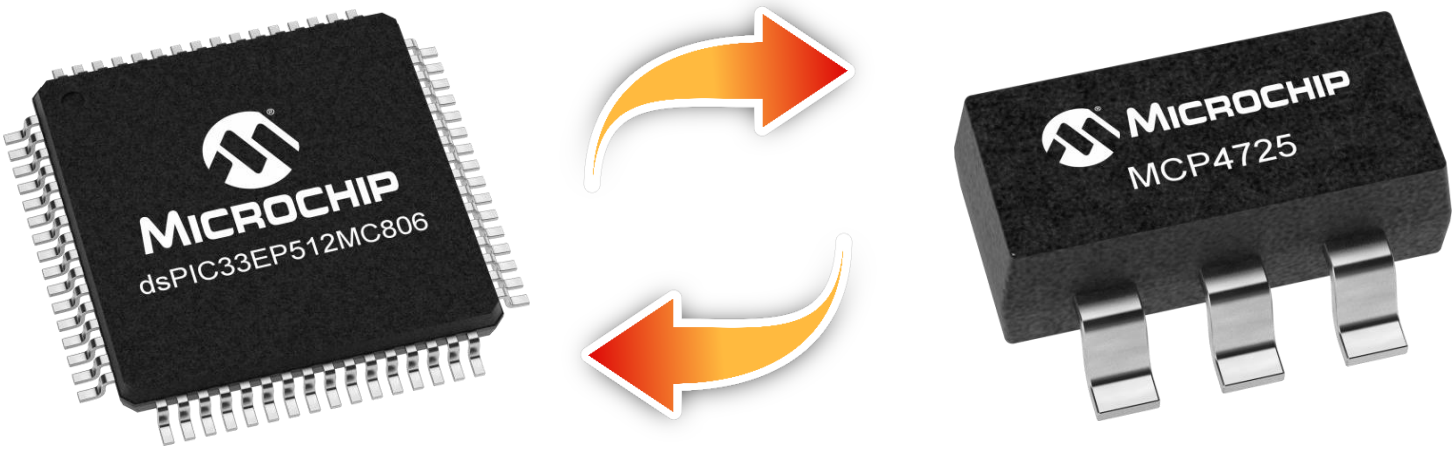
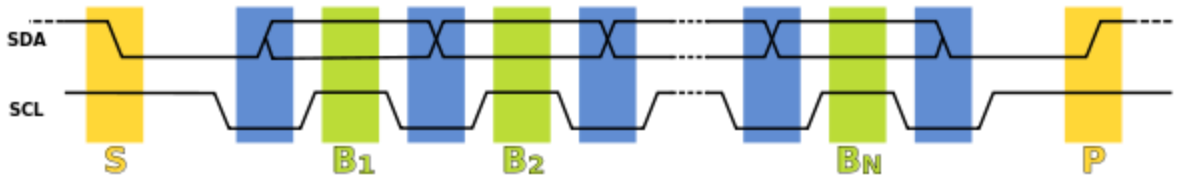
SERİ HABERLEŞME

I^2C SERİ HABERLEŞME PROTOKOLÜ

DİJİTAL ANALOG DÖNÜŞTÜRÜCÜ İÇİN I^2C ARABİRİM TASARIMI





Yusuf ÖZDEMİR

ANKARA

TEMMUZ 2022

İÇİNDEKİLER

➔ KISALTMALAR
➔ ÖZET

1. GİRİŞ
1.1. Çalışmanın Amacı
1.2. Çalışma Sırasında İzlenen Yol

2. SAYISAL HABERLEŞME SİSTEMLERİ
2.1. Sayısal Haberleşme Sistemleri Hakkında
2.2. Paralel Haberleşme Sistemleri
2.3. Seri Haberleşme Sistemleri
2.3.1. Asenkron Haberleşme
2.3.2. Senkron Haberleşme
2.4. Seri Haberleşme ve Paralel Haberleşme Sistemlerinin Farkları

3. HABERLEŞME PROTOKOLLERİ
3.1. Haberleşme Protokolleri Hakkında
3.1.1. Sistemler Arası Haberleşme Protokolleri
3.1.2. Sistem İçi Haberleşme Protokolleri

4. I^2C SERİ HABERLEŞME PROTOKOLÜ
4.1. I^2C Seri Haberleşme Protokolü Hakkında
4.2. I^2C
4.3. I^2C İletişim Adımları
4.3.1. Hattın Durum Sorgusu
4.3.2. Başlangıç Aşaması
4.3.3. Adresleme Aşaması
4.3.4. Veri İletim Aşaması
4.3.5. Sonlandırma Aşaması
4.3.6. ACK/NACK

5. C DİLİ İLE I^2C
5.1. C DİLİ İLE I2C SERİ HABERLEŞME ARABİRİMİNİN GERÇEKLEŞTİRİLMESİ
5.2. Kodlar
5.3. Kodların Açıklaması
5.3.1. Main Dosyası ve Mikrodenetleyicinin Konfigürasyon Ayarları
5.3.2. I^2C Kütüphanesi ve Fonksiyonlar
5.3.2.1. start() Fonksiyonu
5.3.2.2. stop() Fonksiyonu
5.3.2.3. write() Fonksiyonu

➔ SONUÇ
➔ REFERANSLAR
➔ KULLANILAN PROGRAMLAR

KISALTMALAR

GND : Ground

USB : Universal Serial Bus

UART : Universal Asynchronous Receiver/Transmitter

USART : Universal Synchronous Asynchronous Receiver/Transmitter

I^2C : Inter Integrated Circuit

SPI : Serial Peripheral Interface

CAN : Controller Area Network

GPU : Graphics Processing Unit

BIT : Binary Digit

SDA : Serial Data

SCL : Serial Clock

ACK : Acknowledge

ÖZET

Günümüzde haberleşme sistemlerinde "Seri Haberleşme" yaygın olarak kullanılmaktadır. Bunun en büyük nedenlerinden biriside paralel haberleşme sistemlerinde gönderilen bir bytelık verinin her biti ayrı kanaldan aynı anda gönderilmektedir. Bu işlem iletişimde her ne kadar bize hız kazandırsa da maliyeti çok arttırmaktadır. Seri haberleşmeye geldiğimizde durumun çok daha farklı olduğunu görüyoruz, seri haberleşmede gönderilen bir bytelık veride her bit aynı kanaldan farklı zamanlarda gönderilmektedir. Bu işlem teknik olarak paralel haberleşmeye göre ciddi zaman kaybı oluşturuyor olsa da bizim maliyetimizi de ciddi oranda aşağı çekmektedir.

Seri haberleşmede iki tür iletişim vardır bunlardan ilki asenkron haberleşmedir. Asenkron haberleşmede tek kanaldan iletilen veriler için alıcı ve verici tarafında bir saniyede iletilen veri miktarı aynı olmalıdır bu orana "Baud Rate" denmektedir. Aksi durumda veri kaybı meydana gelecektir ve haberleşmemiz başarısız olacaktır.

Diğer seri haberleşme türümüz ise senkron haberleşmedir. Senkron haberleşmede bir adet "DATA" hattı ve bir adet "CLOCK" hattı bulunmaktadır. Veriler iletilirken cihazlar verileri senkron olarak okuyup yazarlar bundan dolayı asenkron haberleşmeye göre çok daha hızlı veri iletimine sahiptirler. I^2C ve SPI haberleşme protokolleri bu haberleşme türüne örnek olarak verilebilir.

Biz bu yazımızda I^2C haberleşme protokolünü ele alacağız. I^2C haberleşme protokolü nedir, nerelerde kullanılır, avantajları nedir, dezavantajları nedir gibi konu başlıklarını ele alırken aynı zamanda I^2C haberleşme protokolü için bir kütüphanenin nasıl yazılacağı hakkında bilgi vereceğiz.

Bu yazımızda "DSPICEP512MC806" mikrodenetleyicisi ile "MCP4725A1T-E/CH" dijital analog dönüştürücüsünün haberleşmesini ele aldık. Farklı "Master" ve "Slave" durumları içinde benzer yollar izlenerek kütüphane yazılabilir.

Anahtar Kelimeler: Seri Haberleşme, I^2C, Mikrodenetleyici, Dijital Analog Converter, C Dili

1. GİRİŞ

1.1. Çalışmanın Amacı

Yapmış olduğumuz bu çalışmada, I^2C ile ilgili literatürdeki Türkçe kaynak eksikliğine alternatif kaynak olmayı amaçladık. Aynı zamanda literatürde, başlangıç seviyesinde ayrıntılı bir şekilde I^2C kütüphanesinin nasıl yazılacağına dair bir kılavuzun yer almaması ve bunun sonucu olarak lisans ve lisans öncesi öğrencilerin araştırma yaparken ciddi zorluklar çekmesinden dolayı yeni başlayan araştırmacılara alternatif bir kaynak olabilmesi için bu yazı yazılmıştır.

1.2. Çalışma Sırasında İzlenen Yol

Yapmış olduğumuz bu çalışmada ilk olarak haberleşme nedir gibi bir genel başlıktan yola çıkarak tümden gelim yöntemi ele alınmıştır. Daha sonrasında seri haberleşme nedir, I^2C seri haberleşme protokolü nedir, C dilinde I^2C seri haberleşme protokolü kütüphanesi nasıl yazılır gibi daha özel başlıklar tek tek incelenmiştir.

2. SAYISAL HABERLEŞME SİSTEMLERİ

2.1 Sayısal Haberleşme Sistemleri Hakkında

Sayısal haberleşme, her türlü bilgi aktarımı veya değiş tokuşudur. Elektronik anlamda haberleşme, belirli mesafeler üzerinden yapay teçhizat kullanarak bilgi aktarımının sağlanması anlamına gelmektedir. Haberleşme sistemleri analog ve sayısal haberleşme sistemleri olarak ikiye ayrılmaktadır. Bir zaman aralığının bütünü yerine sadece belirli zaman anlarında tanımlanmış ve sadece belirli değerleri alabilen işaretler sayısal işaretler olarak adlandırılmaktadır. [1]

Günümüzde sayısal haberleşme sistemleri analog haberleşme sistemlerinin yerini almıştır bunun en büyük nedenlerinden birisi sayısal haberleşme sistemlerinin gürültüden çok çok daha az etkilenmesidir. Bu özellikleri sayesinde veriler daha güvenilir bir şekilde iletilebilmekte ve saklanabilmektedir.

Analog haberleşme sistemlerinde veriler hem zaman uzayında hem de değer uzayında sürekli iken sayısal haberleşme sistemlerinde bir süreklilikten söz edilemez. Zaman uzayı belirli bir frekansa bağlı olarak (f_s) örneklenirken, değer uzayında da kuantalama işlemi gerçekleştirilir. Bu işlemlerde örnekleme frekansı ne kadar yüksek tutularsa verideki kayıp o kadar minumuma inerken verinin miktarı da bir o kadar artmaktadır bu yüzde bu ikisi arasındaki ilişkiyi iyi kurmak önem arz etmektedir.

2.2 Paralel Haberleşme Sistemleri

Paralel haberleşme bir veri içindeki bitlerin aynı anda gönderilmesidir. Paralel veri iletiminde kullanılan verinin her bir biti için ayrı bir kablo verisi bulunur ve her bir bit ayrı bir kablo üzerinden iletilir.

Paralel giriş / çıkışta 8 bit data, 1 bit data hazır, 1 bit data istek, 1 bit GND olmak üzere 11 kablo / 11 tel üzerinden iletişim gerçekleştirilir. Paralel haberleşme hızlı gerçekleşir ancak her bir bit için bir kablo / tel kullanıldığı için maliyetli bir yoldur.

Paralel haberleşme için örnek vermemiz gerekirse günümüz bilgisayarlarından pek bir örnek veremiyorum ancak eski bilgisayarlarda bulunan DB25 portu paralel haberleşme için iyi bir örnek olacaktır. [2]

2.3 Seri Haberleşme Sistemleri

Seri haberleşme bütün verilerin tek bir tel / kablo üzerinden iletildiğini haberleşme protokolüdür. Seri haberleşmede veriler tek bir tel üzerinden aktarıldığı için paralel haberleşmeye göre daha yavaş ancak daha düşük maliyetlidir. Seri haberleşmede 1 adet data ve 1 adet GND olmak üzere 2 kablo / 2 tel üzerinden iletişim gerçekleşir. Bilgisayar üzerinde gerçekleşen iletişim seri iletişim ile sağlanır. Seri iletişimde bir kerede bir karakterin sadece bir biti iletilir bu yüzden alıcı makine ile doğru iletişimin sağlanabilmesi için alıcı makinenin karakter uzunluğu, başla bitir bitlerini ve iletim hızını bilmesi gerekmektedir. Paralel iletimde tüm bitler aynı anda gönderildiği için başla bitir bitlerine ihtiyaç duyulmazken, seri iletimde tüm bitler tek hat üzerinden gittiği için buna ihtiyaç vardır. Seri haberleşme için örnek vermemiz gerekirse bilgisayarlarda bulunan USB, Ethernet ve RS232 portları örnek olarak gösterilebilir. Seri haberleşme ASenkron ve Senkron olmak üzere iki çeşittir. [2]

2.3.1 Asenkron Haberleşme

Alıcı ve vericinin eş zamanlı olarak çalışmasına gerek duyulmayan haberleşme çeşididir. Veri gönderimi olmadığı zaman hat boşta kalır. Senkron iletişime göre daha yavaş gerçekleşir. Her veri grubu ayrı olarak gönderilir, gönderilecek veri bir anda sadece bir karakter olacak şekilde hatta bırakılır. Karakterin başına başlangıç ve sonuna olası hataları tespit etmek için başka bir bit bırakılır. İletişimi başlatmak için başla biti (0) ve iletişimi bitirmek için bitir biti (1) kullanılır. [2]

2.3.2 Senkron Haberleşme

Alıcı ve vericinin eş zamanlı olarak çalışması anlamına gelir. İlk olarak gönderici belirli bir karakter gönderir ve bu iki taraf tarafından bilinen iletişime başlama karakteridir, alıcı bu karakteri okuduğu zaman iletişim başlamış olur. İletişim başladıktan sonra verici taraf verileri göndermeye başlar ve işlem bloku tamamlanana kadar veya iletişim kesilene kadar veri transferi devam eder. Seri ve Paralel haberleşmeyi en iyi açıklayan görsel bu olabilir, yukarıdaki paralel iletişim aşağıdaki seri iletişim örneğidir. [2]

2.4 Seri Haberleşme ve Paralel Haberleşme Sistemlerinin Farkları

- Paralel iletişim seri iletişime göre daha hızlı veri aktarır.
- Paralel iletişim seri iletişime göre daha maliyetlidir.
- Paralel iletişim seri iletişime göre daha çok fiziksel alana ihtiyaç duyar. (Daha fazla tel ile aktarım yapıldığı için tel sayısı arttıkça fiziksel alan ihtiyacı da artmaktadır.)
- Paralel iletişimde başla/bitir bitlerine ihtiyaç yoktur. (Veriler aynı anda aktarıldığı için.)
- Paralel iletişimde her bir bit tek tel / hat üzerinden gönderilirken, seri iletişimde tüm veriler aynı tel / hat üzerinden aktarılır.
- Seri iletişimde başla/bitir bitleri kullanılır.
- Seri iletişimde iletim hızının bilinmesi gerekir. [2]

Sonuç olarak günümüzde yüksek maliyetinden dolayı paralel haberleşme sistemleri tercih edilmemektedir. Geliştirilen farklı seri haberleşme protokolleri ile yüksek hızlarda güvenli bir şekilde veri iletimi gerçekleştirilebilmektedir. Bizde bu yazımızda I^2C seri haberleşme protokolü üzerinden ilerleyeceğiz.

3. HABERLEŞME PROTOKOLLERİ

3.1 Haberleşme Protokolleri Hakkında

Haberleşme Protokolleri, iki veya daha fazla haberleşme sisteminin herhangi bir fiziksel ortam aracılığıyla verileri iletmesine izin veren bir dizi kuraldır. Protokoller hem donanım hem de yazılım veya her ikisinin kombinasyonunda uygulanabilir. Analog ve Dijital Haberleşme Sistemlerinde çeşitli haberleşme protokolleri yaygın olarak kullanılır.

Gömülü Sistem, hem donanım hem de yazılım kullanan elektronik bir sistem veya cihazdır. Bir işlemci, sensörler vb. gibi fiziksel dünya çevre birimlerinden girdi alır, aynısını uygun yazılımla işler ve istenen çıktıyı sağlar. Bu durumda, bileşenlerin beklenen çıktıyı sağlamak için birbirleriyle iletişim kurması gerekir. Bu iletişimde haberleşme protokolleri ile sağlanır.

Haberleşme Protokolleri iki gruba ayrılır. Sistemler Arası Haberleşme Protokolleri, Sistem içi Haberleşme Protokolüdür. [3]

3.1.1 Sistemler Arası Haberleşme Protokolleri

Sistemler arası protokoller, iki iletişim cihazı arasında, yani GPU ile mikroişlemci kiti, geliştirme kartları vb. arasında iletişim kurar. Bu durumda, haberleşme, veri yolu sistemi aracılığıyla sağlanır. Sistemler arası protokol şu şekilde kategorize edilebilir: [3]

- USB Haberleşme Protokolleri
- UART Haberleşme Protokolleri
- USART Haberleşme Protokolleri

3.1.2 Sistem İçi Haberleşme Protokolleri

Elektronik devre kartı içindeki bileşenler arasında iletişim sağlar. Gömülü sistemlerde, denetleyiciye bağlı bileşenlerin sayısını artırır. Bileşenlerdeki artış, devre karmaşıklığına ve güç tüketiminde artışa neden olur. Sistem içi protokol, çevre birimlerden gelen verilere güvenli erişim vardır. Sistem içi protokol şu şekilde kategorize edilebilir: [3]

- I2C Protokolü
- SPI Protokolü
- CAN Protokolü

4. I²C SERİ HABERLEŞME PROTOKOLÜ

4.1 I²C Seri Haberleşme Protokolü Hakkında

Inter Integrated Circuit (I2C), Philips Semiconductors tarafından geliştirilmiş bir seri haberleşme protokolüdür. Bu protokolün temel amacı, çevresel yongaları mikro denetleyici ile bağlamayı kolaylaştırmaktır. Gömülü sistemlerde, tüm çevrebirim aygıtları, mikro denetleyiciye bellek eşlemeli aygıtlar olarak bağlanır.

I2C, cihazlar arasında bilgi taşımak için iki kablo SDA (Seri Veri Hattı) ve SCL (Seri Saat Hattı) gerektirir. Bu iki aktif kablo çift yönlüdür.

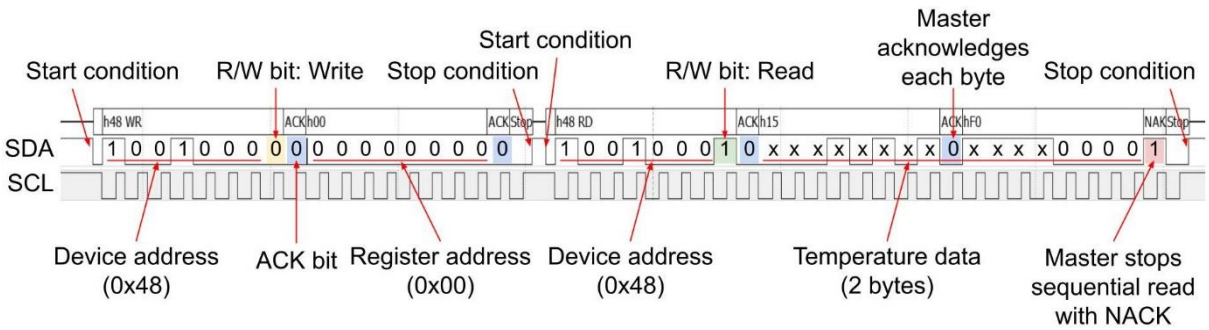
I2C protokolü, Master'dan Slave'e haberleşme protokolüdür. Her Slave'e benzersiz bir adres verilir. İletişim kurmak için, master başlangıçta hedef slave adresini R / W (Oku / Yaz) bayrağıyla birlikte gönderir. Karşılık gelen slave cihaz, diğer cihazları kapalı durumda bırakarak aktif moda geçecektir.

Slave hazır olduğunda, Master ve Slave arasında iletişim başlar. Verici 1 baytlık (8 bit) veri iletirse, alıcı tarafından bir bit alındı bildirimi yanıtlanır. Cihazlar arasındaki iletişimin sonunda bir durdurma koşulu verilir.

I2C Protokollerinin avantajları aşağıdaki gibidir:

- Seyrek erişilen yerleşik cihazlar arasında iyi iletişim sağlar.
- Adresleme mekanizması, master- slave iletişimini kolaylaştırır.
- Maliyet ve devre karmaşıklığı, cihaz sayısına bağlı kalmaz.

I2C İletişim Protokollerinin en büyük dezavantajı sınırlı hızıdır. [3]



4.2 I^2C

I^2C seri haberleşme protokolünde 4 adet pin bulunur bunlar, SDA, SCL, VCC ve GND pinleridir. I^2C protokolü, senkron haberleşme olduğu için clock sinyaline ihtiyaç duyar. SDA pininden verimizi gönderirken SCL pininden de aynı zamanda bir zaman sinyali göndeririz. Bu göndermiş olduğumuz zaman sinyali alıcı ve verici arasındaki senkronizasyonu sağlamaktadır.

I^2C protokolünde genellikle 7 bitlik adresleme kullanılır fakat 10 bitlik adresleme yapmakta mümkündür. Sunmuş olduğu adresleme sistemi, bir yönetici ile birden fazla köleyi yönetmek mümkün kılar. Ayrıyeten bir sistemde birden fazla yönetici olmasına da izin verir. Bu sistemde tek koşul tüm aygıtların tek bir GND hattına bağlı olmasıdır.

I^2C'de farklı hızlarda iletim yapmak mümkündür, bunların en yaygın olanları 100kHz, 400kHz ve 3.4MHz'dir.

I^2C ile uzun mesafeli iletişim yapmak mümkün değildir. Önerilen en uzun iletim mesafesi 4m'dir.

Biz bu çalışmamızda I^2C haberleşme protokolü yardımıyla dijital analog dönüştürücü ile mikrodenetleyicimiz arasında veri aktarımı gerçekleştirdik. Bizim sistemimizde mikrodenetleyicimiz yönetici rolünde iken dönüştürücümüz ise köle rolündedir. Çalışmanın sonunda alınmak istenen sonuç ise gönderdiğimiz veriye göre dönüştürücünün çıkışından istediğimiz gerilim değerini okumaktır. Bunu doğru bir şekilde yapabilmemiz için kullanmış olduğumuz dönüştürücü ve mikrodenetleyicinin kullanıcı kılavuzlarını iyi bir şekilde kavramış olmak önemli. I^2C'de adresleme önemlidir eğer doğru adresi veri yolundan gönderemezsek haberleşmemiz mümkün olmayacaktır. Adreslerin belirli bitleri statik olarak kullanıcı kılavuzlarında yazarken belirli bitleri dinamik olarak kullanıcı tarafından ayarlanabilmektedir.

Bir I^2C seri haberleşme protokolü yazılırken dikkat edilmesi gereken en önemli hususlardan bahsedecek olursak bunlardan ilki zamanlamadır. SDA ve SCL pinlerinin logic level ayarlarından sonrasında atacağımız gecikme süreleri önem arz etmektedir. Diğer önemli bir husus ise kölenin bizden beklediği verilerdir. Bu verilerden ilki ve standart olanı adrestir. Veri hattına bir adres gönderilir ve gönderilen adres hangi köleye aitse o aygıt ile haberleşmeye devam edilir. Adresten sonra cihazın istediği başka herhangi bir veri varsa o gönderilir. Örnek olarak bizim kullanmış olduğumuz dijital analog dönüştürücü bizden adresten sonra bazı konfigürasyona bitleri istemektedir. Bu konular I^2C için temel ve önemli hususlardır.

4.3 I^2C İletişim Adımları

4.3.1 Hattın Durum Sorgusu

Eğer iletim hattında herhangi bir veri transferi gerçekleşmiyorsa bu duruma IDLE denir. Bu durumda SDA ve SCL pinleri lojik 1 durumdur. Haberleşmeye başlamadan önce bu durumun sorgulanması önem arz etmektedir aksi durumda veri transferinde hata meydana gelecektir.

4.3.2 Başlangıç Aşaması

I^2C'de haberleşmeye başlamadan önce başlangıç bilgisi sisteme verilmelidir. Bu işlem SCL pini lojik 1 durumda iken SDA pininin lojik 1 durumdan lojik 0 duruma geçmesidir. Bu işlem yönetici tarafından gerçekleştirilmektedir.

4.3.3 Adresleme Aşaması

Bu aşamada ulaşılmak istenen köleye yönetici tarafından kölenin adresi gönderilir buna ek olarak işlemin okumamı yoksa yazmamı olacağının da bilgisi gönderilmektedir. Genelde 7 bit adres iken sonuncu bit okumamı yoksa yazmamı olacağını köleye bildirmektedir.

4.3.4 Veri İletim Aşaması

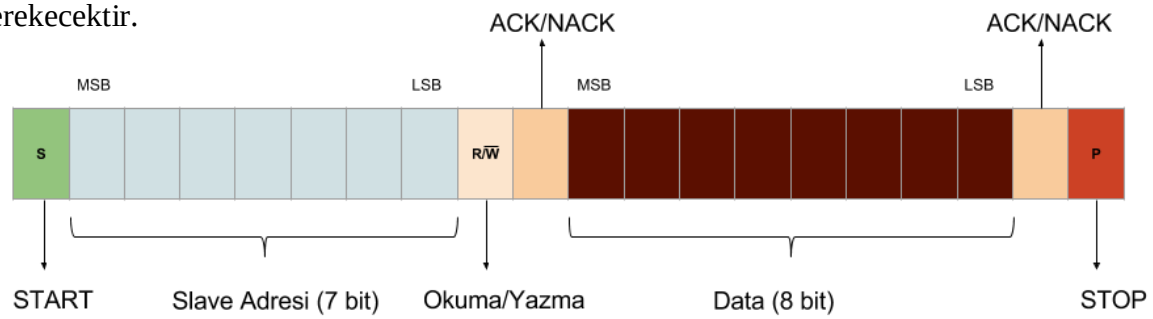
Gönderilen bir baytlık verinin her biti tek tek SDA hattı üzerinden köleye iletilir. Bu iletim aşamasında SCL pini büyük rol oynamaktadır ve iletim sırasında senkronizasyonu sağlamaktadır.

4.3.5 Sonlandırma Aşaması

Veri gönderildikten sonra bağlantının koparılması için işlemin bitmiş olduğuna dair bilgi sisteme verilir. Bu işlem SCL lojik 1 durumda iken SDA hattının lojik 0 durumdan lojik 1 duruma geçmesi ile gerçekleşir. Bu işlemde yine yönetici tarafından gerçekleştirilir.

4.3.6 ACK/NACK

Her bir adımdan sonra karşı taraftan ACK biti beklenir bu bit bize verinin doğrulandığının bilgisini verir. Eğer ACK biti dönmezse (NACK durumu) işlem sırasında herhangi bir hata meydana geldiğini anlayabiliriz. Ve haberleşme işlemi o kısımda kesilir işlemin tekrarlanması gerekecektir.



5. C DİLİ İLE I^2C

5.1 C DİLİ İLE I^2C SERİ HABERLEŞME ARABİRİMİNİN GERÇEKLEŞTİRİLMESİ

Biz bu çalışmamızda "MCP4725" serisi olan dijital analog dönüştürücümüzü "DSPIC33EP512MC806" mikrodenetleyicimiz ile kontrol ettik. Bu kontrol aşamasında I^2C seri haberleşme protokolünü kullandık. Normalde çoğu derleyici hazır fonksiyon olarak bu protokolleri kullanıcıya sunmakta fakat biz bu çalışmada bir seri haberleşme protokolünün nasıl işlediğine değineceğiz ve bir haberleşme protokolünün bir gömülü sistemde nasıl kodlanabileceğini anlatacağız.

Bu çalışmada C dili kullanılmıştır bunun gerekçesi gömülü sistemlerde bize sunmuş olduğu çalışma hızından dolayı C dili tercih edilmektedir. Normalde daha hızlı diller mevcuttur fakat bu dillerin kullanımı çok daha zor olduğu için günümüzde tercih edilmemektedir. Bir dil makine diline ne kadar yakınsa o kadar hızlı bir dil diyebilir fakat makine dili insanlar için kullanımı zor bir dildir bu yüzden C dili günümüzde halen kullanılmaya devam eden makine dili ve gelişmiş diller arasında bulunan hem kolaylık hem de hız sağlayan programlama dilidir.

Bu çalışmada kodlar "CCS C" derleyicisinde yazıldı fakat herhangi bir başka derleyicide kullanılabilir. Derleyiciden derleyiciye bazı temel değişiklikler meydana gelebilmektedir bundan dolayı farklı bir derleyicide bu çalışmadaki kodları çalıştırmadan önce kullanacağınız derleyicinin kullanıcı kılavuzunu okumanız önem arz etmektedir.

Yazmış olduğumuz kodu mikrodenetleyicimize PICKIT-4 yardımı ile attılmıştır. Sizde farklı bir programlayıcı kullanabilirsiniz. Programlamak için ise MPLAB X IPE kullanılmıştır.

Adımlar tek tek kontrol edilmek için ayrıyeten RS-232 bağlantısı ile RealTerm arayüzü üzerinden veriler tek tek takip edilmiştir. Bunu yapmak mecburi olmasa da yaptığımız işlemlerin doğruluğundan emin olmak için veya bize yol göstermesi amacıyla kullanabiliriz.

Bu yazımızda mikrodenetleyici nasıl programlanır, uygulamalar nasıl kullanılır gibi konulara değinilmeyecektir. Bu yazıda sadece I^2C seri haberleşme protokolünün C dili ile nasıl gerçekleştirilebileceği anlatılacaktır.

5.2 Kodlar

I2C_sample.c

I2C_sample.c* | setup.h | i2c_lib.c

```
1  #include"setup.h"                          //SETUP
2  #include"i2c_lib.c"                        //I2C KUTUPHANESI
3
4  #define dac_address          0x23          //DAC ADDRESS
5  #define SS1_A0               PIN_B11       //DAC1 A0
6  #define SS2_A0               PIN_B10       //DAC2 A0
7
8  void main()
9  {
10    unsigned int8 data = 0xFF;               //Veri
11    int1 ret;                                //ACK sorgulama için değişken
12
13    output_bit(SS1_A0, 1);                   //DAC A0 pini adres ayarı
14    output_bit(SS2_A0, 0);
15
16    output_high(SDA);                        //Başlangıçta SDA ve SCL çıkışlarını HIGH duruma getir
17    output_high(SCL);
18
19    start();                                 //Başlangıç bitini yolla
20
21    ret = write(dac_address);                //Adresi yolla
22    if (!ret) stop();                        //ACK bitini sorgula
23
24    write(0xF0);                             //Fast mod setup settings
25    write(data);                             //Veriyi yolla
26
27    stop();                                  //Stop bitini yolla
28
29    while(TRUE);
30 }
```

setup.h

I2C_sample.c* | setup.h* | i2c_lib.c

```
1  #ifndef SETUP_H_
2  #define SETUP_H_
3
4  #include<33EP512MC806.h>                   //Mikrodenetleyicinin kütüphanesini tanımla
5  #use delay(internal = 16MHz)               //Mikrodenetleyici için çalışma frekansı tanımla (DAHILI OSILATOR)
6
7  #endif
```

İ2c_lib.c

I2C_sample.c* | setup.h* | i2c_lib.c*

```
#ifndef I2CLIB_H_
#define I2CLIB_H_

#define SDA                 PIN_F4              //SDA ve SCL pinlerini belirle
#define SCL                 PIN_F5
#define DELAY               delay_us(5);        //Haberleşme hızına göre gecikme süresi belirle

int start(void)                                 //Başlama bitini gönderecek olan fonksiyon
{
   output_high(SDA);
   output_high(SCL);
   DELAY

   if(input_state(SDA) == 0) return 0;          //Hattın durumunu sorgular

   output_low(SDA);
   DELAY
   output_low(SCL);
}
```

```
void stop(void)                                 //Durdurma bitini gönderecek olan fonksiyon
{
   output_low(SDA);
   output_high(SCL);
   DELAY
   while(input_state(SCL) == 0);                //SCL pininin HIGH duruma geçmesini bekler
   output_high(SDA);
   DELAY
}
```

```
int write(unsigned int8 data)                   //DAC register veya eeprom a veri yazma fonksiyonu
{
   int1 ret;                                    //ACK ve NACK bitlerini döndürmesi için değişken

   for(int i=0; i<8; i++)                       //Veriyi bit bit yollayacak olan for döngüsü
   {
      if (data & 0x01) output_high(SDA);        //Verinin ilk bitini sorgular
      else output_low(SDA);

      DELAY
      output_high(SCL);
      DELAY

      data = data >> 1;                         //Veriyi bir bit sağa ötele

      output_low(SCL);
   }

   output_high(SDA);
   DELAY
   output_high(SCL);
   DELAY

   ret = input_state(SDA);                      //ACK veya NACK bitini ret değişkenine yaz

   output_low(SCL);
   DELAY

   return ret;                                  //ACK bitini döndür
}

#endif                          //I2C.lib
```

5.3 Kodların Açıklaması

5.3.1 Main Dosyası ve Mikrodenetleyicinin Konfigürasyon Ayarları

Yukarda resimde vermiş olduğumuz kodları tek tek açıklayacak olursak, sırasıyla gidelim; İlk başta bir adet main dosyasıyla tanımlayalım. Bu dosya bizim ana dosyamız olacak ve yapacağımız tüm işlemleri bu dosya üzerinden gerçekleştireceğiz. Kod okunurluğu açısından mikrodenetleyicinin konfigürasyon ayarlarını başka bir .h uzantılı dosyada gerçekleştirip o dosyayı main dosyamızda çağırdık. Bunu yapmak zorunda değiliz direk main dosyamızda da tanımlamamız mümkündür fakat kod okunurluğu açısından pek tercih edilmez. İkinci olarak yazmış olduğumuz I^2C haberleşme kütüphanesini çağırdık fakat buraya şimdi değinmeyeceğiz sonrasında uzun uzun anlatılacaktır bu kütüphane. Sonrasında sabitler olarak bir adet adres tanımladık, normalde adres verisi dışardan girdi olarak alınabilir fakat biz tek bir köle kullandığımız için adres verimizi sabit olarak tanımladık. Bu adres verisini kullanacağınız kölenin kullanıcı kılavuzundan edinebilirsiniz. Aynı zamanda bazı cihazlar adresten sonra farklı verilerde yöneticiden isteyebiliyor bu durumu da yine cihazın kullanıcı kılavuzundan öğrenebilirsiniz. Hemen altında ise dijital analog dönüştürücümüzün dinamik adres belirleyici pinleri yer almaktadır. Bu pinler ile kölemizin adresini ayarlıyoruz. Genel tanımlamalar bittikten sonra main() fonksiyonumuzun içine giriyoruz. Fonksiyonumuzun içinde verimiz için bir değişken ve ACK biti için bir değişken yer almaktadır. Altında ise dönüştürücünün adresi belirlenmiştir. Bu sizin kullanacağınız kölede farklılık gösterebilir fakat kullanıcı kılavuzundan yardım alabilirsiniz. Devamında SDA ve SCL pinlerinin lojik 1 duruma getiriyoruz. Standart olarak başlangıçta her iki pinde lojik 1 konumunda olmalıdır. Sonrasında sırasıyla I^2C kütüphanemizin içinde tanımlamış olduğumuz fonksiyonları tek tek çağırıyoruz. Sırasıyla başlangıç verisini yolla, adresi yolla, ACK bitini bekle, konfigürasyon ayarlarını yolla, veriyi yolla ve son olarak durdurma verisini yolla ve işlemi tamamla. Şimdi bu fonksiyonların içine derinlemesine girelim ve bir I^2C haberleşme kütüphanesinin nasıl yazıldığını hep beraber inceleyelim.

5.3.2 I^2C Kütüphanesi ve Fonksiyonlar

Kütüphanemizde ilk olarak SDA ve SCL pinlerini birer sabit olarak tanımlıyoruz. Bu pinler kullandığınız mikrodenetleyicide farklı bacaklar olabilir bunu yine denetleyicinizin kullanıcı kılavuzundan öğrenebilirsiniz. Onun ardından bir adet sabit olarak gecikme süresi

tanımlıyoruz. Bu gecikme süresi sizin haberleşmeyi kaç “kbps” hızda yaptığına göre değişecektir. Bu gecikmenin hesabını yapabilirsiniz veya sabit olarak bizimde kullanmış olduğumuz hız olan 100kbps hızını kullanabilirsiniz. Bu hız için ideal gecikme süresi 5 mikro saniyedir, “delay” komutu ile bu gecikmeyi vermek mümkündür.

5.3.2.1 start() Fonksiyonu

Bu fonksiyonumuz hattın dolu veya boş olduğunu sorgulayıp tekrar return ettiği için int türünde tanımlanmıştır. İlk olarak iletim hattımızı kontrol ediyoruz. Eğer iletim hattında herhangi bir veri transferi gerçekleşmiyorsa işlemimize devam ediyoruz eğer gerçekleşiyorsa işlemi baştan yapmak üzere return ediyoruz. Veri transferi olmadığı ve işlemin devam edeceğini varsayarak devam edecek olursak, başlangıç koşulumuz olan SDA ve SCL lojik 1 durumda iken SDA pinimizi lojik 1 durumundan lojik 0 durumuna getiriyoruz ve beklemeye giriyoruz bu sürede bizim başlangıç verimiz hattan iletilmiş oluyor sonrasında ise SCL pinimizi kullanıma hazır hale getirmek için lojik 0 durumuna getiriyoruz ve böylece başlangıç fonksiyonumuzu bitiriyoruz. Bu fonksiyon haberleşmenin başlayabilmesi için ilk ve en önemli koşuldur. Bu fonksiyonda yapılacak bir hatada diğer fonksiyonlarımız doğru dahi olsa haberleşme gerçekleşemeyecektir. Yazılması basit bir fonksiyondur dikkat edilmesi gereken en önemli nokta gecikmelerin doğru yere atılmasıdır. Şimdi sırasıyla diğer fonksiyonlarımızdan devam edelim.

5.3.2.2 stop() Fonksiyonu

Bu fonksiyonumuz haberleşme işlemimiz bittikten sonra haberleşmeyi kesmek amacıyla çağrılmaktadır. İlk olarak durdurma koşulumuz olan SDA pinimizi lojik 0 durumdan lojik 1 duruma geçirmeden önce SDA pinimizi lojik 0 duruma alıyoruz. Sonrasında SCL pinimize lojik 1 ataması yapıyoruz ve gecikme süremizi ekliyoruz. Sonrasında ise SCL pinimizin tekrardan 1 duruma geçtiğinden emin olana dek sorgulama işlemine devam ediyoruz. Artık SDA pinimizi lojik 0 durumdan lojik 1 durumuna geçirebiliriz. Bu işlemden sonrada gecikme süremizi ekliyoruz ve fonksiyonumuz tamamlanıyor. Bu fonksiyonu önemli kılan nokta ise özellikle çoklu köleye sahip olan sistemlerde bir köle ile haberleştikten sonra bağlantıyı durdurmazsak başka bir köleye veri gönderme işleminde hataya düşme ihtimalimiz yüksek olacaktır.

5.3.2.3 write() Fonksiyonu

Son ve en uzun fonksiyonumuz, normalde bir I^2C kütüphanesinde okumak içinde bir fonksiyon yer alır fakat biz bu çalışmamızda o konuyu ele almayacağız. Bu fonksiyona benzer şekilde okuma fonksiyonu da yazmak mümkündür temel yapısı aynıdır. Şimdi veri yazma fonksiyonumuzu inceleyecek olursak, int türünde bir fonksiyon tanımlıyoruz ilk olarak. Bu fonksiyonumuz parametre olarak 8 bitlik veri almaktadır. Bu veri adres olabilir, konfigürasyon ayalarının bilgisi olabilir veya iletmek istediğimiz ana bilgi olabilir. Tüm durumlar için bu fonksiyonu çağırarak işlemimizi gerçekleştiriyoruz. Fonksiyonumuzun içinde ilk olarak ACK bitini döndürmesi için bir adet değişken tanımlıyoruz. Bu değişken bize işlemin başarılımı başarısız mı olduğunun bilgisini döndürecektir. Daha sonrasında 1 baytlık verimizi bit bit göndermek için bir for döngüsü tanımlıyoruz ve döngümüzün içinde verinin birinci bitinin lojik durumunu sorgulatıyoruz. Verimizin lojik durumuna göre de SDA yani veri gönderme hattımızı lojik 1 veya lojik 0 durumuna getiriyoruz ve gecikme süremizi ekliyoruz. Daha sonrasında SCL pinimizi lojik 1 duruma getiriyoruz ve tekrar gecikmeye giriyoruz. Aynı işlemi her bit için tekrarlamak için verimizi bir bit sağa kaydırıyoruz ve artık verimizin ikinci biti birinci biti haline geliyor ve sonraki döngüde artık ikinci biti almış oluyoruz böylece bayt olarak gelen verimizi bitler halinde SDA hattından iletmiş oluyoruz son olarakta tekrardan SCL pinimizi lojik 0 durumuna getiriyoruz ve ilk bitimizi karşı tarafa göndermiş oluyoruz bu işlemi 8 bit içinde tek tek tekrarlıyoruz for döngümüz sayesinde. Veri gönderme işlemi sonrasında SDA ve SCL hatları gerekli gecikmeler eklenerek tekrardan lojik 1 durumuna getiriliyor ve ACK biti sorgulanıyor, SCL pini tekrardan lojik 0 durumuna getirilip gecikme eklendikten sonra ACK bitimizi return ederek fonksiyonumuzu tamamlamış oluyoruz. Artık fonksiyonumuza parametre olarak girmiş olduğumuz veri karşı tarafa iletilmiş oldu eğer işlemler sırasında herhangi bir hata meydana gelmediyse.

Bu fonksiyonumuzu ana dosyamızda görüldüğü üzere üç kere çağırdık. Bunlardan ilki dönüştürücünün adresi için, ikincisi dönüştürücünün konfigürasyon ayarı için üçüncüsü ise dönüştürücüye göndermek istediğimiz veri içindir. Bu işlemler sizin kullanmış olduğunuz köle aygıtta farklılık gösterebilir bundan dolayı kullanmış olduğunuz köle aygıtın kullanıcı kılavuzunu okumanız önem arz etmektedir.

Son fonksiyonumuzu da tamamladığımıza göre artık haberleşmeye başlamaya hazırız. Sizde benzer adımları takip ederek kendinize bir haberleşme sistemi kurabilir kendi kütüphanenizi yazabilirsiniz.

SONUÇ

Yapmış olduğumuz bu çalışmanın sonucunda denetleyicimizi yönetici olarak seçip, köle olan dönüştürücümüzü yönettik. Dönüştürücümüze gönderdiğimiz veriler ile dönüştürücümüzün çıkış gerilimini ayarladık. Bizim devremizde dönüştürücümüz 3.3V'da çalıştığı için verinin tüm bitlerinin 1 olduğu durumda 3.3V çıkış alırken verimizin tüm bitlerinin lojik 0 olduğu durumda ise çıkış gerilimini 0V olarak okuduk. Haberleşmeyi başarılı bir şekilde gerçekleştirmiş olduk. Bu çalışmayı geliştirilerek kendi projelerinizde kullanabilirsiniz. Bu alanda literatürdeki eksik olan Türkçe kaynaklara bir yeni alternatif kaynak hazırlamış olduk. Bu çalışma ile birlikte haberleşme sistemleri ve protokolleri hakkında yeni araştırma yapan bir lisans veya lisans öncesi öğrencinin konuyu kolayca kavrayabileceği bir kılavuzda yazmış olduk.

REFERANSLAR

[1] wikipedia.com

[2] urhoba.net

[3] cennttceylnn.medium.com

[4] demirten.gitbooks.io

[5] “DSPIC33EP512MC806” User Manuel

[6] “MCP4725A1T-E/CH” User Manuel

[7] CCS C User Manuel

KULLANILAN PROGRAMLAR

[1] CCS C

[2] MPLAB X IPE

[3] REALTERM