YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CAN HABERLEŞME PROTOKOLÜNÜN İNCELENMESİ VE BİR SICAKLIK KONTROL SİSTEMİNE UYGULANMASI

Elektronik ve Haberleşme Mühendisi İsmail KARA

FBE Elektrik Mühendisliği Anabilim Dalı Kontrol ve Otomasyon Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Şeref Naci ENGİN

İSTANBUL, 2009

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

ADC	Analog sayısal dönüştürücü (Analog Digital Converter)
CAN	Kontrol Alan Ağı Veri Yolu (Controller Area Network Bus)
CPU	Merkezi işlem birimi (Central Processing Unit),
SISO	Tek giriş tek çıkış (Single İnput Single Output)
MCU	Mikro denetleyici birimi (MicroController Unit)
LCD	Likit kristal gösterge
PC	Kişisel bilgisayar (Personal Computer)
RAM	Rasgele erşilebilen bellek (Random Access Memory)
ROM	Sadece okunabilir bellek (Read Only Memory)

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Çalışmalarım sırasında bana her zaman destek olan beni motive eden sevgili eşim İffet Kara'ya ve yardımlarını esirgemeyerek bilgi ve tecrübeleri ile bana her zaman destek olan değerli hocam ve danışmanım Sayın Yrd. Doç. Dr. Şeref Naci Engin'e sonsuz teşekkürlerimi sunarım.

Mayıs 2009

ÖZET

CAN HABERLEŞME PROTOKOLÜNÜN İNCELENMESİ ve BİR SICAKLIK KONTROL SİSTEMİNE UYGULANMASI

İsmail KARA

Elektrik Mühendisliği, Yüksek Lisans Tezi

Günümüzde birçok kontrol uygulaması birden çok kontrolcüye ihtiyaç duymaktadır. Bu kontrolcülerin birbirleri ile güvenli, hızlı ve düşük maliyetli haberleşmesi büyük önem taşımaktadır. Bu ihtiyaca en iyi cevap veren CAN haberleşme protokolü ile bir kontrol uygulaması yapılmıştır. Bu uygulama gerçeklenirken her bir CAN düğümünde sistem güvenliği açısında zaman tetiklemeli işletim sistemi kullanılmıştır. Kontrol uygulaması olarak sıcaklık kontrolü seçilmiş ve sıcaklık sabit tutmak için PID kontrol kullanılmıştır. Tasarımda gerçeklenen yazılımın daha sonra farklı sistemlerde de kolaylıkla uygulanabilmesi amacıyla, modüler, esnek ve anlaşılabilir olmasına dikkat edilmiştir.

Anahtar Kelimeler: CAN haberleşme protokolü, zaman tetiklemeli işletim sistemi, PID kontrol

JÜRİ

1. Yrd. Doç. Dr. Şeref Naci ENGİN
2. Prof. Dr. Herman SEDEF
3. Yrd. Doç. Dr. Kayhan GÜLEZ

Kabul Tarihi: Haziran 2009

Sayfa Sayısı: 66

ABSTRACT

A REVIEW OF CAN COMMUNICATION PROTOCOL and ITS APPLICATION TO A TEMPERATURE CONTROL SYSTEM

Nowadays many control applications need much more than one controller. There are tremendous efforts in the industry to establish communication between these microcontrollers safely, rapidly and economically. The problem is solved with CAN communication protocol in this thesis. When the application is realized, time triggered operating system is used in each CAN node to provide system security. The temperature control application is chosen as a case study and a PID controller is implemented to keep temperature in the set point. In order to make the system portable, so that the designed system could be integrated with other systems conveniently, modularity, readability and flexibility are taken into consideration in the development of the source code.

Keywords: CAN communication protocol, Time Triggered Operating Systems, PID control

JURY

1. Asst. Prof. Dr. Şeref Naci ENGİN
2. Prof. Dr.Herman SEDEF
3. Asst.Prof.Dr. Kayhan GÜLEZ

Acceptance Date: June 2009
Number of Page: 66

1. GİRİŞ

Bu tezde CAN haberleşme protokolünün incelenmesi, veri yolu kullanarak bir mikro denetleyici ağ yapısının kurulması ve kurulan bu ağ yapısı üzerinde bulunan denetim sistemlerinin bir ana bilgisayar tarafından kontrol edilmesi hedeflenmiştir. Bu ağ yapısı üzerinde kullanılan gömülü yazılımın gerçeklenmesi içinde zaman tetiklemeli bir işletim sistemi tasarlanmıştır.

CAN haberleşme protokolü Robert Bosch tarafından otomotiv sektöründe kullanılmak üzere 1980'li yılların sonuna doğru geliştirilmiş 'akıllı' aygıtları bir sistem veya alt-sistem içinde birbirleriyle haberleştiren seri haberleşme protokolüdür. Multi-Master yöntemi uyumlu sistemlerde kullanılır. Her CAN düğüm (aygıt) birbirleriyle haberleşebilir ve eşzamanlı cevaplar alabilir. Gerçek-zamanlı sistemlerde de kullanılır. ISO 11898 ile uluslararası standartları belirlenmiştir ve ISO/OSI referans modelinin en alt iki katmanının da temelini oluşturur. CAN ağlarında abonelerin veya istasyonların geleneksel anlamda adreslenmesi yapılmaz, onun yerine öncelikli mesaj iletimi yapılır.

Bir verici bütün CAN düğümlerine mesaj yollayabilir. Her düğüm mesajın tanım kısmına bakarak işleyebileceği bir mesaj olup olmadığım anlar. Tanım kısmı ayrıca mesajın önceliğini de belli eder. CAN protokolünün kolay tarafı, az maliyet ve eforla kişisel uygulamalarda kullanılabilmesidir. CAN çip ara yüzleri uygulama programlamasını kolayca yapar. Tanıtım kursları, fonksiyon kütüphaneleri, başlangıç kitleri, I/O ara yüzleri ve araçları, ucuz maliyetli olduğundan CAN ağ yapısı tercih edilerek yapılır.

Kontrol edilecek denetim sistemi olarak bir alüminyum kütlesinin sıcaklığının sabit tutulması amaçlanmıştır. Ağ yapısı üzerinde üç adet CAN düğüm oluşturulmuş ve bunların birbirleriyle haberleşmeleri sağlanmıştır.

Yukarıda bahsedilen sistemin kurulması için oluşturulan yazılım da bu tezin önemli bir kısmını kapsamaktadır. Gömülü sistemlerin çoğunda zamanlama da önemli olduğundan gömülü sistemlerde kullanılan işletim sistemlerinde zamanlama önemli bir faktör olarak işin içine girdi. Gerçek zamanlı işletim sistemleri gömülü sistemlerde kullanılmaya başlanması ile sonsuz döngülü klasik bir yazılım mantığı yerine çoklu görevlerle uğraşabilen, sanki birden fazla işlemci varmış gibi çalışan yazılımlarla tasarım yapılmaya

başlandı. Gerçek zamanlı işletim sistemleri artık 8 bitlik küçük işlemcilerle dahi kullanılabilmektedir. Bu tezin bir diğer amacı da böyle bir işletim sistemi ve gömülü sistem tasarımı üzerine yöneliktir. 3. bölümünde gömülü sistem tasarımında karşılaşılan kritik zamanlamaların yakalanması ve tasarım sürecinin kısaltılması gibi problemlere çözüm aranmaya çalışılacaktır.

CAN haberleşme protokolü hakkında detaylı bilgi 2. bölümde verilmektedir. Bu bölümde veri yolunu kullanmanın avantaj ve dezavantajları üzerinde durularak, veri yolunun yapısı donanımsal ve yazılımsal yönleri belirtilecek ve veri yolunun standartlarından bahsedilecektir. Gömülü sistemler bu sistemlerin alt bölümü olan zaman tetiklemeli işletim sistemleri 3. ve 4. bölümde; yapılan deneysel düzeneğin gerçeklenmesi burada kullanılan kontrol sisteminin açıklanması 5. bölümde; en son olarak deneyde elde edilen sonuçlar, CAN veri yolunun sunduğu avantaj ve dezavantajlar sonuçlar bölümü olan 6. bölümde yer almaktadır.

2. CAN HABERLEŞME PROTOKOLÜNÜN İNCELENMESİ

Bu bölümde CAN haberleşme protokolü üzerinde detaylı bilgi verilecek, veri yolu topolojisi açıklanacak ve haberleşme de kullanılan CAN çerçeve yapıları anlatılacaktır. En son bölümde ise CAN hata denetiminden bahsedilecektir.

2.1 CAN Haberleşme Protokolünün Tanımı

Kelime anlamı olarak Controller Area Network kelimelerinin kısaltılması anlamına gelir. Aşağıda CAN haberleşme protokolünün genel özelliklerinden bahsedilmiştir.

- 2 kablo üzerinden seri haberleşme metodu kullanılır.
- 1980'lerin ortasında Robert Bosch tarafından otomotiv endüstrisi için geliştirilmiştir.
- Gerçek zamanlı uygulamalarda ağ elemanları arasında yüksek kapasiteli veri alış verişi sağlamak ve bunu yaparken mümkün olduğunca ekonomik çözümler sunmak CAN haberleşme protokolünün ana hedefidir.
- CAN haberleşme protokolü tüm dünyada standartlarla belirlenmiştir. Bu standartlara örnek olarak
    - CAN 2.0A: ISO 11519 – düşük hızlı
    - CAN 2.0B: ISO 11898 – yüksek hızlı
    - CAN onaylama: ISO 16845
- CAN haberleşme protokolünün kullanımı 200 milyon düğüme ulaşmıştır ve her geçen yıl bu pazar %30 büyümektedir. CAN sadece otomotiv sanayinde kullanılmakla kalmayıp birçok endüstri alanında kullanılmaktadır. Bunlardan en önemli iki tanesi otomotiv ve endüstriyel makinelerdir.

İki kablo üzerinden kontrolcülerin birbirleriyle iki yönlü olarak haberleşmeleri için kurulan network ağı "The Controller Area Network (CAN)" olarak isimlendirilir. CAN haberleşme protokolü 1980'li yılların ortalarında Robert BOSCH tarafından noktadan noktaya haberleşmenin yerini almak için geliştirilmiştir.

Gerçek zamanlı uygulamalarda yüksek kapasiteli ve doğrulukta veri transferi CAN haberleşme protokolünün genel karakteristiğini oluşturur. Ayrıca gürültülü ortamlarda yüksek doğrulukta güvenilir veri transferi sağlar.

CAN haberleşme protokolünün özellikleri dünya standartlarında iki versiyon olarak kullanılmaktadır. Birinci versiyon, CAN 2.0A düşük hızlı versiyonudur ve temel CAN ya da standart CAN olarak isimlendirilir. Bu standart ISO'nun ISO11519 standartında belirtilmiştir. Diğer versiyon olarak CAN 2.0B ise yüksek hızlı versiyondur ve bu versiyon tam CAN ya da genişletilmiş CAN olarak da isimlendirilir. Bu standart da ISO11898'de tanımlanmıştır. ISO16845 ise CAN haberleşme protokolünün içermesi gereken bilgilerin tanımlı olduğu standardıdır.

CANBUS otomotiv sektörü haricinde birçok alanda kullanılmakta olup kullanım miktarı her geçen yıla oranla %30 artmaktadır. 2001 yılına kadar 200 milyon CAN düğümü kullanılmıştır.

2.2 CAN Haberleşme Protokolünün Avantajları, Tarihçesi ve Uygulama Alanları

Bu bölümde CAN haberleşme protokolünün gelişim süreçlerine değinilecek, neden bu protokolün kullanılması gerektiği üzerinde durulacak ve bugün birçok endüstride kullanılan bu protokolün kullanım alanları anlatılacaktır.

2.2.1 CAN Haberleşme Protokolünü Kullanmak İçin Ana Sebepler

- Güvenilir.
    - Hatasız haberleşme sağlaması
- Ekonomik
    - Kablo masraflarını azaltması
    - Düşük donanım maliyeti sağlaması
- Ölçeklendirilebilirlik
    - Kolay genişletilebilirlik
    - Düğümlerin birbirine bağlama maliyetini düşürmesi
- Uygulanabilirlik

- Çoğu kontrolcünün CAN portuna sahip olması
- Birçok yazılım aracı tarafından desteklenmesi
- Yüksek seviyeli protokol

- Bilinirlik
  - Geliştirme ortamı için geniş kaynak imkanı

Yukarda saydığımız 5 sebep üzerinde kısaca açıklama yapalım.

Güvenilirlik: Hatasız veri alışverişi sağlamak, ileri dereceli güvenilir veri gönderip almak CAN tasarlanırken en üst seviyede sağlanmıştır. Bu sebep bazı güvenlik uygulamalarında ve gömülü sistemlerde olmazsa olmaz bir koşuldur.

Ekonomik: CAN haberleşme için sadece 2 adet kablo kullandığı için network ağı maliyeti oldukça düşüktür. Ayrıca, mikro kontrolör içinde CAN çevre birimleri fazla yer kaplamaz böylece daha düşük maliyetli kontrolcü üretme imkanı oluşur.

Ölçeklendirilebilirlik: CAN ağ yapısı her zaman genişletmeye müsaittir. Veri yolunun herhangi bir noktasına düğümler eklenebilir ve bu ekleme diğer düğümlerde değişiklik gerektirmez. CAN ağ yapısında düğümü network yapısına bağlamak için sadece iki bağlantı noktası yeterlidir. Buda şüphesiz bir avantajdır. Diğer sistemlerde bir ağ yapısına bir düğüm eklemek için çok daha fazla kablo kullanmak gerekir.

Uygulanabilirlik: Gün geçtikçe CAN özellikli mikro kontrolör sayısı artmaktadır. Bununla birlikte yazılım kısmında da CAN hata ayıklama araçları geliştirilmektedir. Bunların yanında yüksek seviyeli standartlaşmış protokoller karmaşık sistemlerin tasarımı için yazılımcıya büyük kolaylık sağlar.

Bilinirlik: CAN haberleşme protokolü 15 yılı aşkın bir süredir kullanılmaktadır ve bu konu üzerinde bilgi düzeyi oldukça artmıştır ve her gecen gün veri tabanı büyümektedir.

2.2.2 CAN Haberleşme Protokolünün Tarihçesi

Açılımı “Controller Area Network Bus” olan yani “Kontrol Alan Ağı Veri yolu” dur.1980’lerde Robert Bosch tarafından otomotivde kablo yumağı yerine bir kablodan yazılım kontrollü veri transferini sağlamak amacıyla geliştirilmiştir. Böylelikle kablolama maliyetleri düşmüş otomobiller daha ucuza imal edilmeye başlamıştır. CAN, otomotiv

endüstrisindeki en bilinen haberleşme sistemidir. Her ne kadar başlangıçta yalnızca otomotiv uygulamaları için tasarlanmış olsa da yüksek performansı güvenirliliğinden dolayı birçok dağıtık (distrubuted) endüstriyel kontrol uygulamalarında yaygın olarak kullanılmaktadır. 90'larda CAN'in gelişimi sonucunda CiA denen (CAN in Automation) kuruldu. Bu bağımsız grup CAN özelliklerini belirlemekte ve gerçekleştirmektedir. Daha sonra DeviceNET geliştirildi. DeviceNet endüstriyel cihazlarını (sensörs, aktuatör) yük seviye cihazlarına (kontrolör) bağlamaya yarayan düşük seviye networktür. DeviceNet özellikle düşük maliyet üzerine yoğunlaşmıştır. 95'de CAN2.0B geliştirilerekten bir CAN sistemine bağlı ünite sayısı teorik olarak 500 milyona çıkarılmıştır. 1996'da CANopen geliştirilmiştir. Böylece uygulamada kullanılabilirliği daha da artmıştır.

2.2.3 CAN Haberleşme Protokolünün Uygulama Alanları

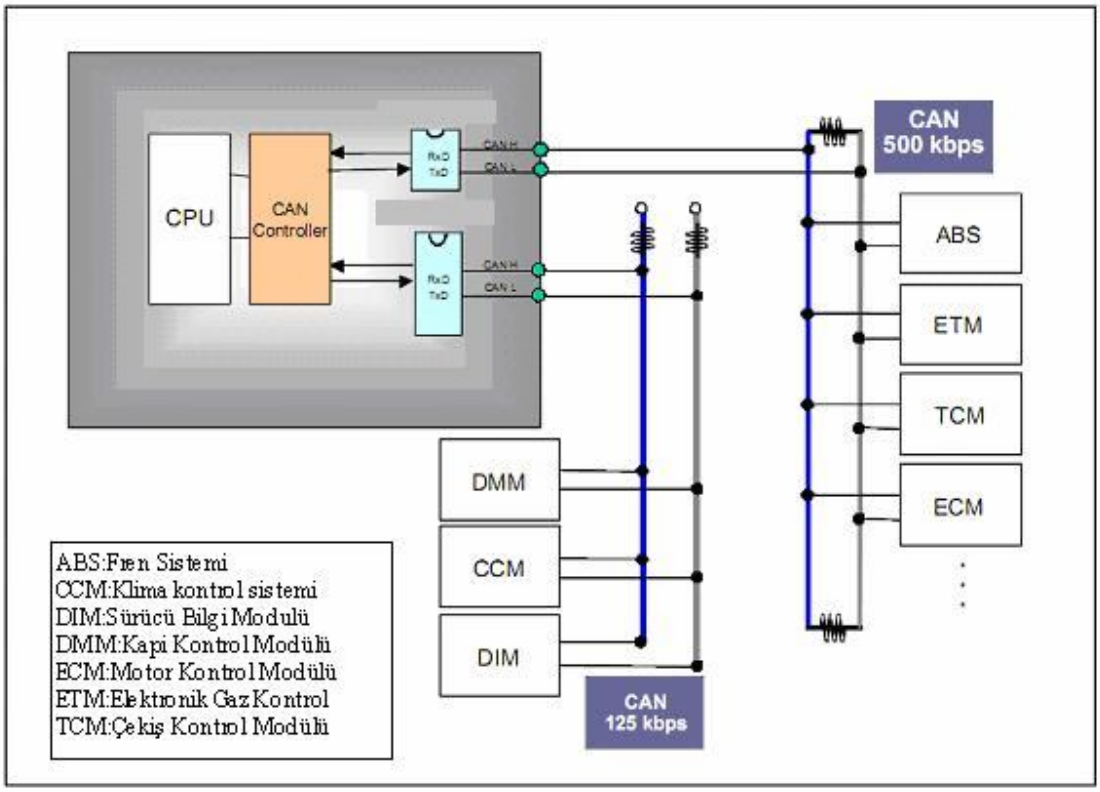
- Bütün elektronikle kontrol edilen cihazlar ve paketleme makinelerinde
- Endüstriyel dondurucular ve yazıcılarda
- Gemiler, trenler ve raylı sistemlerde
- Tarım ve inşaat makinelerinde
- Yarıiletken üretici cihazlarında
- Bina otomasyonunda, HVAC sistemleri, kat asansörleri
- Hastanelerdeki hasta takip kontrolünde

Daha birçok alanda CAN haberleşme protokolü kullanılır. Yukarıdaki liste CAN haberleşme protokolünün ana olarak kullanıldığı otomotiv endüstrisi dışındaki kullanım alanlarını göstermektedir.

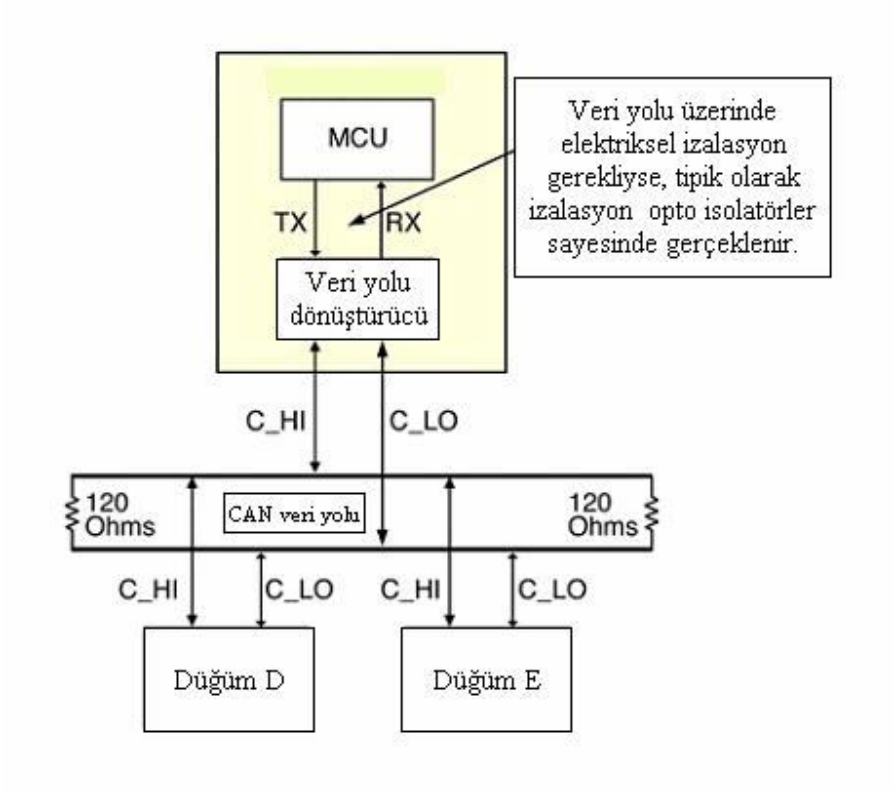
2.3 Veri Yolu Topolojisi

CAN haberleşme sisteminde kontroller iki kablo arasındaki potansiyel farka bakarak gelen veri hakkında karar verir. Veri yolu üzerinde iki seviye mevcuttur. Bunlar baskın ve çekinik seviyelerdir ve bu iki seviye aynı zamanda veri yolu üzerinde olabilirler. Gönderen kontrolcü, bu veri yolu seviyelerini değiştirerek alıcıya mesajını gönderir. Şekil 2.1'de tipik bir CAN bağlantı şekli gösterilmektedir. Veri yolu üzerine bağlı tüm

düğümler veri yoluna mesaj gönderebilirler. Eğer aynı anda iki düğüm veri yoluna mesaj gönderirse, mesaj önceliği olan düğümün mesajı gönderilir ve diğer düğümler alıcı durumundadırlar.



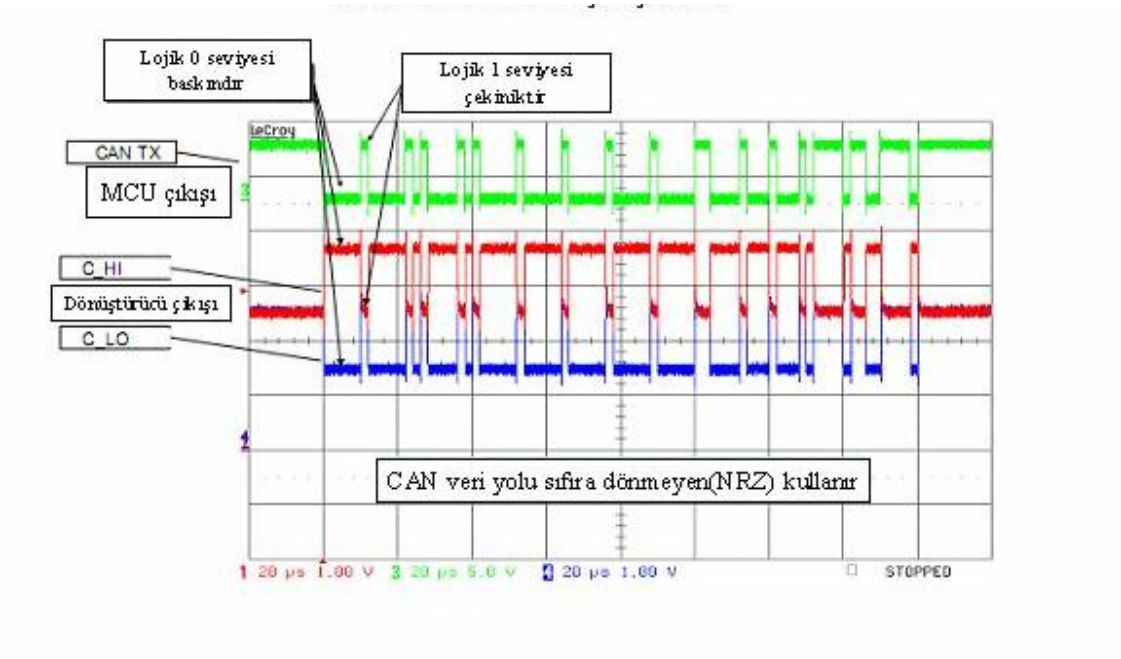
Şekil 2.1 Tipik bir otomobil CAN bağlantı şekli



Şekil 2.2 Fiziksel bir CAN düğümü

Şekil 2.2'de gösterilen fiziksel ara yüzde baskın seviyeyi elde etmek için C_Hi hattı yüksek seviyede, C_LO hattı da düşük seviyede olacak şekilde aynı anda elektriksel sinyal uygulanır. Bunun sonucunda dinamik olarak iki hat ucu arasında potansiyel fark oluşur. Çekinik durumda ise C_Hi ve C_LO hatları yüksek empedans durumundadır. Bu durum veri yoluna veri gönderilmiyorsa her zaman görülür. Buradaki çekinik durumunu elde etmek için veri yolu 120 Ohm'luk bir sonlandırma direnciyle sonlandırılmalıdır. Bu direnç değeri ISO 11898-2 standartında belirtilmiştir.

Çoğu uygulamada buradaki veri yolunun elektriksel olarak izole edilmesi istenmektedir. Bu da optik izolatörler aracılığıyla gerçeklenir. Burada amaç işlemciyi ve veri yolunu birbirinden izole etmektir. Burada normal olarak araya konan izolatörler CAN 2.0B'de standart olan 500 Kbps hızını azaltacaktır. Şekil 2.3'de veri çerçevesine bir örnek verilmiştir.



Şekil 2.3 Örnek veri çerçevesi

Yukarıda yakalanan çerçeve gerçek bir veri çerçevesidir. Yeşil renkle gösterilen en üstteki işaret işlemcinin CAN transmit ucundan çıkıp dönüştürücüye gelen sinyali gösterir. Diğer aşağıda gösterilen sinyaller ise bizim dönüştürücü ünitemizden çıkıp veri yoluna bağlanan hatları göstermektedir.

Bir baskın değer üretmek için (CAN_Tx= Logic Low) C_HI ve C_LO uçları arasında pozitif bir potansiyel fark oluşur. Çekinik seviyenin oluşması içinse C_HI ve C_LO uçları yüksek empedans durumundadırlar. Bu durumda her iki uç arasındaki potansiyel fark sıfır olacağından dolayı sonlandırma dirençleri üzerine gerilim düşmez.

Alıcı üniteler C_HI ve C_LO uçları arasındaki potansiyel fark 900 mV'un üzeride ise bunu baskın seviye olarak kabul ederler. Eğer bu iki uç arasındaki potansiyel fark 500 mV'tan az ise bunu da çekinik seviye olarak algılarlar.

Şekil 2.3'e baktığımızda ilk olarak işlemciden çıkan transmit sinyali normal lojik sevilerinde çalıştığı görülür. Burada 2.5 V'luk bir ofset gerilimi mevcuttur. Bu ofset seviyesi ISO 11898-2 standartlarını sağlamak için gereklidir. Burada CAN, NRZ (non-return-to-zero) denilen sıfıra dönmeyen bir transmit metodu kullanmaktadır. Bunlar hep dışarıdan gelen gürültüleri en aza indirmek amacıyla standartlara yerleştirilmiştir.

2.3.1 CAN Haberleşme Protokolünün Ana Özellikleri

(1) Çoklu Master: Çoklu master özelliği veri yolu boş iken bütün düğümler veri gönderebilirler anlamına gelir. Veri yoluna ilk veri gönderenin verisi kesin olarak gönderileceği garanti edilmiştir. Eğer iki düğüm aynı anda mesaj göndermeye çalışırlarsa, mesaj ID'si yüksek olan mesajın iletileceği protokolde garanti edilmiştir.

(2) Mesaj Transmisyonu: CAN haberleşme protokolünde mesajlar daha önceden karar verilmiş bir formatta gönderilirler. Eğer veri yolu boş ise veri yoluna tüm üniteler mesaj gönderebilirler. Eğer tüm üniteler aynı anda mesaj göndermeye çalışırlarsa, öncelik sırası bir belirleyici (ID) tarafından çözümlenir. Bu ID mesajın hangi düğüme gideceği konusunda herhangi bir bilgi içermez. Bu ID sadece mesajların hangi sıraya göre veri yolunda gönderileceğini belirler. Eğer iki veya daha fazla düğüm mesaj göndermeye karar verirse, buradaki mesaj ID'lerin bit bit karşılaştırılması sonucu karar verilir. Hangi düğüm bu karşılaştırma sonucunu kazanırsa onun mesajı gönderilir ve kaybedenler hemen dinleme konumuna geçerler.

(3) Sistem Esnekliği: Veri yoluna bağlı düğümler herhangi bir adres gibi belirtici ID ye sahip değillerdir. Bu sebepten dolayı, veri yoluna bir düğüm eklenip çıkarılması durumunda herhangi bir yazılım değişimine gerek yoktur. Bunun yanında diğer düğümlerin donanımlarında, uygulama alanlarında herhangi bir değişiklik yapmayı ortadan kaldırır. Bu da tasarlanan sistemin kolaylıkla genişletilmesini sağlar.

(4) Haberleşme Hızı: Network büyüklüğüne ve mesafelere göre haberleşme hızı ayarlanabilir. Aynı networkde bulunan tüm düğümler aynı haberleşme hızına sahip olmalıdırlar. Eğer herhangi bir düğüm farklı bir haberleşme hızı kullanırsa, haberleşmeyi engellemek için bir hata mesajı üretilir. Bu hata mesajı diğer networklerde bulunan düğümlere iletilmez.

(5) Uzaktan Veri İsteği: Diğer düğümlerden bir veri alma isteği gelebilir. İlgili düğüm istekte bulunan düğüme istediği bilgiyi gönderir.

(6) **Hata Belirleme, Hata Bildirme, Hatayı Düzeltme Fonksiyonları:** Ağ yapısına bağlı tüm düğümler hata belirleme özelliğine sahiptir (Hata Belirleme). Hatayı belirleyen herhangi bir düğüm bunu diğer düğümlere bildirebilir (Hata Bildirme). Eğer bir düğüm mesaj gönderirken bir hata belirlerse, mesaj transmisyonunu durdurmaya zorlayabilir ya da diğer düğümleri haberdar edebilir. Gönderdiği mesajı mesaj normal gönderilinceye kadar gönderebilir (Hata Düzeltme).

(7) **Hatayı Hapsetme:** CAN veri yolunda iki çeşit hata oluşabilir. Birincisi etrafta oluşan gürültüden veya başka sebeplerden oluşan düzensiz hatalar. İkincisi sürücü hatalarından, donanımın zarar görmesinden ya da buna benzer sebeplerden oluşan devamlı hatalar. CAN protokolü bu iki hata türünü birbirinden ayırt edebilen fonksiyona sahiptir. Bu fonksiyon hatalı düğümden gelen mesajlara düşük öncelik verir, düğümden devamlı bir hata üretiliyorsa bu üniteyi veri yolundan ayırt ederek düğümden gelen hata mesajlarını dikkate almaz.

(8) **Bağlantı:** CAN network yapısı birden fazla ünitenin aynı anda ağ yapısına bağlanmasına izin verir. Burada bağlanabilen ünite açısından mantıksal bir sınır yoktur. Fakat gerçekte ağ yapısına bağlanabilen ünite sayısı gecikme zamanlarına ve o anda ağ yapısında oluşan elektriksel yüke bağlı olarak değişebilir. Haberleşme hızı azaltılarak daha fazla ünite ağ yapısına bağlanabilirler. Bunun aksine, haberleşme hızı arttırılırsa ağ yapısına bağlanabilen ünite sayısı azalır.

2.4 Diğer Veri Yolu Sistemleri ve CAN Haberleşme Protokolü

CAN haberleşme protokolü haricinde kullanılan en yaygın protokol UART protokolleridir. Bunlardan en çok bilinenleri RS232 ve RS485'tir. Bunlar CAN protokolü ile karşılaştırıldığında bazı avantaj ve dezavantajlar içermektedir. Hemen her mikro kontrolcü UART protokolünü destekleyen bir donanıma sahiptir. Bu haberleşme noktadan noktaya haberleşmede kullanılır. Ekonomik olarak düşük maliyetli olduğu için birçok yerde kullanılır. Aşağıda bu sistemin iki olumsuz yönünden bahsedilecektir.

- UART sadece byte temelli haberleşme olanağı sağlar. Tek seferde birden fazla byte gönderme imkanı yoktur.

- UART'ın donanım yapısı hata denetimine sahip değildir. Burada hata denetimi için yazılım kullanılır bu da daha yavaş bir sistem anlamına gelmektedir.

1970'lerin sonlarında otomotiv endüstrisi bazı sorunlarla karşı karşıya kaldı. Noktadan noktaya haberleşme olduğu için kablolama maliyetleri, fazla kablo kullanıldığında araçların daha ağırlaşması ve daha fazla hacim gerekmesi yeni bir haberleşme protokolü ihtiyacını doğurdu. Burada araç üreticileri düşük maliyetli birden çok işlevi yapan kontrolcülerin bağlanabileceği bir seri veri yolu sistemi hakkından araştırmaya başladılar. Buna en uygun veri yolu sistemi olan CAN birçok ünlü otomobil üreticisi tarafından kullanılmaya başlandı.

Aşağıda CAN haberleşme protokolünün diğer haberleşme protokolleri ile karşılaştırıldığında öne çıkan özellikleri verilmiştir.

- CAN mesaj tabanlı haberleşme protokolüdür ve bir mesaj uzunluğu 8 byte'a kadar olabilmektedir.
- CAN donanımı hata denetleme özelliğine sahip olduğundan yazılım yükü azaltılmış olur.
- Bugün endüstride kullanılan birçok mikro kontrolcü bu protokolü desteklemektedir.
- CAN hem lokal hem de ayrışık sistem mimarilerinde kullanılabilir.
- 1 km'ye kadar veri taşımak mümkündür.

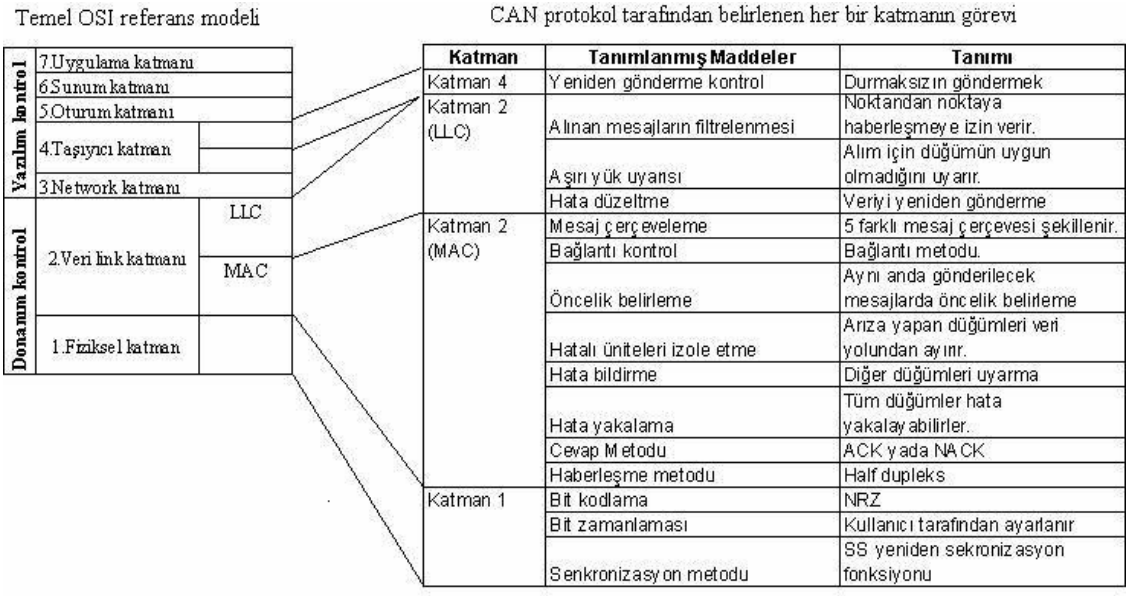
2.5 CAN OSI Model

CAN haberleşme protokolü OSI (Open System Interconnection)'de taşıyıcı katman, veri linki katmanı ve fiziksel katman olarak sınıflandırılmıştır. CAN haberleşme protokolü içerisinde her bir katmanın ne işe yaradığı Şekil 2.4'de ayrıntılı bir şekilde anlatılmıştır.

Veri linki MAC (Medium Access Control) ve LLC (Logical Link Control). Burada MAC katmanı CAN haberleşme protokolünün çekirdeğini oluşturur. Veri katmanı ise fiziksel katmandan mesajları daha anlamlı hale getirerekten örneğin bu bir transfer hata mesajıdır gibi, bir üst katmana iletmektir. Bunlara ek olarak aldığı bit mesajlarını çerçeve şekline, karşılaştırıcı çerçeveye ve benzeri veri mesajlarına dönüştürmek bu katmanın görevidir.

Bütün bu fonksiyonlar veri katmanında CAN kontrolcüsü tarafından donanımsal olarak gerçekleştirilir.

Fiziksel katman da ise bit zamanlamaları, bit çözümleme, senkronizasyon fonksiyonları gibi fonksiyonlar çalışır. Bununla birlikte, sinyalleşme seviyeleri, haberleşme hızı, örnekleme noktası değerleri, sürücü ve elektriksel karakteristikler ya da bağlantı elemanları CAN haberleşme protokolünde tanımlanmıştır anlamına gelmez. Burada bahsedilen bütün karakteristikler kullanıcı tarafından esnek bir şekilde belirlenir. Bosch'un standartlarında CAN haberleşme protokolünün elektriksel karakteristiği ve elektriksel sinyal dönüştürücülerin nasıl olacağı gibi konularda herhangi bir açıklama yapılmamıştır. ISO'nun ISO11898 ve ISO11519 standartlarında ise bütün bu ayrıntılar açıklanmıştır.



Şekil 2.4 OSI ve CAN protokolün temel referans modeli

2.6 CAN Protokol Standartları

CAN protokol standartları bölümünde, ISO tarafından belirlenen elektriksel ve protokole özel standartlardan bahsedilecektir.

2.6.1 ISO tarafından belirlenen CAN standartları

CAN haberleşme protokolünün standartlarını belirleyen kurum ISO'dur. Burada ISO'nun belirlediği birkaç standart vardır. Bunlara örnek olarak ISO11898 ve ISO11592-2

verilebilir. Bu iki standartta Data Link Layer da bir farklılık yoktur. Buradaki farklılık fiziksel katmanda görülmektedir.

(1) ISO11898

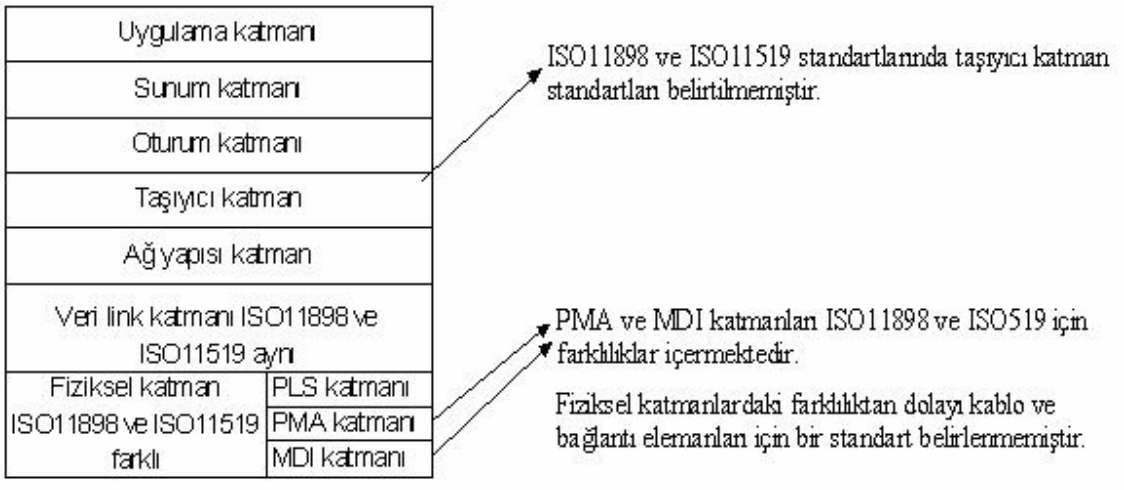
Bu standart yüksek hızlı CAN veri yolu için tanımlanmıştır. Bu standartta ilk başlarda sadece ISO11898 olarak adlandırılmıştır. Data ve fiziksel katmanlar aynı standart içerisinde tanımlanmıştır. Daha sonra bu iki katman birbirinden ayrılarak ISO11898-1 data katmanı, ISO11898-2 fiziksel katman olarak iki farklı biçimde tanımlanmıştır.

(2) ISO11592

Bu standartta düşük hızlı CAN veri yolunu tanımlamak amacıyla belirlenmiştir. Burada maksimum hız 125 Kbps olabilir.

2.6.2 ISO11898 ve ISO11592 Arasındaki Farklılıklar

Şekil 2.5'de ISO11898 ve ISO11592-2 de olması gereken CAN protokolleri gösterilmiştir. Fiziksel katmanda 3 alt katmandan oluşmaktadır. Bunlar PLS, PMA ve MDI alt katmanlarıdır. PMA ve MDI alt katmaları ISO'nun bu iki standartında farklılık göstermektedir. Bu farklılık Şekil 2.5' de gösterilmektedir.



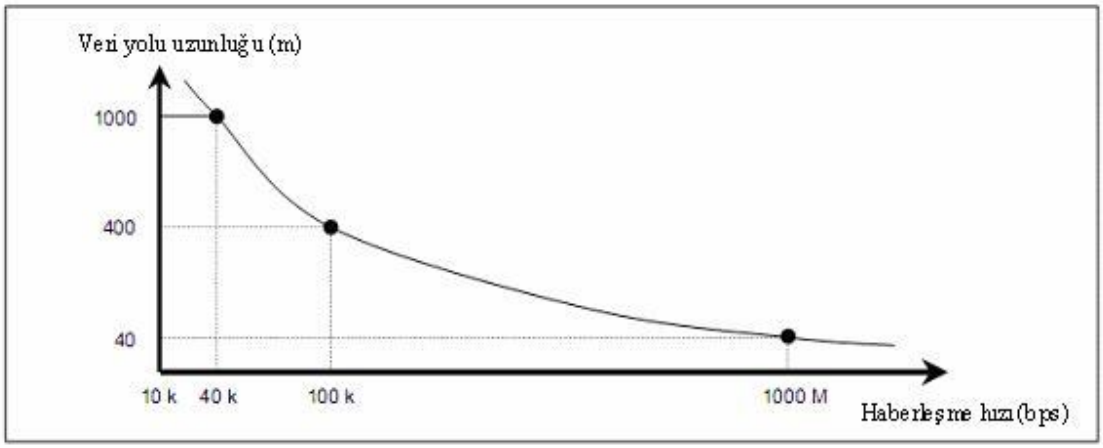
Şekil 2.5 ISO tarafından belirlenen ISO protokolü (OSI 2000)

Çizelge 2.1'de bu iki standart arasındaki fiziksel katmandaki farklılık daha iyi görülmektedir. Şekil 2.6'da ise veri yolu uzadıkça haberleşme hızının azaldığı

gösterilmektedir. Burada iki ünite arasındaki uzaklık göz önünde bulundurularak haberleşme hızını ayarlamak sistemi tasarlayanın sorumluluğundadır.

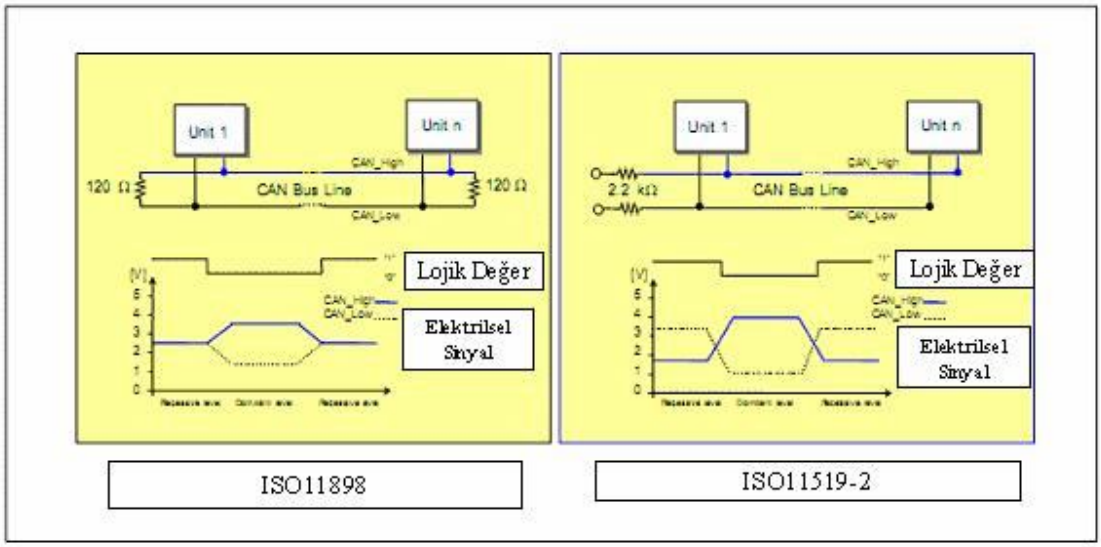
Çizelge 2.1 ISO 11898 ve ISO 11519-2 arasındaki temel farklılıklar

Fiziksel katman	**ISO 11898**						**ISO 11519-2**					
Haberleşme hızı	Maksimum 1 Mbps						Maksimum 125 kbps					
Maksimum hat uzunluğu	40 m/ 1 Mbps						1 km / 40 kbps					
Bağlanabilir düğüm sayısı	Maksimum 30						Maksimum 20					
Elektriksel	**Çekinik**			**Baskın**			**Çekinik**			**Baskın**		
Karakteristikler	Min.	Opt.	Max.	Min.	Opt.	Max.	Min.	Opt.	Max.	Min.	Opt.	Max.
CAN_H (V)	2.00	2.50	3.00	2.75	3.50	4.50	1.60	1.75	1.90	3.85	4.00	5.00
CAN_L (V)	2.00	2.50	3.00	0.50	1.50	2.25	3.10	3.25	3.40	0.00	1.00	1.15
Potansiyel Fark (V)	-0.50	0.00	0.05	1.50	2.00	3.00	-0.30	-1.50	-	0.30	3.00	-
	Kapalı veri yolu						Açık veri yolu					
	Empedans Z:120 Ohm						Empedans Z:120 Ohm					
	Sonlandırma Direnci 120 Ohm						Sonlandırma Direnci 2.2 Ohm					



Şekil 2.6 Haberleşme hızı ve veri yolu uzunluğu arasındaki ilişki

Sonlandırma dirençlerinde ve CAN_H ve CAN_L arasındaki potansiyel farklar arasında bu iki standartda farklılıklar mevcuttur. Şekil 2.7'de ise bu iki protokol için fiziksel bağlantıların nasıl olduğu gösterilmiştir. Çizelge 2.2'de ise bu iki protokolde kullanılabilecek elektriksel sinyal dönüştürücü entegrelerine örnekler verilmiştir.



Şekil 2.7 ISO11898 ve ISO11519-2 arasındaki fiziksel katmanın özellikleri

Çizelge 2.2 ISO11898 ve ISO11519- 2 uyumlu entegrelere örnek

	ISO 11898	ISO 11519-2
Sinyal Dönüştürücü Entegre	TJA1050T(Philips)	TJA1054T(Philips)
	TLE6250(İnfineon)	TLE6254(Infineon)
	CF150C(Bosch)	
	HA13721RPJE(Renesas)	

2.6.3 CAN ve Diğer Standartlar

CAN haberleşme protokolü ISO standartlarının yanında SAE (Society of Automotive Engineers) denilen endüstri organizasyonları tarafından da standartlaştırılmıştır. Bu standartlar yanında araştırma merkezleri ve hatta şirketlere özel standartlarda vardır. Çizelge 2.3'deki standartlarda endüstri göz önünde bulundurulmuştur. Çizelge 2.4'de ise bir otomobilde kullanılan CAN veri yollarının haberleşme hızlarına göre sınıflandırılması gösterilmektedir.

Çizelge 2.3 SAE tarafından belirlenen standartlar

Standart Adı	Haberleşme Hızı (bps)	Özellik	Uygulama Alanı
SAE J1939-11	250k	İki kablo üzerinden izalasyonlu	Kamyon,Otobüs
SAE J1939-12	250k	İki kablo üzerinden izalasyonlu 12V Güç Kaynağı	Ziraat Makineleri
SAE J2284	500k	iki kablo üzerinden izalasyonsuz	Otomobil
SAE J2411	33.3k, 83.3k	Tek kablo üzerinden	Otomobil
NMEA-2000	62.5k, 125k, 250k	İki kablo üzerinden izalasyonlu	Gemiler
DeviceNet	125k, 250k, 500k	İki kablo üzerinden izalasyonlu	Endüstriyel Makinelerde

Çizelge 2.4 Örnek bir otomobildeki CAN hızları

Sınıf	Haberleşme Hızı	Kullanım Amacı	Kullanılan Protokoller	
			CAN Protokol	Diğer Protokoller
Sınıf A	10 Kbps hızına kadar	Işıklandırma kontrol, kapı kontrol, kilit sistemi kontrol	Düşük hız	LIN protokol
Sınıf B	10-125 Kbps	Sürüş bilgileri, hata denetimleri, klima kontrol		J1850, VAN
Sınıf C	125Kbps ile 1 Mbps arası	Motor Kontrol, Fren Kontrol vb.	Yüksek hız	
Sınıf D	5 Mbps ve üzeri	Multimedya araçları		FlexRay, IEEE 1394, MOST

2.7 CAN Protokol

Bu bölümde CAN haberleşme protokolünde kullanılan çerçeve tipleri detaylı bir şekilde açıklanacak, her bir çerçeve tipinin hangi amaçlarla kullanıldığı hakkında bilgi verilecektir.

2.7.1 Çerçeve Tipleri

Düğümler arasında haberleşme aşağıdaki 5 tip çerçeve kullanılarak yapılır.

- Veri Çerçevesi

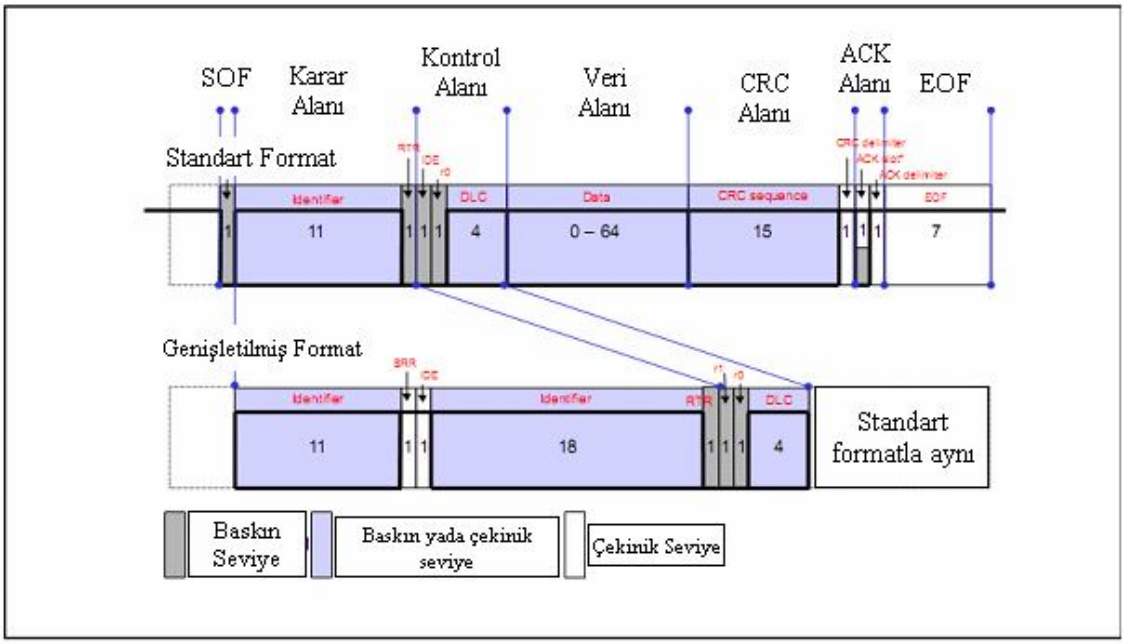
- Uzak Çerçevesi
- Hata Çerçevesi
- Aşırı Yük Çerçevesi
- Ayırıcı Boşluk

Buradaki veri ve uzak çerçevesi kullanıcı tarafından ayarlanmak zorundadır. Fakat diğer çerçeveler CAN veri yolunun donanımsal yapısı içinde ayarlanmaktadırlar.

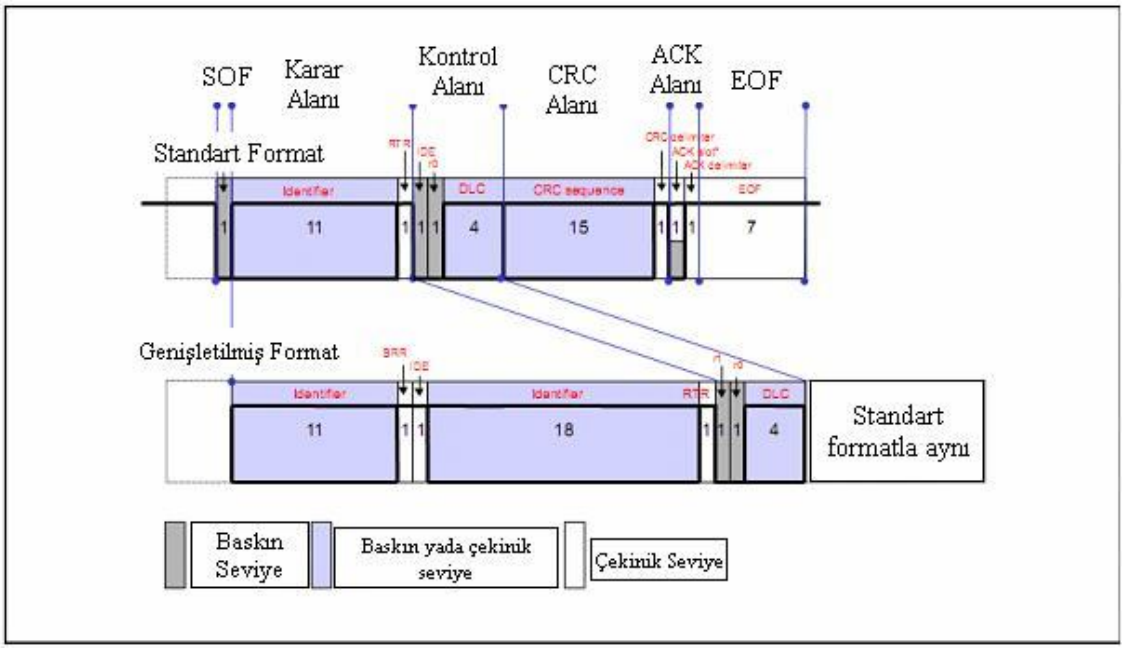
Veri ve uzak çerçeveleri iki çerçeve formatında olabilirler. Bunlar standart ve genişletilmiş formatta olabilirler. Standart formatta tanımlayıcı (ID) 11 bittir. Genişletilmiş formatta ise 29 bit olarak belirlenmiştir. Her bir çerçevenin rolü Çizelge 2.5'de belirtilmiştir. Şekil 2.8 -2.12'de her bir çerçevenin yapısı gösterilmiştir.

Çizelge 2.5 Çerçeve tipleri ve her bir çerçevenin görevleri

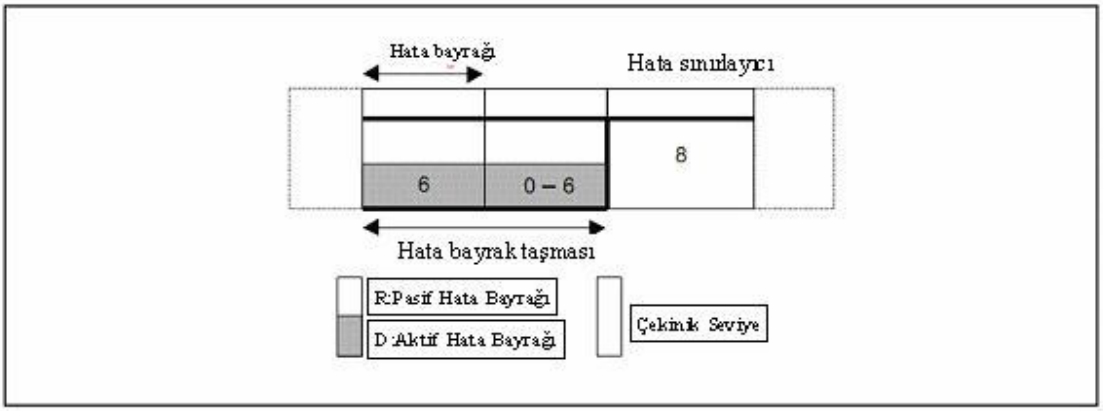
Çerçeve	Çerçevelerin Görevleri	Kullanıcı Ayarları
Veri Çerçevesi	Bu çerçeve transmit ünite tarafından kullanılır. Bu çerçeve ile alıcı ünitelere gerekli veriler gönderilir.	Yapılmalı
Uzak Çerçeve	Bu çerçeve uzaktaki bir üniteden veri isteğinde bulunmak için kullanılır. Uzaktaki ünite aynı mesaj ID' si ile cevap gönderir.	Yapılmalı
Hata Çerçevesi	Bir ünite hata algıladığı zaman diğer üniteleri de haberdar etmek için bu çerçeveyi kullanır.	Gerek Yok
Aşırı Yük Çerçevesi	Bu çerçeve alıcı ünite tarafından üretilir ve alıcı ünite veriyi almaya hazır olmadığını belirtir.	Gerek Yok
Ayırıcı Boşluk	Veri yada uzak çerçeveyi kendisinden önce gelen çerçeveyi ayırt etmek için kullanılır.	Gerek Yok



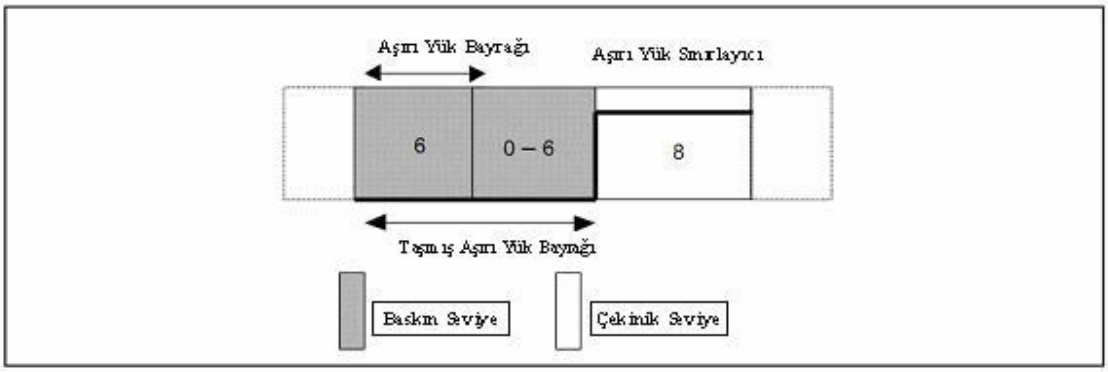
Şekil 2.8 Veri çerçevesi yapısı



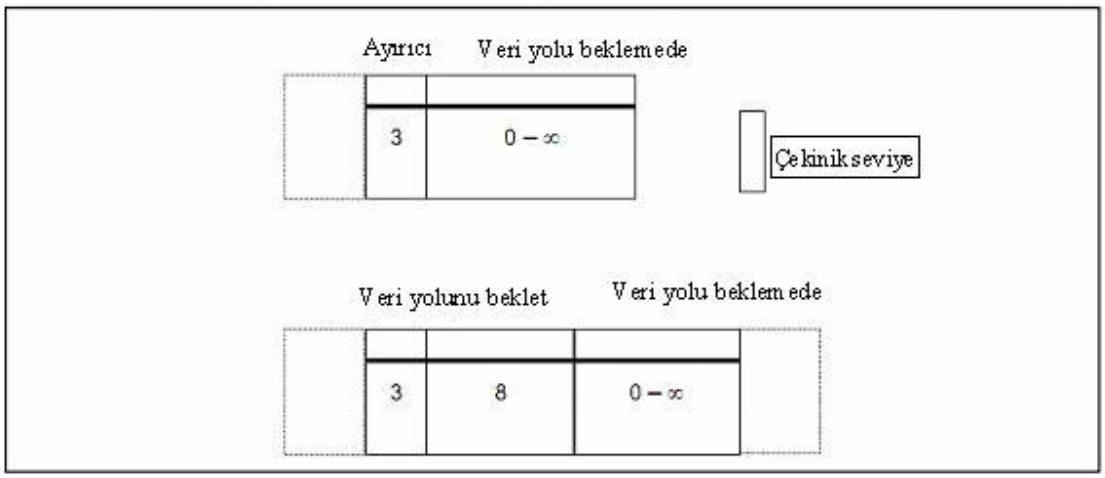
Şekil 2.9 Uzak çerçeve yapısı



Şekil 2.10 Hata çerçeve yapısı



Şekil 2.11 Aşırı yük çerçeve yapısı



Şekil 2.12 Ayırıcı boşluk çerçeve yapısı

2.7.2 Veri Çerçevesi

Veri çerçevesi bir mesajı transfer düğümünden alıcı üniteye göndermek için kullanılan çerçevedir. Bu çerçeve CAN veri yolu ağı için en önemli çerçevedir. Data çerçevesi yedi adet alandan oluşur. Şekil 2.13 veri çerçevesinin yapısını göstermektedir.

(1) Çerçeve başlangıç biti (SOF)

Bu alan veri çerçevesinin başladığını işaret eder.

(2) Öncelik karar alanı(Arbitration Field)

Bu alan çerçevenin öncelik alanını belirlemek amacıyla kullanılır.

(3) Kontrol alanı (Control Field)

Bu alan kaç byte verinin gönderileceğini belirtir. Bu alanda rezerve edilmiş bitlerde mevcuttur.

(4) Veri alanı

Bu alanda gönderilecek veriler tutulur. Bu alanda maksimum 8 byte veri gönderilebilir.

(5) CRC alanı

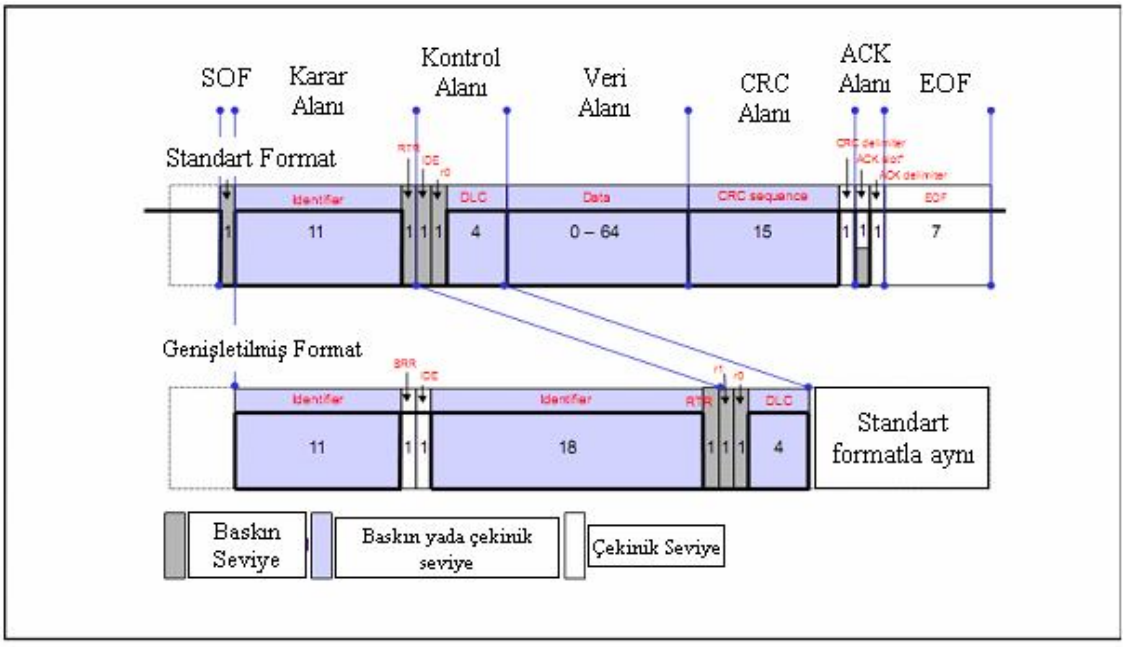
Bu alan veri transferi esnasında çerçevede meydana gelen hataları tespit etmek için kullanılır.

(6) ACK alanı (Acknowledge Field)

Bu alan alıcı tarafından verinin sağlıklı bir şekilde alındığını belirtmek için kullanılır.

(7) Çerçeve sonu alanı (EOF)

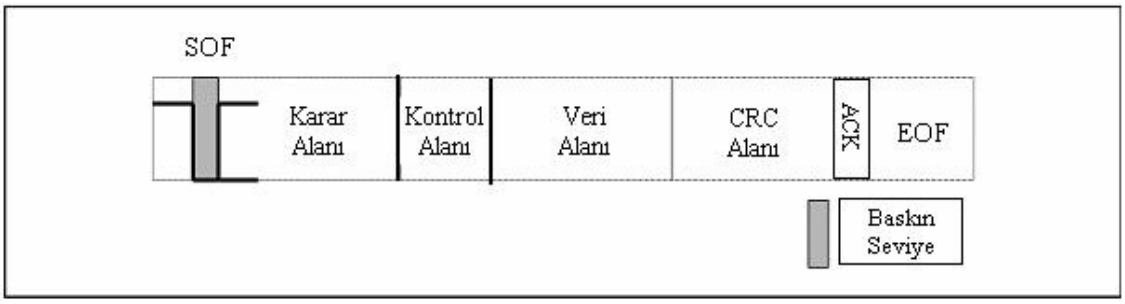
Bu alan veri çerçevesinin sonu olduğunu belirtmek için kullanılır.



Şekil 2.13 Veri çerçevesi ve alanları

(1) Çerçeve başlangıç biti (SOF)

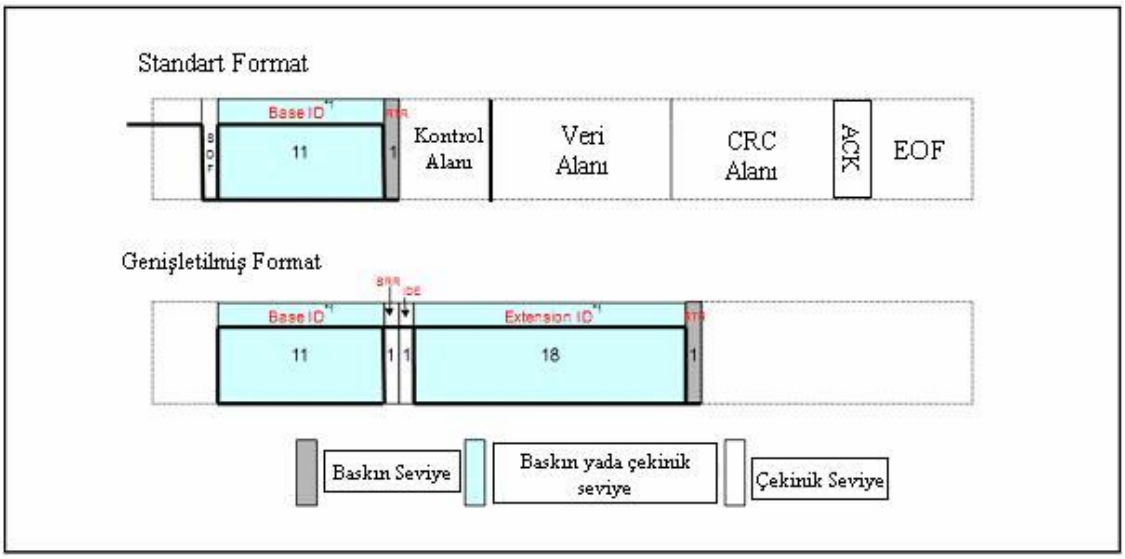
Bu alan veri çerçevesinin başladığını işaret eder. Bu bit her zaman baskın (0) bittir. Bu alan genişletilmiş ve standart formatlar için ortaktır.



Şekil 2.14 Çerçeve başlangıç biti gösterilişi

(2) Öncelik karar alanı (Arbitration Field)

Bu alan verinin önceliğini belirtir. Bu alan standart ve genişletilmiş formatta farklıdır.

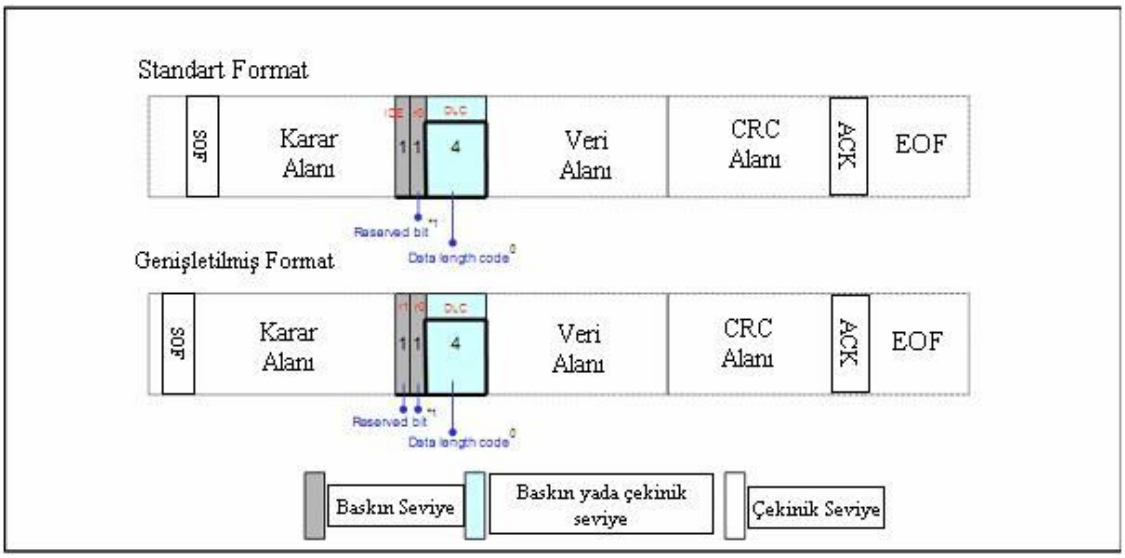


Şekil 2.15 Öncelik karar alanı gösterilişi

Standart formatta 11 bit mevcuttur. Burada bitler MSB(Most Significant Bit)'den başlayarak ardışık olarak transfer edilirler. Burada en yüksek yedi bit hepsi çekinik(1) olamaz. Bu standartlar tarafından yasaklanmıştır. 2048 -16=2032 (1111111xxxx gibi bir transfer yasaklanmıştır) farklı tanımlayıcı (ID) mesaj gönderilebilir. Genişletilmiş format ise 11 temel bit ve 18 genişletilmiş bitten oluşur. Burada da yine en yüksek yedi bit çekinik (1) olamaz bu durum yasaklanmıştır. $2032*2^{18}$ farklı mesaj tanımlanabilir. Hiç bir durumda farklı üniteler aynı anda aynı mesaj tanımlayıcısı (ID) gönderemezler. Bu kısım oldukça önemlidir.

(3) Kontrol alanı (Control Field)

Bu alandaki altı bit kaç byte'ın transfer edileceğini bildirir. Bu alanın yapısı standart ve genişletilmiş formatlarda farklıdır.



Şekil 2.16 Kontrol alanın gösterilişi

Rezerve edilmiş bitler r0 ve r1'dır. Bunlar gönderilirken her ikisi de baskın (0) olma zorunluluğu vardır. Ancak alıcı tarafta bunların çekinik ya da baskın olması çok önemli değildir.

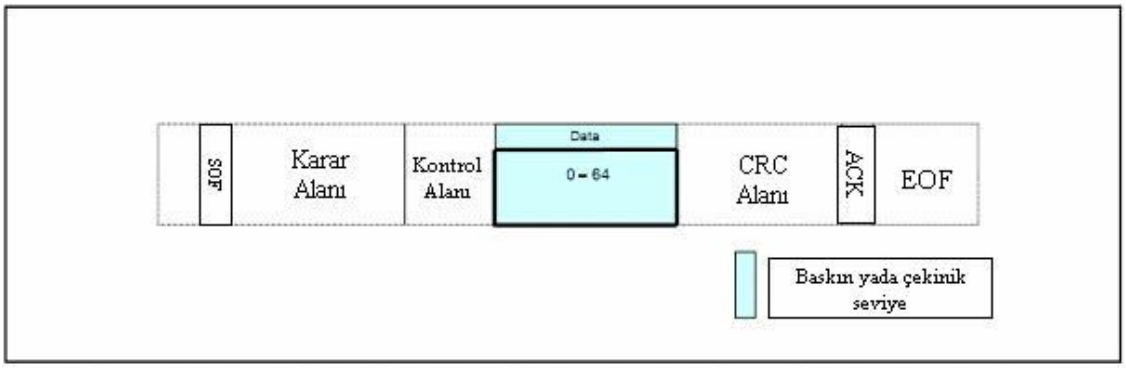
Buradaki ikinci kısım veri uzunluk kodu (DLC) olarak isimlendirilir ve dört bitten oluşur ve gönderilecek verinin uzunluğu bu bitlere bakılarak karar verilir. Karar verme için gerekli bit tablosu aşağıda Çizelge 2.6'de verilmiştir.

Çizelge 2.6 Veri uzunluk kodu (DLC) ve byte sayısı

Veri Byte Sayısı	Veri uzunluk kodu			
	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D yada R	D yada R	D yada R

(4) Veri alanı

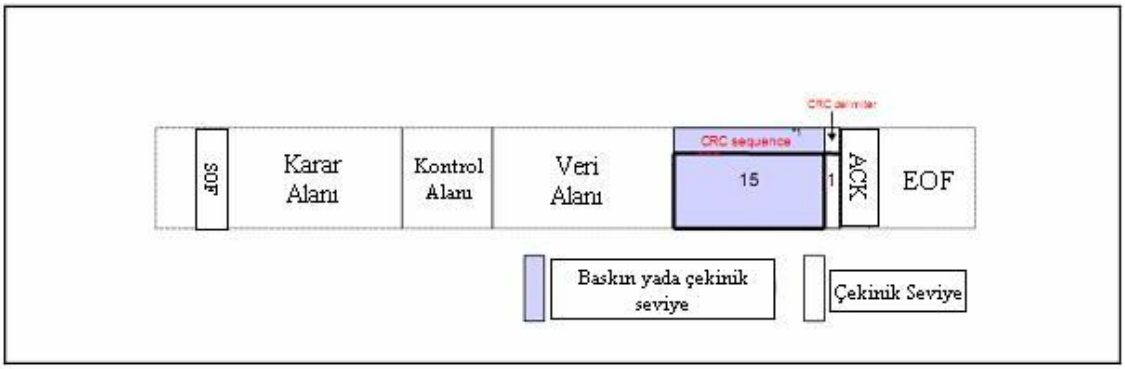
Bu alanda gönderilecek verinin içeriği mevcuttur. Sıfırdan sekiz byte'a kadar gönderilecek verinin sayısı kontrol alanında belirlenir. Data transferi tanımlayıcı (ID) alanında olduğu gibi yüksek bitten düşük bite doğrudur.



Şekil 2.17 Veri alanı gösterilişi

(5) CRC alanı

Bu alan çerçeveyi transfer hatalarına karşı denetlemek amacıyla kullanılırv e 16 bitten oluşur. 15 bit CRC alanıdır ve son bit ise ayırma biti olarak adlandırılır.



Şekil 2.18 CRC alanı

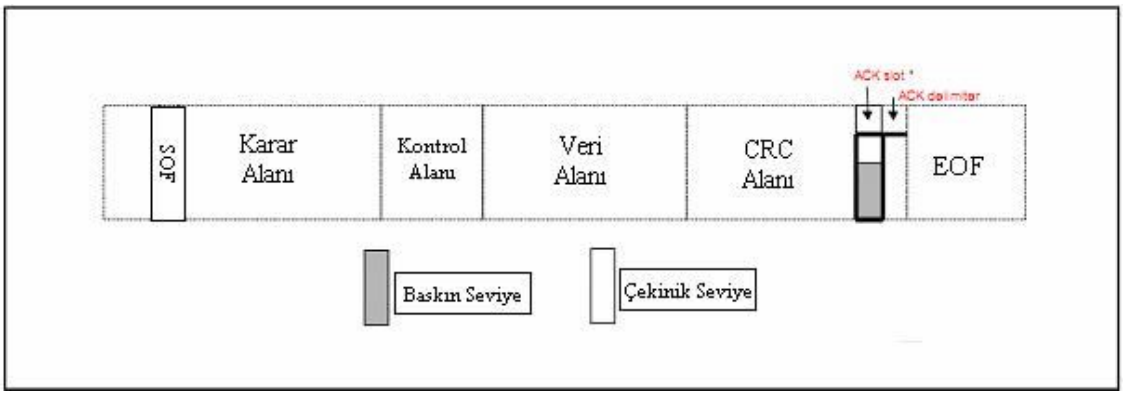
CRC bilgisi hesaplarken bir poligon fonksiyonu olan $P(x)$ kullanılır. $P(x)$ fonksiyonu aşağıda tanımlanmıştır.

$$P(x)= x^{15}+ x^{14}+ x^{10}+ x^{8}+ x^{7}+ x^{4}+ x^{3}+1$$

Veri gönderilirken ve alınırken bu CRC değeri karşılaştırılır eğer değer aynı değilse bir hata mesajı üretilir.

(6) ACK alanı (Acknowledge Field)

Bu alan gönderilen verinin normal bir şekilde gönderildiğini belirtmek amacıyla oluşturulmuştur. Bu alan iki bitten oluşur. Birincisi ack biti, diğeri ise ayırıcı sınırlayıcı bittir.



Şekil 2.19 ACK alanı

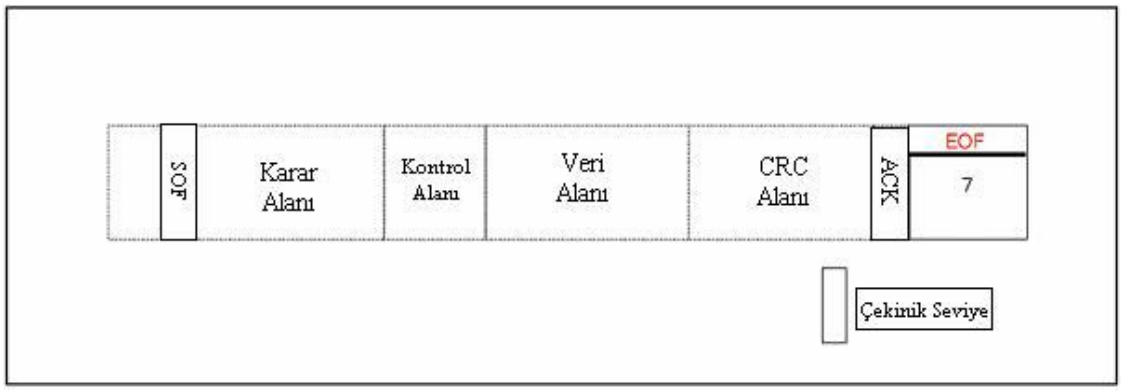
1) Eğer bu alan gönderici tarafından gönderiliyorsa her iki bitte çekinik (1) konumundadır.

2) Eğer alıcı ünite mesajı normal bir şekilde almış ise buradaki ack bitini baskın bit (0) olarak gönderir. Sınırlayıcı bit ise çekinik olarak kalır. ACK bitinin baskın olduğunu gören gönderici mesajın doğru bir şekilde iletildiğini anlar.

Devre dışı kalmış üniteler ve uyku durumunda olan üniteler hariç, diğer alıcı üniteler mesajı doğru aldıktan sonra ACK gönderebilirler. Eğer gönderici üniteden hariç başka bir ünite yoksa ACK mesajı geri dönmeye gerek yoktur. Ama gönderici üniteden başka ünitelerde measj varsa haberleşmesinin sağlanması için ACK göndermek mecburidir. Kısaca veri yolu üzerinde ikiden fazla ünite varsa bunlar normal bir şekilde mesajı aldıklarında gönderen üniteye ACK gönderirler.

(7) Çerçeve sonu alanı (EOF)

Bu alan çerçevenin sonu olduğunu bildirir. Bu alan 7 adet çekinik (0) bitten oluşur.



Şekil 2.20 Çerçeve sonu alanı gösterilişi

2.7.3 Uzak Çerçevesi

Bu çerçeve ile diğer ünitelerden mesaj gönderme isteğinde bulunulur. Bu çerçeve altı adet alandan oluşur. Veri alanı dışında uzak çerçevesi de aynı alanları içerir.

(1) Çerçeve başlangıç biti (SOF)

Bu alan veri çerçevesinin başladığını işaret eder.

(2) Öncelik karar alanı (Arbitration Field)

Bu alan çerçevenin öncelik alanını belirlemek amacıyla kullanılır.

(3) Kontrol alanı (Control Field)

Bu alan kaç byte verinin gönderileceğini belirtir.Bu alanda rezerve edilmiş bitlerde mevcuttur.

(4) CRC alanı

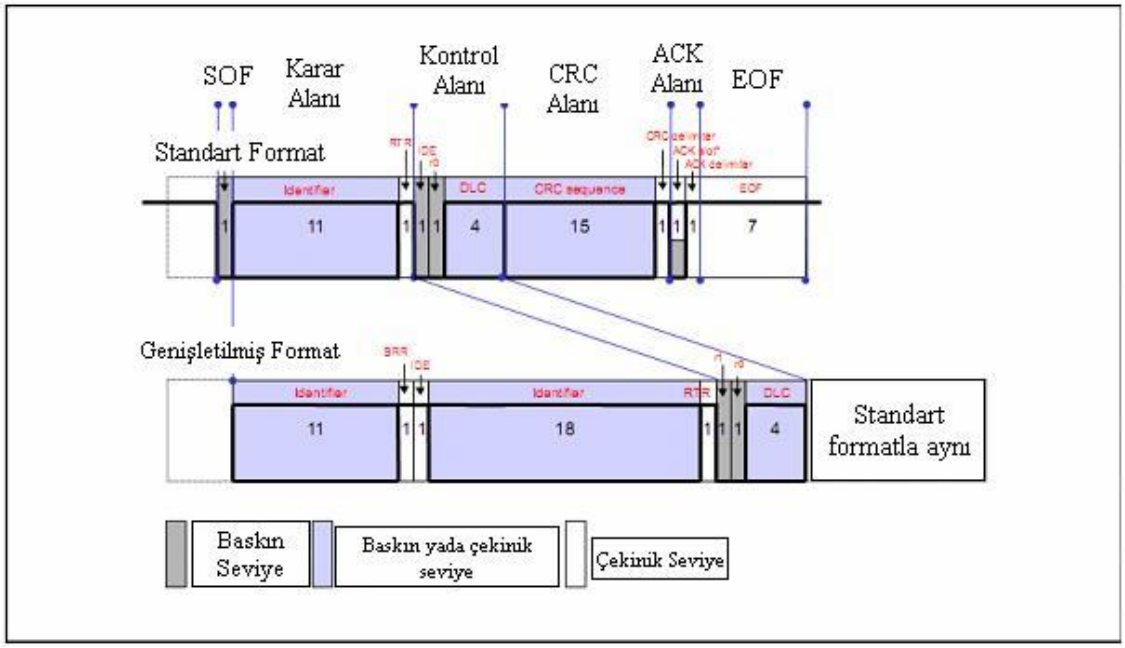
Bu alan data transferi esnasında çerçevede meydana gelen hataları tespit etmek için kullanılır.

(5) ACK alanı (Acknowledge Field)

Bu alan alıcı tarafından verinin sağlıklı bir şekilde alındığını belirtmek için kullanılır.

(6) Çerçeve sonu alanı (EOF)

Bu alan veri çerçevesinin sonu olduğunu belirtmek için kullanılır.



Şekil 2.21 Uzak çerçeve yapısı

Veri Çerçevesi ve Uzak Çerçevesi

- İki çerçeve arasındaki farklar
    - Uzak çerçevesi data alanı içermez ve öncelik karar alanındaki RTR biti çekinik (1)'tir.
    - Veri çerçevesinde eğer veri alanı yoksa bu RTR biti ile ayırt edilebilir.
- Uzak çerçevesindeki kontrol alanında belirtilen veri adedi (byte olarak) veriyi gönderecek olan ünitenin kaç byte'lık veri göndermesi gerektiğini belirtir.
- Aklımıza şöyle bir soru gelebilir. Veri içermeyen veri çerçevesi ne işe yarar?
    - Bu veri çerçevesi belki de üniteler arasında haberleşmenin olup olmadığını periyodik olarak kontrol etmek amacıyla kullanılabilir. Ya da sadece öncelik alanını veri yolu üzerinde gerekli ünitelere ulaştırmak için kullanılabilir.

2.7.4 Hata Çerçevesi

Bu çerçeve veri transferi esnasında hata meydana gelmesi halinde oluşur. Hata çerçevesi hata bayrağı ve hata ayıracından oluşur. Hata çerçeveleri CAN donanımı tarafından oluşturulurlar. Şekil 2.22'de hata çerçevesi gösterilmiştir.

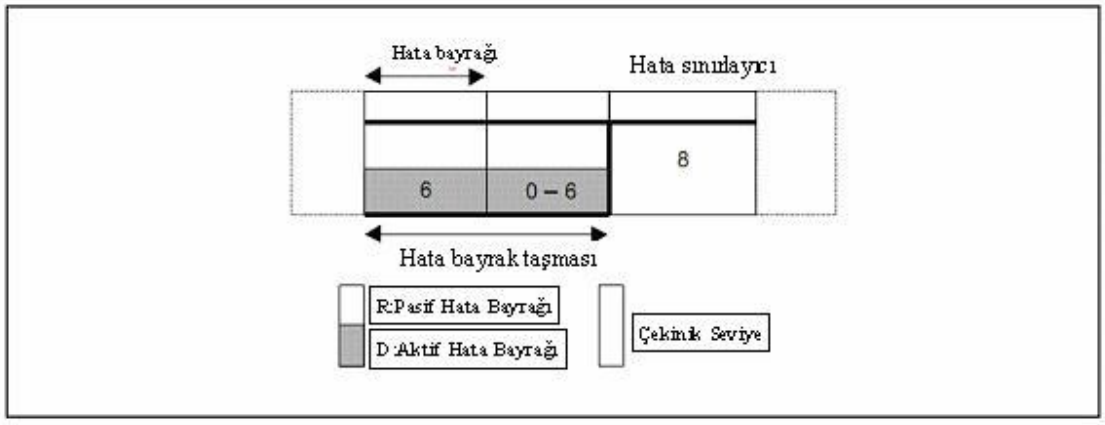
(1) Hata Bayrağı

Burada iki tip hata bayrağı olabilir: aktif hata bayrağı, pasif hata bayrağı.

- Aktif hata bayrağı: 6 adet baskın (0) bit
- Pasif hata bayrağı: 6 adet çekinik (1) bit

(2) Hata ayıracı

8 adet çekinik (1) bitten oluşur.



Şekil 2.22 Hata çerçevesi gösterilişi

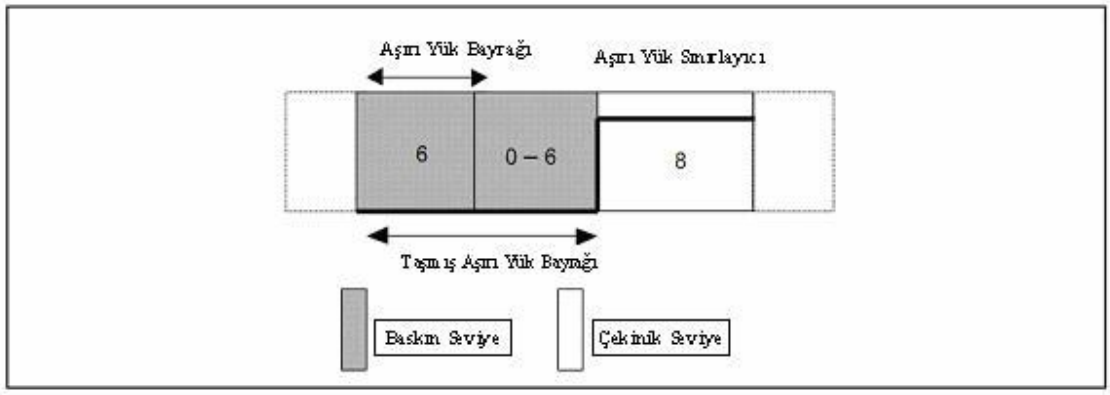
Veri yoluna bağlı ünitelerden her biri zamana bağlı olarak bir hata belirleyebilir. Toplamda 12 bit olmak üzere hata bayrağı 12 bite kadar genişleyebilir. Bir ünite pasif hata durumunda bir hata belirlemiş ise pasif hata oluşur. Bir ünite aktif hata durumunda bir hata belirlemiş ise aktif hata oluşur. Bu konu CAN protokol hataları bölümünde ayrıntılı bir şekilde incelenecektir.

2.7.5 Aşırı Yük Çerçevesi

Bu çerçeve alıcı ünitesinin veriyi almak için henüz hazır olmadığını belirtmek için oluşturulur. Hata çerçevesinde olduğu gibi aşırı yük bayrağı ve aşırı yük ayıracından oluşur. Şekil 2.23'de aşırı yük çerçevesinin yapısı görülmektedir.

(1) Aşırı yük bayrağı

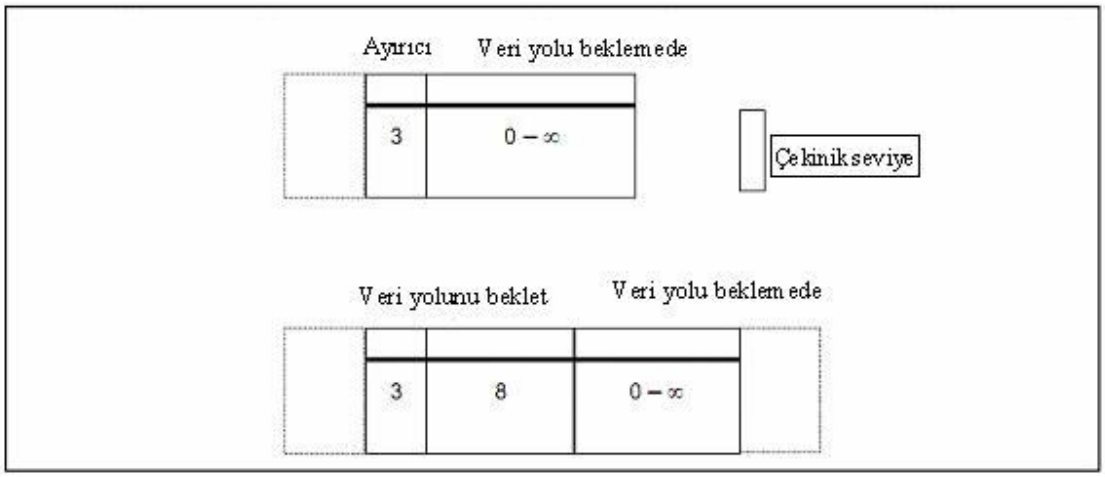
6 adet baskın (0) bitten oluşur. Bu bayrak hata çerçevesinin aktif hata durumuna benzer şekilde oluşturulmuştur. Benzer biçimde aşırı yük ayıracı da 8 çekinik bitten oluşur. Bu yapıda hata çerçevesine benzer.



Şekil 2.23 Aşırı yük çerçevesi gösterilişi

2.7.6 Ayırıcı Boşluk

Bu çerçeve veri ve uzak çerçeveleri birbirinden ayırmak amacıyla kullanılır. Veri ve uzak çerçeveleri, kendilerinden önce hangi çerçeve gelirse gelsin ayırıcı boşluk ile ayrılır. Buna karşılık hata ya da aşırı yük çerçeveleri ayırıcı boşluk kullanmazlar. Şekil 2.24'de ayırıcı boşluk yapısı gösterilmiştir.



Şekil 2.24 Ayırıcı boşluk gösterilişi

(1) **Aralık verme (Intermission)**

Bu çerçeve içerisindeki alan 3 adet çekinik (1) bitten oluşur. Eğer bu alan içerisinde herhangi bir baskın (0) bit belirlenirse aşırı yük çerçevesi hemen veri yoluna gönderilmelidir. Fakat aralık verme bitlerinin üçüncüsü baskın (0) bit ise yeni bir çerçeve başlangıcı (SOF) algılanmış olur.

(2) **Veri yolu bekleme konumunda (Bus idle)**

Bu konumda tüm bitler çekinik (1) konumundadır ve bu bitler sonsuz uzunlukta olabilir. Bu da su anlama gelir; herhangi bir ünite veri yoluna herhangi bir çerçeve gönderebilir.

(3) **Transferi bir sureliğine dondur (Suspend transmission)**

8 çekinik (1) bitten oluşur. Gönderici ünite pasif hata durumunda ise veri mesajı gönderilmeden önce bu çerçeve veri yoluna gönderilir.

2.8 CAN Haberleşme Protokol Hataları

Ağ yapısına bağlı tüm düğümler hata belirleme özelliğine sahiptir (Hata Belirleme). Hatayı belirleyen herhangi bir düğüm bunu diğer düğümlere bildirebilir (Hata Bildirme). Eğer bir düğüm mesaj gönderirken bir hata belirlerse, mesaj transmisyonunu durdurmaya zorlayabilir ya da diğer düğümleri haberdar edebilir. Gönderdiği mesajı mesaj normal gönderilinceye kadar gönderebilir (Hata Düzeltme).

2.8.1 Hata Durumları

Bir ünite aşağıda belirtilen 3 durumda bulunabilir.

(1) Hata aktif durumu

Bu durumda veri yoluna bağlı olan ünite normal bir şekilde ağa bağlanabilir. Veri gönderip alabilir. Bu ünite hatasız bir şekilde çalışmaktadır. Eğer bu ünite bir hata tespit ederse, aktif hata bayraklı bir hata çerçevesi gönderir.

(2) Hata pasif durumu

Bu gruba dahil olan üniteler hata yapmaya eğilimli olarak adlandırılırlar. Bu üniteler veri yolu üzerinde diğer ünitelerle haberleşmeye devam ederler. Diğer üniteleri haberleşmelerini engellemek için hata mesajları konusunda uyaramazlar. Bunun yanında

bu hata durumunda olan bir ünite bir hata tespit etse bile diğer üniteler tarafından bu hata algılanmazsa veri yolunda hiç bir hata oluşmamış gibi algılanır. Veri yolu üzerinde bu ünite hata tespit ettiğinde bunu pasif hata bayrağı ile hata çerçevesiyle veri yoluna gönderir. Bu durumda bulunan ünite bir mesajı gönderdikten sonra hemen ardından mesaj gönderemez, mesaj göndermesi için hattın bekletme alanını içeren ayırma boşluğunu beklemek mecburiyetindedir.

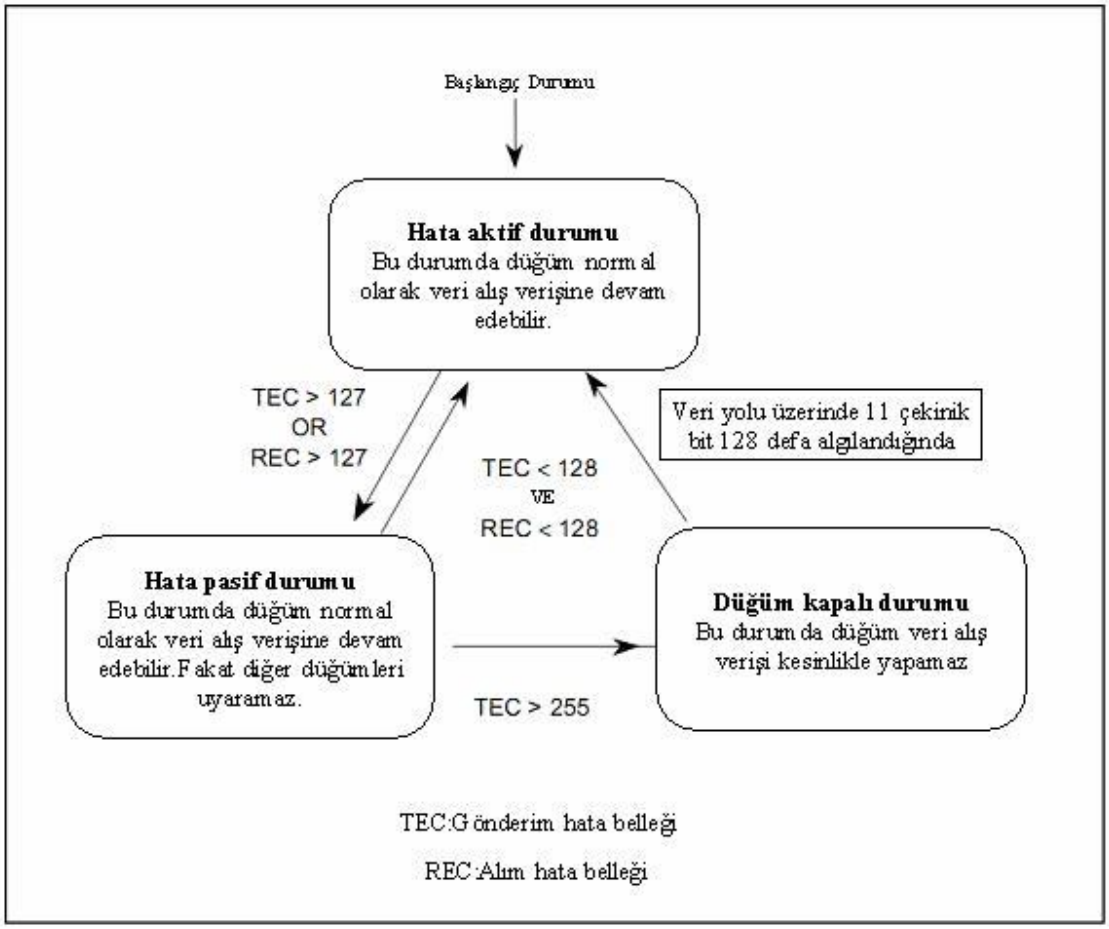
(3) Veri yolu kapalı durumu

Bu durumda olan bir ünite normal bir şekilde diğer ünitelerle haberleşemez. Çünkü ünitenin bütün mesajlaşma kısımları kapalı durumdadır.

Yukarıda bahsedilen bütün durumlar gönderici hata sayacı ve alıcı hata sayacı tarafından kontrol edilmektedir. İlgili durum kararı bu iki sayaca bakılarak karar verilir. Çizelge 2.7'de sayaçlar ve hata durumları arasındaki ilişki gösterilmiştir.

Çizelge 2.7 Hata durumları ve sayıcı değerleri (Renesas 2004)

Hata Tipi	Çıkış Zamanlayıcısı
Bit hatası	Bu tip hatalar hata tespit edildikten hemen sonra hata bayrağı biti aktif olur
ACK hatası	
Format hatası	
CRC hatası	ACK sınırlayıcısı geldikten sonra hemen set edilen hata bayrağıdır.



Şekil 2.25 Hata sayaçları ve hata durumları arasındaki ilişki (Renesas, 2004)

2.8.2 Hata Sayaçlarının Değerleri

Hata sayaçlarının artış değerleri daha önceden tanımlanmış kıstaslara göre belirlenir. Çizelge 2.8 hangi hata sayacı hangi koşullarda nasıl değiştiğini gösteriyor. Tek bir veri gönderme ya da alma durumunda birden fazla durum söz konusu olabilir. Buradaki hata sayaçlarının artma zamanı hata bayrağının ilk bitinin veri yoluna gönderildiği anda oluşur.

Çizelge 2.8 Hata sayacı değeri değişim durumları

	Transmit / Receive Ünite Hata Sayaçları Değişim Durumları	Transmit Hata Sayacı (TEC)	Receive Hata Sayacı (REC)
1	Alıcı ünite bir hata algıladığında.	-	+1
2	Alıcı ünite alınan bitlerin ilk bitinde bir baskın seviye algıladığında hemen hata bayrağı gönderir.	-	+8
3	Gönderen ünite bir hata bayrağı algıladığında	+8	-
4	Gönderici ünite bir bit hatası algıladığında	+8	-
5	Alıcı ünite bir bit hatası algıladığında	-	+8
6	Ağ üzerindeki herhangi bir ünite aktif hata ya da aşırı yük bayrağından sonra 14 ardışık baskın bit algılarsa bu olaydan sonra 8 ardışık bit algılarsa	+8	+8
7	Ağ üzerindeki herhangi bir ünite pasif hata bayrağından sonra 8 ardışık baskın bit algılarsa bu olaydan sonra 8 ardışık bit algılarsa	+8	+8
8	Gönderici ünite normal bir mesaj gönderirse	-1	-
9	Alıcı ünite normal şekilde bir mesaj alırsa.	-	-1 1<=REC<=127 0 REC=0 REC>127 ise REC=119 ile 127 arasında
10	Ağa yeni eklenen ünite 11 ardışık çekinik biti 128 defa algılarsa	TEC=0	REC=0

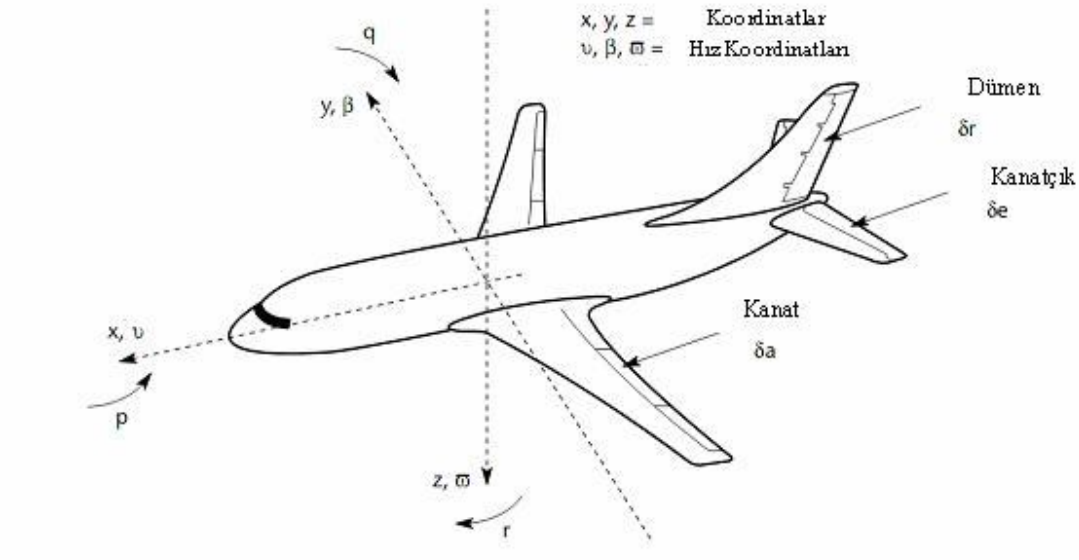
- Aşağıdaki durumlarda gönderici durum sayacı değişmez.
    - Pasif hata durumundaki bir ünite veri gönderdiği anda bir ACK hatası algılanırsa sayıcının içeriği değişmez. Bu hatanın sebebi, ACK biti görülemeyebilir ya da görülse bile baskın seviyeler algılanamazsa bir hata oluşur, fakat sayıcı arttırılmaz.
    - Baskın seviye gönderilmesine rağmen alıcı tarafta çekinik değer algılansa bile bu sayacı arttırmaz.

3. GÖMÜLÜ YAZILIM ve GERÇEK ZAMANLI İŞLETİM SİSTEMLERİ

Gömülü sistemler günümüzde hayatın her alanında geniş bir kullanım alanına sahiptir. Gömülü sistemlerin çok basit sistemlerden çok karmaşık sistemlere kadar birçok uygulama alanı vardır. Son zamanlarda teknolojideki gelişmeler ve mikroişlemcilerin yeteneklerinin de artmasıyla gerçek zamanlı işletim sistemleri gömülü sistemlerde de kullanılmaya başlandı. Gömülü sistemlerin birçoğu da kritik zamanlama beklentilerine sahiptir. Bu sistemlerin en önemli problemleri kritik zamanlı işlemlerin doğru bir şekilde yapılabilmesi ve kısa zamanda tasarlanıp sorunsuz çalışmasıdır. Gerçek zamanlı işletim sistemleri de kritik zamanlama ve hızlı tasarım problemini çözmek için kullanılır.

3.1 Gerçek Zamanlı İşletim Sistemleri

Birçok yazılım sistemi kullanıcısı uygulamalarına daha çabuk cevap verilmesini ister. Bu durum enformasyon sistemlerinde ve masaüstü bilgisayarlarda gerekli bir işlem gibi algılanır. Fakat çabuk cevap verme olayı gerçek zamanlı sistemlerde olmazsa olmaz koşuldur. Şekil 3.1'de verilen otomatik pilot sisteminin basitleştirilmiş hali gösterilmektedir. Bu şekil göz önüne alındığında diyelim ki pilot bir hava burgacına (türbülansına) girdiğinde uçağı düzgün bir rotada tutmak için otomatik pilot sistemi uçağın kanatlarını kuyruk kontrol kanatlarına belli açılar vermesi gerekir.



Şekil 3.1 Basit bir otomatik pilot sistemin görünümü

Buradaki sistemin en büyük karakteristiği sistemin giriş ve çıkışlarına hızlı cevap verebilmesidir. Bu süre yaklaşık olarak milisaniyeler mertebesindedir. Mesela burada ani değişim gerektiren bir durumda gecikme olursa uçak salınım girebilir ve sistem böyle devam ederse kaza oluşur. Bazı gerçek zamanlı sistem uygulamalarında zamandan çok doğru karar verme algoritmalarının istenen zaman dilimi içerisinde koşması gerekir. Yapılan çalışmalarda göstermektedir ki verilen görev zamanında bitirilmelidir ve bunu denemek için ikinci bir şans yoktur.

Birçok uygulamada belirli örnekleme aralıklarıyla veriler toplanır ve dijital değerlere çevrilir. Alınan bu değerler bir kontrol algoritması ile değerlendirilip çıkışa bir dijital analog çevirici (DAC) yardımıyla uygulanır. Tabi ki bu işlemin bir sonraki örnekleme zamanına kadar bitirilmiş olması gerekir.

3.2 Gömülü Sistemler

Gömülü sistemler donanım ve yazılımın sıkı bir birliktelik içinde çalıştığı ve başka çevre birimlerinin de olduğu belirli bir amaç için tasarlanmış sistemlerdir. Sistemin belirli bir amaç için tasarlanmış olması önemlidir, çünkü özel bir amaç için tasarlanmamış ise o sisteme gömülü sistem denemez. Evlerdeki kişisel bilgisayarlar bu durum için güzel bir örnektir. Bir kişisel bilgisayar da donanım, yazılım ve mekanik parçalara sahiptir (disk sürücüleri gibi). Ancak bir kişisel bilgisayar belirli bir amaç için tasarlanmamıştır. Üretici kişisel bilgisayarın ne amaçla kullanılacağını bilmez. Bir kullanıcı network server olarak kullanırken bir diğeri sadece oyun oynamak için kullanabilir.

Gömülü (embedded) sözcüğü, bu tip sistemlerin genelde daha büyük sistemlerin parçası olduğu için kullanılır. Büyük sistemin içine gömülmüş küçük bir sistem. Olabilecek durumların çoğunda gömülü sistemler tamamen gömülüdür, yani sistemlerin içindeki sistemlerdir. Çoğunlukla kendi başlarına çalışamazlar (Barr, 1999).

Günümüzde evlerde bulunan “digital set-top box” (DST) ‘lar örnek alınırsa, DST’nin bir iç parçası olan A/V decoder olarak adlandırılan sayısal ses/görüntü çözümleme sistemi bir gömülü sistemdir. A/V decoder tek bir multimedya veri dizisi alır, ses ve görüntü olarak çıkışına verir. Uydudan DST’nin aldığı sinyalin içinde birçok veri ve kanal mevcuttur. Bu yüzden A/V decoder transport veri dizisi decoder’i ile ilişkili çalışır ki bu

da bir embedded sistemdir. Transport veri dizisi decoder'i gelen multimedya veri dizisini kanallara ayırır ve seçilen kanalı A/V decoder'e verir. Birbiriyle etkileşimli çalışan bir gömülü sisteme örnekte otomobillerdeki elektronik kontrol mekanizmalarıdır. Bu gelişmiş araçlar birçok gömülü sisteme sahiptir. Bu sistemlerden biri frenlerin kilitlenmemesini kontrol ederken, diğeri hava yastığı ile ilgili kontrolü yapar, bir diğeri ise göstergede bilgiler gösterir (Qing ve Yao, 2003). Buradaki gömülü sistemler arasındaki haberleşme genellikle CAN haberleşme protokolü ile sağlanmaktadır.

Bazı durumlarda gömülü sistemler tek başlarına da çalışabilirler. Bir network router'i tek başına çalışan bir gömülü sistemdir. Özel bir işlemci, bellek, network portlarından ve gelen paketleri göndermek için yazılmış algoritmalardan oluşur. Diğer bir deyişle network router'i tek başına çalışan programlanmış bir algoritmaya dayalı olarak bir portundan gelen veriyi diğer portuna veren bir gömülü sistemdir.

Gömülü sistemler genel olarak üç ana kategoriye ayrılabilir. Bunlar olay tetiklemeli, zaman tetiklemeli ve hibrid gömülü sistemler olarak adlandırılabilirler.

3.2.1 Olay Tetiklemeli Sistemler

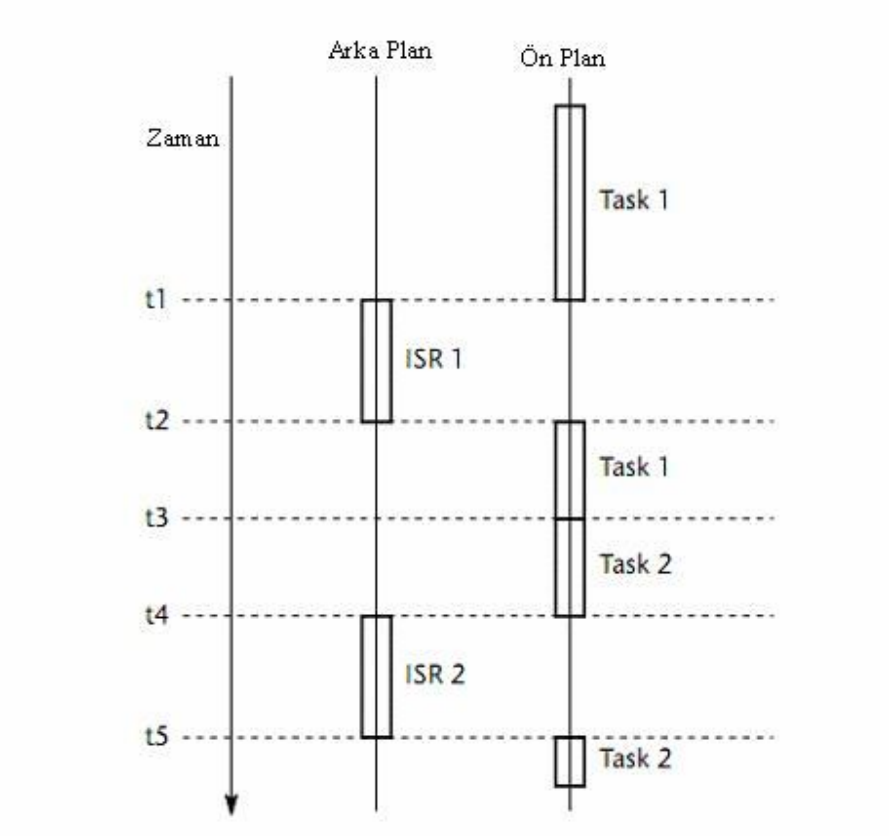
Bugün kullanılan birçok sistem olay tetiklemeli ya da olay sürmeli sistem olarak bilinmektedir. Mesela masaüstü bilgisayarımızda farenin tıklanması ya da klavyenin tuşuna basılması bu olaya bir örnektir.

Olay tetiklemeli sistemler genel olarak kesme yapısını kullanarak gerçeklenir. Burada sistem mimarı birden fazla kesmeyi kontrol etmek için alt programlar yazmak ve bunları yönetmek zorundadır.

3.2.1.1 Donanım Kesmeleri

Düşük seviyeden bakıldığında kesme bir donanımsal mekanizmadır. Bu mekanizma bazen dış olaylarla mesela fare tıklanması bazen iç olaylarla mesela timer taşması gibi olaylarla tetiklenir.

Yüksek seviyeden bakıldığında kesme aynı zaman diliminde birden fazla görevin aynı anda işletilmesi anlamına gelir. Bu Şekil 3.2'de gösterilmiştir.



Şekil 3.2 Bir gömülü sistemde kesmenin şematik olarak gösterimi

Yukarıdaki şekilde ön planda iki adet görev işletilmektedir. Bunlar Task-1 ve Task-2'dir. Task-1'in işletilmesi esnasında birinci kesme gelmiş ve program sayacı ilgili kesme adresinden kesme alt programını çalıştırmıştır. Task-2'nin işletilmesi sırasında da 2. kesme programı işletilmiştir.

Yazılımcının bakış açısından ise kesme özellikli bir donanımsal olay sonucunda mikro kontroller tarafından çağrılan bir alt programdır.

3.2.2 Zaman Tetiklemeli Sistemler

Olay tetiklemeli kontrol sistemlerinin ana alternatifi zaman tetiklemeli kontrol sistemleridir. Olay tetiklemeli ve zaman tetiklemeli sistemler hem masaüstü bilgisayarlarda hem de gömülü sistemlerde kullanılırlar. Burada zaman tetiklemeli ve olay tetiklemeli sistemler arasındaki farkı anlamak için bir hastanede çalışan bir doktoru

düşünebiliriz. Hastanede doktor gece boyunca hemşirelerle birlikte 10 adet hastaya bakmak zorunda olduğunu düşünelim.

- Eğer doktor herhangi bir hastada spesifik bir problem olursa kendisini uyandırması için bir hemşireyi görevlendirir. Bu olay tetiklemeli sisteme örnektir.
- İkinci örnekte ise doktor saatinin alarmını her saat başına kurar ve her saat uyandığında hemşireye hastaları dolaşmasını söyler eğer gerekli bir durum varsa hastaya müdahale eder. Bu da zaman tetiklemeli sisteme bir örnektir.

Tabii ki bizim birçok hastanemizde doktorlar uyumayı yeğledikleri için olay tetiklemeli sistemi tercih ederler.

Birçok gömülü sistemde uyumaya gerek duyulmadığı için zaman tetiklemeli işletim sistemi kullanılır. Zaman tetiklemeli sistemler birçok avantaja sahiptirler. Özellikle olay tetiklemeli işletim sistemlerine göre daha güvenlidirler. Zaman tetiklemeli sistemler önceden görülebilir sistemler olduğu için hata yapma riski daha azdır. Az önceki hastane örneğimizi düşünürsek eğer doktor derin bir uykuda kalıp hasta hastalandığında uyandırılması ya da mesela hemşire hastada çok büyük bir problem yok deyip atlaması bazı verilerin kaybolmasına neden olabilir. Diyelim ki hemşire ufak problemleri olan dört hasta tespit etti ve doktoru uyandırmaya karar verdi. Doktor kontrolleri sonucunda iki hastanın acil ameliyat olması gerektiği sonucuna vardı. Aynı anda iki ameliyat yapamayacağı için bu durumda hastalardan biri kaybedilir. Fakat zaman tetiklemeli sistem göz önüne alındığında doktor her saat başı hastalarını kontrol edecek ve durumu acil gördüğünü ameliyat edecek ve tüm hastalar hayatta kalabileceklerdir.

Bu olay gömülü sistemlerde de benzer biçimde olmaktadır. Örneğin klavyenin iki tuşuna aynı anda basıldığında birinin atlanması yüksek bir olasılıktır. Renesas mikro kontrolcüsü göz önünde bulundurulduğunda diğer mikro kontrolcüler gibi burada da kesmelerin yüksek ve düşük gibi iki öncelik seviyeleri vardır. Eğer iki kesmenin aynı anda oluştuğunu düşünürsek aşağıdaki durumlar oluşur. Burada kesmeler kesme-1 ve kesme-2 olarak isimlendirilmiştir.

- Eğer kesme-1 düşük öncelikli kesme-2 yüksek öncelikli ise:

Eğer düşük öncelikli kesme gelmiş ise ve bu alt program koşuyor ise yüksek öncelikli kesme tam bu esnada gelirse yüksek öncelikli kesme alt programı koşmaya başlar. Bu durumda düşük öncelikli alt programı durdurulur ve yüksek alt program işler. Yüksek öncelikli alt programın koşması tamamlandığında tekrar düşük öncelikli alt program çalışmaya başlar ve bitirilir. Buradaki çoğu durumda herhangi bir aksama olmadan sistem çalışır.

- Eğer kesme-1 ve kesme-2 düşük öncelikli ise:

  Düşük öncelikli bir kesme koşuyor ise bu kesmenin diğer bir düşük öncelikli kesme tarafından görevi durdurulamaz. Sonuç olarak ikinci kesme biraz gecikerek çalışacaktır ya da bazı durumlarda çalışamayacaktır.

- Eğer kesme-1 yüksek öncelikli kesme-2 düşük öncelikli ise:

  Yüksek öncelikli bir kesme alt programı çalışıyor iken düşük öncelikli bir alt kesme programı bunu durduramaz. Sonuç olarak ikinci kesme biraz gecikerek çalışacaktır ya da bazı durumlarda çalışamayacaktır.

- Eğer kesme-1 ve kesme-2 yüksek öncelikli ise:

  Yüksek öncelikli bir kesme alt programı koşarken başka bir yüksek öncelikli kesme alt programı bunun çalışmasını durduramaz. Sonuç olarak ikinci kesme biraz gecikerek çalışacaktır ya da bazı durumlarda çalışamayacaktır.

Yukarıdaki sonuçlar göz önüne alındığında olay tetiklemeli sistemlerde birden fazla olayın aynı anda meydana gelmesi durumunda sistem daha fazla karmaşık olacak ve sistem güvenilirliği azalacaktır. Zaman tetiklemeli kontrol sistemlerinde ise tasarımcı sistem tarafından bir zaman dilimi içerisinde hangi olayın işletileceğini bilir ve ona göre planlamasını yapar.

Daha öncede bahsedildiği gibi sistem güvenliğinin ön planda olduğu uygulamalarda zaman tetiklemeli sistemler daha güvenlidir. Bunun yanında zaman tetiklemeli sistemlerin kullanılması CPU üzerindeki yükü azaltır ve daha az hafiza kullanılmasına olanak sağlar. Bizde bu tezin yazılım kısmında mümkün olduğunca zaman tetiklemeli sisteme uygun bir tasarım kullanacağız.

4. ZAMAN TETİKLEMELİ SİSTEMLERİN GERÇEKLENMESİ

Zaman tetiklemeli işletim sistemi gerçeklenirken takvim planlayıcılarına (scheduler) ihtiyaç duyulmaktadır. Bu takvim zamanlayıcıları farklı şekillerde olabilir. Yazılımımızı gerçeklerken yardımcı (co-operative) takvim zamanlayıcılarını kullanacağız. İlerleyen bölümlerde bu zamanlayıcılar hakkında geniş bilgi verilecektir. Bu zamanlayıcıları kullanmanın avantajları, güvenilirliği sisteme adaptasyonu gibi konular ele alınacaktır.

4.1 Yardımcı Takvim Zamanlayıcıları

Yardımcı takvim zamanlayıcıları sadece tek bir görevin çalışmasını garanti eden bir mimaridir. Bu zamanlayıcının gerçeklenmesi oldukça basittir ve yüksek doğrulukla çalışır. Bu zamanlayıcının gerçeklenmesi bütünüyle C programlama dili kullanılarak gerçeklenmiştir. Böylelikle sistem daha okunabilir, geliştirilmesi ve bakımı daha kolay ve diğer uygulamalara kolaylıkla adapte olabilecek bir yapıdadır. Task başına oldukça küçük byte kullanımı ile CPU yükünü azaltır. Bu yapıda en önemli rolü C'de kullanılan fonksiyon işaretçileri alır.

4.2 Fonksiyon İşaretçileri (Function Pointers)

C programcılarının alışık olmadıkları bir alanda fonksiyon işaretçileridir. Bu işaretçiler masaüstü uygulamalarında pek sık kullanılmaz ancak yardımcı takvim zamanlayıcılarının temelini oluşturacağından bunlara değinme zorunluluğu vardır.

Buradaki anahtar nokta program belleğinde yer alan bir dizi kod parçasının yerinin karar verilmesidir. Bu kod parçası genellikle bir göreve özellikli bir fonksiyonun başlangıç adresini belirtir. Bu adres fonksiyonun işaretçisi olarak adlandırılır ve fonksiyonu çağırmak amacıyla kullanılır.

Dikkatli kullanıldığında tasarımı daha anlaşılır ve yapılması kolaydır. Böylelikle çok daha karmaşık dizaynlar basitleştirilerek gerçeklenmiş olur. Örneğin bir kritik görevi yerine getiren bir sistemde, acil bir durumla karşılaşıldığında sistem hemen kapatılmaz. Bunun yerine sistem durumuna uygun olan fonksiyonu çağırmak gerekir. Burada doğru olan gerekli iyileştirme fonksiyonlarının ve gerekli fonksiyon işaretçisini tanımlamaktır.

Sistem durumu değiştikçe ilgili fonksiyon işaretçisinin gösterdiği ilgili iyileştirme fonksiyonu çağrılarak sistem daha iyi kontrol edilebilir.

4.3 Yardımcı Takvim Zamanlayıcılarının Anahtar Yapıları

- Takvim zamanlayıcısı veri yapısı
- Bir başlangıç fonksiyonu
- Kesme alt programı: belirli zaman aralıklarında koşan update fonksiyonu
- Yeni tasklar eklemek için fonksiyon
- Zamanı geldiğinde görevleri çalıştırmak için fonksiyon
- Görevleri takvim zamanlayıcısından çıkarmak için gerekli delete fonksiyonu

Bu bölümde yukarıda adı geçen bütün fonksiyonlara açıklanacaktır.

4.3.1 Genel Bakış

Yukarıda bahsedilen bileşenleri açıklamadan önce yardımcı takvim zamanlayıcılarına bir örnek verelim. Bu örnekte en basit uygulamalardan bir LED'in bir saniye açık bir saniye kapalı olması ve bu olayı belli aralıklarla tekrarlaması açıklanacaktır. Şekil 4.1'de örnek programın C dilindeki yazılımı gösterilmiştir.

```
void main(void)
{
// takvim zamanlayıcısını başlat
SCH_Init ();
// 'Flash_LED' görevini hazırla
LED_Flash_Init();
// 'Flash LED' görevini ekle (~1000ms açık, ~1000ms kapalı)
// zamanlamalar darbe şeklindedir. (1 ms her bir darbe zamanı)
SCH_Add_Task(LED_Flash_Update, 0, 1000);
// takvim zamanlayıcıyı başlat
SCH_Start();
while(1)
{
SCH_Dispatch_Tasks();
}
}
#pragma INTERRUPT SCH_Update
void SCH_Update (void)
{
// görev listesini güncelle
...
}
```

Şekil 4.1 LED yakıp söndürme programı

Yukarıdaki kod parçası incelendiğinde aşağıdaki sonuçlar çıkarılabilir:

1) Burada LED'in belirli aralık açık ve belirli aralık kapalı tutmak için LED_Flash_Update fonksiyonu kullanılır. Bu görevi yerine getirmek için bu fonksiyonun periyodik olarak bir saniye aralıklarla çağrılması gerekir.
2) SCH_Init fonksiyonu kullanarak zamanlayıcı ayarlanır.
3) Zamanlayıcı fonksiyonu ayarlandıktan sonra SCH_Add_Task fonksiyonu ile yukarıdaki LED fonksiyonu görev listesine eklenir. Buradaki fonksiyonda aynı zamanda yineleme zamanlamalarda belirtilmiştir.

   SCH_Add_Task(LED_Flash_Update, 0, 1000);

4) LED_Flash_Update fonksiyonu zamanlayıcı kesmesi tarafından hangi zamanda çağrılacağı SCH_Update kesmesi tarafından kontrol edilir.

   #pragma INTERRUPT SCH_Update

```
void SCH_Update (void){
        // görev listesini güncelle
...}
```

5) Update fonksiyonu ilgili görevi çalıştırmaz. Bu fonksiyon içerisinde sadece runme bayrağı ayarlanır ve görev main loop içerisinde çalıştırılır. Bu bayrak arzu edilen değere ayarlandığın da görev SCH_Dispatch_Tasks sıraya konur ve koşturulur.

```
while(1)
{
SCH_Dispatch_Tasks();
}
```

Bileşenleri açıklamaya geçmeden önce elbette ki sadece bir LED'i yakıp söndürmek için böyle bir program yapısının oluşturulması oldukça maliyetli bir iştir. Fakat bu yapı birçok görevden oluşan karmaşık yapıların çalıştırılması için temel oluşturmaktadır.

Yukarıda söylenenlere ek olarak bu yardımcı takvim zamanlayıcısı oldukça düşük maliyetlidir. Bu yapı CPU kaynaklarının oldukça az miktarını harcar. Bunlara ek olarak her bir görev sadece yedi byte'lık bir hafiza alanı kullanır. Tipik uygulamalar ortalama olarak altı görev kullanacakları için buradaki hafiza tüketimi 40 byte'ı pek aşmaz. Bu değer 256 byte'lık bir hafizaya sahip 8 bitlik bir mikro denetleyici için oldukça iyi değerlerdir.

4.3.2 Takvim Zamanlayıcısı Veri Yapısı ve Görev Dizisi

Bu yapının merkezinde takvim zamanlayıcısı veri yapısı bulunmaktadır. Bu kullanıcı tanımlı görev tipi her bir göreve ait özellikleri taşır. Bu görev yapısına bakıldığında, her bir görevi atama yapmak için bir fonksiyon göstericisi, başlangıçtaki zaman gecikmesini ayarlamak için 'Delay' isimli bir değişken tanımlanmıştır. Yine her bir görevin yapısını belirlemek için Period değişkeni ve görevlerin zamanı geldiğinde çalıştırmak için 'RunMe' bayrağı tanımlanmıştır. Şekil 4.2'de zaman tetiklemeli yazılım için tasarlanan veri yapısı gösterilmektedir.

```
// toplam bellek kullanımı 7 bytes
typedef data struct
{
// görevler için fonksiyon gösterici
void (code * pTask)(void);
//başlangıç geciktiricisi
tWord Delay;
//görevlerin hangi aralıklarla koşacağı belirlenir
tWord Period;
//görevin çalıştırma zamanı geldiğinde bayrak ayarlanır
tByte RunMe;
} sTask;
```

Şekil 4.2 Zaman tetiklemeli işletim sistemi veri yapısı

4.3.3 Başlangıç Fonksiyonu

Birçok görev fonksiyonunda olduğu gibi takvim zamanlayıcı fonksiyonunda da bir başlangıç (initialization) fonksiyonuna sahiptir. Bu yapıda tanımlanan görev dizilerinin başlangıç değerleri atanır. Kullanılacak timer kesmesinin başlangıç değerleri ve ilgili belleklere değerler atanır. Burada ayrıca her bir darbe zamanı da belirlenmiş olur. Bizim tasarladığımız fonksiyonda zamanlayıcı A kullanılmıştır. Ek 6 kısmında programın akış şeması verilmiştir. Bu akış şeması incelendiğinde ilk olarak her bir görevin varsa başlangıç atamaları yapılır. Daha sonra zamanlayıcı A'yı çalıştırmak ve darbe zamanının ayarlamak için gerekli bellek gözlerine Renesas tarafından belirlenen matematiksel hesaplamalarda yapıldıktan sonra gerekli atamalar yapılır. Zamanlayıcı A'nın kesme öncelik sırası bir olarak ayarlanır.

Buradaki başlangıç değerleri yüklenirken kullanılan osilatör frekansının 20 MHz olduğu ve darbe zamanlarının ise 5 ms olarak ayarlandığı görülür.

Geçerli olan kural ise her bir mikro denetleyici için sadece bir kesme alt programı olması ön koşuldur. Birden fazla kesme olan sistemlerde takvim zamanlayıcısı çalışması garanti edilemez ve sistem güvenirliliği azalır.

4.3.4 Güncelleme Fonksiyonu (Update)

Güncelleme fonksiyonu bir kesme alt programıdır. Zamanlayıcı taşması olduğunda kesme alt programı çağrılır ve her bir göreve ait periyot bir darbe zamanı arttırılır. Güncelleme fonksiyonu kompleks bir yapıda değildir. Güncelleme fonksiyonunun ilgili göreve ait çalıştırma bayrağını ayarlar. Böylece görev "SCH_Dispatch_Tasks" fonksiyonu tarafından koşulur. Ek 7'de güncelleme fonksiyonu alt programının akış diyagramı gösterilmiştir.

4.3.5 Yeni Tasklar Eklemek İçin Fonksiyon (Add Task)

Adından da anlaşılacağı üzere bu fonksiyon yapıya yeni görev fonksiyonları eklemek ve onları belirli zaman aralıklarında koşturmak için oluşturulmuştur. Bu fonksiyona örnek verdiğimizde SCH_Add_Task (Do_X,1000,0) olarak verilebilir. Buradaki Do_X görevi 1000 zaman darbesi gecikmesi ile bir kere çalıştırılır.

Task_ID = SCH_Add_Task (Do_X,1000,0) bu fonksiyon ise task id'si tutuluyor ki daha sonra bu task delete fonksiyonu ile silinebilsin. SCH_Add_Task (Do_X,0,1000) bu örnekte ise task hemen başlatılıyor ve her 1000 darbe zamanında bir periyodik olarak çalıştırılıyor. Sch_Add_Task (Task_Name, Initial_Delay, Period) buradaki fonksiyon parametrelerini lojik anlamları gösterilmektedir. Ek 8'de görev ekleme fonksiyonu akış diyagramı gösterilmiştir.

4.3.6 Zamanı Geldiğinde Görevleri Çalıştırmak İçin Fonksiyon (Dispatcher)

Daha önceki bölümlerde de bahsedildiği gibi update fonksiyonun görevleri koşturma yeteneği yoktur. Görevler "dispatcher fonksiyonu" sayesinde çalışır.Bu fonksiyon sonsuz bir "while" döngüsü içerisinde devamlı olarak çağrılır. Görev dizisi içerisindeki görevler zamanı geldiğinde zamanlayıcı A kesmesi sayesinde görev çalıştırma bayrağı bir yapılarak görevin çalıştırılması sağlanır. Ek 9'da programın akış diyagramı gösterilmiştir.

4.3.7 Görevleri Takvim Zamanlayıcısından Çıkarmak İçin Gerekli Fonksiyon

SCH_Add_Task fonksiyonu ile bir görev eklendiğinde bu fonksiyon bir task ID döndürür. Bu ID görevi silmek içinde kullanılır. Çalışma zamanında koşan görevleri

silmek için birde görev silme fonksiyonuna ihtiyaç duyulmaktadır. Bu fonksiyon kendisine gönderilen ilgili görevi görev dizisinden çıkarır. Ek 10'da görev silme alt programının akış diyagramı verilmiştir.

4.4 Darbe Zamanının Belirlenmesi

Hazırlanan tezdeki geliştirilen uygulamada kullanılan görevler genellikle milisaniyeler mertebesindedir. Takvim zamanlayıcısına eklenen görevler mesela 5 ms, 10 ms veya 1000 ms olabilir. Genellikle yapılan uygulamalarda kesme zamanını 1 ms olarak ayarlamak uygun gibi görülebilir.

Burada seçilen kesme zamanı oldukça önemlidir. Bu süre ne kadar kısa tutulursa CPU üzerindeki yük o derece daha fazla olacaktır. Tasarlanan sistemlerde bu süre mümkün olduğunca fazla tutulmalıdır. Bu sürenin uzak tutulması demek daha az CPU yükü ve daha güç tüketimi sağlayacaktır.

Daha az yük tüketimi ve daha efektif bir sistem tasarımı yapılacaksa eğer burada görevlendirilen taskların ortak bölenlerinin en büyüğü alınarak kesme zamanı tespit edilirse sistem optimum şekilde tasarlanmış olur.

Örnek olarak 3 adet görevimiz olduğunu varsayalım. Bunlar X, Y, Z olsunlar. Görev X her 10 ms 'de bir çağrılsın, görev Y her 30 ms 'de bir çağrılsın ve görev Z 'de her 25ms de bir çağrılsın. Kesme zamanı ayarlaması yapılırken aşağıdaki faktörler göz önünde bulundurulmalıdır.

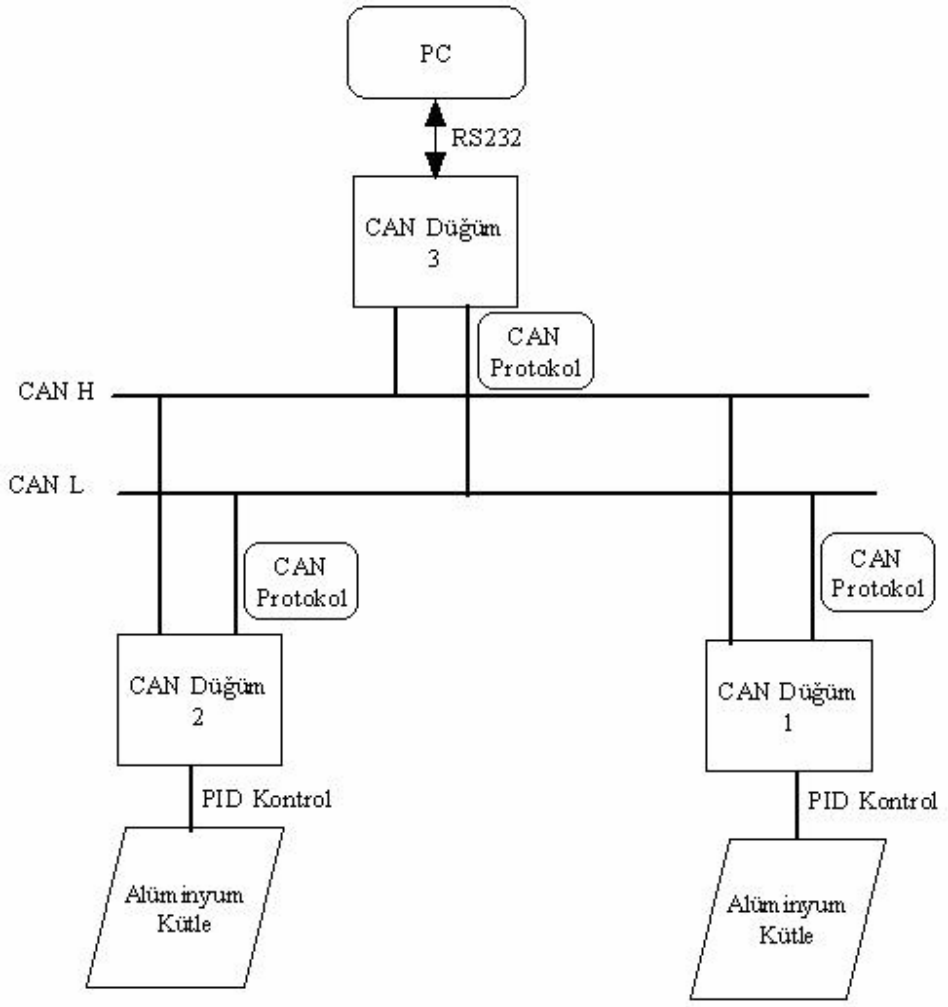
- Görev X 'in tam bölenleri 1 ms, 2 ms, 5 ms ve 10 ms.
- Benzer olarak görev Y 'nin tam bölenleri 1 ms, 2 ms, 3 ms, 5 ms, 6 ms, 10 ms, 15 ms ve 30 ms.
- Son olarak görev Z 'nin tam bölenleri ise 1 ms, 5 ms ve 25 ms.

Yukarıdaki faktörler göz önüne alındığında ortak bölenlerin en büyüğü 5ms 'dir. Bu da bizim kesme zamanımızı belirlemektedir.

Belirlenmiş görevlerde birde başlangıç gecikmesi varsa, bu başlangıç gecikmesi de göz önüne alınmalıdır. Örneğin başlangıç gecikmesi 1 ms ise yukarıda örnek verilen tasarım yok sayılarak kesme zamanı 1 ms olarak belirlenmesi zorunludur.

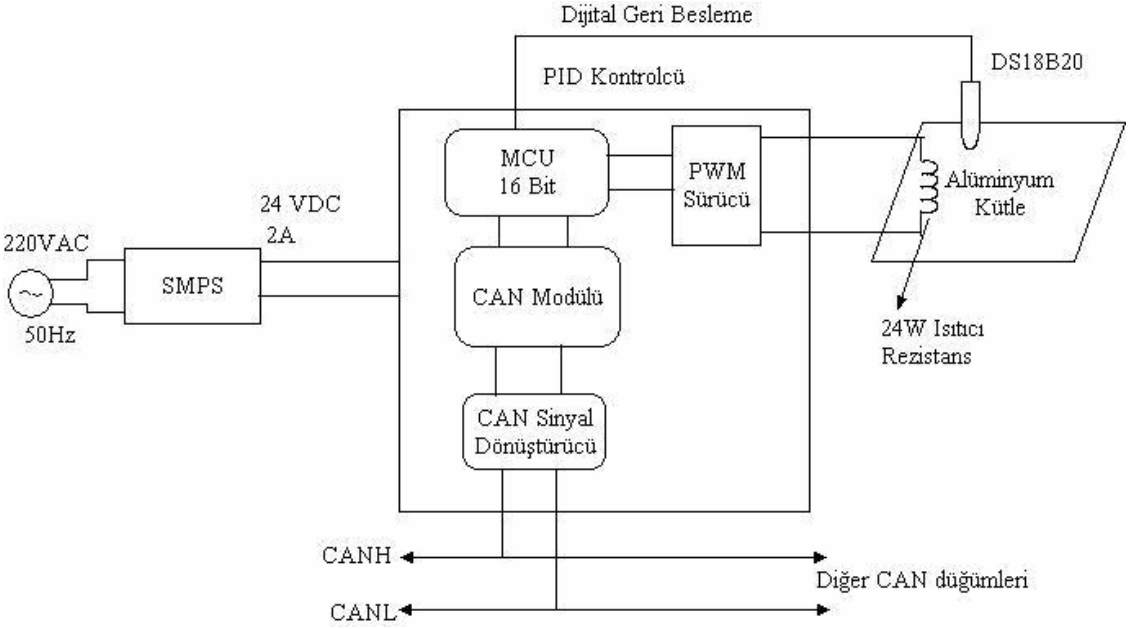
5. CANBUS HABERLEŞME PROTOKOLÜ KULLANARAK BİR ZAMAN TETİKLEMELİ SİSTEM TASARIMI VE KONTROL UYGULAMASI

CAN veri yolu haberleşmesinin üstünlüğünü göstermek için basit bir deney düzeneği kurulmuştur. Bu deneyde iki alüminyum bloğunun iç sıcaklıklarının arzu edilen değerlerde sabit tutulması amaçlanmıştır. Böylece klasik PID denetleyicisi tasarımı gerçekleştirilmiştir. Haberleşme mimarisi temel olarak olarak üç adet CAN düğümüyle oluşturulmuştur. Düğümlerden birinci ve ikincisi alüminyum kütlelerin sıcaklılarını bilgisayardan gelen giriş (referans, setpoint) değerlerinde tutmakla sorumludur. Üçüncü düğüm ise bir dönüştürücü gibi çalışmakta ve CAN protokolünü bilgisayarla bağlantı yapılan RS232 protokolüne dönüştürmekle görevlidir. Şekil 5.1'de sistemin blok diyagramı görülmektedir.



Şekil 5.1 Sistemin blok diyagramı

Bu deneysel çalışmada sıcaklık kontrol sisteminin seçilmesindeki temel düşünce bir binanın farklı noktalarındaki sıcaklık kontrolünün CAN protokolü kullanılarak yapılabileceğinin ve bu tasarımın bir endüstriyel ürüne dönüştürülebilirliğinin gösterilmesidir. Bu amaçla Renesas 16 bit R8C22 mikrodenetleyici ailesi seçilmiştir. Tabii ki bu mikrodenetleyicinin seçilmesindeki en önemli etken CAN protokolünü destekliyor olmasıdır. Yazılım C dilinde geliştirilip derlendikten sonra "flash" belleğe yüklenmiştir. Çalışmada yazılım geliştirme ortamı olarak yine Renesas tarafından desteklenen HEW geliştirme ortamı kullanılmıştır. Sistemin blok diyagramı içerisinde görülen CAN düğüm-1'in ayrıntılı blok diyagramı Şekil 5.2'de gösterilmektedir.



Şekil 5.2 CAN düğüm-1 ayrıntılı blok diyagramı

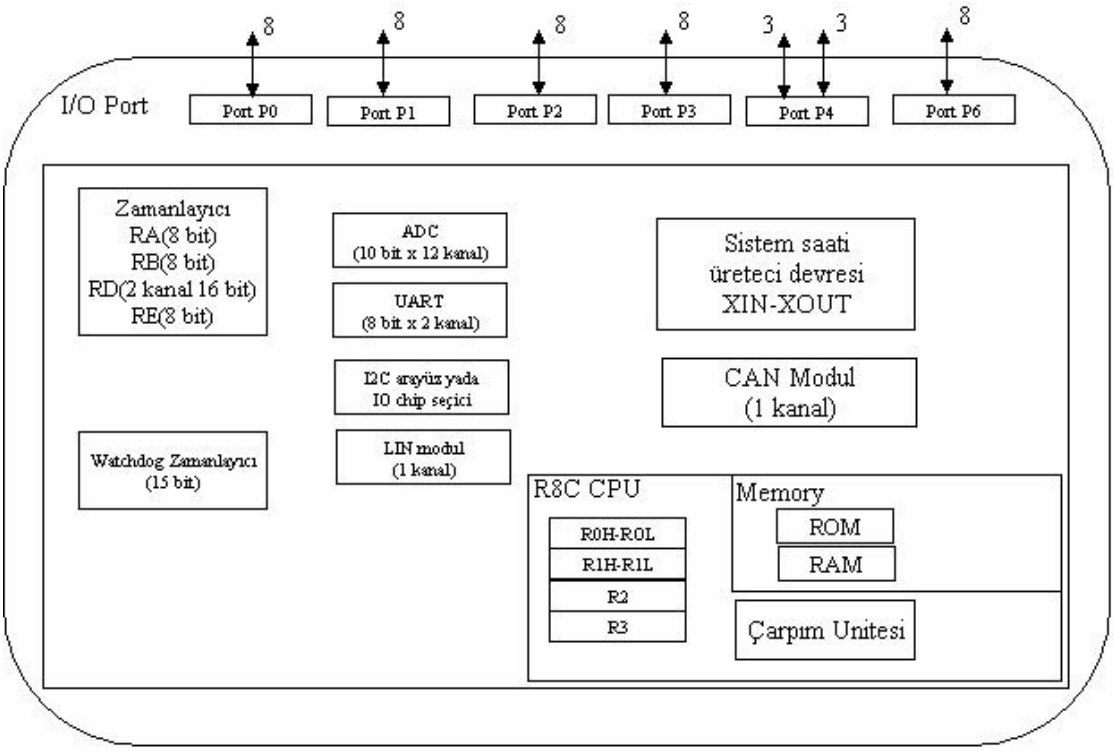
5.1 Uygulamada Kullanılan Donanımın Açıklanması

Bu bölümde uygulamada gerçeklenen elektronik kontrol devresinin donanım elemanlarından bahsedilecektir.

5.1.1 Kullanılan Mikro Denetleyici

Sistemde 3 adet Renesas tarafından üretilmiş R8C mikro denetleyici ailesine ait R5F21227JFP tip numaralı ürün kullanılmıştır. Bu mikro denetleyici 48 adet pinli, 48

Kbyte Eprom, 2.5 Kbyte RAM kapasitesine sahip, bir adet CAN portu bulunan bir mikro denetleyicidir. Bu denetleyicinin en önemli özelliği düşük işlem zamanına sahip olması ve CPU çekirdeğinin 16 bit olmasıdır. Şekil 5.3'de mikro denetleyicinin blok diyagramı gösterilmektedir.



Şekil 5.3 Mikro denetleyici blok diyagramı (Renesas, R8C23 donanım el kitabı)

Yukarıdaki şekle baktığımızda 4 adet donanım zamanlayıcısına sahip bunlar RA (8 bit), RB (8 bit) , RD (16bit 2 kanal) ve RE (8 bit) zamanlayıcılarıdır. Bunun yanında 12 kanal 10 bit ADC, kendi üzerinde osilatör devresi, 1 adet CAN ve LIN modülü matematiksel işlemleri daha hızlı yapmak için çarpıcı donanımı ve 16 bitlik CPU çekirdeği ile oldukça modern bir mikro denetleyicidir.

5.1.2 Gösterge Kartı

Kullanıcı ara yüzü olarak 4x20 toplamda 80 karakterlik bir LCD ekran kullanılmıştır. Data bitlerinin kontrolü için 4 bitlik bir veri yolu kullanılmıştır. Bu da bize 4 bitlik bir port kazancı sağlamıştır.

5.1.3 CAN Veri Yolu Dönüştürücü (Transceiver)

Bölüm 2'de bahsedilen CAN protokolüne uygun Maxim firmasının ürettiği max 3050 entegresi kullanılmıştır. Bu entegre 2 Mbps hıza kadar haberleşmeyi desteklemekte ayrıca el tipi cihazlar için veri yolu bekleme konumunda iken kendini uyku moduna alma ve hat üzerinde yüksek gerilimlere karşı koruma özelliklerine sahip oldukça gelişmiş bir dönüştürücüdür. Bu entegre CAN bölümünde anlatılan ISO11898 standartlarını desteklemektedir.

5.1.4 RS232 Haberleşme Protokolü

Bu bölümde kısaca blok diyagramında görülen PC ile bağlantımız sağlayan RS232 protokolü üzerinde genel bilgiler verilecek ve kullanılan entegreden bahsedilecektir.

RS232 standartı ile iletim seri bir şekilde ve asenkron olarak yapılmaktadır. RS232 hatları TTL sinyal seviyelerini (+5V, OV) taşımazlar. Tipik olarak gerilim seviyeleri +12 V ve -12V'dur. Fakat RS232 hatları, +25V DC'ye kadar yüksek olan sinyal seviyeleri ile -25 VDC'ye kadar düşük olan sinyalleri taşıyabilir. Bilindiği üzere bilgisayardaki veri iletimi ikilik sistemde olmaktadır. Lojik 1'e +5V karşılık gelirken, lojik 0'a OV seviyesi denk gelir. Bu tür bir çevrime TTL (Transistor, Transistor Lojik Level) çevrimi denir. Bu, bilgisayar içindeki haberleşme standartı kabul edilir. Bilgisayar içindeki veri transferlerinde TTL seviyeli sinyallerin kullanılması birkaç sebepten dolayı avantajlıdır. Bunlar:

- Güç yönünden
- Isı dağılımının az olmasından

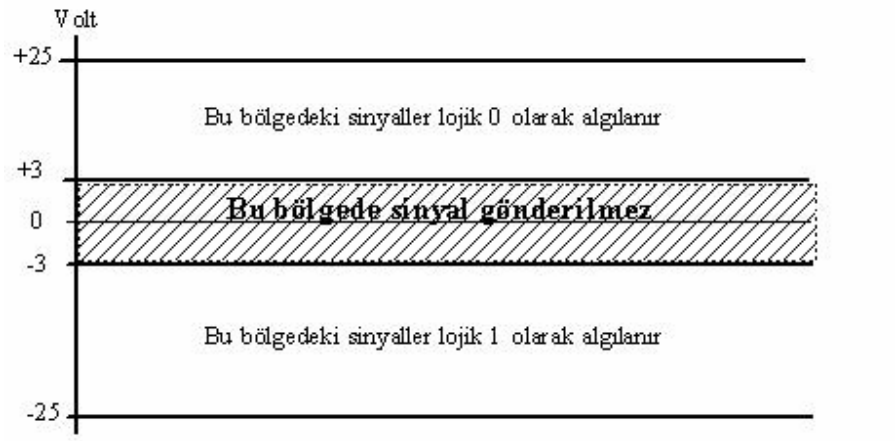
Bu tür çalışan aletler için sürücüye ve alıcıya ihtiyaç duyulmadan doğrudan bağlantı yapılabilir. TTL aletler yüksek hızda çalışabilir. Bu durum bilgisayar içindeki veri transferleri için çok uygundur.

TTL haberleşmesinde mesafe artması durumunda çok ciddi problemler ortaya çıkmaktadır. Ayrıca TTL, dışarıdan gelen sinyallerden çok çabuk etkilenir. Dolayısıyla sinyaldeki birkaç voltluk kayıp sinyalin belirsiz bölgeye düşmesine sebep olur.

Normalde bir konektörün pinleri 4 kısımda incelenebilir.

- Data
- Kontrol
- Zamanlama
- Asenkron

PC'lerde tek gerilim (genellikle 5V) lojik 1 olarak ve 0 V ve toprak da lojik 0 olarak belirtildiğini söylemiştik. Bu tür iletimler bilgisayar içinde sorun çıkarmaktadır. RS232 hatları gürültülü hatlar üzerinde çok uzun mesafelere sinyalleri göndermek zorunda kalabilmektedir. +5 V, uzak mesafelere göndermede bir zayıflamaya maruz kalacaktır. Başarılı bir iletimin sağlanabilmesi için RS232 sinyalleri pozitif bir sinyal için +5 V ile +15 V arasında ve negatif bir sinyal içinde -5 V ile-15 V arasında ve negatif bir sinyal içinse -5V ile -15 V arasında bir değer almalıdır. Bu aralığı bu şekilde tutarak, gürültüden dolayı oluşan gerilim dalgalanmalarından etkilenmesini de minimuma indirmiş oluruz. Bu şekilde bir sinyal göndericiden gönderildiği durumda, RS232 alıcısı için bu sinyal aralığı +3 V'dan yukarısı için pozitif sinyal, -3 V ve bundan aşağı için ise negatif sinyal olarak anlaşılır. Burada -3 V ile +3 V arasındaki bölgede kararsız bir bölgedir. Bu bölgede bulunan bir sinyal, gürültü olarak kabul edilir. Sonuç olarak RS232 hatları pozitif olarak +25 V'a kadar, negatif olarak da -25V'a kadar olan sinyalleri taşıyabilir.



Şekil 5.4 RS232 hatlarındaki verilerin gerilime göre lojik yapısı

Yukarıda bahsedilen fiziksel yapıyı sağlamak için mikro denetleyiciden gelen TTL seviyesini RS232 elektriksel karakteristiğe dönüştürmek için MAX232 entegresi kullanılmıştır.

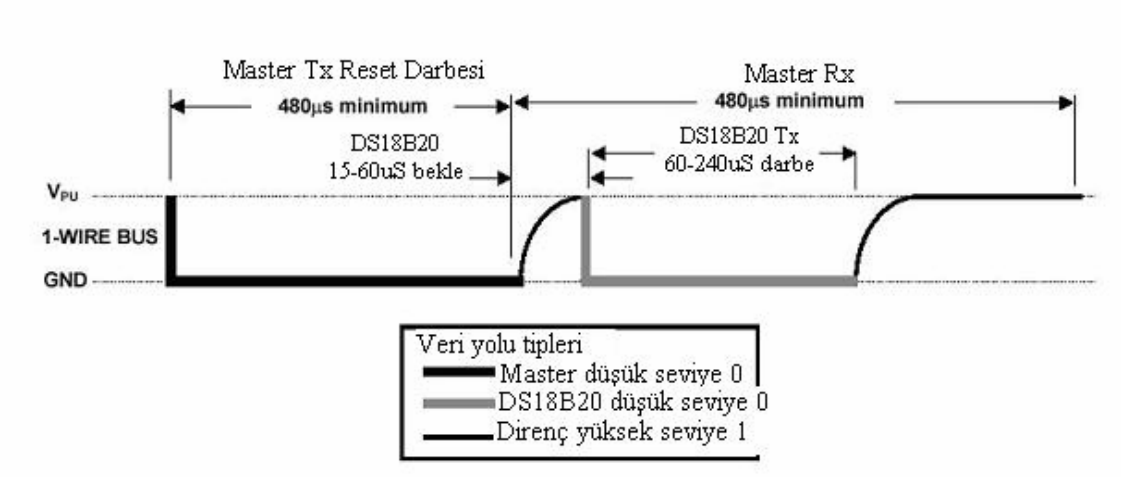
5.1.5 Sıcaklık Sensörleri ve Isıtıcı Rezistans

Sistemde alüminyum kütlelerin sıcaklıklarını ölçen iki adet sıcaklık sensörü bulunmaktadır. Sıcaklık ölçümü için Dallas DS18B20 bir-telli sayısal sıcaklık sensörleri seçilmiştir. Cihazla iletişim için tek bir kapı bacağı gerektiğinden bir-telli yol kullanımı oldukça pratiktir. İletişim protokolü darbe genişliklerine dayanır. Sıcaklık yarım derece hassasiyetle ölçülebilmektedir. Bu yonga -55 ile +125 derece arasındaki sıcaklıklarda kullanılabilmektedir. Bir-telli yol protokolü, aynı hat üzerinden birden fazla cihaz konulmasına olanak sağlayabilir, fakat yazılım karmaşıklığı daha fazladır ve her bir sensörün aynı hat üzerinde ilk olarak adreslenmesi gerekmektedir.

İki kablolu yol ile haberleşmeden farkı daha düşük veri hızı ve daha düşük maliyetli olmasıdır. Genel olarak sayısal termometreler, hava cihazları gibi pahalı olmayan, küçük cihazların haberleşmesinde kullanılır.

Haberleşme sinyalleri darbe genişliğine dayanır. Yol yöneticisi tarafından gönderilen 480 mikrosaniyelik bir darbe, yol üzerinde bulunan cihazların kendilerini sıfırlamasını sağlar. Eğer yol üzerinde çalışan bir cihaz varsa, sıfırlama sinyaline yanıt olarak yaklaşık 200 mikrosaniyelik bir varlık bildirme sinyali gönderir. Böylece yol yöneticisi sıfırlama

işleminin başarıyla gerçekleştiğini ve yol üzerinde aktif cihaz olduğunu anlar. Şekil 5.4 bir kablolu yol üzerindeki sıfırlama sinyalini zamanlama bilgileri ile birlikte göstermektedir.



Şekil 5.5 Bir kablolu yol sıfırlama sinyali (Dallas, 1999)

Veri transferi için ise yine sinyal genişlikleri kullanılır. Yol üzerindeki 60-120 mikro saniye arasındaki bir darbe mantıksal SIFIR anlamına gelirken, 1 mikro saniyeden biraz uzun bir sinyal mantıksal BİR anlamına gelmektedir.

Bir kablolu yol üzerindeki bütün veri iletişimi yol yöneticisi tarafından başlatılır ve yönetilir. Veri iletimi önemli olan bit (MSB) ilk gönderilecek şekilde yapılır.

Bir kablolu yol üzerinde aynı anda birçok cihaz bulunabilir. Cihazların birbiri ile karışmamasını sağlamak için her cihazın sadece kendisine ait, 64 bitlik tanıtım numarası vardır. Yol üzerinde bulunan bütün cihazları tanımlayabilen belirli bir arama algoritması vardır. Bu algoritma kullanılarak, tek kablolu yol üzerinde bulunan bütün cihazlar tanımlanarak indekslenir. Tanımlama sırasında cihazın tanıtım numarasının içinden cihazın ne tür bir cihaz olduğu hakkında gerekli bilgi de alınır. Daha sonra yol üzerindeki istenen cihaza erişmek için o cihazın adresi kullanılır.

Eğer tek kablolu yol üzerinde yalnızca bir adet cihaz bulunuyorsa, bu durumda cihazla haberleşmek için adresini bilmeye ve kullanmaya gerek yoktur. Haberleşme başladıktan sonra gönderilen “SKIP ROM” komutunu alan cihaz bundan sonra yapılan haberleşmenin doğrudan kendisine geldiğini bilir.

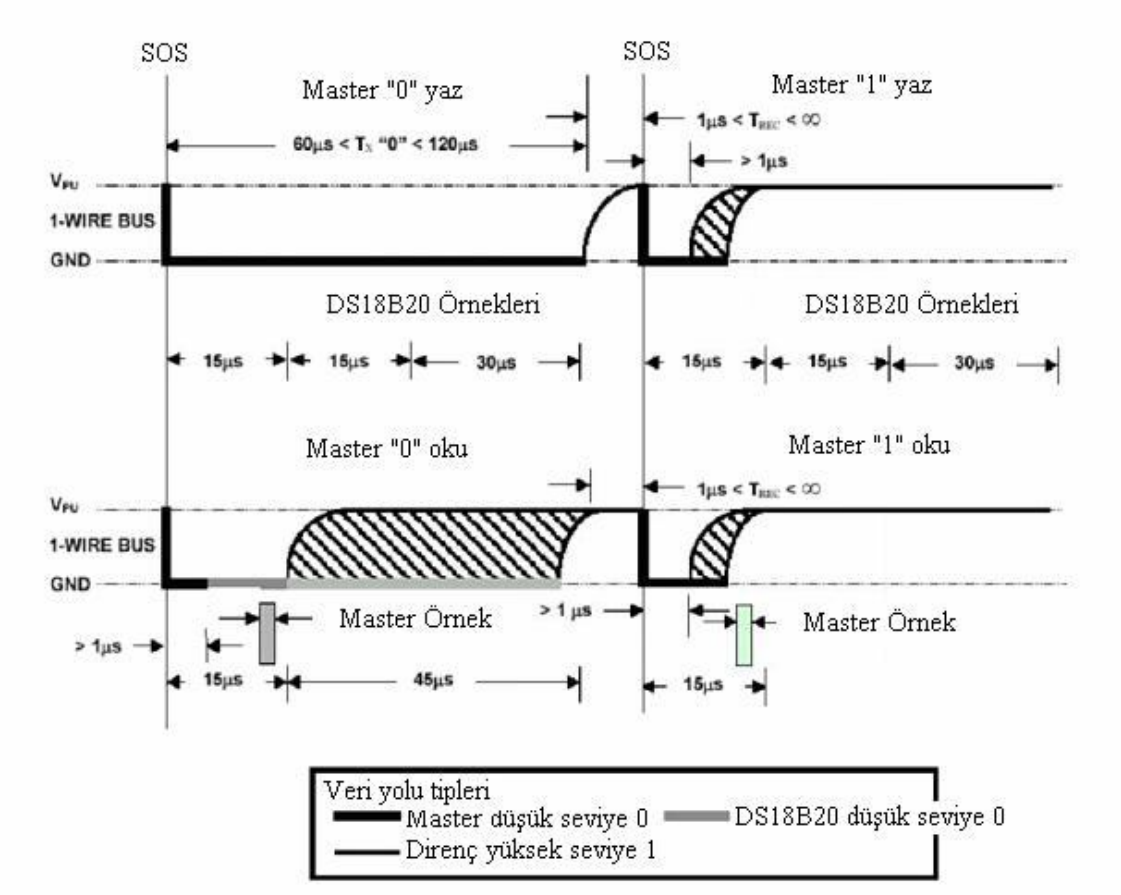
Bir kablolu yol protokolünün en küçük gerçeklemesi kapsamında yazılımda aşağıdaki fonksiyonlar kullanılmıştır.

char init_trans_1wire(void); Bir kablolu yolu sıfırlar. Dönüş değeri olarak yol üzerindeki cihazdan varlık sinyalini alamazsa hata verir.

unsigned char read_byte(void); Yol üzerinde bulunan cihazdan bir bayt veri okur.

void write_byte(unsigned char byte_data); Yol üzerinde bulunan cihaza bir bayt veri azar.

Şekil 5.5'de bir kablolu yol üzerindeki veri bitlerinin transferini zamanlama bilgileri ile birlikte göstermektedir.



Şekil 5.6 Bir kablolu yol veri sinyalleri (Dallas, 1999)

Bir kablolu yol sürücüsü gerçeklenirken gerek duyulan zamanlamalar, boş bekleme döngüleri ile sağlanmıştır. Bekleme sürelerinin mikro saniyeler düzeyinde olması

nedeniyle, bu döngüler genel sistem performansını olumsuz yönde etkilemezler. Ancak yine de sistem verimini artırmak adına sensör okumaları saniyede bir defa yapılır. Bekleme döngüsü sistemin şu anki çalışma frekansına uygun olarak ayarlanmıştır. Bu nedenle sistemin çalışma frekansı değiştirilirse tekrar ince ayar yapılması gereklidir. Aksi takdirde tek kablolu yol sürücüsü yol üzerindeki cihazlara erişemeyecektir.

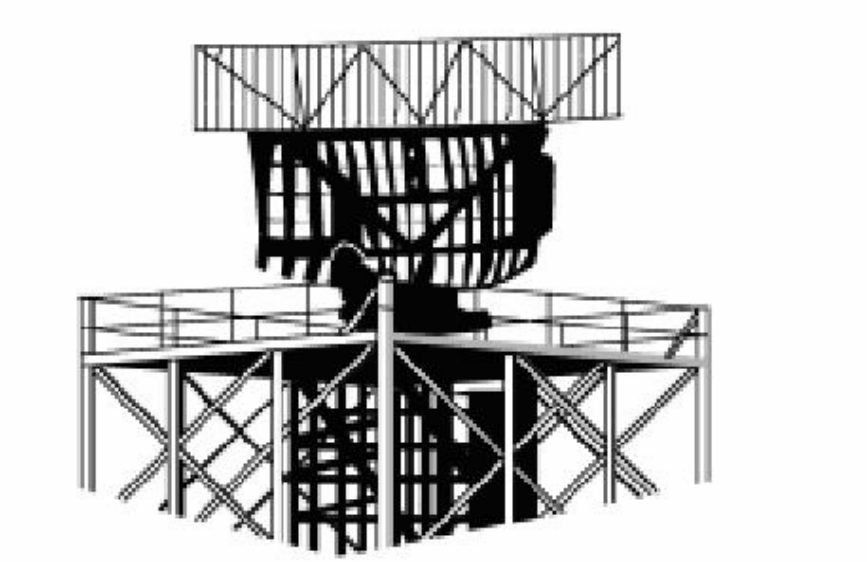
Bu uygulamada kullanılan alüminyum kütleleri ısıtmak için 24 V DC gerilimle çalışan 24 W'lık bir ısıtıcı rezistans kullanılmıştır.

5.2 Sıcaklığı Sabit Tutmak İçin Tasarlanan PID Algoritması

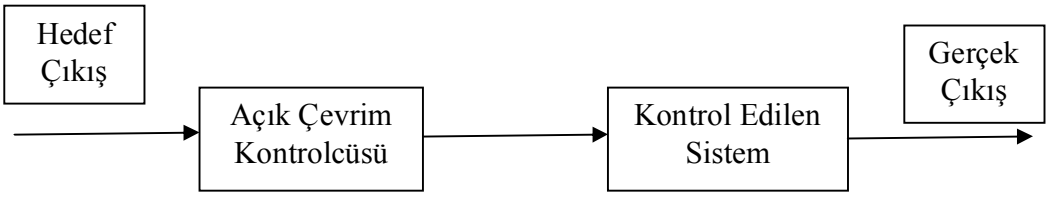
Sürekli zamanda PID denetleyicisi girişindeki istenen sıcaklıkla gerçek sıcaklık arasındaki farkı temsil eden $e(t)$ hata işaretini, oran, K, integral, K_i, ve türev, K_d, katsayıları ile oransal, türevsel ve tümlevsel olarak işleyerek denetim işareti olarak kullanılacak $u(t)$ çıkış işaretini üretir. Bu bölümde kısaca, proses kontrol işleminde kullanılan PID kontrolcüsü hakkında bilgi verilecektir. Bunlara ek olarak kullanılan kontrol algoritmasının temellerinden bahsedilecektir. PID algoritması oldukça kullanışlı, gerçeklenmesi kolay ve kontrol sistemlerinde çok sık kullanılan bir tekniktir. Ayrıca bu algoritmanın tasarımı ve gömülü bir sisteme uygulanması üzerinde duracağız.

5.2.1 Açık Çevrim Kontrol

Örnek olarak hava trafik denetlemede kullanılan bir radar sisteminde var olan bir DC motorun kontrol edilmesini ele alalım. Radar sistemi Şekil 5.7'de görülmektedir. Motor hızını değiştirmek için kontrolcü çıkışında DAC'ın sayısal değerlerini değiştirerek ayarlamaya çalışalım.



Şekil 5.7 Hava trafik uygulamasında kullanılan bir radar sistemi



Şekil 5.8 Açık çevrim kontrol şematik gösterilimi

Açık çevrim kontrol sistemlerinde çıkış değerini belirlemek için sistem parametrelerinin bilinmesi gereklidir. Örnek verilen radar kontrol sistemlerinde DC motorun yapısı, radar donanımı ve motor sürücü devresi hakkında tam bir bilgi sahibi olduktan sonra motor hızı kontrol edilebilir.

Gerçek hayatta bu tip bir açık kontrol devresinin gerçeklenmesi oldukça basittir. İstenilen çıkış hızı için bir look-up tablosu yapılabilir. Mesela, bir navigasyon sisteminde istenilen değer look-up tablosunda okunur ve digital analog çevirici (DAC) çıkışına uygulanır.

Çizelge 5.1 Hava radar kontrol sistemi look-up tablosu

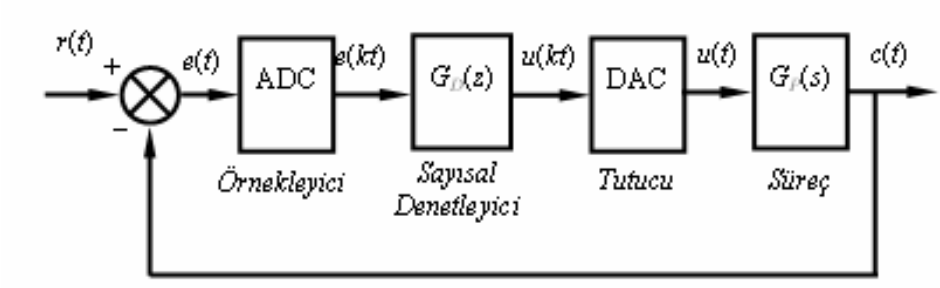
Motor Dönüş Hızı (Rpm)	DAC çıkışı (8 bit)
0	0

2	51
4	102
6	153
8	204
10	255

Bu sistem tek giriş tek çıkış (SISO) olarak isimlendirilir. Ne yazık ki pratik uygulamalarda bu tip lineer sistemler oldukça az sayıdadır. Radar sisteminde verilen look-up tablosu rüzgarsız bir ortam göz önünde bulunarak gerçeklenmiştir. Rüzgar parametresi de göz önüne alındığında her bir rüzgar hızına göre yeni bir tablo yapmak gerekmektedir. Bu da gerçek hayatta pek uygulanabilir bir durum değildir.

5.2.2 Kapalı Çevrim Kontrol

Klasik tasarım yöntemleri dolaylı veya doğrudan olmak üzere iki biçimde uygulanır. Dolaylı uygulamada, süreç dinamiklerine uygun bir analog PID denetleyici tasarlanır ve daha sonra bu denetleyici ayrıklaştırılarak fark denklemi halinde sayısal işlemciye yazılım olarak aktarılır. Doğrudan uygulamada ise sürecin ayrık modeli elde edilerek yine analog denetleyici tasarımlarında kullanılan yöntemlerle ayrık PID denetleyici tasarımı gerçeklenir. Ele alınan örnek uygulama için de geçerli genel bir sayısal kontrol sistemi blok çizimi Şekil 5.9'da gösterilmiştir. Açık çevrim kazancındaki ana problem kontrolcünün şuursuz olmasıdır. Yani kontrolcü kontrol etmek istediği sistem hakkında hiç bir geri besleme bilgisine sahip değildir.



Şekil 5.9 Kapalı çevrim sayısal kontrol sistemi

Belirtildiği gibi, PID denetleyici, endüstriyel kontrol sistemlerinde çokça kullanılan geri beslemeli bir kontrol sistemidir ve temel olarak üç ayrı parametre içerir. Bunlar, oran kazancı (K_p), integral kazancı (K_i) ve türev kazancıdır (K_d). Bu katsayıların denetlenecek sürece, maruz kalınan iç ve dış bozuculara göre seçilmesi ve ayarlanması literatürde önemli bir yer teşkil eder. Esas olarak, oran kazancı cevabın yükselme süresini kısaltır, fakat fazla büyük olması aşımlara sebep olabilir; integral kazancı kalıcı-hal hatasını, e_{ss}, azaltmaya yararken iyi ayarlanmaması yerleşme süresini uzatabilir; türev kazancı aşımlara yol açmadan cevabı hızlandırmaya yarar. Bu üçlünün birbirini karşılıklı etkilemesi de söz konusu olduğundan PID katsayılarının seçilmesi ve ayarlanması önemli bir çalışma konusudur.

5.2.3 PID Kontrol Tanımı

Bir kontrol kitabını açtığımızda (5.1) verilen eşitlik görülecektir. Burada PID eşitliği görülmektedir. Bu eşitlik oldukça karmaşık görünmesine rağmen neyi hedeflediği çok iyi anlaşılırsa, gerçeklenmesi oldukça basit olacaktır. Sürekli zamanda PID denetim işlemi oran, K, integral, K_i, ve türev, K_d, katsayıları cinsinden zaman düzleminde (5.1) denklemindeki gibi ifade edilir (N.S. Nise,2004; Carnegie Mellon University).

Aşağıda bütün PID algoritmasının eşitliği ve terimleri verilmiştir.

```
// Oransal kısım
Change_in_controller_output = PID_KP * Error;
// Integral kısım
Sum += Error;
Change_in_controller_output += PID_KI * Sum;
// Türev kısım
Change_in_controller_output += (PID_KD * SAMPLE_RATE * (Error - Old_error));
```

$$u(t) = Ke(t) + K_i \int_{t_0}^{t} e(t)\mathrm{d}t + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \qquad (5.1)$$

Ancak, daha anlamlı bulunduğundan yukarıdaki ifade pratikte saniye cinsinden integral (reset) zamanı, T_i, ve türev (hız) zamanı, T_d, değişkenleriyle (5.2) denklemindeki gibi formüle edilir (K. Ogata, 2002).

$$u(t) = K\left[e(t) + \frac{1}{T_i} \int_{t_0}^{t} e(t)\mathrm{d}t + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right] \qquad (5.2)$$

PID algoritması üç temel bileşenden oluşur. P oransal , I İntegral ve D Türev parçalarından oluşur.

Başlangıç olarak P yani oransal bileşen üzerinde duracağım. Bu parametre PID kontrolün ana bileşenini oluşturur. Kontrol üzerinde en fazla etkisi olan parametredir. Yalnız bu parametrenin kullanılıp diğerlerinin sıfırlandığı kontrol sistemleri yalnız P veya yalnız oransal kontrol olarak adlandırılır. I ve D parametreleri kontrol sisteminin ince ayarlarını yapmak amacıyla kullanılırlar.

Bu algoritmayı anlamak için günlük yaşamdan örnek verecek olursak; otobanda 30 km/sa hızla ilerlerken 35 km/sa hıza çıkmak istersek çok hafif bir şekilde gaz pedalına basarız ama 70 km/sa hıza çıkmak istersek oldukça sert bir şekilde gaz pedalına basmamız gerekir.

Yukarıdaki durum oransal kontrolün temelini oluşturur. Burada her zaman istenilen ve şu anki hız ölçülerek ikisi arasındaki hata değeri hesaplanır. Oransal kontrolün amacı bu hatayı sıfıra indirgemektir. Bunu gerçeklemek için hatayı belli bir oranla çarpıp çıkışa bir gerilim uygulanır. Kontrolcü çıkışını yukarıda söylenenler doğrultusunda şu şekilde olmalıdır.

Change_in_controller_output = PID_KP * Error;
PID_KP oransal kazanç olarak adlandırılır. Bu parametre her bir kontrol algoritmasına özel olarak kullanıcı tarafından ayarlanması gerekmektedir. Ek 11'de PID algoritmasının akış diyagramı görülmektedir.

5.2.4 Pencere Koruması

Yukarıda örnek olarak verilen algoritma küçük değişiklikler ile oldukça iyi sonuç vermektedir. Integral elementine baktığımızda sürekli bir hata toplamı söz konusudur (integral windup). Eğer bu artım kontrol altına alınmazsa çıkışta uygulanan PWM değeri sürekli artacak ve bu da sistemi karasız hale sokacaktır.

Sum_G += Error;
Control_new += PID_KI * Sum_G;
Sistem çıkışına %100 "duty cycle" sahip PWM sinyali uygulanıyorsa ya da %0 uygulanıyorsa bunun bir koruma altına alınıp buradaki Sum_G değeri tekrar çıkarılarak

yeni kontrol çıkışı ya tam ya da sıfır değerine ayarlanmalıdır. Böylelikle çıkış limiti son değere ulaştığında, bunun daha fazla artmasını önlemekle sistem performansı arttırılmış olur.

5.2.5 Kontrol Parametrelerinin Ayarlanması

PID algoritması uygulanırken iki önemli nokta özellikle dikkate alınmalıdır. Bunlardan birincisi PID çıkış işaretinin sınırlarının belirlenmesi ve dolayısıyla integral birikmesi"ni engelleyici tedbirin algoritmaya alınmasıdır. Diğer önemli nokta ise PID katsayılarının ayarlanmasıdır. Birinci sorunun giderilmesi için çeşitli yöntemler geliştirilmiştir, yukarıda belirtildiği gibi, basit bir yolla bu sorun çözülmüştür. Diğer önemli nokta olan katsayıların belirlenmesi hususunda klasik Ziegler Nichols yönteminden yararlanılmıştır (K. Ogata, 2002). Endüstride yarım yüzyılı aşkın bir süredir yaygın olarak kullanılan bu yöntem PID katsayılarının hiç değilse başlangıç değerlerinin belirlenmesi bakımından hâlâ oldukça yararlıdır. Bu yöntemden çok basitçe aşağıdaki gibi yararlanılmıştır.

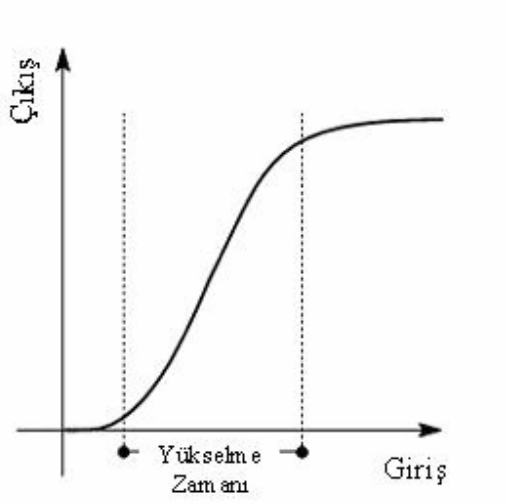
1. K_I ve K_D parametreleri sıfırlanarak işleme oransal kontrolle başlanır.
2. K_P parametresi yavaşça arttırılarak sistemin tam salınıma girmesi sağlanır.
3. Elde edilen K_P değeri yarıya indirilir.
4. Eğer gerekliyse K_D değeri küçük artımlarla arttırılır ve sistem cevabı ölçülür.
5. Eğer gerekliyse K_I değeri arttırılır ve devamlı hata azaltılmaya çalışılır.
6. K_I değeri sıfırdan farklı ise integral birikimini önlemek için her zaman pencere koruması kullanılmalıdır.

Kullanılan yöntem "Ziegler and Nichols" yönteminin basitleştirilmiş versiyonudur.

5.2.6 Örnekleme Zamanın Ayarlanması

Örnekleme zamanın doğru seçilmesi kontrol sisteminin verimliliği açısından oldukça önemli bir noktadır. Düşük bir örnekleme frekansı seçilirse alising denilen örtüşme efekti oluşur. Buradaki örnekleme zamanı sistemin band genişliğinin minimum iki katı olmalıdır. Örneğin bir ses örneklenecekse sesin frekansı 4 KHz kabul edilirse bu minimum 8 KHz ile örneklenmelidir ve 4 KHz üzerindeki işaretler bir anti-alising filtre yardımı ile filtrelenmelidir.

Kontrol sistemlerinde bunun istenilen örnekleme frekansının bulunması biraz daha farklıdır. İlk olarak sistemin yükselme zamanı tespit edilmelidir. Örnek olarak bir aracın motoru rölantide çalışıyorken gaz kelebeği sonuna kadar açıldığında araç maksimum motor devrine ne kadar sürede oluştuğunu bulmak için, zamanın açık çevrim kapalı sistemde deneysel olarak gözlenmesi gereklidir. Buradaki örnekleme zamanı ölçülen yükselme zamanının 1/6'sı olabilir. Şekil 5.10 yükselme zamanı grafiği gösterilmiştir.



Şekil 5.10 Yükselme zamanın ölçülmesi

Bu kabullenmede daha fazla bilgi için Franklin et al., 1994, kitabı incelenmelidir.

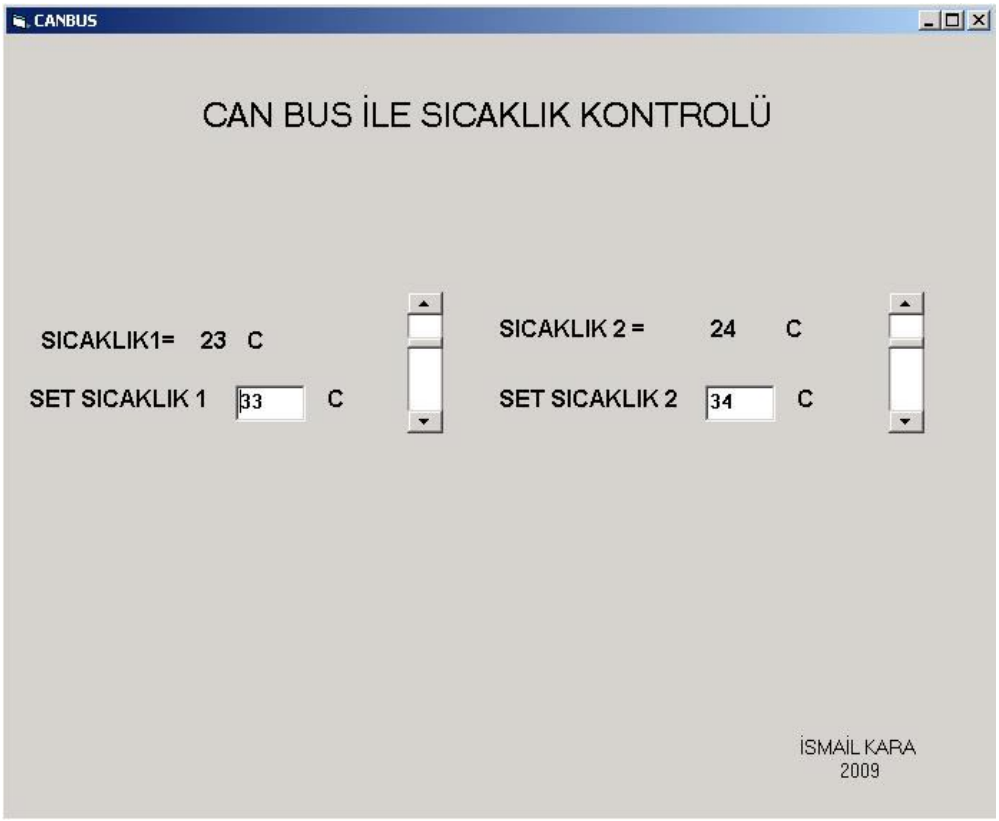
5.2.7 PID Algoritmasının Avantaj ve Dezavantajları

- Birçok tek giriş tek çıkış (SISO) sistemi için kullanışlıdır.
- Genel olarak efektiftir.
- Gerçeklenmesi oldukça kolaydır.
- Çok giriş çok çıkışlı sistemler için uygun değildir.
- PID parametrelerinin ayarlanması zaman kaybına yol açabilir.

5.3 Uygulama Ara yüz Yazılımı

Uygulama geliştirme ortamı olarak Visual Basic 6.0 seçilmiştir. Bu geliştirme ortamının seçilmesindeki en büyük etken RS232 kütüphanelerinin mevcut olması kullanıcı arabirim tasarlamanın oldukça basit olması ve geliştirme sürecinin oldukça kısa olmasıdır.

Verilerin güncellenmesi işleminde bu kütüphanelerde bulunan kesme alt programları kullanılmıştır. Şekil 5.11'de bilgisayardan sıcaklık değerlerini ayarlamak amacıyla tasarlanan kullanıcı ara yüz programı görülmektedir.



Şekil 5.11 Sıcaklık ayar ara yüz programı

5.3 Uygulamada Kullanılan Parametreler

Uygulamada kullanılan CAN veri yolu haberleşme hızı olarak 1Mbps olarak seçilmiştir. Ayrıca genişletilmiş CAN veya CAN 2.0 versiyonu olan en son versiyon kullanılmıştır. Sıcaklık sensörlerinden ölçülen veriler bilgisayar ekranında başarıyla izlenmiş ve

bilgisayar ekranından ayarlanan sıcaklık değerleri ilgili CAN düğümüne başarıyla iletilmiştir. Burada bilgisayar ve dönüştürücü kart arasındaki haberleşme protokolü olarak RS232 kullanılmış ve haberleşme hızı olarak 19200 Kbps olarak seçilmiştir. Buradaki bilgilerden de anlaşılacağı üzere CAN haberleşmesi RS232 haberleşmesine göre yaklaşık 50 kat daha hızlı ve güvenli olarak veri iletimi sağlamaktadır. Uygulamaya ait devre şeması, PCB çizimi ve yapılan kartların resimleri ekler bölümünde verilmiştir. Çizelge 5.2'de RS232 ve CAN haberleşme protokollerinin ayrıntılı olarak karşılaştırma çizelgesi verilmiştir.

Çizelge 5.2 RS232 ve CAN haberleşme protokollerinin karşılaştırılması

Özellikler	CAN	RS232
Operasyon Modu	Farksal sonlamış	Tek sonlanmış
Ağa bağlanacak maksimum düğüm sayısı	32 (ISO11519) Adet	1 Adet
Maksimum kablo uzunluğu	1 km	15 m
Makimum veri hızı	1 Mbps	160 kbps
Hata denetimi	gelişmiş	yok
Tek seferde gönderilen byte sayısı	8	1

Yukarıdaki çizelgede CAN haberleşme protokolünün çoklu mikro denetleyiciye sahip ağ yapıları için oldukça uygun olduğu kesindir. Ayrıca gerilim farkı mantığıyla çalıştığı için gürültüden en az seviyede etkilenmektedir ve bu da daha güvenli bir veri yolu anlamına gelmektedir. Bir diğer üstünlük ise hata denetimi konusundadır. CAN haberleşme protokolüne özgü hata denetimi sayesinde veri güvenilirliği %100'e yakın yani sıfır hata ile sağlanmaktadır.

Tasarlanan devrenin elektriksel karakteristikleri ise şöyledir:

Şekil 5.2'ye baktığımızda girişte 220 VAC şebeke gerilimini 24 VDC gerilime çeviren bir güç kaynağı kullanılmıştır. Bu güç kaynağının çıkışı 24 VDC ve 2A çıkışa sahiptir. Ayrıca tasarlanan kontrol kartının üzerinde 24 VDC gerilimi 5 VDC gerilime düşürmek içim LM317 gerilim regülatörü kullanılmıştır.

6. SONUÇLAR

Bu tez kapsamında, CAN protokolü gerçeklenmiş ve bir uygulama ile simülasyonu yapılmıştır. Bu tezin yapılmasında üç ana hedef üzerinde durulmuştur. Bunlardan birincisi ve en önemlisi CAN protokolünün gerçeklenmesi diğer haberleşme protokollerine göre avantaj ve dezavantajlarının belirlemesi ve sıcaklık kontrolü ile bir kontrol devresine uygulanmasıdır. İkinci önemli hedef ise kullanılan yazılımın güvenilir, daha az bellek harcayan ve geliştirme sürecini kısaltmak maksadıyla yazılan zaman tetiklemeli işletim sistemidir. Bu sistem yazılımın ana gövdesini oluşturmaktadır. En son olarak da sıcaklık kontrolü yapılacak uygulamada kullanılan PID kontrolcüsünün tasarımıdır.

Projede yazılan kaynak kod büyüklüğü yaklaşık olarak 6000 satırdır. Tabi ki burada üç adet CAN düğümü mevcut olduğu için her bir düğüm üzerinde farklı kodlar koşmaktadır.

CAN haberleşme protokolü kullanmanın avantajları olarak aşağıdaki özellikler sıralanabilir;

- CAN haberleşme protokolü mesaj tabanlı bir haberleşme protokolüdür ve her bir mesaj ID' si ile 8 byte uzunlukta veri göndermek mümkündür. Mesaj tabanlı olmasının en büyük avantajı çoklu master desteğini vermesidir.
- Günümüzde Renesas dahil birçok mikro denetleyici firması bu protokolü desteklemektedir. Bu nedenle geliştirme açısından gerekli kaynaklar mevcuttur.
- CAN donanım yapısı oldukça gelişmiş hata denetim mekanizmalarına sahiptir. Buda ağ üzerinde daha güvenli bir veri alışverişi anlamına gelir. Ayrıca diğer ağ yapılarında bu hata denetimleri yazılımsal olarak yapılmak zorunda olduklarından daha fazla yazılım yükü getirirler.
- Diğer kontrol amaçlı kullanılan haberleşme protokolleriyle karşılaştırıldığında oldukça yüksek bir haberleşme hızına sahiptir. CAN 2.0B versiyonu 1 Mbps hızını 100 m'ye kadar desteklemektedir.

Dezavantaj olarak aşağıdaki örnek verilebilir;

- Diğer haberleşme protokolüne sahip mikro denetleyicilerle karşılaştırıldığında maliyet olarak biraz daha pahalıdır. Ancak bu pahalılık CAN protokolünün daha yaygınlaşmasıyla azalacaktır.

Bu uygulama sistemini daha da ileri götürmek isteyenler için gelecekte yapılacak işlerden bazıları şu şekilde sıralanabilir;

- Yazılımda kullanılan kod boyu ve bellek kullanımı açısından optimize edilebilir.
- Burada yapılan uygulama bir binanın kalorifer kontrol sisteminin sıcaklığını kontrol etmek amacıyla kullanılabilir.
- Sıcaklık kontrolü için kullanılan PID kontrol geliştirilebilir.
- Bulanık Mantık tabanlı denetim uygulanabilir.
- Bu sistem Ethernet arabirimi eklenerek internet üzerinden izlenebilir bir sistem haline dönüştürülebilir.

KAYNAKLAR

Barr, M., Programming Embedded Systems in C and C++, O'Reaily, New York, 1999.

Zurel, K., C Programming for Embedded Systems, R&D Books, Kansas, 2000.

Michael J Pont, "Patterns for Time-Triggered Embedded Systems", 2001.

N.S. Nise, "Control Systems Engineering", J. Wiley, 4. baskı, 2004.

Dallas Semiconductors, The 1-WireBus Specification, Dallas, 1999

Maxim Semiconductor, "Max30555 High-Speed CAN Transceiver", Data Sheet, 2003.
Renesas Semiconductors, "CAN Specification", 2004.
R. Bosch, "CAN Specification Version 2.0", Manual, 1991.

Burns, A., Real-time systems and Programming Languages, Addison-Wesley, 2001.

Francesco Balena, "Programming Visual Basic 6.0", Microsoft Press, 1999.

Rodriguez, E. Diaz, J. Vazquez, N. Hurtado, J. Correa, J. "Digitally Addressable Digital Dimming Electronic Ballast Based on CAN Bus", Applied Power Electronics Conference, APEC 2007, Anaheim, CA, USA, pp.281-286, 2007.
S. Altınışık, Ş. N. Engin, "PIC Tabanlı Ayrık Filtre ve Denetleyici Gerçeklemesi için bir Blok Diyagram Derleyici Tasarımı", TOK'08, İTÜ, Maslak, İstanbul.

ÖZGEÇMİŞ

Doğum tarihi 19.10.1981

Doğum yeri Tarsus/ MERSİN

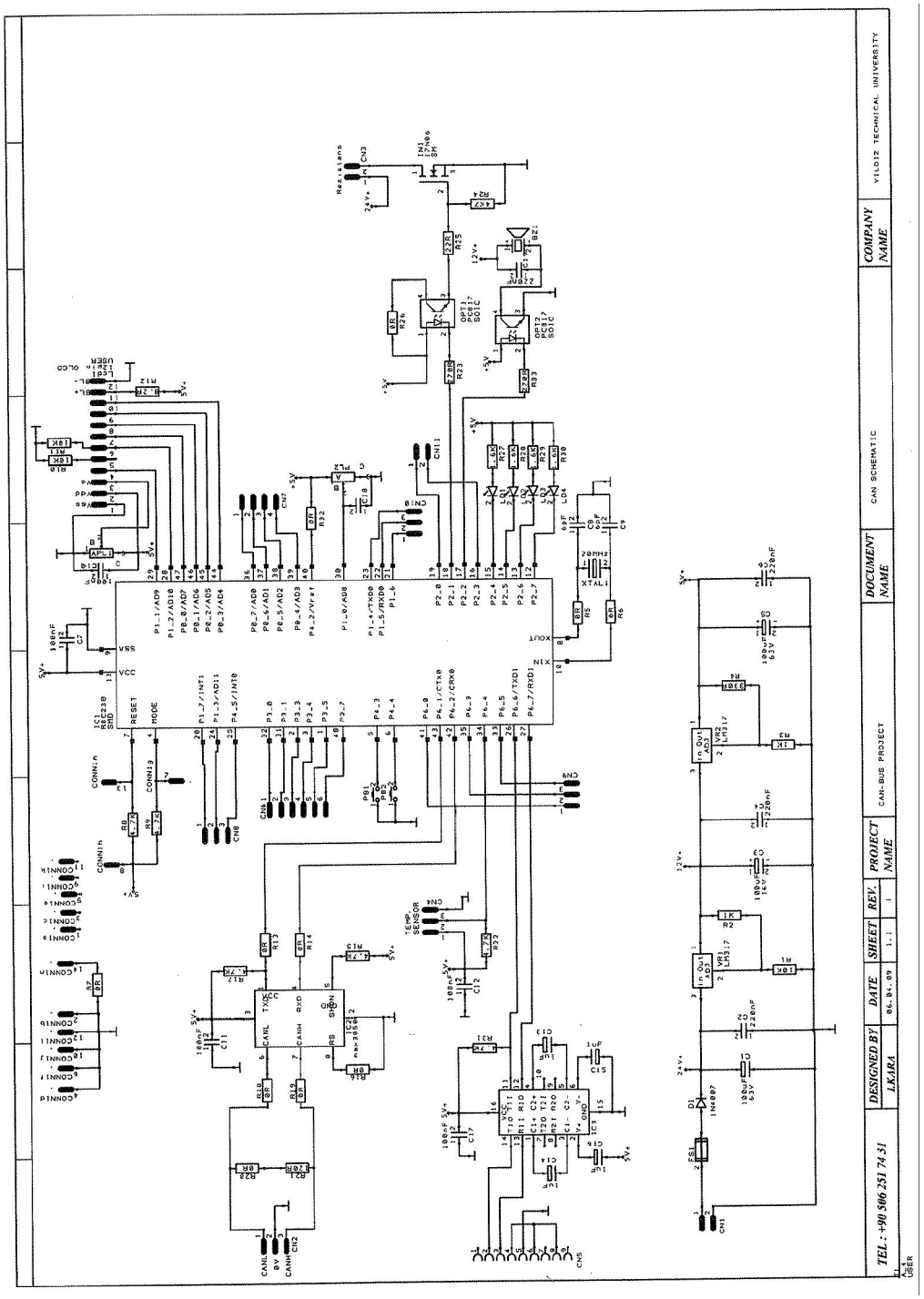
Lise	1997-2000	Tarsus Cengiz Topel Lisesi
Lisans	2000-2005	Yıldız Teknik Üniversitesi Mühendislik Fak. Elektronik ve Haberleşme Mühendisliği Bölümü
Yüksek Lisans	2006-2009	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Kumanda Sistemleri Anabilim Dalı Kontrol ve Otomasyon Programı

Çalıştığı Kurumlar

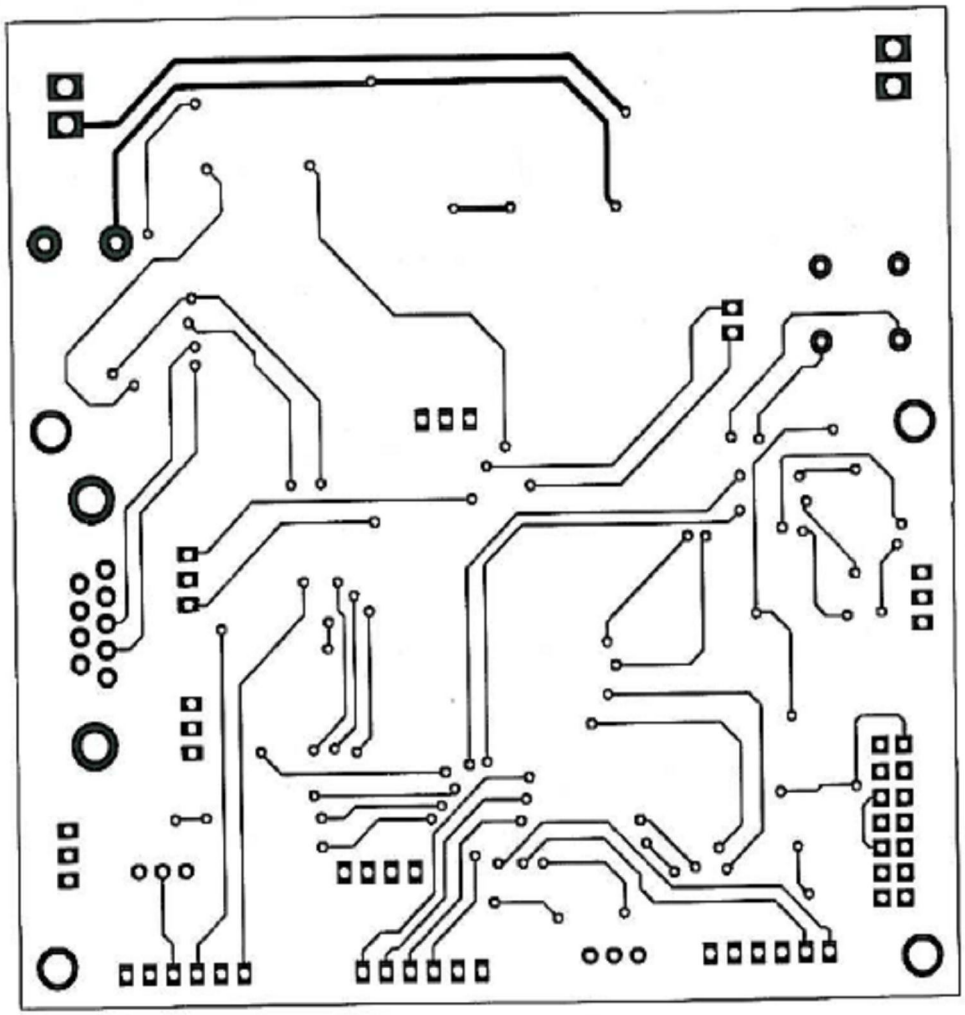
2005 - 2007	İnci Otomasyon Ltd. Şti. Arge Mühendisi
2007- …..	Nortel- Netaş Yazılım Geliştirme Mühendisi (E911 Line Services)

EKLER

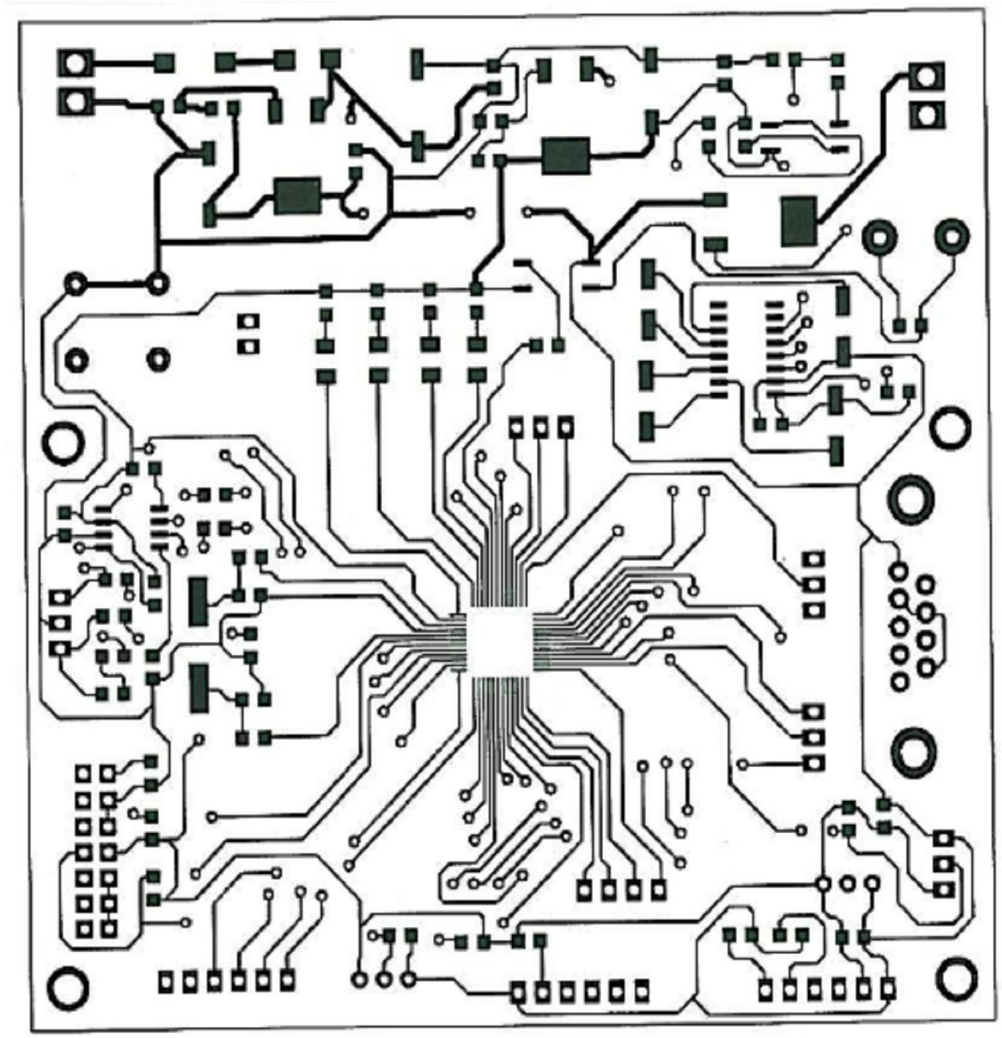
Ek 1 Uygulama Devre Şeması



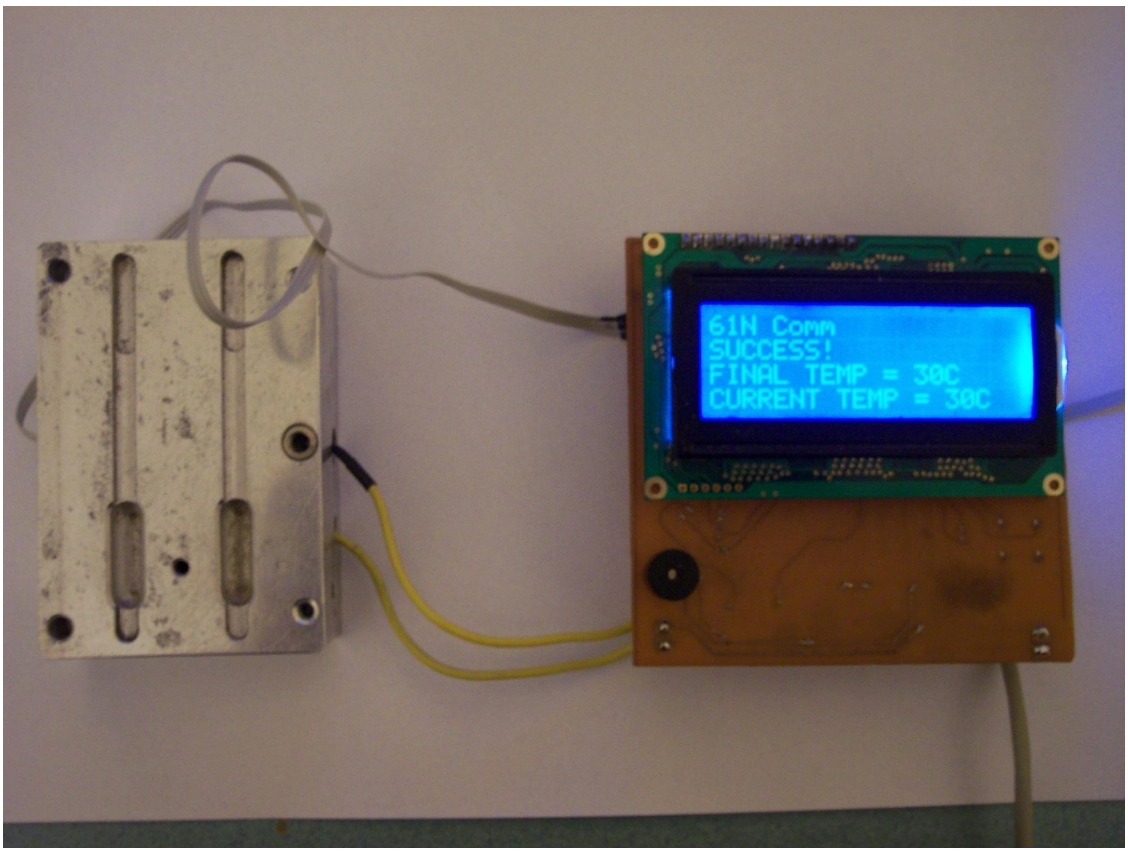
Ek 2 Uygulama Devresi PCB Alt



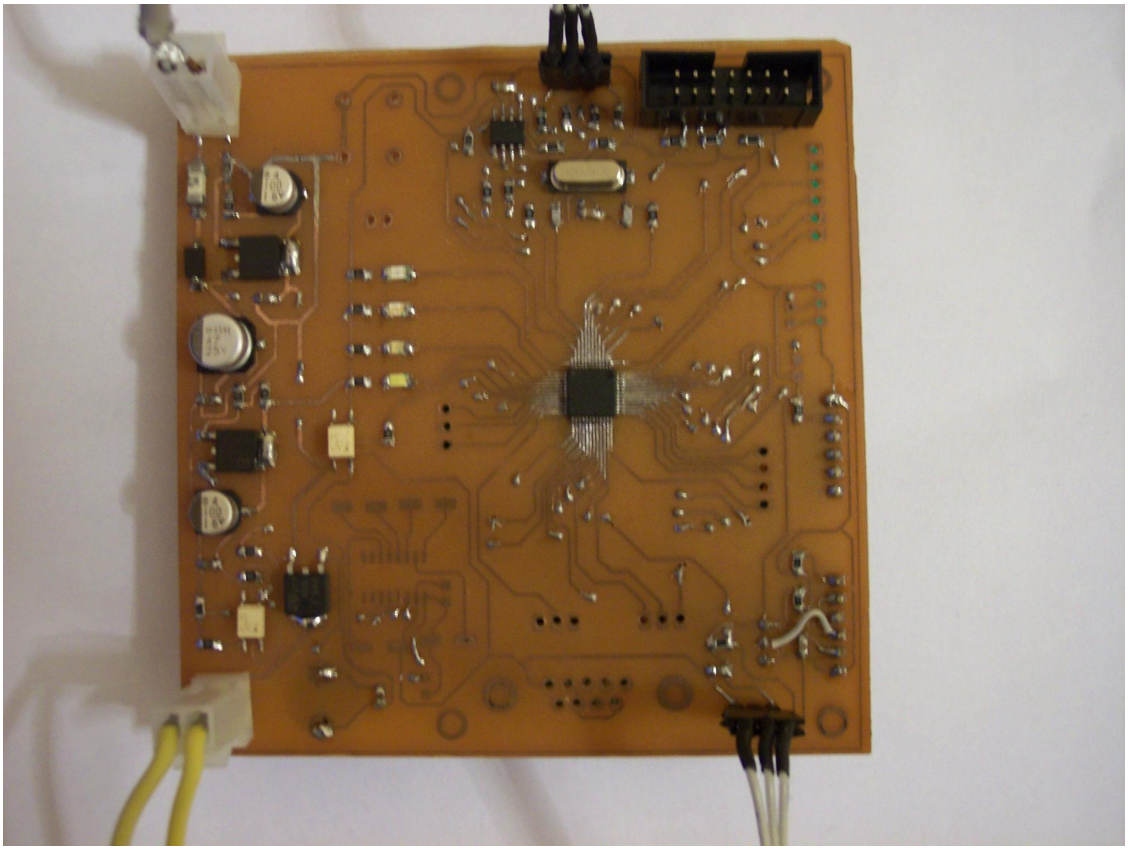
Ek 3 Uygulama Devresi PCB Üst



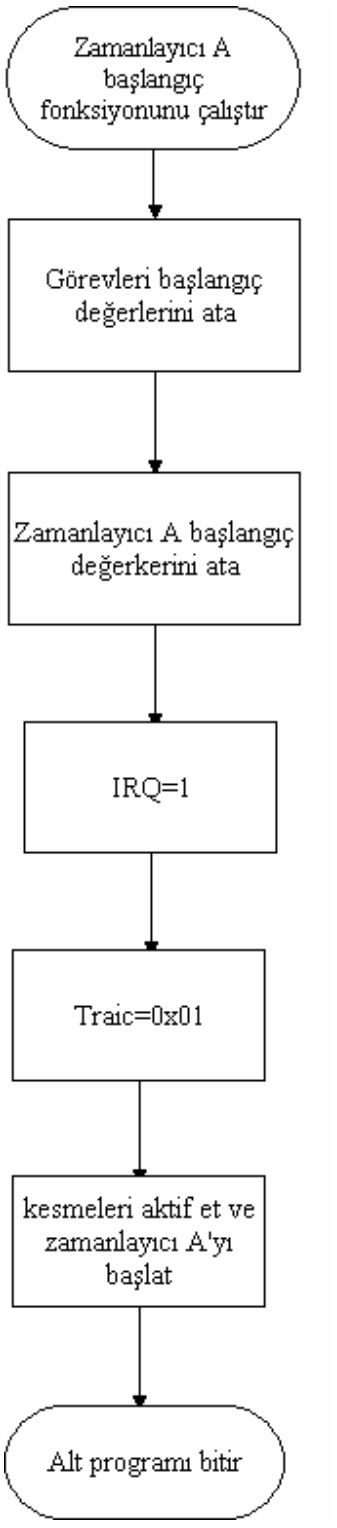
Ek 4 Uygulama Devresi Üstten Görünüm



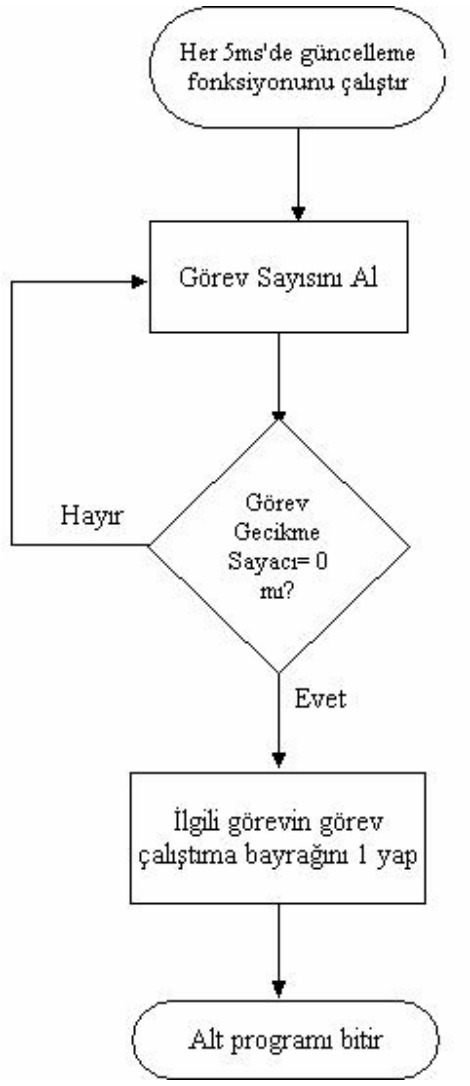
Ek 5 Uygulama Devresi Alttan Görünüm



Ek 6 Zamanlayıcı A Başlangıç Fonksiyonu



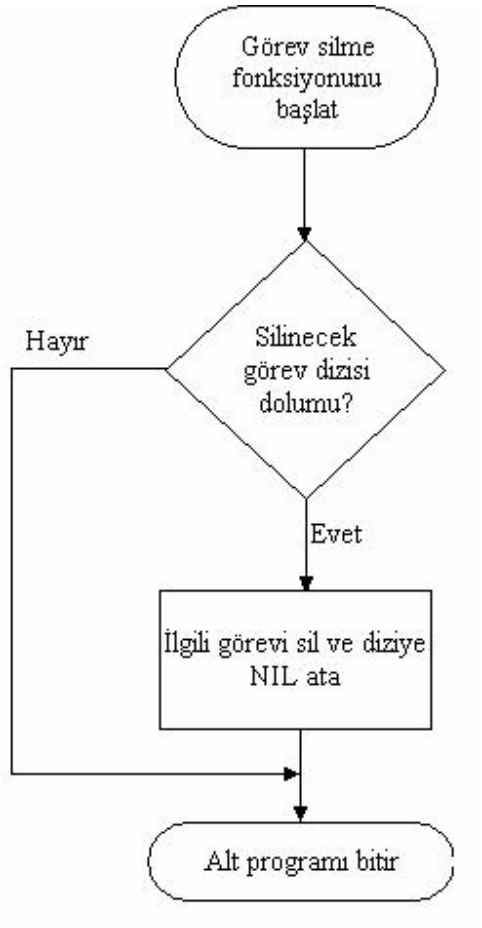
Ek 7 Güncelleme Fonksiyonu



Ek 8 Görev Ekleme Fonksiyonu

Ek 9 Görev Çalıştırma Fonksiyonu

Ek 10 Görevleri Silme Fonksiyonu



Ek 11 PID Kontrol Alt Programı