yaziya_Cevir(wISIHam)+2);      // Sıcaklık değerini
ayırıp ekrana yazdır
        LCD_Karakter_Yaz(0xDF);         // Santigrat işaretini yazdır
        LCD_Karakter_Yaz('C');          // C karakterini yazdır.
        }
}

// Timer0_A0 kesme vektörü
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{       if(++bGecikmeSayac>=100){               // 500 ms oldu mu?
        bGecikmeSayac=0;                        // Zaman sayacını sıfırla
        ADC10CTL0 |= ENC + ADC10SC;             // AD çevrimi başlat
        }
        TA0CCR0 += 5000;                        // Timeri yeniden kur.
}

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

// 0-9999 Arası girilen sayıyı BCD'ye çeviren fonksiyon
void BCDCevir(unsigned short Sayi){
        unsigned char bSayac;
        for(bSayac=0;bSayac<4;bSayac++){
                bBCDSayi[3-bSayac]= Sayi%10;
        Sayi /= 10;
        }
}
```

Görüldüğü gibi kodlarda öncelikle her zamanki gibi başlangıç ayarları ve osilatör ayarları yapılır. Sonrasında DS18B20, LCD, zamanlayıcı ve ADC ayarları yapılır kesmelere izin verilip sonsuz döngüye girilir. Sonsuz döngünün başında işlemci uyku moduna sokulur. Yaklaşık yarım saniye sonra zamanlayıcı kesmesi ADC ile sıcaklık ölçümü başlatılır tekrar uyku moduna geçilir. ADC işlemi bitince işlemci uykudan uyandırılır. Sonrasında dahili sensörden ölçülen sıcaklık değeri hesaplanarak LCD ekranda 1. satırda görüntülenir. Devamında benzer olarak DS18B20 ile sıcaklık ölçümü yapılıarak ölçüm sonucu LCD ekranda 2. satırda görüntülenir. Bu şekilde sürekli olarak program sonsuz döngüde çalışmaya devam eder.

LCD_Yazi_Yaz(DS18B20_Yaziya_Cevir(wISIHam)+2) kod satırında bulunan 2 ilavesi ile DS18B20 ile ölçülüp yazdırılan sonucun ilk iki karakteri yani işareti ile yüzler basamağı gösterilmemiştir. Bunun nedeni karakter LCD mizin sınırlı sayıda karakter göstermesidir. Sıfır dereceninin altında yada 100 derecenin üzerinde çalışmadığımız için bu karakterler ihmal edilebilir.

Uygulama kodlarını derleyip denetleyiciye yüklemeden önce geliştirme kartı ayarlarımızı bir önceki uygulama ile aynı yapmamız gerekmektedir. Ayarları doğru bir şekilde yaptıktan sonra uygulama kodlarını MSP430G2553 denetleyicimize yükeleyebiliriz. Herhangi bir sorun yok ise uygulama çalışınca ekranda ölçüm sonuçları görünecektir.

Şekil 10: Uygulama 8.2 Çalışma Görüntüsü

Şekil 10'da uygulama 8.2'nin ekran çıktısı görülebilir. Görüldüğü gibi 1. satırda dahili olarak belirtilen sıcaklık MSP430G2553'ün içerisinde bulunan sıcaklık sensörü ile ölçülen sıcaklık değeridir. Benzer şekilde harici olarak gösterilen sıcaklık ise DS18B20 ile ölçülen sıcaklık değeridir. Dahili sensör ile noktadan sonra 2 basamak gösterilmiştir. Ölçüm işlemi 10 bit ADC ile yapıldığı için sensör çözünürlüğü DS18B20 kadar değildir.
Görüldüğü gibi kalibrasyon hatasından dolayı sıcaklık değerlerinde yaklaşık 2°C fark vardır. DS18B20 katalog değerlerine göre 0.5°C hata payı olduğundan hatanın çoğunun dahili sıcaklık sensöründen kaynaklandığnı söyleyebiliriz. Her ne kadar dahili sıcaklık sensörüne kalbirasyon ayarı vs. yapılabilse de D18B20 kadar iyi sonuçlar alamayız. Tabi bu arada dahili sensörümüzün bedava olduğunu göz önünde bulunadurmak gerekir. Sonuç olarak uygulamalarınızda sensör seçiminde çok yönlü düşünerek uygulamanın ihtiyaçlarını en iyi şekilde karşılayan asgari maliyetli ürünü seçmek gerekir.