TC
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CAN (CONTROLLER AREA NETWORK) TEMELLİ ALARM SİSTEMİ TASARIMI

HÜSEYİN KAYABAŞI

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

MALATYA
Kasım 2008

Tezin Başlığı: CAN (Controller Area Network) Temelli Alarm Sistemi Tasarımı

Tezi Hazırlayan: Hüseyin KAYABAŞI

Sınav Tarihi: 13 Ekim 2008

Yukarıda adı geçen tez jürimizce değerlendirilerek Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Sınav Jürisi Üyeleri (ilk isim jüri başkanı, ikinci isim tez danışmanı)

Yrd. Doç. Dr. M. Emin TAĞLUK	İnönü Üniversitesi
Yrd. Doç. Dr. Ö. Faruk ÖZGÜVEN	İnönü Üniversitesi
Yrd. Doç. Dr. Müslüm ARKAN	İnönü Üniversitesi

İnönü Üniversitesi Fen Bilimleri Enstitüsü Onayı

Prof. Dr. İsmail ÖZDEMİR
Enstitü Müdürü

Onur Sözü

Yüksek Lisans Tezi olarak sunduğum "CAN (Controller Area Network) Temelli Alarm Sistemi Tasarımı" başlıklı bu çalışmanın bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın tarafımdan yazıldığını ve yararlandığım bütün kaynakların, hem metin içinde hem de kaynakçada yöntemine uygun biçimde gösterilenlerden oluştuğunu belirtir, bunu onurumla doğrularım.

Hüseyin KAYABAŞI

ÖZET

Yüksek Lisans Tezi

CAN (CONTROLLER AREA NETWORK) TEMELLİ
ALARM SİSTEMİ TASARIMI

Hüseyin Kayabaşı

İnönü Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Anabilim Dalı

94+IX sayfa

2008

Danışman: Yrd. Doç. Dr. Ö.Faruk Özgüven

Endüstrinin gelişmesi ile oluşan yeni sistemler için haberleşme protokolleri tanımlanmıştır. CAN protokolü önemli iletişim protokollerinden biridir. Endüstriyel uygulamalar için etkili ve yeterli bir protokoldür. Senkron iletişim için üç tele ihtiyaç vardır. Asenkron seri iletişimde iki telin kullanılır. CAN protokolü sistemlerinin iki telle iletişim yapması maliyeti düşüreceği ve sistemin karmaşıklığını da azaltacağından bu CAN protokolünün kullanımı için önemli bir avantajdır. CAN protokolü olay odaklı protokol olduğu için gerçek zamanlı uygulamalarda kullanılmaz. Bu sorunu çözmek için zaman tetiklemeli CAN (TT-CAN) geliştirilmiştir. TT-CAN protokolünden tezde bahsedilmiştir.

Bu tezde; ilk önce CAN protokolü ile yapılan çalışmalardan bahsedilmiştir. CAN protokolünün yapısı, mesaj iletimi, hata tespiti, iletişim hızı hesabı ve sistem için uygun entegre devre seçimi sonraki bölümde incelenmiştir. Tezin dördüncü kısmında PIC18F458 mikrodenetleyicisinin CAN modülü ve kayıtçılarının ayarlanması üzerinde durulmuştur. Tezin son kısmında ise iki uygulama gerçekleştirilmiştir. Alarm sistemi ve analog dijital dönüştürücü uygulamalarının CAN protokolü ile tasarlanıp gerçekleştirildi. Sonuçları ve uygulamanın önemi bu bölümde incelenmiştir. CAN protokolü ile haberleşen düğümlerin devre şeması ve kodları ile ekte verilmiştir.

ANAHTAR KELİMELER: CAN, LIN, TTCAN, TTP/C, FlexRay, Gerçek Zamanlı Uygulamalar, Alarm Sistemleri, PIC18F458, PCA82C250, MCP2551

DESIGN OF AN ALARM SYSTEM WITH CANBUS

Hüseyin KAYABAŞI

İnönü University

Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

94+IX pages

2008

Supervisor: Assoc. Prof. Dr. Ö. Faruk Özgüven

With the development of industrial technologies, particular communication protocols have been developed. Controller Area Network (CAN) is one of the important communication protocols. It is an effective and sufficient protocol used in variety of industrial applications. Synchronous systems need three wires for serial communication, whereas asynchronous communication systems can serially communicate through two wires. Because of having an asynchronous communication structure CAN protocol systems only need two wires, and this is an advantage from the cost and complexity points of view. Since the CAN protocol have event oriented structure can not be used for controlling real-time systems. To solve such problems time triggered CAN (TT-CAN) has been developed, and are described in the thesis.

In this thesis, first of all CAN applications were extensively discussed. The CAN protocol infrastructure, message transmission, error detection, baud rate calculation and appropriate IC selection for the system were also discussed. In the fourth part 18F458 micro controller CAN module and its registers were analyzed. In the final part of thesis, as an application, an alarm system and an analog to digital converter were designed and realized with CAN protocol. The results and practical considerations were discussed in this section. The designed circuit schemes for two node bidirectional communication with CAN protocol and soft codes were given in appendixes.

KEYWORDS: CAN, LIN, TTCAN, TTP/C, FlexRay, Real Time Applications, Alarm Systems, PIC18F458, 82C250, MCP2551

TEŞEKKÜR

Bu çalışmada, yardımlarını esirgemeyen değerli hocam Yrd. Doç. Dr. Ömer Faruk Özgüven'e ve de katkılarından dolayı Yüksek Lisans Öğrencisi Neşet Baysal'a teşekkürü bir borç bilirim.

Ayrıca, maddi ve manevi desteklerinden dolayı aileme de şükranlarımı sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1.GİRİŞ

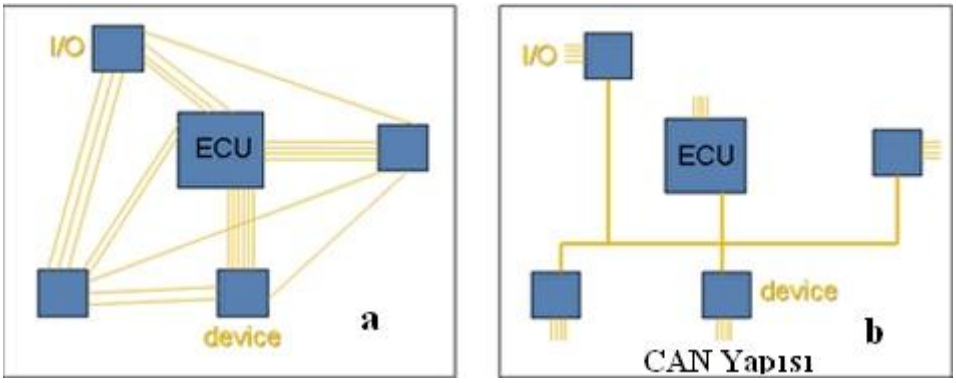
Bu tezde CAN (Controller Area Network) Denetleyici Alan Ağı temelli alarm sistemi tasarımı amaçlandı. Uygulamaya yönelik bir tez çalışması olduğundan CAN protokolünün yapısının bilinmesi gereklidir. İki CAN mikrodenetleyicisi içeren düğümün bilgisayarda ara yüz programı aracılığıyla denetimi hedeflendi. Alarm sisteminin altyapısında CAN modülü içeren PIC18F458 mikrodenetleyicisi, PCA82C250 ve MCP2551 CAN ara yüz entegreleri kullanıldı. Kullanılan CAN mikrodenetleyicisinin çıkışına 2x16 satır 4 bitlik LCD takıldı. Veri yolu üzerindeki mesajlar LCD ve bilgisayardaki ara yüz programında gözlendi. Tezde uygulamasını yaptığımız bir diğer devre ise mikrodenetleyicinin çıkışına bağlı ADC (Analog Digital Converter) Analog Dijital Dönüştürücü uygulamasıdır. Mikrodenetleyicinin çıkışındaki gerilim ayarlı direnç kullanılarak gerilim değerinin değiştirildi. Gerilim değeri CAN mesajı şeklinde LCD ve bilgisayara aktarımı yapıldı.

Alarm sistemi tasarımında kullandığımız yöntem ise mikrodenetleyicilerin gönderdiği mesajların belirleyici kısmına (Identifier ID) göre değerlendirilmesi oldu. Farklı belirleyici kısmı (ID) içeren mesajlar düğümleri ve bu mesajların içerdiği veriler ise düğümlerden gelecek alarm seviyesini belirlemede kullanıldı.

Kullandığımız bu CAN yapısından mesaj iletiminde aldığımız sonuçlarda sistemin doğruluğunun kesin, güvenilirliğin yüksek olduğu gözlendi. CAN protokolü ile haberleşmede mikrodenetleyicilerin iletişim hızlarının farklı olması durumunda haberleşemediği görüldü. Düğümler arası haberleşme hızlarının ayarlanması karşılaştığımız en önemli sorun oldu.

Denetleyici Alan Ağı, hata denetiminin önemli ve veri iletişimin mekanizmasının gelişmiş olduğu bir protokoldür. Dağıtımlı gerçek zaman denetimli uygulamalarda CAN protokolü yüksek seviyede güvenlik sağlar. Robert BOSCH firması tarafından 1983 yılında otomotiv sektöründe kullanılmak üzere tasarlanmıştır. Uluslararası Standartlar Organizasyonu (International Standarts Organization ISO), Açık Sistemler Ara Bağlantı (Open Systems Interconnection OSI) ağ katmanını iletişim protokolleri için referans alır. OSI referans modeli farklı cihaz veya protokollerden gelen bilginin kolayca haberleşebilmesini sağlamak için tanımlanmıştır. ISO, CAN protokolünü 1993 yılında ISO11898 olarak kabul etmiştir. CAN protokolünün kullanım alanları geniştir. CAN protokolü ile farklı sektörlerde gerçekleştirilmiş ticari uygulamalar vardır. Ulaşım, üretim, inşaat, tarım, sağlık, iletişim, finans, eğlence sektörlerinde ve bilimsel uygulamalarda CAN protokolü kullanılmıştır. CAN daha çok otomobil, asansör, medikal cihazlarda tercih edilmektedir. CAN protokolünün tercih nedenleri, hata denetim yöntem sayısının diğer haberleşme protokollerine göre fazla olması,

düşük gecikme zamanı, mikrodenetleyicilerinin ucuz olması, kullanılan kablo sayısının az olması ve maksimum 1Mbps olan iletişim hızıdır. CAN protokolünün kullanılması ile yapılan bir sistem tasarımı Şekil 1.1.'de gösterilmiştir. CAN protokolü sistemde kullanılan kablo sayısını azaltıp sistemin mimari yapısını anlaşılır hale getirmiştir. Şekil 1.1. Sistemin kapladığı alanın önemli olduğu sektörlerde CAN protokolü büyük kolaylık sağlamaktadır.. CAN protokolünde 40 metreye kadar 1 Mbps hızında iletişim sağlanır. Uzun mesafeler için iletişim hızı düşürülür. İletişim hızı 62,5 kbps olarak ayarlandığında mesajın iletileceği mesafe 1000 metre olacaktır. Mesafe arttıkça sinyalde zayıflama olacağından tekrarlayıcı kullanılarak sinyalin zayıflaması azaltılır.



Şekil 1.1. CAN Protokolü Kullanılarak Yapılmış Sistem [1]

CAN protokolü ile yapılmış akademik çalışmalarda ise, yurtdışında Kyung Chang Lee ve Hong-Hee Lee yangın belirleme sistemini [2] CAN protokolü ile gerçekleştirmiştir. Bu çalışma da T89C51cc01 CAN mikrodenetleyicisi, PCA82C250 CAN ara yüz entegresi ile duman ve gaz sensörleri kullanılmıştır. Yangın tespiti için yapılan uygulama sonuçları üzerinde durulmuştur. CAN protokolünün öncelik seviyeleri yangın oluşma evrelerine göre belirlenmiştir.

Ülkemizde CAN haberleşme protokolü üzerine yapılan uygulamalar da C.Bayılmış; kablosuz yerel alan ağlarını kullanarak CAN bölümlerini genişletilmek için arabirim tasarlamıştır. [3] Celal Çeken, kablosuz ATM protokolünü kullanarak CAN mesajlarının iletimi için yeni bir yöntem üzerine çalışmışlardır [4]. Y. Santur, PIC16F877 mikrodenetleyicisini kullanarak CAN protokolü için eğitim amaçlı deney seti tasarımı yapmıştır [5]. Bu çalışmada PIC16F877 mikrodenetleyicisine bağlı MCP2510 CAN denetleyici ve MCP2551 CAN ara yüz entegreleri kullanılmıştır. SPI (Serial Peripheral Interface) eşzamanlı seri iletişim protokolü kullanılarak PIC16F877 ile MCP2510 CAN denetleyicisi haberleştirilmiştir. A. Özdemir, CAN protokolü üzerinde yeni bir uygulama

protokolü CANup tanımlayarak asansörlerde uygulamasını yapmıştır [6]. S.Tuncel, CAN protokolünün Visual Basic programlama dilinde eğitim amaçlı benzetim programı geliştirmiştir [7]. A. Karaca, AT89C51CC01 mikrodenetleyicisini kullanarak bina güvenlik uygulaması yapmıştır [8].

2. CAN PROTOKOLÜ ve DİĞER PROTOKOLLER

Endüstriyel otomasyonda kullanılmak üzere çok sayıda seri haberleşme protokolleri vardır. Her biri hızına, fiyatına ve performansına özgü farklı karakteristik gösterir. En fazla kullanılanlar; HS, LS, FT, tek telli CAN, TTCAN, LIN, I2C, D2B, IEEE 1394, CPL, X-by-Wire, FlexRay, Safe-by-Wire protokolleridir.[9]

2.1. Yüksek Hızlı CAN

Yüksek Hızlı CAN protokolü; daha çok otomobillerin güç bölümündeki bağlantılar, motor kontrol, vites kutusu ve fren gibi önem arz eden yerlerde kullanılır. Bu birimler aracın en önemli ve kritik donatıları olduğundan 250 kbps veya 500 kbps ya da 1 Mbps hızında kontrol edilir. Donanım ve yazılım katmanlarının geliştirilmesi yüksek AR-GE bütçe gereksinimine karşın yeni üretilen araç sistemlerinde yüksek hızlı CAN protokolü kullanılır.

Otomobiller için geliştirilecek uygulamaların OSEK (Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen) standardına uyması gerekir. OSEK'in açılımı Açık Sistemler ve Motorlu Araçlar için Elektronik Ara Yüzleridir. OSEK 1993 yılında BMW, Robert Bosch GmbH, Daimler Chrysler, Opel, Siemens ve Volkswagen firmalarının bir araya gelerek oluşturdukları konsorsiyumdur. Daha sonra 1994'te Renault ve PSA Peugeot Citroen firmaları da katılmıştır [10].

Günümüzde OSEK'in OSEK-VDX adında Avrupa otomobil endüstrisinde kullanılan işletim sistemi vardır [11].

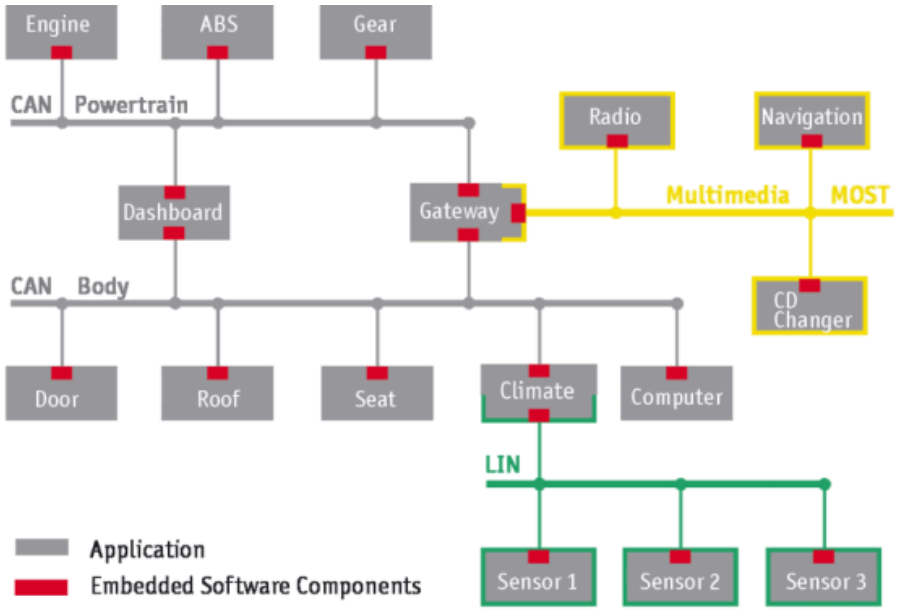
2.2. Düşük Hızlı CAN

Düşük hızlı CAN katmanı ara katmanlardan biridir. Gerçekte, düşük hızlı CAN protokolü 62,5 kbps fakat sıklıkla 125 kbps iletişim hızı kullanılır. Bu hızlar arabaların kapılarını ve tavanını açmak için yeterlidir. Bu ağ yapısı hata toleranslı yapıdadır. Hatalı mesajların pek fazla önemsenmediği durumlarda kullanılmaktadır.

2.3. LIN (Local Interconnect Network)

LIN protokolünün açılımı Yerel Ara Bağlantı Ağıdır. LIN protokolü CAN'in alt protokolü olarak adlandırılır. CAN protokolüne ihtiyaç duyulmadığı durumlarda kullanılır. En fazla 20 kbps kullanılır. Arabaların camlarının denetiminde, korna, direksiyon göstergesi, tepe lambaları, uzaktan kumanda ve lastik basınç göstergesinin izleniminde LIN kullanılması pratiklik sağlar. CAN protokolü ile uyumlu yapıdadır [8]. Şekil 2.1'de LIN protokolünün

CAN protokolü ile birlikte kullanımı gösterilmiştir. Arabanın kliması, LIN protokolünün CAN protokolüne bağlantısını sağlayan ağ geçididir.



Şekil 2.1. CAN ve LIN Protokolü [11]

2.4. CPL (Current Power Line)

Güç hattı üzerinden akım taşıma, 20 ila 50 kbps arasında yüksek frekanslı taşıyıcı içeren ASK veya FSK modülasyonunu kullanan protokoldür. Otomobillerde klima kontrolü, immobilizer, engel sensörü ve kapılarda kullanılır [12].

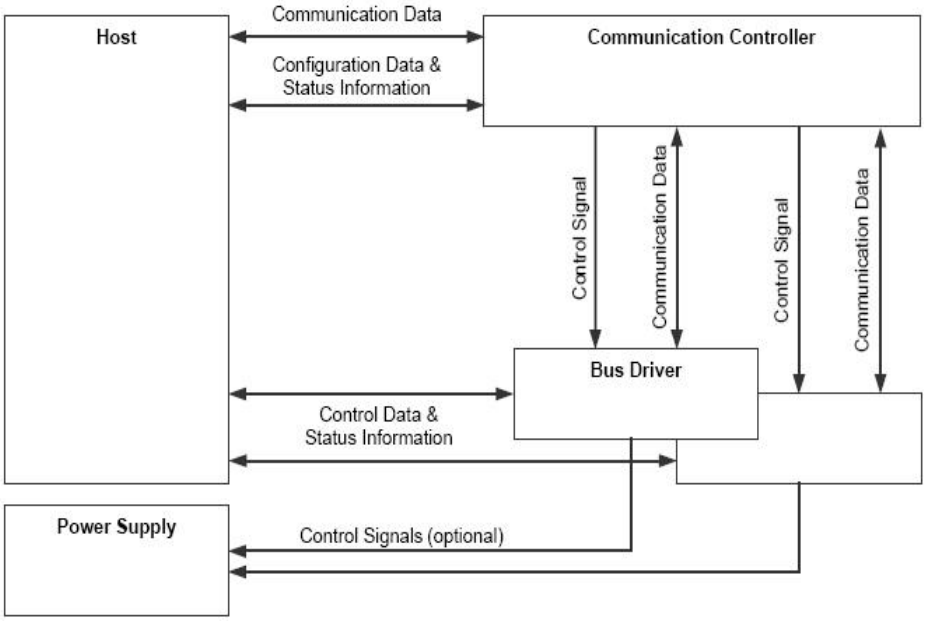
2.5. X-by-Wire

X-by-Wire, otomobillerde mekanik güvenliğin olmadığı hata toleranslı elektronik sistemler için kullanılan protokoldür. X-by-Wire'daki "x" yönetim, fren, çoklu airbag, süspansiyon kontrolü ve güç kullanımın artışı anlamına gelmektedir [13].

X-by-Wire protokolü 7-20 Mbps hızındadır. 6 ila 15 düğüm için tasarlanmıştır.

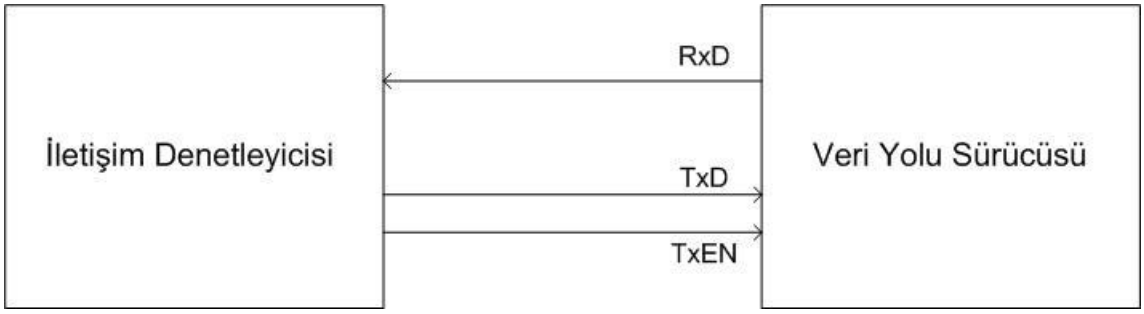
2.6. FlexRay

FlexRay protokolü senkron ve asenkron çerçeve transferi, senkron iletimde garanti edilmiş çerçeve gecikmesi ve baskısı, çoklu ana saat senkronizasyonu, hata belirleme ve sinyalleme, hata içerme ve hata düzeltme özelliklerine sahiptir [14]. FlexRay yapısı Şekil-2.2.'deki gösterilmiştir. Bir iletişim denetleyicisi, bir işlemci, iki veri yolu sürücüsü ve güç kaynağı gereklidir.



Şekil 2.2. FlexRay Yapısı [14]

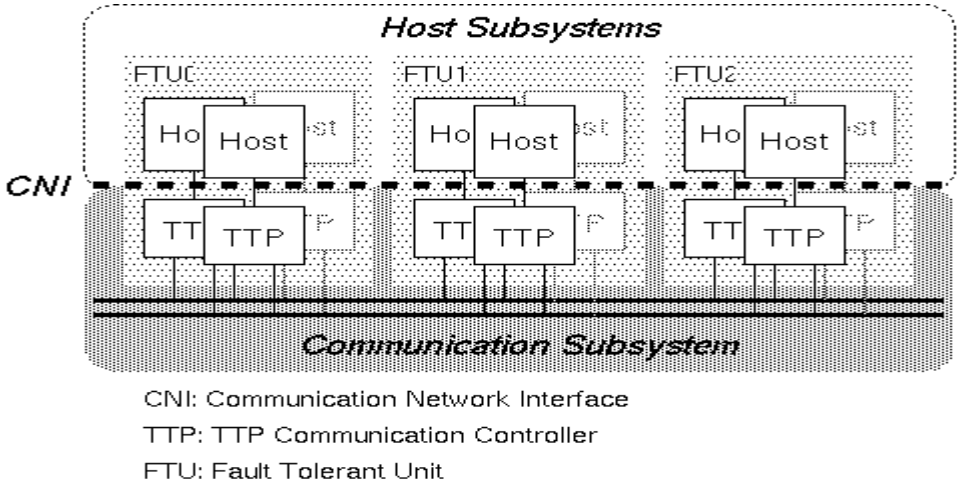
Flexray mimarisinde yer alan iletişim denetleyicisi ile veri yolu sürücüsü arasında üç sinyal vardır. Şekil 2.3'de bu sinyaller gösterilmiştir. Bunlardan TxD ve TxEN iletişim denetleyicisinden RxD ise veri yolu sürücüsünden çıkış sinyalidir.



Şekil 2.3. FlexRay İletişim Denetleyici (FlexRay Communications System Protocol Specification [14]'den değiştirilerek alınmıştır.)

2.7. TTP/C

TTP/C protokolü güvenli ve kritik gerçek zamanlı kontrol sistemleri için zaman tetiklemeli protokoldür [15]. TTP/C'nin yapısı Şekil 2.4.'de gösterilmiştir. TTP/C protokolünde tetikleme ile hata toleransına izin veren birim farklılaştırılmıştır. HOST olarak adlandırılan kısım her düğümdeki işlemcidir. Gerçek zamanlı uygulamalar burada işlenir. İletişim Ağ Ara Yüzü (CNI); iki alt sistem arasında kalan mesajların alınıp verilmesini sağlayan hafıza bölümüdür.



Şekil 2.4. TTP/C Yapısı [15]

2.8. TTCAN (Time Triggered CAN)

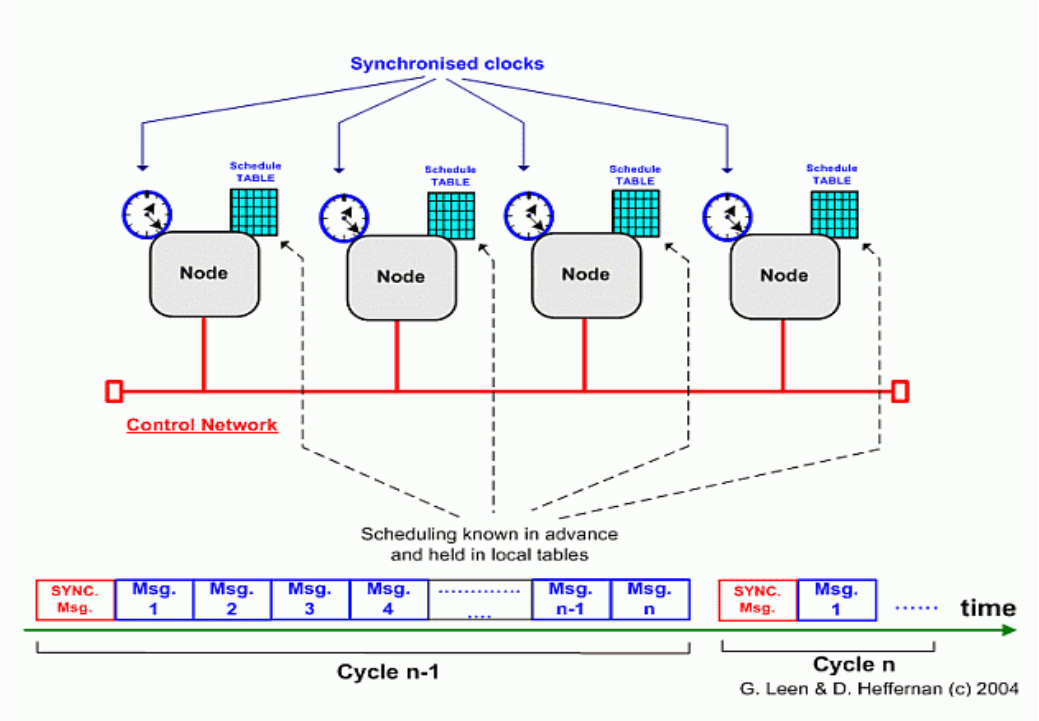
Zaman Tetiklemeli CAN protokolü anlamına gelmektedir. CAN protokolü olay tetiklemeli protokoldür. Bu yüzden gerçek zamanlı sistemlerle uyumlu değildir. Bu eksikliğin giderilmesi için bir üst katman tanımlanmıştır. TTCAN, OSI'nin oturum katmanında tanımlanmıştır. Çizelge 2.1.'de OSI katmanlarında tanımlı protokoller gösterilmiştir [16].

Çizelge 2.1. CAN Protokolünün OSI katmanları İçerisinde Gösterimi (Fuhrman [16]'dan değiştirilerek alınmıştır.)

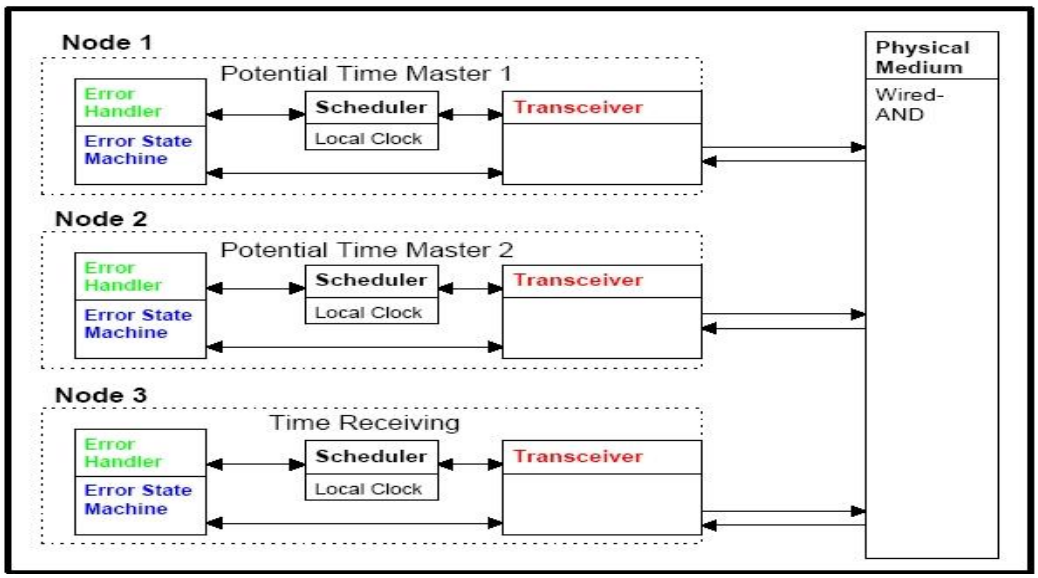
7.KATMAN Uygulama	CAL:CAN uygulama katmanı ve CANopen	Can Kingdom	DeviceNet	Smart Distributed Systems (SDS)
6.KATMAN	Tanımlanmamıştır.			
5.KATMAN	"Zaman Tetiklemeli CAN" ISO 11898-4			
4.KATMAN	Tanımlanmamıştır.			
3.KATMAN	Tanımlanmamıştır.			
2.KATMAN	ISO 11898-1			
1.KATMAN	ISO 11519-2 ISO 11519-3			

TTCAN geçici olaylarla tetiklenir. CAN protokolünde böyle bir durum söz konusu değildir. TTCAN protokolü ile 5-10 Mbps hıza ulaşılmıştır. Zaman tetiklemeli CAN protokolünün yapısı TDMA (Zaman Bölümlemeli Çoklu Geçiş) yapısına benzerdir [16]. Her düğümün yerel saat sinyali vardır. Mesajların iletiminde bu sinyaller kullanılır. Şekil 2.5.'de

TTCAN protokolünün yapısında mesajların sıralanışı gösterilmiştir. TTCAN protokolünün ağ modeli ise Şekil 2.6.'da gösterilmiştir. İki ana düğüm ile zaman bilgisini alan düğümden oluşan bu yapıda her düğümün zamanlayıcısı bulunmaktadır. Mikrodenetleyicinin bilgisi zamanlayıcı kısmında kodlanmıştır. Gönderilen mesajın ulaşacağı adres bu kısım içerisinde saklıdır. Zamanlayıcı kısmında yerel saat bilgisi de taşınmaktadır. Hangi mesajın hangi anda aktif olduğu zamanlayıcı kısmından anlaşılır.



Şekil 2.5. TTCAN Yapısı [16]



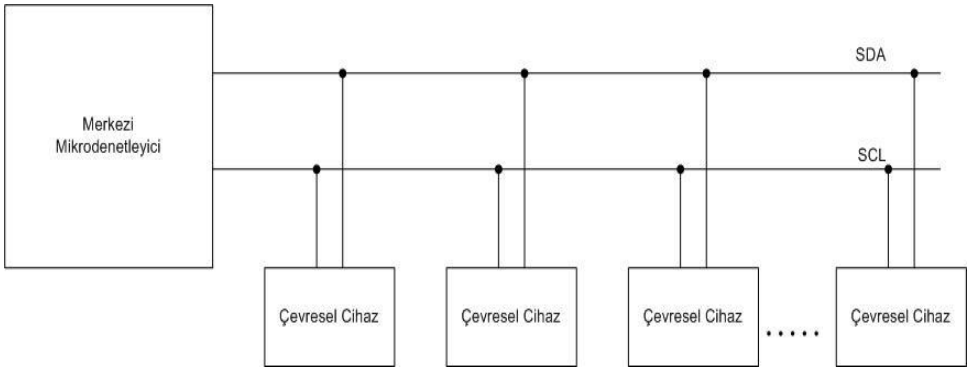
Şekil 2.6. TTCAN Protokolü Ağ Modeli [16]

2.9. Safe-by-Wire

Güvenlik sistemlerinde noktadan noktaya bağlantılar için kullanılır. Airbag ve emniyet kemerinin durumları için Safe-by-Wire Plus ASRB-2 ağ topolojisi tanımlanmıştır. Bu bağlantılar doğrudan araç içerisinde ilgili kısım ile bağlantılı olduğundan kullanılmaktadır [13].

2.10. I^2C (Inter-Integrated Circuit)

I^2C protokolü genelde bilgisayarların ana kartları, cep telefonları, gömülü sistemlerde kullanılır. CAN protokolünün kullanıldığı sistemlerde örneğin araçlarda CAN protokolünü kullanmayan cihazların denetim işlemlerinde kullanılır. CD çaları durdurma, çalıştırma gibi işlemler I^2C protokolü ile yapılır. Şekil 2.7.'de I^2C protokolünün yapısı gösterilmiştir [17].

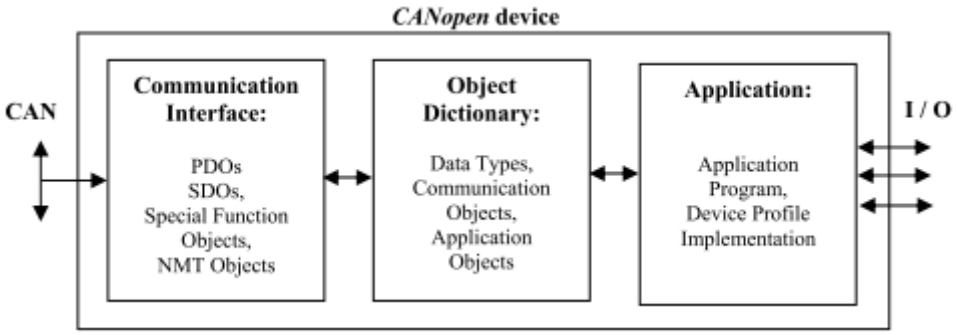


Şekil 2.7. I^2C Donanım Yapısı (http://www.embedded.com/story/OEG20010718S0073 [17]'den değiştirilerek alınmıştır.)

2.12 CANopen

OSI biçiminde uygulama katmanında tanımlanmış alan denetimli bir katmandır. Açık network standardı olarak dünyada kabul görmüştür. Endüstriyel otomasyon alanında kullanılmaktadır. Hareket tabanlı uygulamalar için tasarlanmıştır [18]. Ticari olmayan uygulamaların gerçekleştirilmesinde yaygın kullanımı vardır. Şekil 2.8'de [18] CANopen sisteminin mimarisi gösterilmiştir. CANopen mimari yapısında bulunan Servis Veri Nesne (SDO) protokolü, kullanılan sensör, hareket denetleyicisi gibi cihazlardan gelen değerleri okuma ve kurma için kullanılır. İşlem Veri Nesne (PDO) protokolü, gerçek zamanlı veri değerlerinin işlenmesi için kullanılır. İki çeşit PDO vardır. Bunlar, alıcı ve iletim işlemleri veri nesneleridir. CANopen, CAN mesajının 11 bitlik belirleyici kısmının ilk 7 bitini düğüm

kimliği sonraki 4 bitini ise fonksiyon kodu olarak kullanmaktadır. Şekil 2.9.'da CANopen mesajının belirleyici kısmının biçimi gösterilmiştir.



Şekil 2.8. CANopen Mimari Yapısı [18]

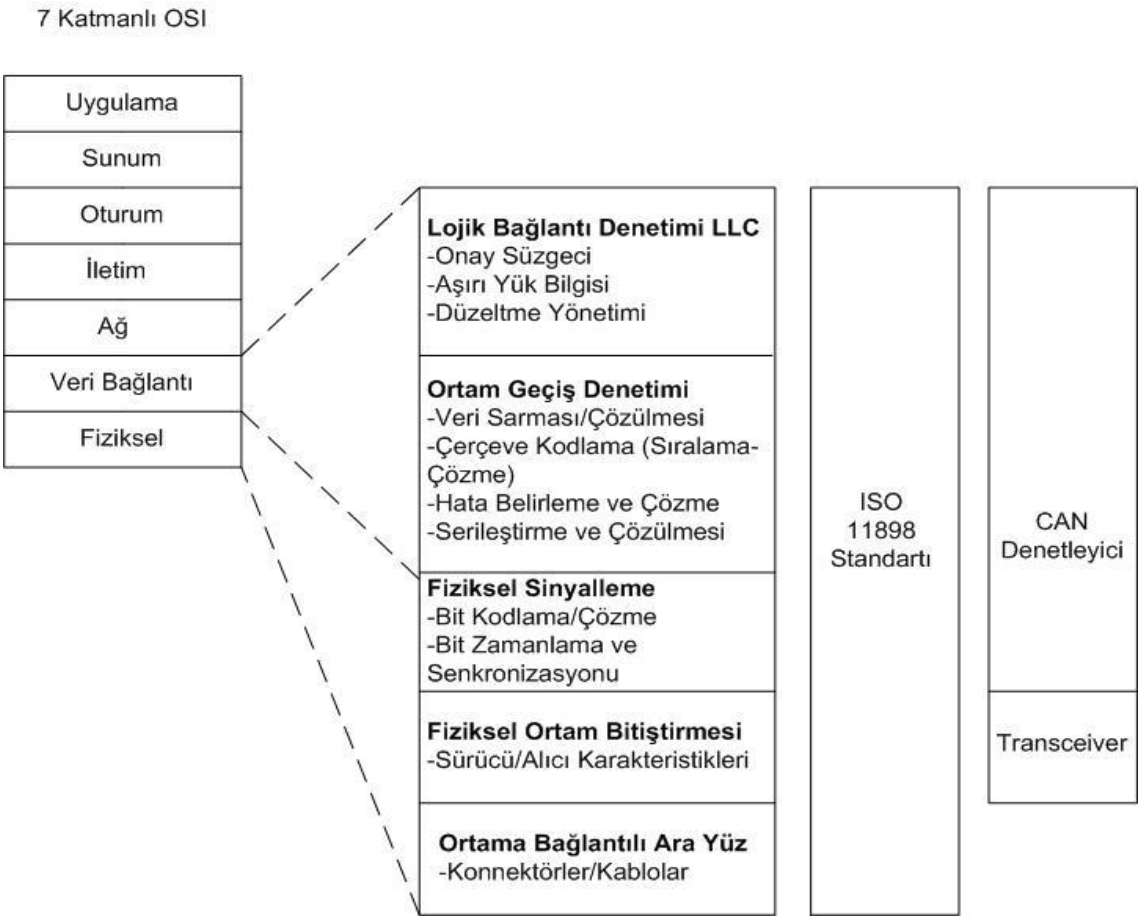
Bit Numarası										
10	9	8	7	6	5	4	3	2	1	0
Fonksiyon Kodu				Düğüm Kimliği						

Şekil 2.9. CANopen Mesajının Belirleyici Kısmı (Boterenbrood [18]'den değiştirilerek alınmıştır.

CANopen protokolünde fonksiyon kodu kullanıcının belirleyeceği değerlerdir. Düğüm kimliği ise verinin hangi düğümden geldiğini belirlemeye yarar. 128 düğüm belirleyici kısmında tanımlanabilir. CANopen ayrıca, kullanıcıya düğümlerden gelebilecek acil mesajların sınıflanmasını da sağlar.

3. CAN PROTOKOLÜ YAPISI

Tüm ağ protokolleri Açık Sistem Ara Bağlantı (OSI) modeli içinde gösterilir. CAN protokolü, OSI'de veri bağlantı katmanı ve fiziksel katman arasında tanımlanır. Veri bağlantı katmanı CAN yapısını oluşturmaktadır. Lojik bağlantı denetimi; aşırı yük denetimi ve belirlenmesi, mesaj filtrelenmesi ve hata yönetim fonksiyonlarını içerir [19]. Fiziksel Ortam Birleştirme (PMA) ve Ortama Bağımlı Ara Yüz (MDI), CAN protokolü içinde bulunmayan fiziksel katmanın iki parçasıdır. Fiziksel sinyalleşme bölümü CAN yapısı ile belirlenir [20].



Şekil 3.1. CAN Protokolü ve OSI Katmanları (Richards [20]'den değiştirilerek alınmıştır.)

CAN protokolü dört ana katman ile belirtilir. Bu katmanlar Çizelge 3.1.'de verilmiştir. Nesne ve iletim katmanı, ISO/OSI modelinde belirtilen veri bağlantı katmanının tüm servis

fonksiyonlarını kapsamalıdır. Nesne katmanı iletilen mesajların bulunduğu, iletim katmanında kullanılan alınan mesajların onay kararının verildiği ve ilgili donanıma bir ara yüz sağlayan katmandır. Nesne katmanı; kısaca mesajın filtrelenmesi, işlenmesi ve durumunun belirlenmesinden sorumludur. Veri bağlantı katmanı, CAN protokolünün yapısını oluşturmaktadır.

Çizelge 3.1. CAN Protokolü Katmanları (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır)

Uygulama Katmanı	
Nesne Katmanı	—Mesaj ve durum işleme
	—Mesaj filtreleme
Transfer Katmanı	—Hata durdurma
	—Hata belirleme ve sinyalleşme
	—Mesaj doğrulama
	—Onaylama
	—Kabul edilebilme (Arbitration)
	—Mesaj çerçeveleme
	—İletim oranı ve zamanlama
Fiziksel Katman	—Sinyal seviyesi ve bit yeniden oluşturma
	—İletim ortamı

Lojik bağlantı denetimi aşırı yükün denetimi, belirtilmesi ile mesaj filtrelemesi ve hata yönetim fonksiyonlarını içerir. Nesne işlemenin belirlenmesinde kolaylık sağlanmıştır. İletim katmanının göstergesi iletim protokolüne bağlıdır. Çerçeve denetlenmesi, kabul edilebilme, hata denetimi, hatanın sinyalleşmesi ve hatanın durdurulması iletim katmanının işlemleridir.

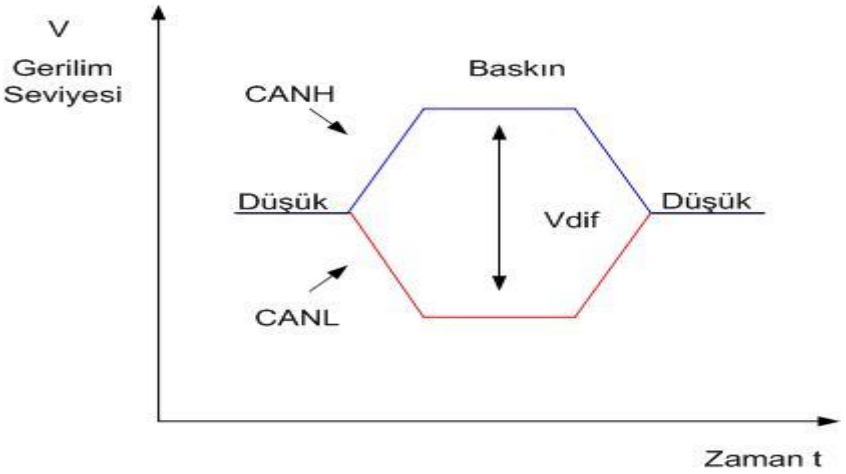
İletim katmanı, CAN protokolünün çekirdek kısmını oluşturmaktadır. Veri yolu boş ise yeni bir iletimin veya başlamış olan veri alımının devam etmesini sağlar. İletim katmanı, Çizelge 3.1.'de verilen görevlerinin yanı sıra bit zamanlamasının bazı genel özellikleri barındırır. İletim katmanı yapı itibariyle değişikliklere açık değildir. Fiziksel katmanın görevi ise farklı düğümler arasındaki bitlerin iletimini sağlamaktır. Fiziksel Ortam Birleştirme (PMA) ve Ortama Bağımlı Ara Yüz (MDI) CAN protokolü tarafından belirlenmeyen fiziksel katmanın iki parçasıdır. Sinyallerin gerçekten iletilip iletilmediğini bu katman belirler. Bu düğümlerin tümünün fiziksel katmanlarının ve elektriksel özelliklerinin aynı olması gerekmektedir. Fiziksel katman, kablo ve konektörlerden oluştuğu için kullanıcı tercih konusunda serbesttir. Fiziksel sinyalleme bölümü CAN yapısı ile belirlenir. CAN protokolünün ISO 11898-2 ile tanımlanan versiyonunda fiziksel katman PMA ve MDI alt katmanlarını belirlemek için yayınlanmıştır.

3.1.Veri Yolu Gerilim Seviyesi

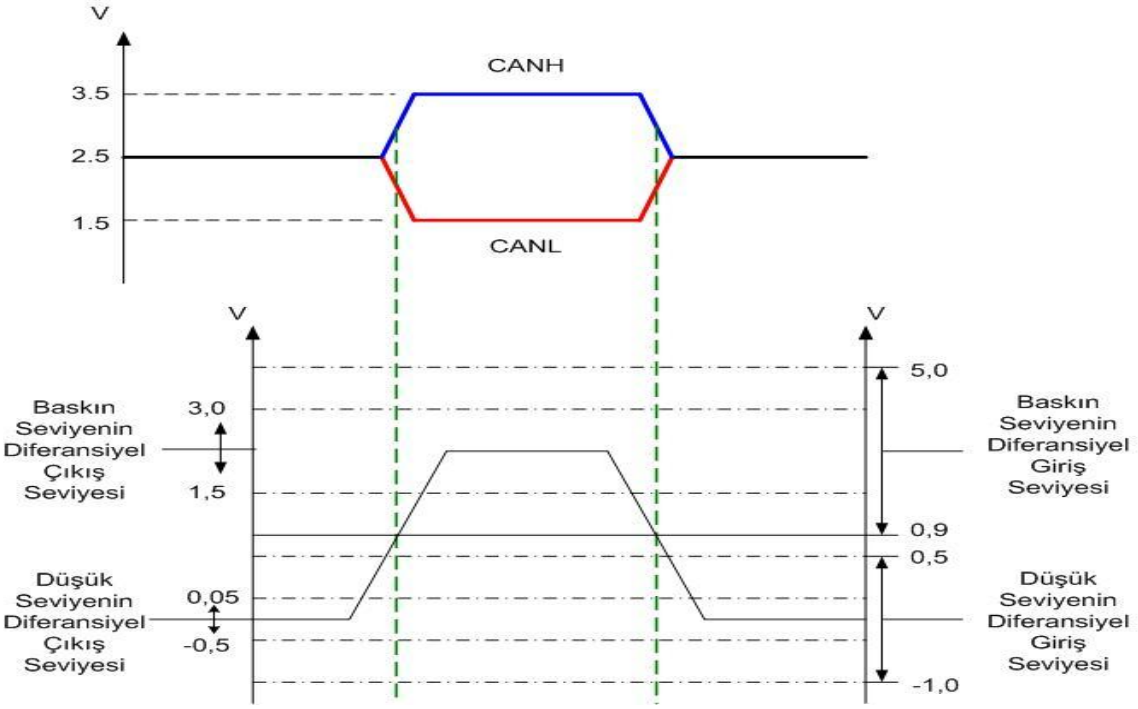
CAN protokolünde düşük ve baskın olarak tanımlanan seviye gerilimleri vardır. Bu gerilim seviyeleri lojik olarak ifade edilir. Düşük seviye geriliminin (Vr) aralığı

$$0{,}5<V_r<1{,}5\ \text{V} \qquad (3.1.)$$

arasındadır. Baskın seviye lojik olarak sıfırdır. CANH ve CANL üzerindeki diferansiyel gerilim (V_{diff}) eşik geriliminden büyüktür. Baskın bit düşük bitin üzerine binerek bit hatasını önler. CAN protokolünde kullanılan baskın ve düşük bitlerin gerilim seviyeleri Şekil 3.2.'de gösterilmiştir.



Şekil 3.2. CAN Protokolü Gerilim Seviyesi (Richards [20]'dan değiştirilerek alınmıştır.)



Şekil 3.3. Baskın ve Düşük Seviyelerin Gerilim Seviyeleri (Richards [20]'dan değiştirilerek alınmıştır.)

3.2. Konektörler ve Teller

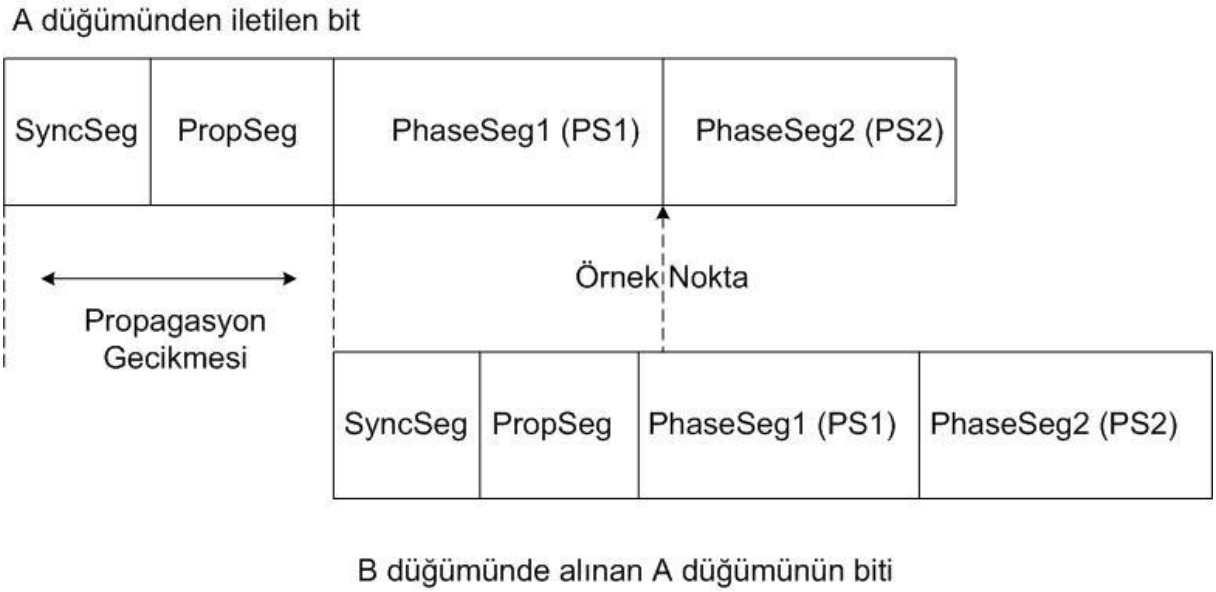
ISO11898-2 standardında kablo ve konektörlerin özellikleri ayrıntılı olarak belirtilmiştir. Buna rağmen protokolde 120Ω nominal direnç veri yolunun sonunda kullanılır. Dirençlerin kullanımı veri yolunda yansıma oluşmasını engellemek içindir.

3.3. Veri Yolu Uzunluğu

ISO11898 standardına göre 40 metreye kadar 1 Mbps verilebilir. Veri yolu uzun olması düşük veri hızında iletiminin gerçekleşmesine neden olacaktır. Veri yolunun en büyük sınırlayıcısı alıcı-vericinin propagasyon gecikmesidir. Cena ve Valenzano [21], CAN protokolünün yüksek performansa sahip CAN benzeri network olarak tanımladığı FastCAN protokolünde 3000 metreye kadar 1 Mbps verilebileceğini açıklamışlardır. İletim gecikmesi arttırılıp bit oranı düşürülerek CAN protokolü uzak mesafelere iletilebilir [21].

3.4. Propagasyon Gecikmesi

Her düğümde tahkimin aynı bit zamanında ve örnek noktada olması gerekir. Örneğin, veri yolu üzerinde zıt uçlarda aynı anda mesaj iletimi başlarsa veri yolunun denetimi için tahkim gereklidir. Bu tahkim aynı bit zamanında gerçekleşiyorsa etkilidir. Şekil 3.4.'de iki düğüm arasında tek yönlü propagasyon gecikmesi gösterilmiştir.



Şekil 3.4. Propagasyon Gecikmesi (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

Aşırı propagasyon gecikmeleri yanlış tahkime neden olacaktır. Bu da veri yolunun uzunluğuna bağlıdır. Bir CAN sisteminin propagasyon gecikmesi sinyalin veri yolu üzerindeki düşme anı (t_{bus}), çıkış süresi gecikmesi (t_{drv}) ve giriş karşılaştırıcı gecikmesine (t_{cmp}) bağlıdır.

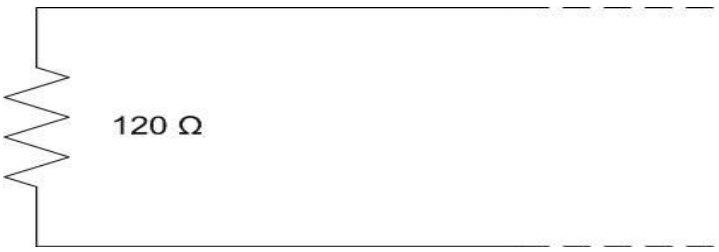
$$t_{prop} = 2 \times t_{bus} + t_{cmp} + t_{drv} \quad (3.2.)$$

3.5. Veri Yolu Sonlandırması

Veri yolunda sinyal yansımasını azaltmak için sonlandırma yapılır. ISO11898 standardına göre hat empedansı 120Ω 'dur. Bundan dolayıdır ki veri yolunun iki tarafında da 120Ω kullanılır. CAN protokolü elektromanyetik sinyallerden etkilenir. Bu etkileşimi azaltmak ve elektromanyetik uyumluluk (EMC) performansını artırmak için çeşitli sonlandırma yöntemleri kullanılır. En fazla kullanılan sonlandırma yöntemi standart sonlandırma yöntemidir.

3.5.1. Standart Sonlandırma

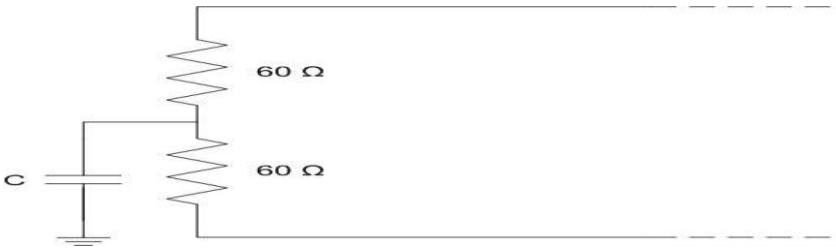
Bu sonlandırma tipinde veri yolunun sonunda tek 120Ω'luk direnç kullanılır. Çoğu CAN sisteminde bu yöntem tercih edilir. Şekil 3.5.'de standart sonlandırma gösterilmiştir.



Şekil 3.5. Standart Sonlandırma (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

3.5.2. Ayrılmış Sonlandırma

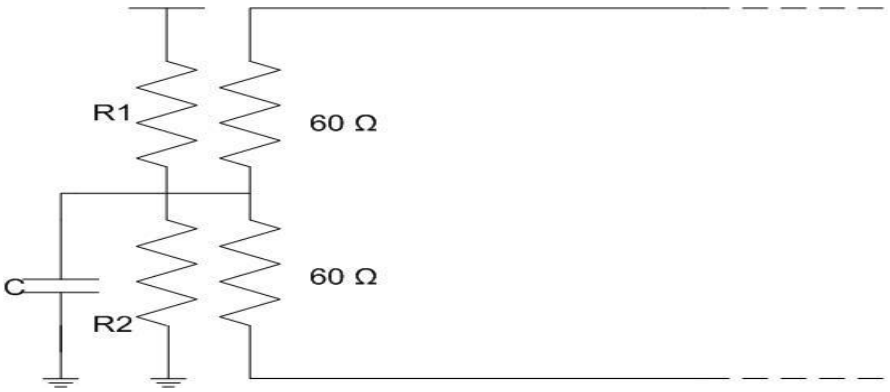
Emisyon azalması kolayca yapıldığından yaygınlaşmaya başlamıştır. Her bir uçta 120Ω'luk dirençler iki 60Ω'luk dirençle sağlanmıştır. Bu iki 60Ω'luk dirençler mümkün oldukça aynı değerde olmalıdır. Şekil 3.6.'da standart sonlandırma gösterilmiştir.



Şekil 3.6. Ayrılmış Sonlandırma (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

3.5.3. Beslemeli Ayrılmış Sonlandırma

Bu sonlandırma yöntemi düşük seviye gerilimini sabit bir değerde tutmak için kullanılır. EMC performansı artar. Bu devre ayrılmış sonlandırmadaki 60Ω'luk dirençler arasına gerilim bölücü devresinin eklenmesinden oluşur [4]. Şekil 3.7.'da beslemeli sonlandırma gösterilmiştir.



Şekil 3.7. Beslemeli Ayrılmış Sonlandırma (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

3.6. CAN Protokolü Basit Kavramları

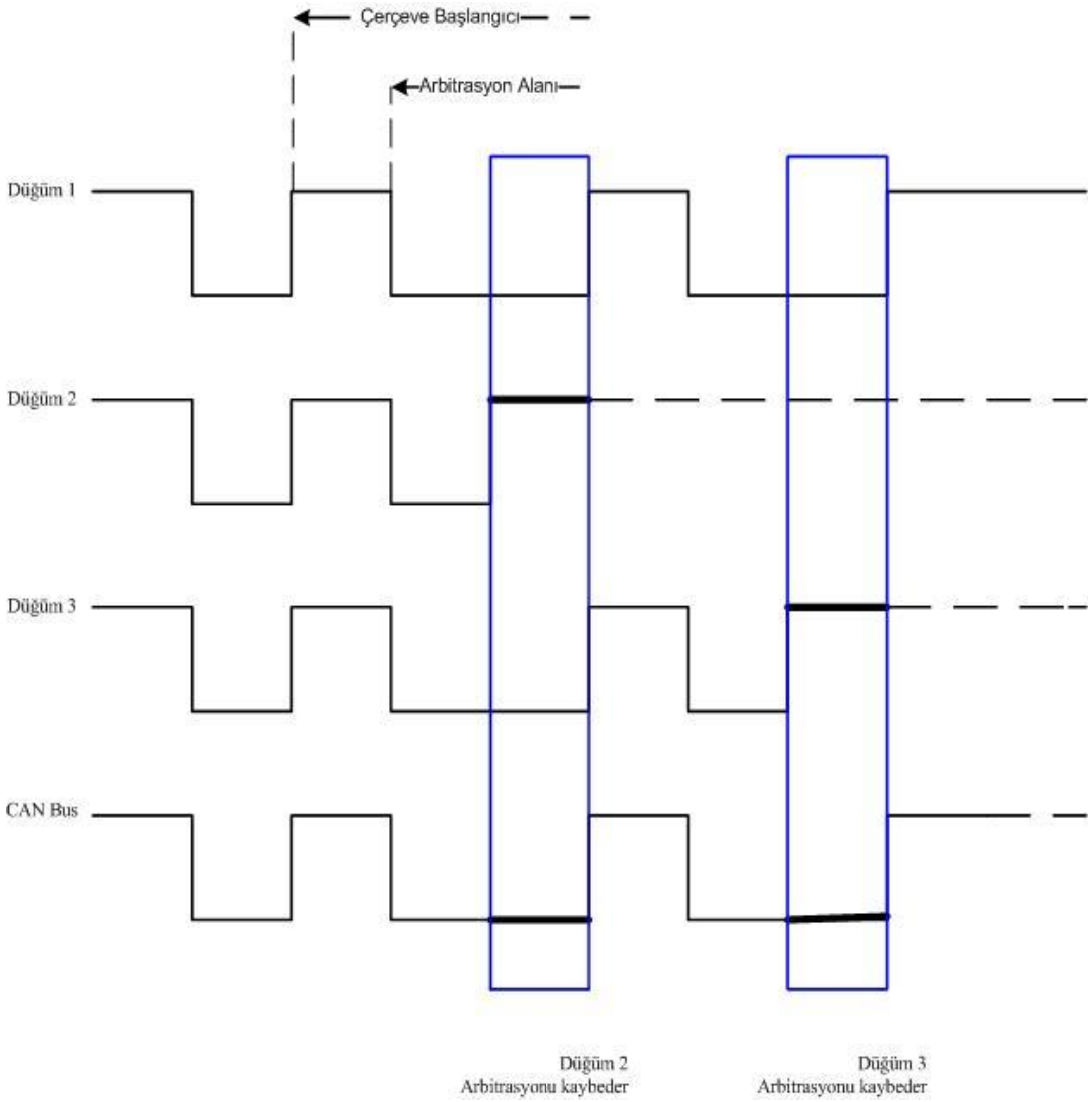
CAN protokolü, şu özelliklere sahiptir.

- Mesajların önceliği
- Gecikme zamanının ayarlanabilmesi
- Yapının esnekliği
- Zaman ile senkronize çoklu yayın
- Gönderilen veri ile tutarlı sistem
- Çoklu ana kullanıcı
- Hata denetim ve sinyalleşme
- Veri yolunun boş olduğu anda eksik mesajların otomatik olarak yeniden iletilmesi
- Düğümlerde oluşan geçici ile kalıcı hataların birbirinden ayrılması ve eksik düğümlerin otomatik olarak kapatılması

Fiziksel katman sinyallerin nasıl iletileceğini belirler. Fiziksel katman, iletim için geçerli olan ortamın özelliğine ve uygulamalar için en iyi sinyal seviyelerine sahip olmayabilir.

İletim katmanı CAN protokolünün çekirdeğini oluşturur. Nesne katmanına iletilmek üzere mesajı alır ve nesne katmanından iletilmek üzere gelen mesajları kabul eder. İletim katmanı bit zamanlaması ve senkronizasyonu, mesaj çerçeveleme, onay, hata denetimi ve sinyalleşmesi ile hatanın giderilmesi işlemlerinin gerçekleştiği katmandır. Nesne katmanı mesaj filtrelemenin yanı sıra mesajın durumunun ve yönetiminin olduğu katmandır. CAN protokolünde bilgi sabit biçimli mesaj olarak gönderilir. Fakat mesaj uzunluğu sınırlıdır. İletim için veri yolu boş olduğunda herhangi bir düğüm mesaj iletir. CAN sistemlerinde, bir CAN düğümü sistem ayarlarını içeren mesaj iletmez. Bunun nedeni sistemin yapısı hakkında gizliliğin korunmasıdır. Düğümler, CAN ağına sistemin donanımsal veya yazılımsal ayarlarında değişiklik yapılmadan eklenebilir. Bir mesajın içeriği Belirleyici olarak adlandırılır. Belirleyici mesajın gideceği adres bilgisini içermez. Fakat verinin ne anlama geldiğini içerir. Böylece ağ üzerindeki tüm düğümler mesajın kendileri ile ilgili olup olmadığını belirlemek için filtreleme yapar. CAN protokolünde çoklu iletim mesaj filtrelemenin önemli kavramlarından birisidir. Ağ üzerinde aynı işlevi gören tüm aygıtlar aynı mesajı alır. CAN yapısında, mesajın ağ üzerindeki tüm düğümler tarafından veya sadece bir düğümün almasını garanti etmek için tasarlanmıştır. Böylece sistemin veri kararlılığı çoklu iletim ve hata denetimi ile sağlanmış olur. CAN protokolü ile tasarlanan ağda iletişim hızı ağ içerisindeki düğümlerde farklı olabilir. Farklı hızlarda iletişimin ağ geçidi tanımlanarak farklı

hızları kullanan düğümlere ağ geçitleri ile ulaşılır. Hız farklı olsa dahi bit oranı sabittir. Mesajın belirleyici kısmı (ID), veri yolu üzerindeki mesaj iletimi boyunca değişmez mesaj önceliğine sahiptir. CAN mesajlarında belirleyici kısmı, düğümler arasında veri iletiminin de öncelikli olanları belirtmeyi sağlar. Uzak çerçeve ana denetleyicinin düğümlerdeki cihazların denetimi için kullanılır. Uzak çerçevenin gönderilmesi ile mesajın gönderildiği düğüm diğer düğüme uzak çerçeve iletim istek (Remote Transfer Request) biti iletir. Veri çerçevesi ile uzak çerçeve aynı belirleyici kısmı (ID) tarafından oluşturulur. CAN protokolünde veri yolu boş olduğunda herhangi bir düğümden mesaj iletilir. CAN mesajında yüksek öncelik biti içeren mesaj veri yolu üzerinde ilk önce gönderilir. Veri yolu boş olduğunda tüm düğümlerden mesaj iletimi gerçekleşeceğinden aynı anda mesaj iletimi veri kaybı olabileceğinden sorun oluşturabilir. Bu durumda belirleyici kısmı ID kullanılarak bit bazında mesaj incelenir ve hangi mesajın iletileceğine karar verilir ve bu olaya tahkim denir. Tahkim mekanizması hem bilginin hem de zamanın kaybolmasını engelleyici özelliğe sahiptir. Eğer veri çerçevesi ve uzak çerçeve aynı belirleyici ile aynı anda başlarsa veri çerçevesi uzak çerçevesinden önce iletilir. Tahkim sürecinde her iletici veri yolu üzerindeki bitlerin seviyelerini karşılaştırır. Eğer bu seviyeler düğümün seviyesi ile aynı ise mesaj iletimi devam eder. Düşük seviyeli bit gönderilip, baskın seviyeli bit veri yolu üzerinde gözlenirse düğüm tahkimi kaybeder ve daha fazla bit göndermeden iletimi keser.



Şekil 3.8. Tahkim İşlemi (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

Şekil 3.8.'de veri yolu üzerinde tahkim işlemi gösterilmiştir. Düğümlerden gelen mesajlar veri yolu üzerindeki mesajla karşılaştırılır. Veri yolundaki iletilen mesajdan farklı mesaj iletim yapan düğüm tahkim işlemi sonucunda iletim yapamaz. Veri iletiminin en sonunda güvenli olarak başarılmasında, her CAN düğümünün hata denetimi, sinyalleşmesi ve kendi kendine kontrol etmesinin payı vardır. CAN protokolünde hataların belirlenmesi için gözlemleme, CRC (Dönüşsel Artıklık Kontrolü), bit sıralama ve mesaj çerçeve kontrolü yöntemleri kullanılır. Hata denetim yöntemlerinin başarılı olması için birtakım özeliklere sahip olması gereklidir. Tüm hataların belirlenmesi, vericilerin yerel hataları belirlemesi, mesajda rasgele beş hatanın belirlenmesi, bir mesajdaki bit hatalarının 15'ten az olması ve bir mesajdaki her hangi tek sayıların hatalarının belirlenmesi gerekir. Hangi düğümden iletildiği tespit

edilemeyen eksik mesajların sistemin geneli için toplam kalan hata olasılığı Denklem (3.3.) ile ifade edilir.

$$p > M_{ho} \times 4.7 * 10^{11} \quad (3.3.)$$

M_{ho} : Mesaj hata oranı

Bir düğüm tarafından tespit edilen hatalarda eksik mesajlar işaretlenir. Bu durumda olan mesajlar iptal edilir ve otomatik olarak tekrardan iletimi sağlanır. Tespit edilmiş bir hatanın kurtarma zamanı, eğer başka bir hata yoksa sonraki mesajın başlaması için gerekli olan 29 bit zamanıdır. 29 bit zamanı genişletilmiş CAN mesajının belirleyici kısmını (ID) ifade etmektedir. CAN düğümleri, kısa süreli karışıklıklar ile kalıcı hataları ayırt etme özelliğine sahiptir. Hatalı düğümler alıcı ve verici sayaçlarının aldığı değerlere göre iletişime kapatılır. CAN protokolü seri iletişim özelliğinde olduğundan çok sayıda cihaz bağlanabilir. Teorik açıdan bir sınırlama yoktur. Fakat uygulamada düğümlerin sayısı gecikme zamanı ve/veya veri yolu hattının elektriksel özelliğine bağlanacak cihazların sayısını etkilemektedir. Veri yolu üzerinde bitler tek kanal üzerinden taşınır. Tek kanal üzerinden taşınmasından dolayı verinin yeniden senkronizasyonu elde edilmelidir. Kanalın özelikleri farklı olabilir. Örneğin; tek tel (ve toprak), iki diferansiyel kablo veya fiber optik kablo olabilir. CAN protokolünde veriler iki tamamlayıcı lojik değere sahiptir. Baskın veya düşük bit. Baskın ve düşük bitlerin eş zamanlı iletiminde veri yolu baskın bit değeri alacaktır. Örneğin, tek tel üzerinden iletim yapmak için tasarlanmış CAN sisteminde baskın bit seviyesi ‘0’, düşük bit seviyesi ‘1’ olarak alınılabilir. Bit seviyelerini ortamın fiziksel durumları belirler. CAN protokolünde alıcılar mesajın kararlılığını kontrol eder. Onaylama biti mesajın doğru biçimde alındığını belirlemede kullanılır. Kararsız mesajlarda onay biti hata oluştuğunu göstermek için kullanılır. Sistemin güç tüketimini azaltmak için CAN cihazı uyku moduna geçer. Uyku modunda cihazın etkinliği yoktur ve veri yolu sürücüleri ile bağlantısı kesilir. Sistemin iç durumunda bir değişiklik veya veri yolunda hareketlilik tespit edildiğinde sistem uyku modundan uyanma moduna geçer. Uyanma modunda cihazın etkinliği tekrar başlar. Buna rağmen iletim katmanı sistemin salınımının kararlı hale geçmesini ve veri yolu etkinliğinin kendiliğinden senkronize olması bekler. Veri yolunun senkronize olması ardışık 11 düşük bitin denetimi sonucunda olur. Tüm bu işlemlerin olması için veri yolu sürücülerinin etkin durumda olması gerekir. Sistemdeki diğer düğümlerinin uyanma moduna geçmesi için ise uyku modunda özel bir uyanma modu mesajının iletilmesi gerekir. Sistemin tasarımında en düşük belirleyici kısmı kullanılmalıdır. (Örneğin; ddd dddb dddd, d = düşük, b = baskın)

3.6.1. CSMA/CD

Taşıyıcı Duyarlı Çoklu Geçiş/ Çarpışma Belirleyici anlamına gelmektedir. Bu sistemle çeşitli düğümler veri yolu uygun olduğunda mesaj iletiminde bulunmaya çalışır. Ağ üzerindeki mesajlar aynı anda iletimde olursa sorun oluşur. Belirli bir periyottan sonra her düğüm ağa ulaşmaya devam edecektir. Veri transferinin iptalleri teorikte ağın taşıma kapasitesinin düşmesine neden olur. Bu durumda şebekenin veri trafiğinin tepe değerlerini aldığı zaman bloklar. Gerçek zamanlı uygulamalar çalıştıran ağlar için bu istenmeyen durumdur.

3.6.2. CSMA/CA

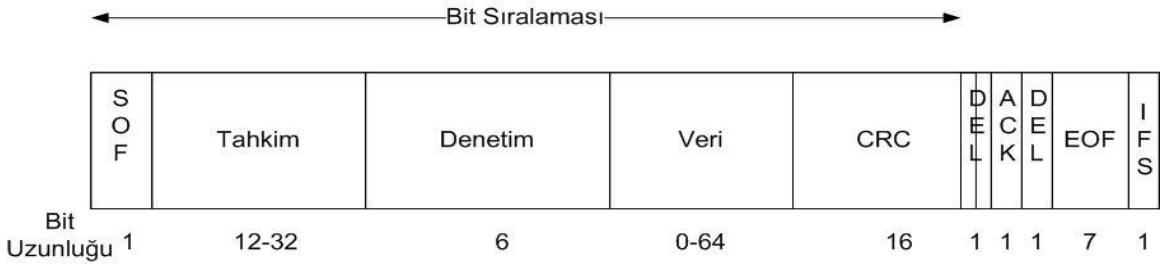
Taşıyıcı Duyarlı Çoklu Geçiş anlamına gelmektedir. Veri yolu üzerinde veri paketlerinin çarpışmasının önlenmesi amaçlanmıştır. CSMA/CD protokolünde karşılaşılan problemler incelenerek geliştirilmiştir. Bu protokolde ağ üzerindeki düğüm çarpışma olmaması için veri yolunu gözlemler. Veri yolu boş olduğunda mesaj iletimi yapar. CSMA/CA, CAN protokolünde kullanılmaktadır [19].

3.7. Mesaj İletimi

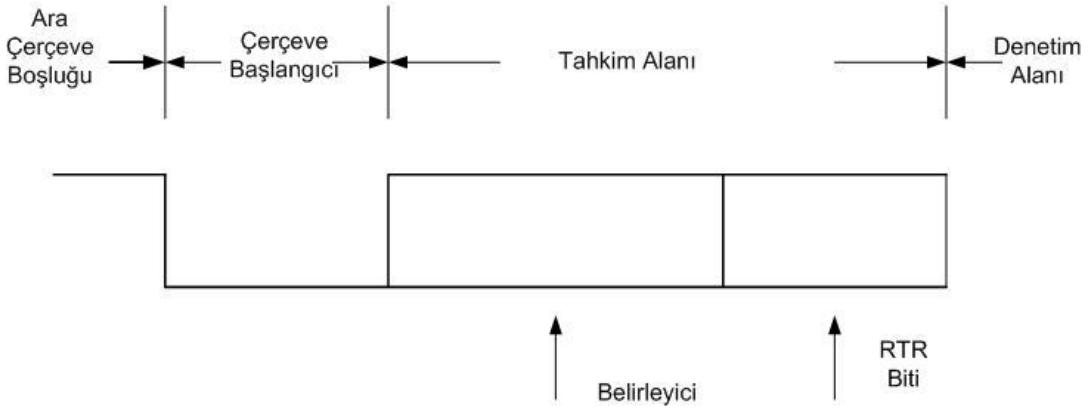
CAN protokolünde mesajın iletimi konusunda kullanılan mesajların biçimleri ve mesaj çevrelerinin türleri önemlidir.

3.7.1.Çerçeve Türleri

Mesaj iletimin belirlenmiş ve denetlenmiş dört tip çerçeve vardır. Veri çerçevesi, veriyi vericilerden alıcılara taşımada kullanılır. Veri çerçevesi yedi farklı bit alanından oluşur. Çerçeve Başlangıcı, Tahkim Alanı, Denetleme Alanı, Veri Alanı, CRC (Dönüşsel Artıklık Denetimi) alanı, ACK (Onay) alanı ve Çerçeve bitiş alanı. Alarm sistemlerinde, alarm meydana gelen düğümlerin sadece belirleyici kısmı alarmın oluştuğu bildirmeye yeterlidir. Bu durumda veri gönderilmeyeceği için veri alanının uzunluğu sıfır olur. Şekil 3.9.'da standart CAN mesajının biçimi gösterilmiştir. Çerçeve Başlangıç biti veri ve uzak çerçevelerin başlangıcıdır. Tek bir baskın bit içerir. Bir düğüm veri yol boş ise iletime başlar. Tüm düğümler Çerçeve Başlangıç Bitinin (SOF) yükselen kenarı ile senkronize olmalıdır. Bir diğer çerçeve olan tahkim alanı, belirleyici kısmı ID ve RTR (Uzak Çerçeve İletim İsteği) bitini içerir. Şekil 3.10.'da tahkim alanı gösterilmiştir.



Şekil 3.9. CAN Mesajı Veri Çerçevesi (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)



Şekil 3.10. Tahkim Alanı (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

CAN mesaj çerçevesinde belirleyici kısmının uzunluğu 11 bittir. Bu bitler ID-10'dan ID-0'a kadar olan bölümdür. En az önemli biti ID-0'dır. ID-10 ile ID-4'ten oluşan bu önemli yedi bit düşük seviyeli olmamalıdır. Uzak çerçeve iletim istek biti (RTR), veri çerçevesi içinde baskın durumunda olmalıdır. Uzak çerçevede ise düşük seviyeli olmalıdır.

3.7.2. Denetim Alanı

Denetim alanı 6 bitten oluşur. Veri uzunluk kodu (DLC) ve iki bit genişletilmiş CAN düzeni için kullanılacaktır. Denetim alanındaki baytların sayısı veri uzunluk kodu olarak belirlenir. Veri uzunluk kodu 4 bittir ve kontrol alanı içinde iletilir. Şekil 3.11'de denetim alanı ve içerdiği bitler gösterilmiştir. Çizelge 3.2.'de veri uzunluk kodu ve aldığı değerler gösterilmiştir. Baskın bit (Dominant) d, düşük bit (Recessive) r ile gösterilmiştir

Şekil 3.11. Denetim Alanı

.

Çizelge 3.2. Veri Uzunluk Kodu

Veri Bayt Sayısı	Veri Uzunluk Kodu			
	DLC3	DLC2	DLC1	DLC
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	d	d

Çizelge 3.2.'de kullanılan veri baytı için kullanılan sayılar 0 ila 8 arasındadır. Diğer sayılar kullanılmaz.

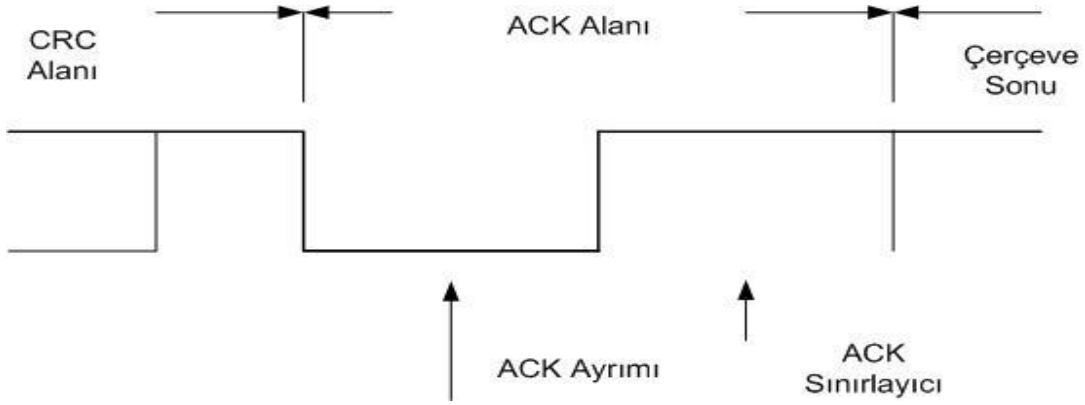
3.7.3. CRC Alanı

Cyclic Redundancy Check (CRC), dönüşsel artıklık denetiminin kısaltımıdır. Dönüşsel artıklık kodu denetimi mikrodenetleyici tarafından hesaplanır. Seri iletişim protokollerinde hata denetiminde kullanılan yöntemlerin başında gelmektedir. 127 bitten aşağısı (BCH kodu) için en iyi yöntemdir. CRC hata denetim yönteminde gönderilen bit dizisi CAN protokolü için tanımlanan polinoma bölünerek bölüm sonucu kalan değerlere bakılarak denetim sağlanır. Bu polinomun katsayıları SOF, Tahkim Alanı, Kontrol Alanı,Veri alanıdır. Polinomun katsayıları CAN mesajındaki alanların başlangıcını belirlemektedir.

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1 \qquad (3.4.)$$

3.7.4. ACK Alanı

ACK alanı 2 bit uzunluğunda ve ACK bölümü ve ACK sınırlayıcısını içerir. ACK alanında verici düğümü iki düşük bit gönderir. Bir alıcı mesajı doğru aldığında bunu vericiye baskın bit gönderir. Gönderilen bu baskın bit ACK bölümü alanındadır. Kısaca bu işleme ACK gönderimi denir. Şekil 3.12.'de onay bölümü gösterilmiştir.



Şekil 3.12. ACK Bölümü (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

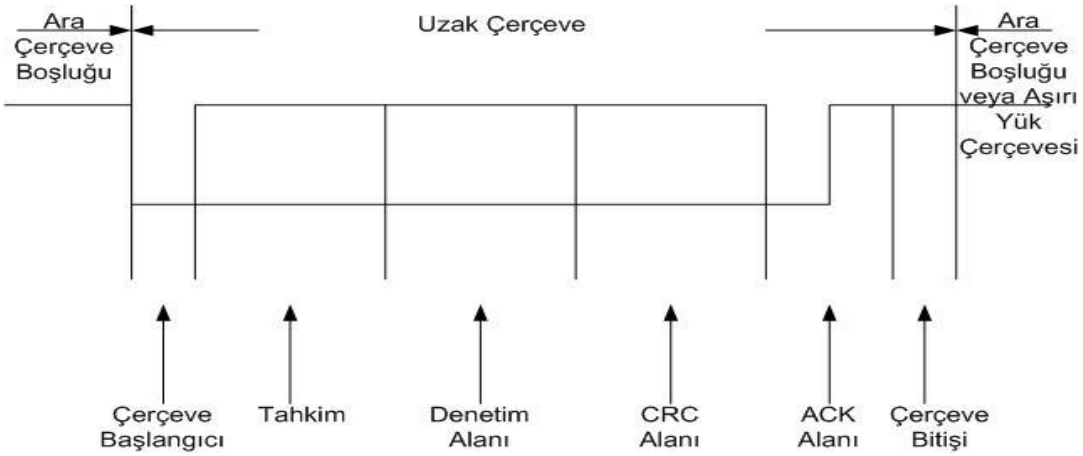
Tüm düğümler eşlenmiş CRC dizisini alır. İçinde ACK bölümünde bulunan vericinin düşük bitini baskın bit olarak iletir. ACK alanını CRC alanından ayırmak için kullanılır. Bu bitin kullanım amacı CRC alanının fark edilmesini sağlamaktır. ACK alanının ikinci bitidir. Düşük seviyeli bit olmalıdır. Sonuç olarak ACK bölüm iki düşük seviyeli bit tarafından (CRC sınırlayıcı, ACK sınırlayıcı) çevrelenmiştir.

3.7.5. Çerçeve Bitiş Biti

CAN sisteminde, her veri ve uzak çerçeve 7 düşük seviyeli bit içeren işaret dizisiyle sınırlanır.

3.7.6. Uzak Çerçeve

Uzak çerçeve, veri yolu üzerindeki bir düğüm tarafından aynı belirleyici kısmı içeren veri çerçevesi için iletilir. Bir düğüm belirli bir veriyi kendisi için gerekli olan durumun gerçekleşmesini sağlamak için uzak çerçeve gönderir. Uzak çerçeve altı farklı bit alanından oluşur. Bu alanlar çerçeve başlangıcı, Tahkim alanı, Kontrol Alanı, CRC Alanı, ACK alanı ve Çerçeve bitidir. Veri çerçevelerinin aksine, Uzak Çerçevede RTR biti düşük seviyededir. Uzak çerçevede veri alanı yoktur. Veri uzunluk kodunun değerlerinden bağımsızdır. 0 ila 8 arasında uygun değer alabilir. Veri uzunluk kodu veri çerçevesiyle uyumludur. Şekil 3.13.'de uzak çerçevenin yapısı gösterilmiştir.

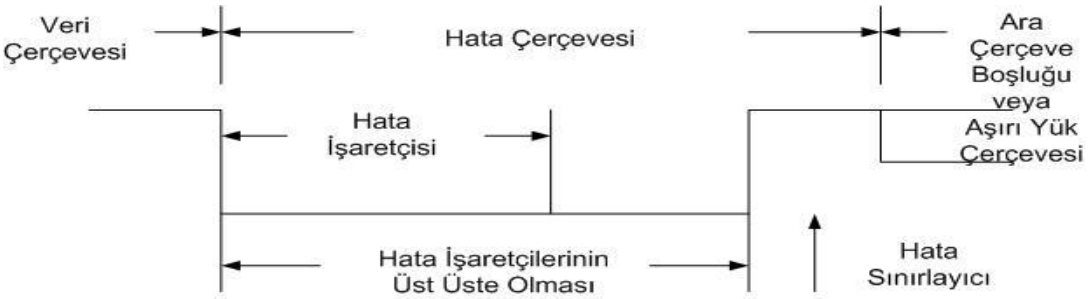


Şekil 3.13. Uzak Çerçeve Yapısı (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

RTR bitinin polaritesi veri çerçevesi (RTR biti baskın seviyeli) veya uzak çerçevesinin (RTR biti düşük seviye) iletildiğini belirler.

3.7.7. Hata Çerçevesi

Hata çerçevesi, veri yolu üzerindeki her hangi bir düğüm tarafından veri yolu hatası algılandığında iletilir. Hata çerçevesi iki farklı alan içerir. İlk alan farklı düğümden gelen hata işaretçilerinden oluşur. İkinci alan ise hata sınırlayıcı bölümüdür. Şekil 3.14.'de hata çerçevesi ve bitleri gösterilmiştir.



Şekil 3.14. Hata Çerçevesi Yapısı (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

Hata çerçevesinin doğru işlenmesi için hata pasif düğümünde veri yolunun en az 3 bit zamanı kadar boş olmasını gerekir. Bunun için hata pasif alıcısındaki hata sadece bu düğüm için olmalıdır. Hatanın olduğu düğümün haricinde veri iletilmeyeceği için veri yolu üzerinde mesaj olmayacaktır.

Hata işaretçisinin aktif hata ve pasif hata olmak üzere iki türü vardır.

i. Aktif hata işaretçisi 6 ardışık baskın bitten oluşmalıdır.

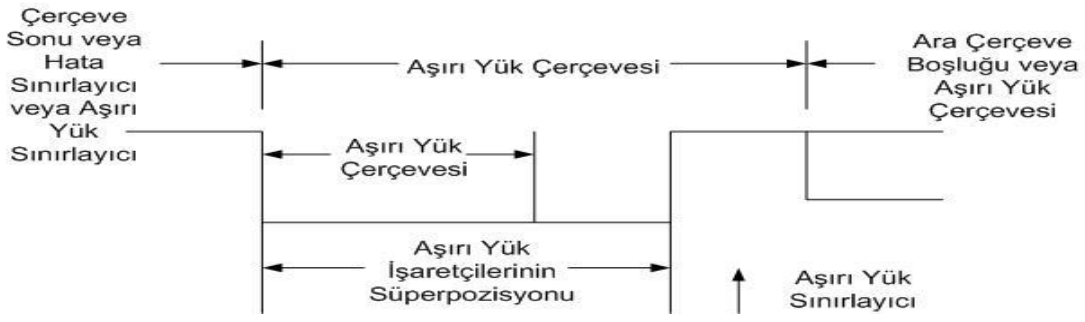
ii. Pasif hata işaretçisi 6 düşük seviyeli bitten oluşmalıdır. Diğer düğümler tarafından düşük seviyeli bitlerin üzerine baskın bitlerin yazılmamalıdır.

Aktif hata işaretçisinin iletimiyle hatanın oluştuğu düğüm hata durumlarının sinyallerini belirler. Hata işaretçisinin formu bit sıralama kontrolünü bozar. Çerçeve başlangıç bitinden CRC sınırlayıcısına veya ACK alanından çerçeve bitiş bitine kadar olan kısım hata işaretçisinin etkilediği bölümdür. Hata denetiminde diğer tüm düğümler hata durumunu algılar ve hata işaretçi bitini iletir. Baskın bitlerin sıralı olması farklı düğümlerden gelen hata işaretçilerinin rahatça belirlenmesini sağlar. Sıralı bitlerin uzunluğu 6 ila 12 bit arasında değişir. Hatanın oluşmadığı düğüm kendi hata durumunu belirler. Pasif hata işaretçisinin iletimiyle iletişimin sağlanmasını dener. Pasif hata işaretçisi sıralı 6 bitin tespit edilmesiyle tamamlanır. Hata sınırlayıcı bölümü sekiz düşük seviyeli bitten oluşur. Hata işaretçisinin iletiminden sonra her düğüm düşük seviyeli bit gönderir ve veri yolunu düşük seviyeli biti tespit etmek için gözlemlemeye başlar. Sonrasında yedi veya daha fazla düşük seviyeli biti gönderir.

3.8. Aşırı Yük Çerçevesi

Aşırı yük çerçevesi, veri çerçevesi ile uzak çerçeve arasında fazladan gecikme zamanı sağlar. Aşırı yük işaretçisi ile aşırı yük sınırlayıcısından oluşur. Alıcının içsel durumları, sonraki veri veya uzak çerçeveleri arasında gecikmeye ihtiyaç duyar. Ara zamanında baskın bitin gözlenmesi.

Aşırı yük çerçevesinin başlaması iki durumda olur. Aşırı yük işaretçisinin birinci durumunda ara zamanında gönderilen ilk bitin tespit edilmesinde veya aşırı yük işaretçisinin ikinci durumu ara zamanında gönderilen ilk bitten sonra baskın bitin gözlenmesinde aşırı yük çerçevesi başlar. Şekil 3.15.'de aşırı yük çerçevesi ve bitleri gösterilmiştir.



Şekil 3.15. Aşırı Yük Yapısı (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

3.8.1. Aşırı Yük İşaretçisi

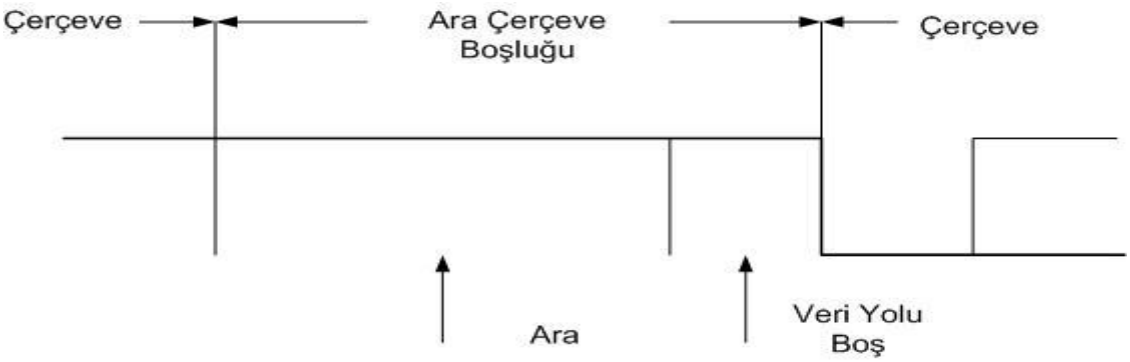
Aşırı yük işaretçisi 6 baskın durumda olan bit içerir. Aşırı yük çerçevesinin biçimi ara bölümün yapısını bozmaktadır. Sonuç olarak diğer tüm düğümler aşırı yük durumunu tespit eder ve aşırı yük işaretçilerini iletir.

3.8.2. Aşırı Yük Sınırlayıcısı

Aşırı yük sınırlayıcısı sekiz düşük seviyeli bitten oluşur. Aşırı yük sınırlayıcısı hata sınırlayıcısıyla aynı yapıdadır. Aşırı yük işaretçisinin iletiminden sonra düğüm baskın bitten sonra düşük seviyeli bitin iletimi için veri yolunu gözlemler. Bu noktada her veri yolu üzerindeki düğüm aşırı yük işaretçisinin iletimini bitirir ve aynı anda yedi veya daha fazla düşük seviyeli bitin iletimine başlar.

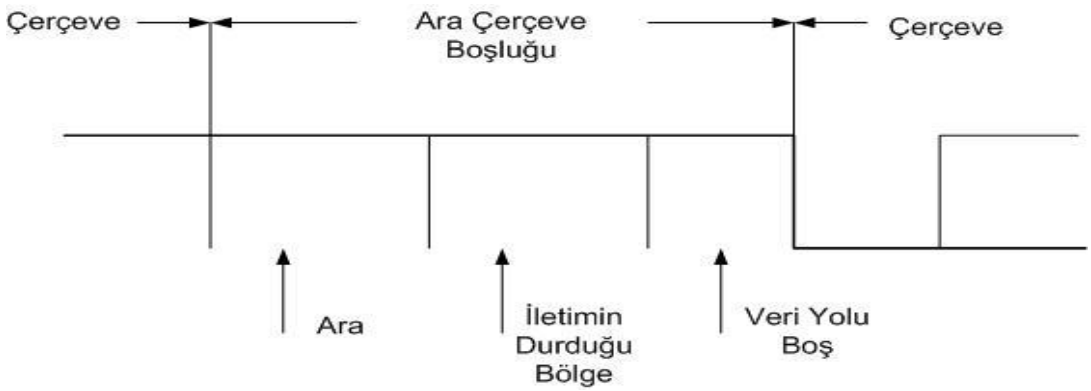
3.9. Ara Çerçeve Boşluğu

Veri ve uzak çerçeveleri arasında ara çerçeve boşluğu bulunmaktadır. Aşırı yük çerçeveleri ile hata çerçeveleri arasında veya çok sayıda aşırı yük çerçeveleri arasında ara çerçeve boşluğu bulunmaz. Ara çerçeve boşluğu ara zamanı ve veri yolu boş bit alanlarını içermektedir. Hatanın olmadığı düğümler için önceki mesajın ileticisi veya iletimini engelleyici anlamı taşır. Hatanın olmadığı veya önceki mesajın alıcısı için Şekil 3.16'da gösterilmiştir.



Şekil 3.16. Hatanın Olmadığı veya Önceki Mesajın Alıcısı için Ara Çerçeve Boşluğu (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

Hatanın olmadığı veya önceki mesajın vericisi için ara çerçeve boşluğu ise Şekil 3.17'de gösterilmiştir.



Şekil 3.17. Hatanın Olmadığı veya Önceki Mesajın Vericisi için Ara Çerçeve Boşluğu (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır.)

Şekil 3.17.'deki ara zamanı üç düşük seviyeli bit içerir. Ara zamanında hiçbir düğüm veri veya uzak çerçeve iletemez. Sadece aşırı yük sinyali iletilebilir.

3.9.1. Veri Yolu Boş Bölümü

Veri yolunun boş olma bölümü tahkim bölgesinin uzunluğuna bağlıdır. Bu bölümde veri yolu iletim için uygundur. Böylece her düğüm veri yoluna ulaşabilir. Diğer mesajın iletimi için geçen sürede bekleyen bir mesaj izleyen ara zamanında ilk bitini iletir. Veri yolu üzerinde tespit edilen baskın bit çerçeve başlangıç bitidir.

3.9.2. İletim Erteleme Bölümü

Hatanın olmadığı düğüm mesaj ilettikten sonra 8 düşük seviyeli bit sonraki ara zamanında gönderilir. Düşük seviyeli bitlerin gönderilmesinden önce veri yolunun iletim için boş olduğu bilgisi iletilmelidir. Eğer bu arada iletim başlarsa, düğüm bu mesajın alıcısı olacaktır.

3.10. Alıcı / Verici Tanımları

Mesajı ileten düğüme verici denir. Bu düğüm veri yolunun boş veya düğümün tahkim sonucunda iletimin geçersiz olmasıyla verici olmaktan çıkar. Mesajın vericisi ve veri yolu boş değilse düğüm alıcı olarak tanımlanır.

3.11. Mesaj Doğrulaması (Ana Başlık)

Mesajın doğru olarak alındığı zaman verici ve alıcı için farklıdır. Vericide mesaj doğru ve çerçeve bitiş bitine kadar hata yoktur. Eğer mesaj kesikse, yeniden iletim öncelik bilgisiyle otomatik olarak başlayacaktır. Diğer mesajların veri yolu üzerinden iletimi veri yolu boş

olduğunda hemen başlayacaktır. Alıcıda ise çerçeve bitiş bitine kadar hata yoksa mesaj doğrudur.

3.12. CAN Kodlaması

Bit sıralama yönteminde çerçeve başlangıç, tahkim alanı, kontrol alanı, veri alanı ve CRC dizisi bitleri sıralanmıştır. Bir verici beş ardışık bitleri tespit ettiğinde, otomatik olarak iletilecek bit akışına tamamlayıcı bir bit ekler. Veri veya uzak çerçevenin kalan bit alanları (CRC sınırlayıcısı, ACK alanı ve Çerçeve bitiş biti) sabit biçimdedir. Hata ve aşırı yük çerçevesi de sabit biçimdedir. Fakat bit sıralama yönteminde kodlanmamıştır.

Mesajdaki bit akışı NRZ (Ters Sıfıra Dönüşsüz) kodlamasına uygundur. Bunun anlamı toplam bit zamanında oluşturulan bit zamanı baskın veya düşük seviyeli değildir [16]. Şekil 3.18.'de NRZ kodlama örneği gösterilmiştir.

Şekil 3.18. NRZ Kodlama (Fuhrman [16]'dan değiştirilerek alınmıştır.)

3.13. Hata İşleme

Hata İşleme, hatanın tespit edilmesi ile başlar. Tespit edilen hatanın hangi düğümden ve ne anlama geldiği işlenerek tamamlanır.

3.13.1. Hata Belirleme

ISO 11898 standardına göre tanımlanmış beş farklı hata türü vardır. Bunlardan ilki bit hatasıdır. Bir düğüm veri yoluna bit gönderir ve veri yolunu gözlemler. Gönderilen bit ile gözlemlenen bit farklı olduğunda bit hatası tespit edilir. Tahkim alanının veya ACK bölümünün bit akışında düşük seviyeli bit iletiminde istisnai durum oluşur. Sonrasında veri yolu üzerinde baskın bit gözlendiğinde bit hatası oluşur. Bir verici pasif hata işaretçi bitini gönderir ve baskın bit gözler. Bu durum bit hatası olarak yorumlanmaz. Sıralama hatası, mesaj alanının 6. ardışık bit zamanında tespit edilir. Bunun için mesajın bit sıralama yöntemiyle kodlanması gerekir. CRC dizisinde vericinin, CRC hesaplama sonucu vardır. Alıcılar da vericiler gibi CRC değerinin hesaplar. CRC hatası hesaplanan sonucun alınan CRC dizisiyle aynı olmadığında oluşur. Form hatası da bir veya birden fazla sabit yanlış bit tespit

edildiğinde oluşur. Onaylama hatası ise verici tarafından ACK bölümü zamanında baskın bit gözlenmediğinde oluşur.

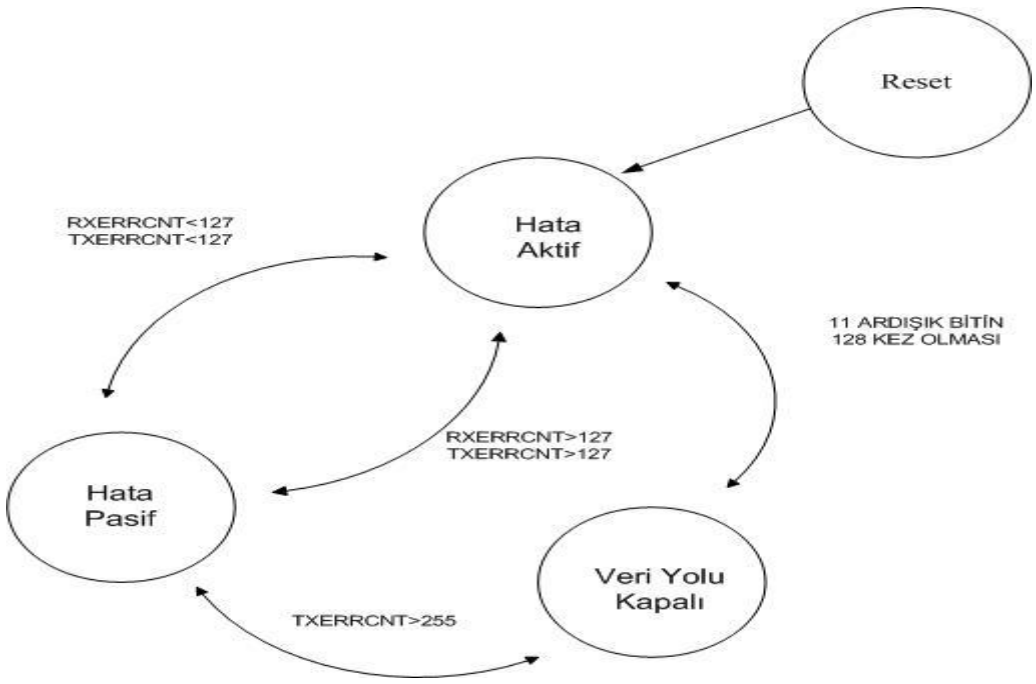
3.14. Hata Sinyalleşmesi

Bir düğüm hata işaretçisi ileterek hata durumunu bildirir. Hatanın olduğu düğümde aktif hata işaretçisi, hatanın olmadığı düğümde ise pasif hata işaretçisi iletilir. Bit hatası, sıralama hatası, biçim hatası veya onay hatası bir düğüm tarafından tespit edildiğinde hata işaretçisinin iletimi kendi düğümünde sonraki bitle başlar.

CRC hatası tespit edildiğinde, hata işaretçisinin iletimi ACK sınırlayıcısından sonra gelen bitle başlar. Bunun için hata işaretçisinin diğer durumlar için iletilmemesi gerekir.

3.15. Hata Düzeltme

Bir düğümde hata düzeltme 3 durumda olur. Hatanın olduğu düğüm veri yolu üzerinde olan mesajı alır ve aktif hata işaretçisini gönderir. Bu duruma hata aktif durumu denir. Hatanın pasif durumda olması durumunda düğüm hata aktif işareti göndermez. Veri yolundaki üzerindeki tüm mesajlar alınır. Hata oluştuğunu belirtmek için hata pasif işareti gönderilir. İletimden sonra hatanın oluşmadığı düğüm iletim yapmadan önce bekler. Üçüncü hata düzeltme durumu ise veri yolunun kapalı olması durumudur. Kapalı durumda olan düğüm veri yolundaki mesaj iletimini etkilemez. Veri yolu düğümlerinde hata düzeltme için iki sayacı vardır. Bunlar alıcı ve iletim hata sayaçlarıdır.



Şekil 3.19. Hata Sayaçları BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır)

Bu sayıcılar kurallara göre düzenlenir. Şekil 3.19'da hata sayacının algoritması gösterilmektedir. Birden fazla kural iletilecek mesajda kullanılır.

3.15.1. Hata Sayaçlarının Çalışma Kuralları

i. Alıcı hata tespit ettiğinde, alıcı hata sayıcısı 1 artar. Aktif hata işaretçisi veya bir aşırı yük işaretçisinin iletiminde bit hatası tespit edildiğinde sayaç artmaz.

ii. Alıcı hata işaretçisinin ilk bitinin iletiminde baskın bit tespit ettiğinde alıcı hata sayacı 8 artar.

iii. Verici, hata işaretçisi gönderdiğinde iletim hata sayacı 8 artar.

Hata sayaçları için iki istisnai durum bulunmaktadır. Eğer verici de hata yoksa ve onaylama hata biti tespit edilmişse bunun nedeni ACK biti ile pasif hata işaretçisinin iletiminde baskın bit tespit edilememesidir. Diğer durumda ise verici tahkim sırasında sırlama hatası oluştuğunda hata işaretçisi yayınlar. Bunun için RTR bitinden önce sıralı bitler gelir. Sıralı bitler düşük seviyeli olmalı ve iletilmeli fakat baskın bit olarak algılanmalı. Bu iki istisnai durum iletim hata sayacının değerini değiştirmez.

iv. Eğer bir verici aktif hata işaretçisi veya aşırı yük işaretçisi iletirken bit hatası belirlerse verici hata sayacı 8 artar.

v. Eğer bir alıcı aktif hata işaretçisi veya aşırı yük işaretçisi iletirken bit hatası belirlerse alıcı hata sayısı 8 artar.

vi. Herhangi bir düğüm aktif hata işaretçisi pasif hata işaretçisi veya aşırı yük işaretçisinden sonraki 7 ardışık baskın biti idare eder. 14. ardışık baskın bitin tespit edilmesinden sonra veya pasif hata işaretçisinden sonraki 8.baskın bitin tespiti ve her vericinin 8 ardışık baskın bit dizisinden sonra verici ve alıcı hata sayacı 8 artar.

vii. Bir mesajın başarılı iletiminden sonra (ACK biti alınmış ve çerçeve bitiş bitine kadar hata yoksa) verici hata sayacı 1 azalır. Verici hata sayacı 0 ise azalma olmaz.

viii. Bir mesajın başarılı alımından sonra (ACK bölümüne kadar hata yok ve ACK bit başarılı bir şekilde iletilmişse) alıcı hata sayacı 1 ila 127 arasında ise 1 azalır. Eğer alıcı hata sayacı 0 ise sayaç değeri 0 olur. 127'den büyük ise sayaç değeri 119 ila 127 arasında değer alır.

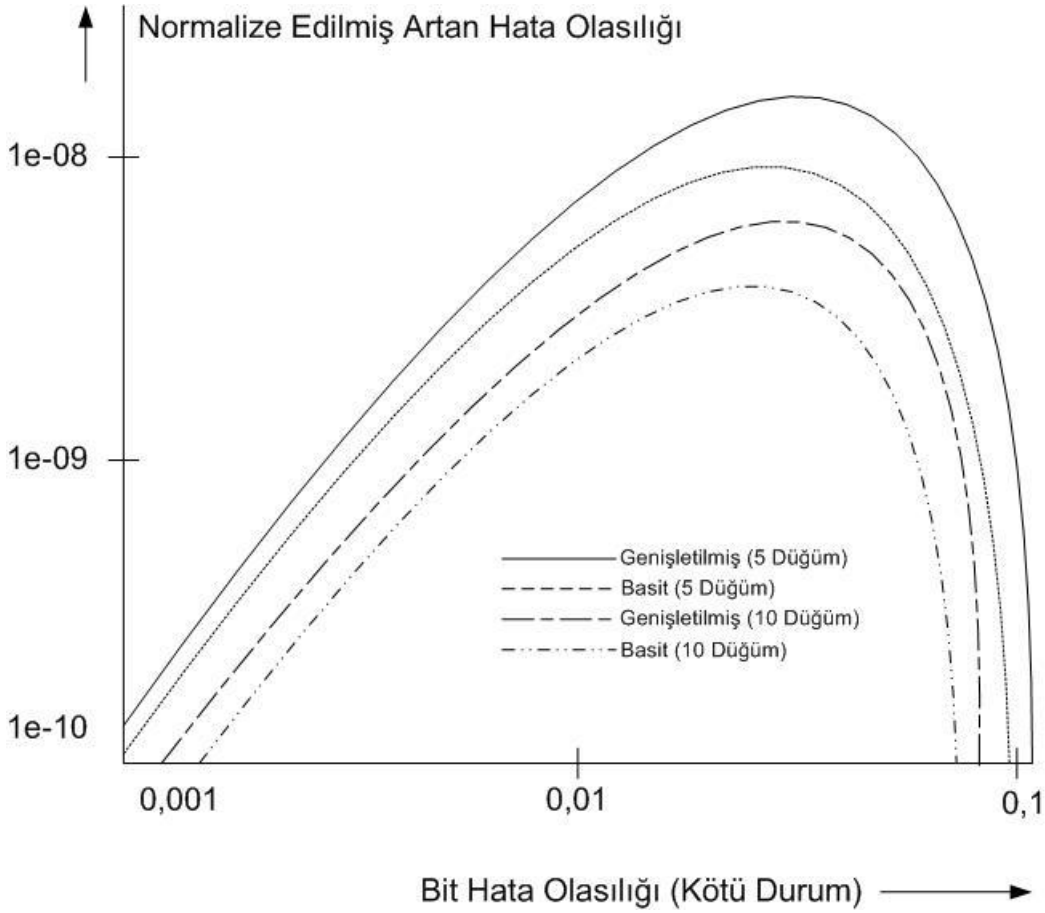
ix. Alıcı ve verici hata sayacı 128 veya 128'den büyükse bu düğüm pasif hatalı kabul edilir. Bir hata oluştuğunda pasif hatalı olur ve aktif hata işareti iletilir.

x. Verici hata sayacı 256'ya eşit veya fazla olursa düğüm kapalı duruma gelir.

xi. Verici ve alıcı hata sayaçları 127'ye eşit veya daha az olursa pasif olan düğüm tekrar aktif hatalı duruma gelir.

xii. Kapalı bir düğümün hata aktif durumuna geçmesi için, hata sayaçlarının 0'a ardından 128'e ayarlanmalı ve sonrasında 11 ardışık düşük seviyeli bit veri yolunda gözlenmelidir. Başlangıç anında sadece 1 düğüm aktifse ve bu düğüm bazı mesajları iletiyorsa, bu düğüm onaylama biti almayacak, hatayı bulup mesajı tekrarlayacaktır. Veri yolu kapalı durumuna gelmez fakat hata pasif durumuna gelebilir.

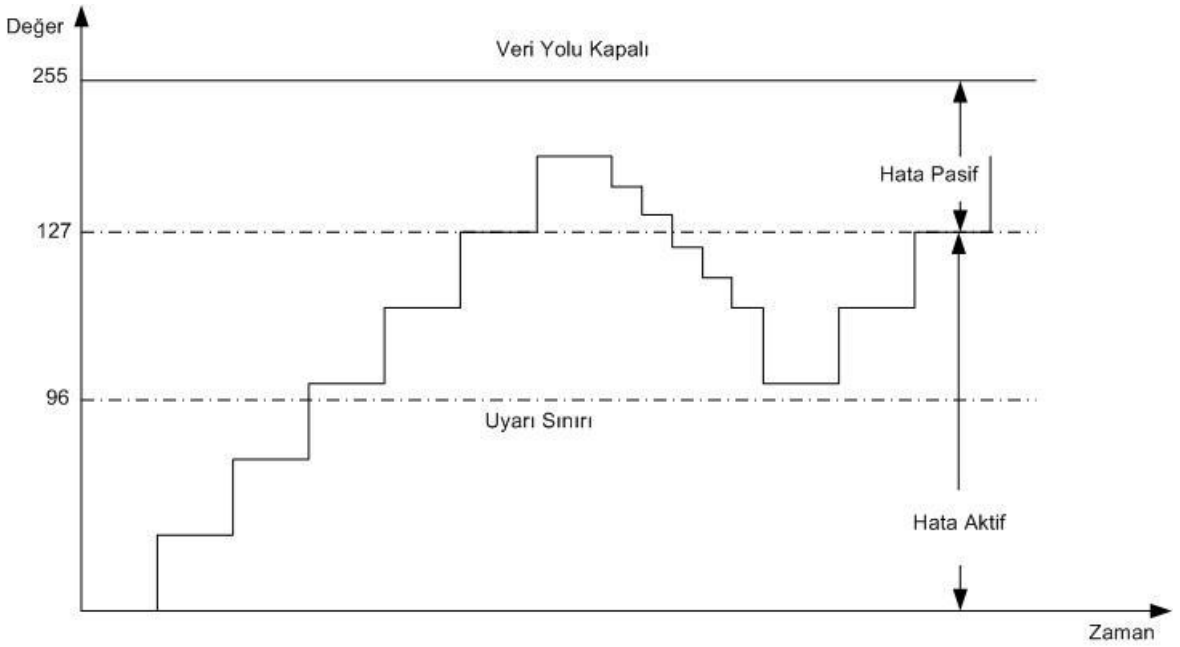
Şekil 3.20.'de CAN protokolünün basit ve genişletilmiş sürümünde düğüm sayısına göre hata belirleme performansı gösterilmiştir [5].



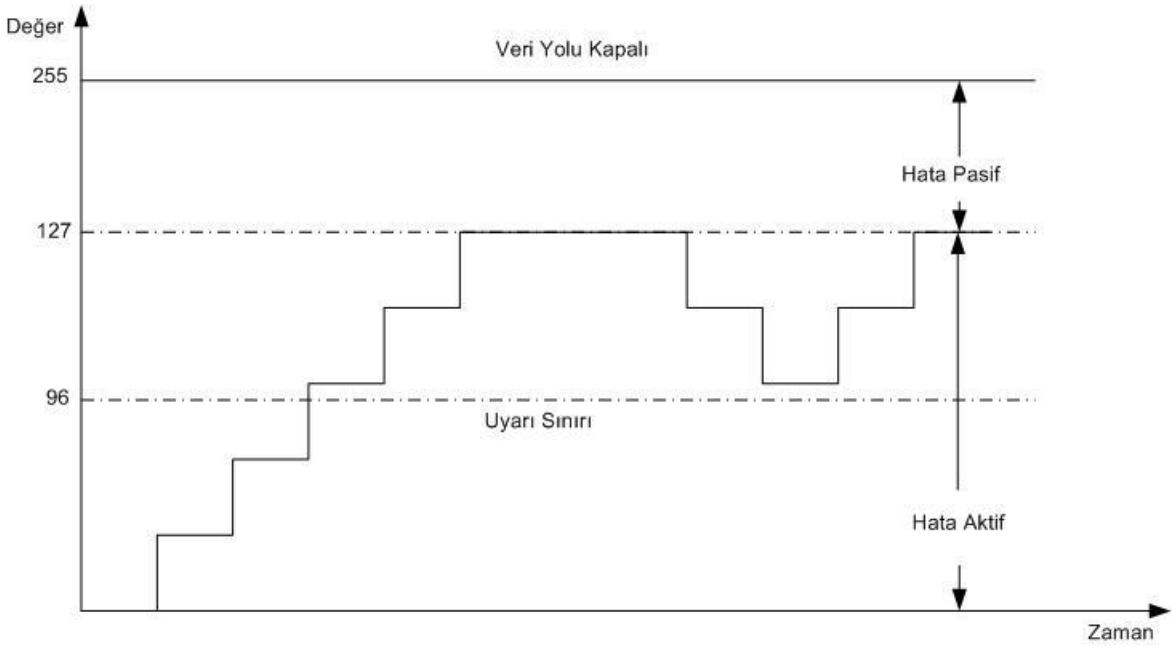
Şekil 3.20. Hata Belirleme Performansı (Richards [20]'den değiştirilerek alınmıştır.)

Hata sayaçlarının aldığı değere göre gösterimi Şekil 3.21. ve Şekil 3.22.'de gösterilmiştir. Alıcı hata sayacının grafiği ise Şekil 3.22.'de gösterilmiştir. Şekil 3.19.'da gösterilen hata pasif durumunun alıcı sayaç değerinin 127'yi geçmesi durumunda gerçekleşeceği görülmektedir. Alıcı veya verici hata sayaçlarının değeri 96 veya daha fazla ise veri yolu aşırı güvensizdir. Hata sayaçlarının 255 değerine ulaşması durumunda sayacın ait olduğu düğümdeki CAN mikrodenetleyicisinin mesaj iletimi gerçekleşmeyecektir. Hata sayaçlarının

değerlerine bakılarak sistemin düzgün çalışıp çalışmadığı gözlenebilir. Devre dışı kalan düğümler belirlenip alarm sinyali üretilir.



Şekil 3.21. İletim Hata Sayacı (Fuhrman [16]'dan değiştirilerek alınmıştır.)



Şekil 3.22. Alıcı Hata Sayacı (Fuhrman [16]'dan değiştirilerek alınmıştır.)

3.16. Bit Zamanlama

Bit zamanlaması, kullanılan CAN mikrodenetleyicinin iletişim hızının hesaplanmasında kullanılır.

3.16.1. Nominal Bit Oranı

Nominal Bit Oranı saniye başına iletilmiş bit sayısıdır. Tanımlamada ideal verici tarafından yeniden senkronize olma durumundaki kayıplar göz önünde bulundurulmuştur.

3.16.2. Nominal Bit Zamanı

Nominal bit zamanı ve nominal bit oranı (3.5.) ve (3.6.) denklemlerinde verilmiştir.

$$\tau_{bit} = \tau_{SyncSeg} + \tau_{PropSeg} + \tau_{PS1} + \tau_{PS2} \quad (3.5.)$$

$$f_{NBT} = \frac{1}{\tau_{NBT}} \quad (3.6.)$$

Nominal bit zamanı üst üste binmeyen ayrı zaman bölümlerinden oluşmuştur. Bu bölümler CAN iletişim hızının ayarlanmasında kullanılır [19].

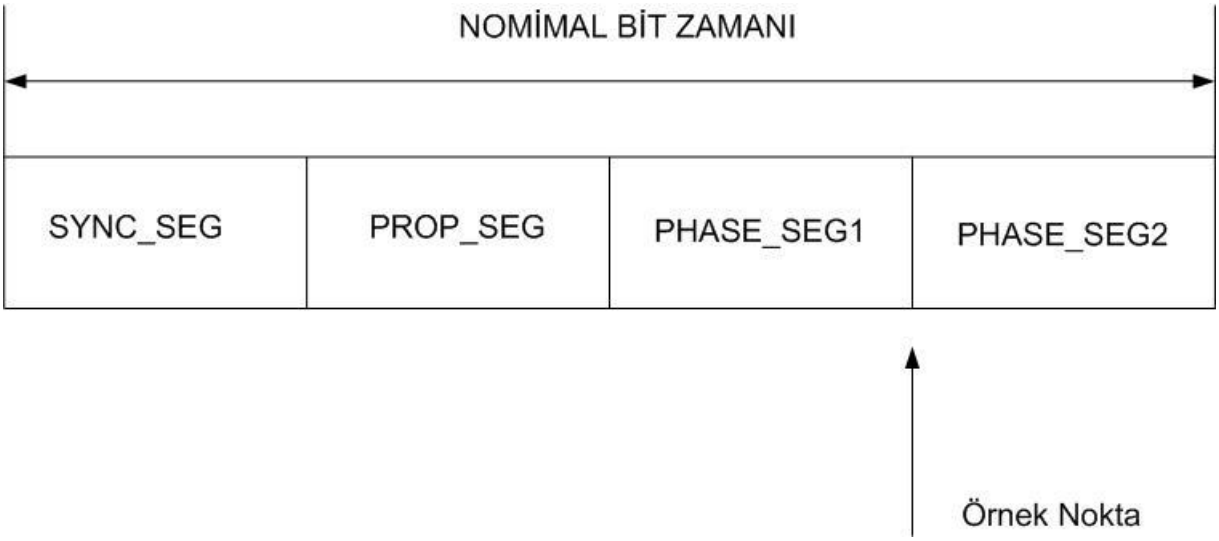
i. Senkronizasyon Kısmı (SYNC_SEG)

ii. Propagasyon Zaman Kısmı (PROP_SEG)

iii. Faz Tamponu 1. Kısım (PHASE _SEG1)

iv. Faz Tamponu 2. Kısım (PHASE_SEG2)

Şekil 3.23.'de nominal bit zamanı ve kısımları gösterilmiştir.



Şekil 3.23. Nominal Bit Zamanı (BOSCH CAN Specification [19]'dan değiştirilerek alınmıştır)

Bit zamanın bu bölümü veri yolu üzerindeki çeşitli düğümleri senkronize yapmada kullanılır. Bu bölümün sinyalinde bir kenar olması beklenir.

Bit zamanın bu bölümü ağın fiziksel gecikme zamanını ayarlamada kullanılır. Veri yolu üzerindeki sinyalin iletim zamanının toplamının iki katıdır. Giriş karşılaştırıcı gecikmesi sinyallerinden oluşmaktadır.

Faz tampon bölümleri 1 ve 2, faz hatalarının kenarlarının ayarlanmasında kullanılır. Bu bölümler yeniden senkronizasyon ile uzatılmış veya kısaltılmış olabilir.

Örnek nokta, veri yolu seviyesinin okunduğu ve istenilen birim değerinin anlaşıldığı noktadır. Veri yolu üzerinde bir kez veya üç kez örneklenir. 1. Faz tampon bölümünün sonunda yer almaktadır.

Bilgi işleme zamanı örnek noktadan başlar. Bu bit seviye hesaplamasından sonra kullanılmak üzere ayrılmıştır.

Parçacık zamanı, osilatör periyodundan türetilmiş sabit zaman aralığıdır. Programlanabilir ön çarpan frekans bölücüsü 1 ila 32 arasındadır. Minimum zaman parçacığının başlamasıyla zaman parçacığı şöyledir.

$$t_q = m \times t_{mq} \qquad (3.6.)$$

t_q : zaman parçacığı.

m : ön çarpan değeridir.

t_{mq} : minimum zaman parçacığıdır.

Senkronizasyon kısmı (SYNC_SEG) 1 zaman parçacığı uzunluğundadır. Propagasyon zamanı (PROP_SEG) ile Faz tampon bölümü1 (PHASE_SEG1) 1,2,….8 zaman parçacığı uzunluğuna programlanabilir. Faz tampon bölümü 2 ise PHASE_SEG1'in ve bilgi işleme zamanının maksimum olduğu değerdir. Bilgi işleme zamanı 2 zaman parçacığı uzunluğunda veya daha az olmalıdır.

Toplam zaman parçacıklarının sayısı ise 8 ila 25 arasında ve programlanabilir olmalıdır.

3.17. Senkronizasyon

CAN sinyalinin, tüm düğümler tarafından alınması için senkronizasyona ihtiyaç duyulmaktadır.

3.17.1. Sabit Senkronizasyon

Sabit senkronizasyondan sonra içsel bit zamanı SYNC_SEG1 ile tekrar başlar. Sabit senkronizasyonda sinyalin kenar kısmının senkronizasyon bölümünde uzanmasını yeniden başlayan bit zorlar.

3.17.2. Yeniden Senkronizasyon Sıçrama Genişliği

Yeniden senkronizasyonun bir sonucu olarak PHASE_SEG1 uzun veya PHASE_SEG2 kısa olur. Faz tampon bölümlerinin uzunluk veya kısalıklarının üst sınırı yeniden senkronizasyon sıçrama genişliği ile belirlenir. Yeniden senkronizasyon sıçrama genişliği 1 ila min(4,PHASE_SEG1) arasında programlanabilir. Saat bilgisi bir bit değerinden diğerine geçişten türetilir. Veri yolundaki bir düğümün yeniden senkronizasyonunda çerçeve boyunca bit akışı olur. Bu akış esnasında maksimum başarılı bit sayısı aynı değerdedir. İki durum arasındaki maksimum bit uzunluğu 29 bit zamanıdır. Yeniden senkronizasyonun oluşabilmesi için bu süre gereklidir.

3.17.3. Kenarın Faz Hatası

Bir kenarın faz hatası seviyelendirme zamanında SYNC_SEG'in kenarının pozisyonunu verir. Faz hatasının işaretinin anlamı şudur:

i. e = 0 kenar SYNC_SEG ile uyumlu

ii. e > 0 kenar örneklem noktasından önce yükseliyor.

iii. e < 0 kenar örnekleme noktasından önceki bitten sonra yükseliyor.

3.17.4. Yeniden Senkronizasyon

Kenarın faz hatasının büyüklüğü yeniden senkronizasyon değerinden az veya eşit olursa bu durumda, yeniden senkronizasyonun etkisi sıkı senkronizasyon ile aynıdır. Faz hatasının değeri yeniden senkronizasyon sıçrama genişliğinden büyük olduğunda

i. Faz hatası pozitif ise PHASE_SEG1 yeniden senkronizasyon sıçrama değerine eşit olacak şekilde genişletilmelidir.

ii. Faz hatası negatif ise, PHASE_SEG2 yeniden senkronizasyon sıçrama değerine eşit olacak şekilde kısaltılmalıdır.

3.18. Senkronizasyon Kuralları

Sabit senkronizasyon ve yeniden senkronizasyon senkronizasyonun iki türüdür. Bu iki tür şu kurallara uymalıdır:

i. Bir bit zamanında yalnızca bir senkronizasyon yapılabilir.

ii. Senkronizasyon için bir kenar kullanılır. Bunun için önceki örnek nokta (önceki okunan veri yolu değeri) da tespit edilecek değer veri yolu üzerindekinden farklıysa bu kenardan hemen sonrakinde senkronizasyon gerçekleşir.

iii. Sabit Senkronizasyon veri yolu boş olduğunda düşük seviyeden baskın seviyeye geçiş kenarında kullanılır.

iv. Diğer tüm düşük seviyeli sinyalin düşen kenarından yükselen tarafına (Yükselen tarafından düşen tarafına ise düşük bit oranlarında geçerlidir.) yeniden senkronizasyon için 1. ve 2. kurallar uygulanır. Bir düğüm yeniden senkronizasyon için baskın seviyeli bit göndermeyecektir. Bu pozitif faz hatasının düşen kenarının yükselmesinden kaynaklanmaktadır. Bu durumda yeniden senkronizasyon için düşük kenarın yükselen tarafı alınacaktır.

3.19. CAN Osilatörünün Toleransının Yükselmesi

CAN sistemi için bit zamanlama referans frekansından (f_{OSC}) türetilir Bu da, düğümler arasındaki ideal osilasyon toleransından daha az bir durum oluşturur. Faz kayması ve osilasyon sırasında gerçekleşir.

Maksimum osilatör toleransını %0,5'ten mümkün olan %1,5'a yükseltmek için CAN protokolünde değişimler gerekliyse yapılabilir. CAN protokol standardında en kötü osilatör toleransı 125 kbps için %1,58'dir [20].

i. Eğer bir CAN düğümü; ara zamanının 3.baskın bitini örneklerse bu biti çerçeve başlangıç biti olarak yorumlar.

ii. Eğer bir CAN düğümü iletmek için bir mesajı bekliyorsa ve ara zamanının 3.bitini çerçeve başlangıç biti olarak almış sonraki biti ise iletilen mesajın belirleyicisi olarak iletir. Belirleyicinin ilk iletilen biti çerçeve başlangıç biti içermez ve doğal olarak alıcıda da alınmaz.

iii. Eğer bir CAN düğümü hata sınırlayıcısının 8. ve son bitini örnekliyorsa sonraki bit aşırı yük çerçevesi hata çerçevesi olmadan iletir. Bu durumda hata sayaçları artmaz.

iv. Sadece sinyalin düşen kenarının yükselen kısmı senkronizasyonda kullanılır.

Bu özellikler sağlanmışsa şu kurallar geçerlidir:

v. Tüm CAN denetleyiciler sabit senkronizasyona çerçeve başlangıç biti ile geçer.

vi. Ara zamanında 3 düşük seviyeli bit sayılmadan hiçbir CAN denetleyicisi çerçeve başlangıç biti göndermez.

Bu ayarlar maksimum osilatör toleransı %1,58 olan seramik rezonatör ve veri yolu hızı 125 kbps olduğunda geçerlidir. CAN 2.0 B protokolünün uyumluluğunu artırmak için şu yapılmalıdır.

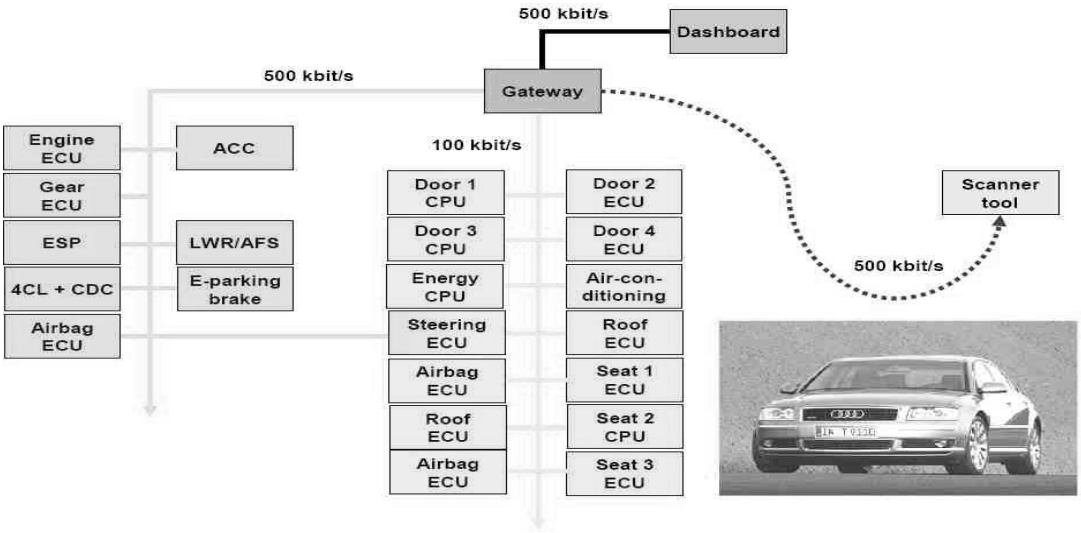
vii. CAN denetleyicisinin aynı ağ üzerinde kullanımı için quartz osilatör kullanılmalıdır.

Osilatör duyarlığı en yüksek olan işlemci diğer tüm düğümlere de osilatör duyarlığını sağlamalıdır. Seramik rezonatör sadece genişletilmiş CAN protokolündeki tüm düğümler için kullanılmalıdır.

3.20. CAN Protokolünün Uygulamaları

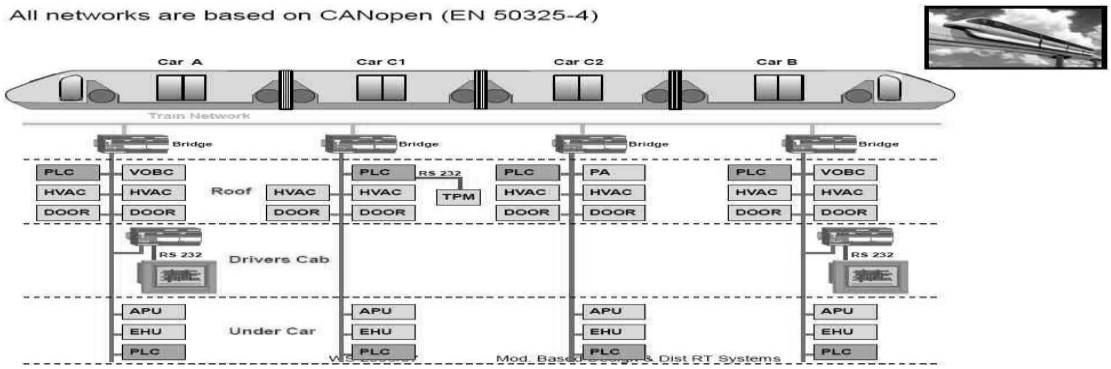
CAN Protokolünün otomobiller dışında, deniz hava, medikal, mekatronik alanlarında da uygulamaları vardır.

Şekil 3.24.'de bir otomobilin CAN protokolü ile yapılmış sistemi gösterilmektedir[16].

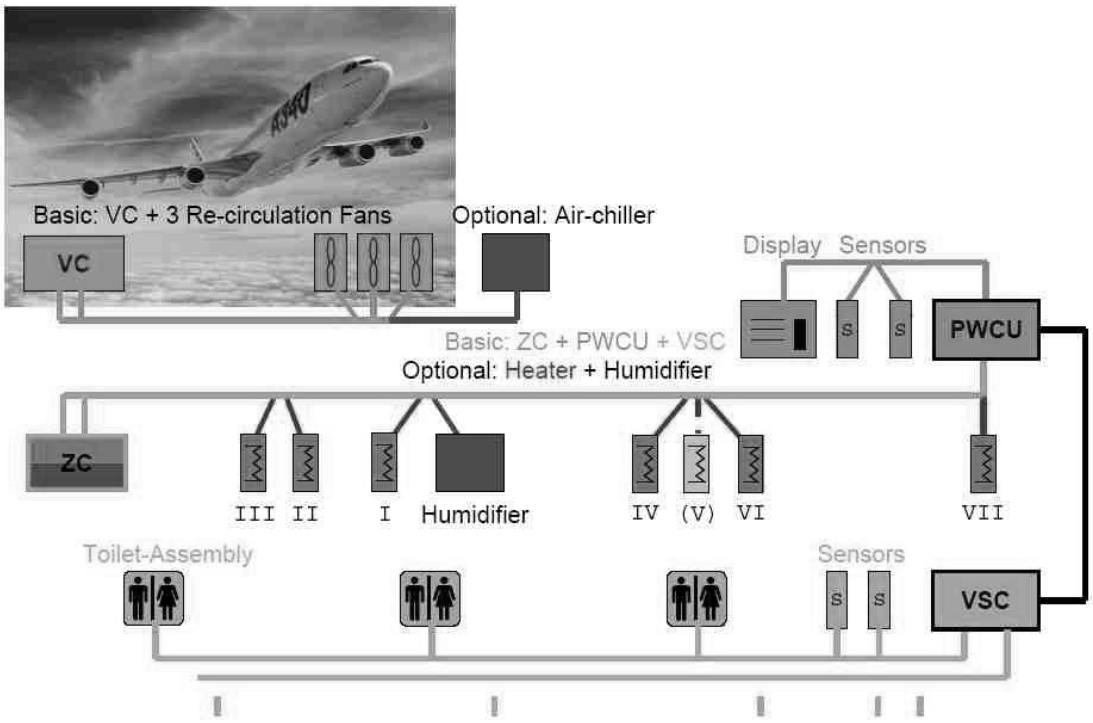


Şekil 3.24. Audi-A8 CAN Yapısı (Fuhrman [16]'dan değiştirilerek alınmıştır.)

Şekil 3.24.,25.,26.,27., 28.'de ise değişik CAN protokol uygulamaları gösterilmiştir.

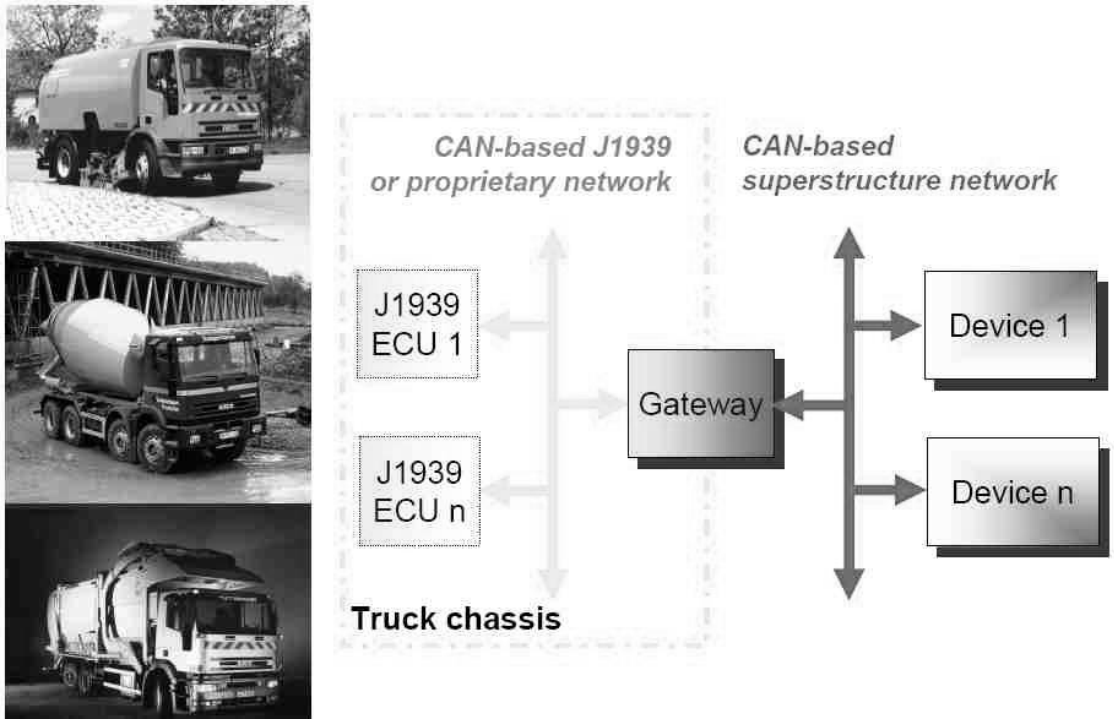


Şekil 3.25 Metronun CAN Yapısı (Fuhrman [16]'dan değiştirilerek alınmıştır.)

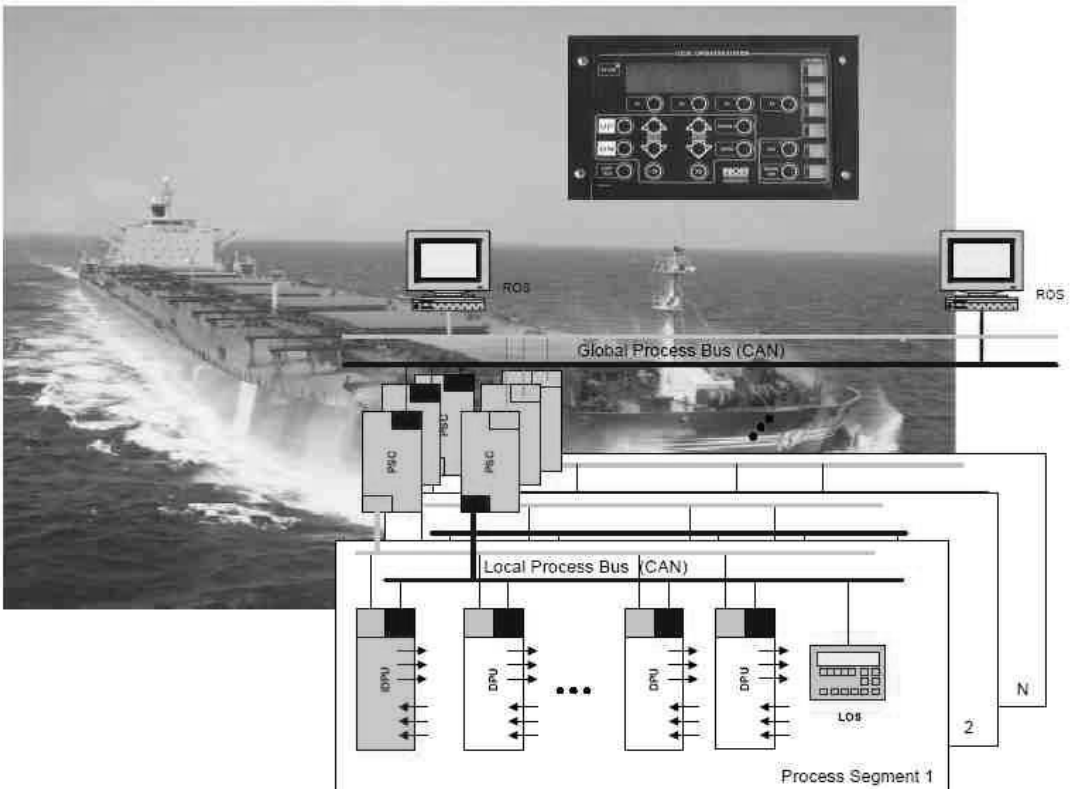


Şekil 3.26 Uçağın CAN Yapısı [16]

Kamyonlar için tasarlanmış bir CAN protokolü türevi olan J1939'un yapısı Şekil 3.25'de gösterilmiştir.



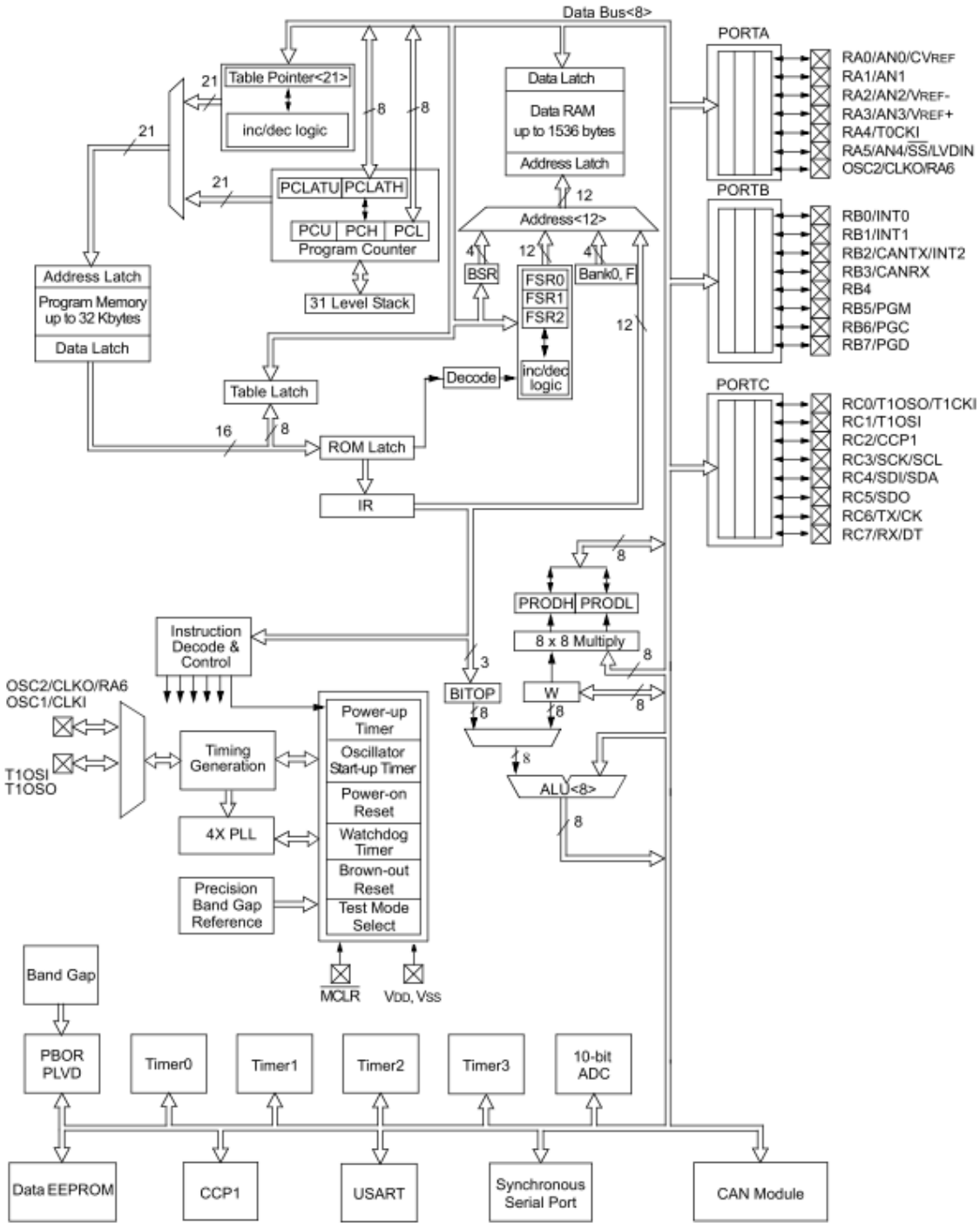
Şekil 3.27. J1939 Protokol Yapısı [16]



Şekil 3.28. Gemi İçin Tasarlanmış CAN Yapısı [16]

4. PIC18F458 CAN MODÜLÜ YAPISI

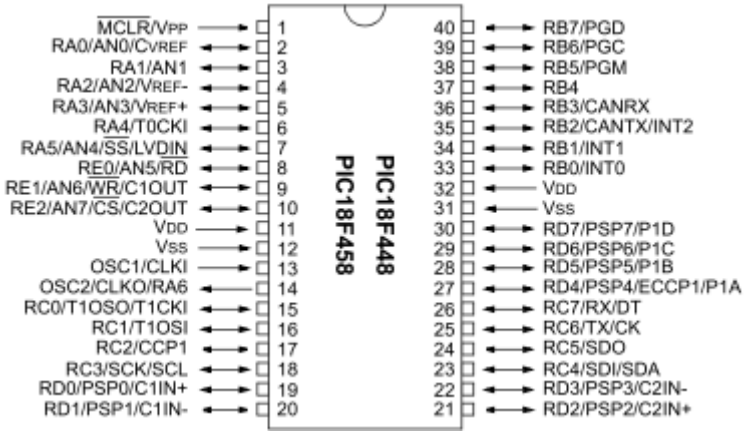
PIC18F458 mikrodenetleyicisi, 32Kbyte dahili program hafızasına sahiptir. Mikrodenetleyicinin 5 giriş/çıkış portu , 8 kanallı analog-dijital çeviricisi, CAN modülü ve diğer modülleri de vardır.



Şekil 4.1. PIC18F458 Mimari Yapısı [23]

CAN protokolü ile haberleşme de Atmel T89C51CC01 mikrodenetleyicisi de kullanılmaktadır. PIC18F458 mikrodenetleyicisinin CAN modülünün bulunması ve programlanması T89C51CC01 mikrodenetleyicisine göre kolay olduğundan tercih edilmiştir.

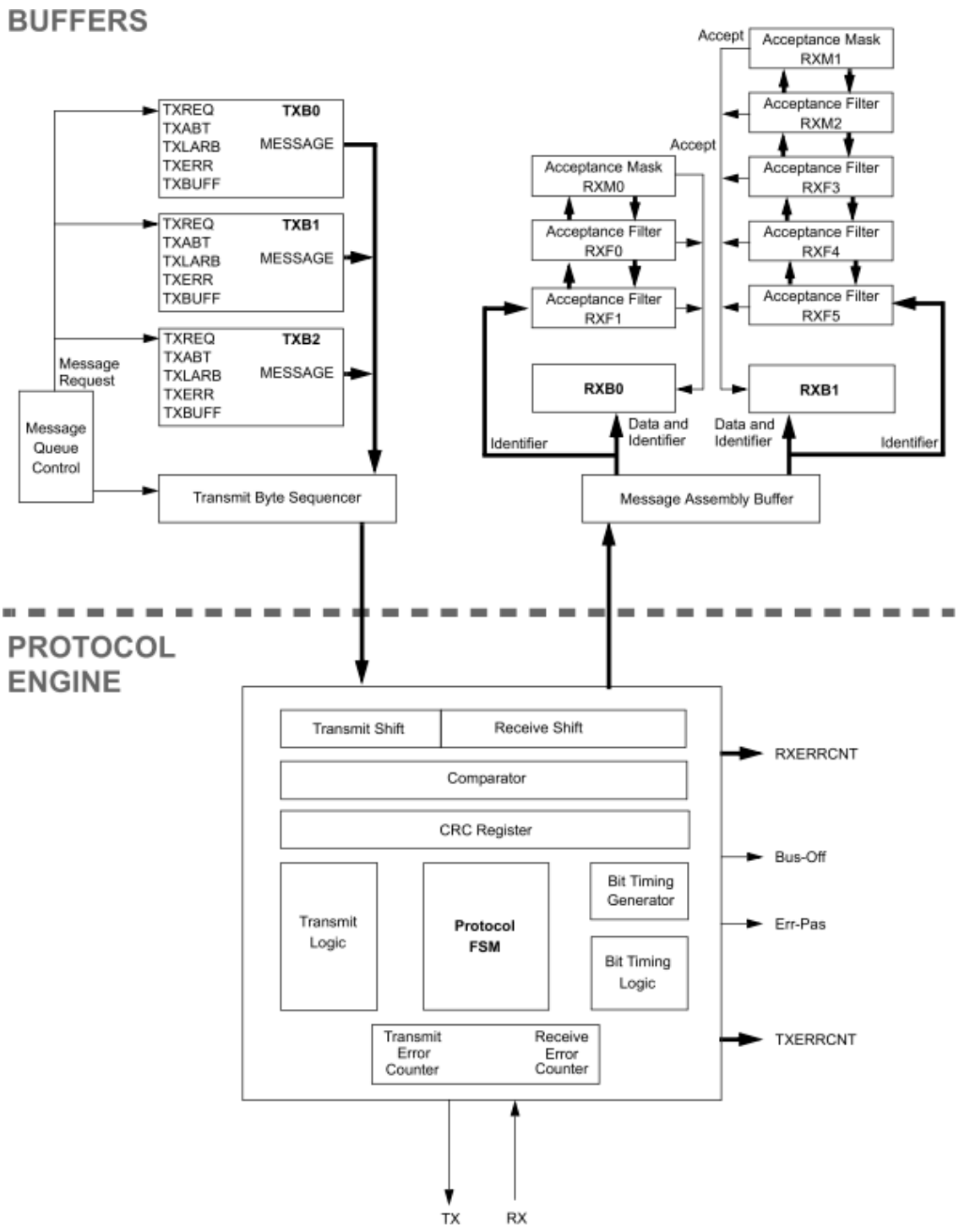
PIC18F458 entegresinin uçlarının diyagramı Şekil 4.2.'de gösterilmiştir. Mikrodenetleyicinin 35. ve 36. uçları CAN sinyal uçlarıdır. Mikrodenetleyicinin içindeki CAN modülü Port B üzerindedir. Port A Analog-Dijital Çevirici uygulamaları için kullanılmaktadır.



Şekil 4.2. PIC18F458 Uç Diyagramı [23]

4.1. CAN Modülü Özellikleri

Şekil 4.3.'de 18F458 mikrodenetleyicisinin CAN modülünün yapısı gösterilmiştir. PIC18F458 mikrodenetleyicisinin CAN modülü ISO CAN uyum testinden geçmiştir. CAN 2.0 B protokolünü destekler. 3 İletim tamponu ve 2 alıcı tamponu içermektedir. Ayarlanabilir saat kaynak bilgisi ve düşük güç uyku modunda çalışabilme özelliklerine sahiptir. CAN modülünde iletim ve alıcı tamponları dışında kalan kısım CAN protokolü mekanizması olarak adlandırılır. CAN protokolündeki hata tespiti ve diğer denetimler bu blok içerisindedir.



Şekil 4.3. PIC18F458 CAN Modülü Blok Diyagramı [23]

4.2. CAN Çalışma Modları

PIC18F458 mikrodenetleyicisinde 6 çalışma modu bulunmaktadır. Bu modlar ayar modu, kapalı mod, normal mod, izleme modu, geri çevrim modu ve hata tanıma modudur. Tüm modlar, hata tanıma modu da dahil CANCON kayıtçısında bulunan REQOP bitlerinin set edilmesiyle ayarlanır. CANCON kayıtçısında bit değerleri Çizelge 4.1.'deki gibidir.

Çizelge 4.1. CANCON Kayıtçısı (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	-
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

REQOP2:REQOP0 bitlerinin alacağı değerlere göre istenilen CAN modu değeri Çizelge 4.2.'deki gibi olacaktır.

Çizelge 4.2. CAN Çalışma Modları (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

REQOP2	REQOP1	REQOP0	MOD
1	X	X	Ayar Modu
0	1	1	İzleme Modu
0	1	0	Kapalı Çevrim Modu
0	0	1	Kapalı Mod
0	0	0	Normal Mod

CANCON kayıtçısının ABAT (Bekleyen tüm mesajları iptal et) biti veri yolu üzerinde iletişim için tamponlarda bekleyen tüm mesajları durdurmayı sağlar. ABAT bitinin reset edilmesi iletimin normal olmasını sağlar. CANCON kayıtçısında bulunan WIN bitleri kesme oluştuğunda mesajların doğru tamponlara gitmesi için ICODE2:ICODE0 bitlerinin kopyalandığı bitlerdir.PIC18F458 mikrodenetleyicisinin hangi modda çalıştığı CANSTAT kayıtçısının OPMODE bitlerinin onaylanmasıyla olur. Modların değişiminde, bekleyen tüm mesajların iletimleri sağlanmadığı sürece mod değişmeyecektir. Kullanıcı, işlemlerin gerçekleştiğini mod değişmeden doğrulamalıdır. Opmode bitleri CANSTAT kayıtçısının içerisinde yer almaktadır. Çizelge 4.3.'de CANSTAT kayıtçısının bitleri verilmiştir.

Çizelge 4.3. CANSTAT Kayıtçısı (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

OPMODE2	OPMODE1	OPMODE0	---	ICODE2	ICODE1	ICODE0	---
Bit 7	Bit 6	Bit 5	Bit4	Bit 3	Bit 2	Bit1	Bit0

OPMODE2	OPMODE1	OPMODE0	MOD
1	1	1	Kullanılmıyor
1	1	0	Kullanılmıyor
1	0	1	Kullanılmıyor
1	0	0	Ayar Modu
0	1	1	İzleme Modu
0	1	0	Kapalı Çevrim Modu
0	0	1	Kapalı Mod
0	0	0	Normal Mod

4.2.1. Ayar Modu

Mikrodenetleyicinin CAN modülünün çalıştırılmadan önce silinmesi gerekir. Bu ancak modülün, ayar modunda olduğunda gerçekleşebilir. Ayar Modu, CANCON kayıtçısının REQOP2 bitinin set edilmesiyle olur. CANSTAT kayıtçısı içindeki OPMODE2 bitinin set edilmesi ile çalışılan mod belirtilmiş olur. CAN modülünde, haberleşme için yapılan ayarlardan sonra REQOP denetim bitlerinin sıfıra set edilmesiyle modül etkinleştirilmiş olur. CAN protokolünün kullanıcı hatasından kaynaklanan programlama hatalarından etkilenmemesi için bu yöntem geliştirilmiştir. Ayar modunun tüm denetim kayıtçıları modül çalışırken değiştirilemez. CAN modülü iletimde iken ayar moduna geçemez. Ayar modunda, CAN modülü veri alımı veya iletimi yapamaz. Hata sayıcılar sıfırlanır ve kesme işaretçileri değişmez. Programcı diğer modlarda iken ayar kayıtçılarına ulaşabilecektir. Mikrodenetleyicinin CAN modülünün haberleşme hızını ayarlamaya yönelik 3 tane kayıtçısı vardır. Bunlar BRGCON1, BRGCON2, BRGCON3 kayıtçılarıdır. Baud Rate Control Kayıtçısı 1, (BRGCON1) senkronizasyon için sıçrama genişliğini ve baud oranı ön çarpım bitlerinin ayarlanmasını sağlar. BRGCON1 kayıtçısının bitleri Çizelge 4.4.'de gösterilmiştir.

Çizelge 4.4. BRGCON1 Kayıtçısı (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BRGCON1 kayıtçısındaki bitlerin açıklaması Çizelge 4.5.'de gösterilmiştir.

Çizelge 4.5. BRGCON1 Kayıtçısının Bitlerinin Açıklanması (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

bit 7-6	SJW1:SJW0: Senkronizasyon için sıçrama bit zamanı
	11 = Senkronizasyon için sıçrama genişliği bit zamanı = 4 x TQ
	10 = Senkronizasyon için sıçrama genişliği bit zamanı = 3 x TQ
	01 = Senkronizasyon için sıçrama genişliği bit zamanı = 2 x TQ
	00 = Senkronizasyon için sıçrama genişliği bit zamanı = 1 x TQ
bit 5-0	BRP5:BRP0: Baud oranı ön çarpan bitleri
	111111 = TQ = (2 x 64)/FOSC
	111110 = TQ = (2 x 63)/FOSC
	:
	:
	000001 = TQ = (2 x 2)/FOSC
	000000 = TQ = (2 x 1)/FOSC

Baud Oranı Denetim Kayıtçısı 2, CAN veri yolu üzerindeki örnekleme zamanını, faz bölümü 1 ve 2'nin zaman bitleri ile propagasyon zamanının ayarlanmasını sağlar. Çizelge 4.6'da BRGCON2 kayıtçısının bitleri ve Çizelge 4.7.'de ise açıklaması verilmiştir. BRGCON2 kayıtçısı CAN mesajının 1. Faz bölümü ve propagasyon gecikmesi süresinin değerleri taşır. Veri yolu üzerinde yapılacak örnekleme sayısı bilgisi de BRGCON2 kayıtçısında saklanır. BRGCON2 kayıtçısının 7. Biti SEG2PHTS biti ise BRGCON3 kayıtçısının 2. Faz bölümünün hesaplaması için kullanılır. BRGCON2 kayıtçısının 6. Biti SAM veri yolu üzerinde örnekleme sayısını ayarlamaya yarar. SAM bitinin değeri 0 olduğunda veri yolu üzerinde bir kez bitin değeri 1 olduğunda ise üç kez örnekleme yapılır. 1. Faz bölümünün ayarlanmasında BRGCON2 kayıtçısının 3.,4. ve 5. bitleri kullanılır.

Çizelge 4.6. BRGCON2 Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Çizelge 4.7. BRGCON2 Kayıtçısı Bitlerinin Açıklanması (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

Bit 7	SEG2PHTS: 2.Faz bölümünün zaman seçim bitleri
	1 = Serbestçe programlanabilir
	0 = Faz Bölümü 1 değeri veya Bilgi İşlem Zamanı (BİZ), hangisi büyükse
Bit 6	SAM: CAN veri yolu üzerinde örnekleme biti
	1 = Örnekleme noktasından önce 3 kez örnekleme
	0 = Örnekleme noktasında 1 kez örnekleme
Bit 5-3	SEG1PH2:SEG1PH0: Faz Bölümü 1 Bitleri
	111 = Faz Bölümü 1 Zamanı = 8 x TQ
	110 = Faz Bölümü 1 Zamanı = 7 x TQ
	101 = Faz Bölümü 1 Zamanı = 6 x TQ
	100 = Faz Bölümü 1 Zamanı = 5 x TQ
	011 = Faz Bölümü 1 Zamanı = 4 x TQ
	010 = Faz Bölümü 1 Zamanı = 3 x TQ
	001 = Faz Bölümü 1 Zamanı = 2 x TQ
	000 = Faz Bölümü 1 Zamanı = 1 x TQ
Bit 2-0	PRSEG2:PRSEG0: Propagasyon Zamanı seçim bitleri
	111 = Propagasyon Zamanı = 8 x TQ
	110 = Propagasyon Zamanı = 7 x TQ
	101 = Propagasyon Zamanı = 6 x TQ
	100 = Propagasyon Zamanı = 5 x TQ
	011 = Propagasyon Zamanı = 4 x TQ
	010 = Propagasyon Zamanı = 3 x TQ
	001 = Propagasyon Zamanı = 2 x TQ
	000 = Propagasyon Zamanı = 1 x TQ

BRGCON3 kayıtçısı BRGCON2 kayıtçısının 7. biti 0 iken 2.Faz Bölümünün zaman seçim bitlerini ayarlamaya ve veri yolu üzerinde mesaj alışverişi olduğunda hattı aktifleştirmek için filtrenin kullanılmasını sağlar. Çizelge 4.8.'de BRGCON3 kayıtçısı, Çizelge 4.9.'de BRGCON3 kayıtçısının bitleri açıklanmıştır. BRGCON3 kayıtçısının 6. Biti olan WAKFIL bitinin set edilmesiyle veri yolu üzerinde iletimin olmadığı durumda iletimi başlatır.

Çizelge 4.8. BRGCON3 Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

---	WAKFIL	---	---	---	SEG2PH2	SEG2PH1	SEG2PH0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

BRGCON3 kayıtçısında 3.,4. ve 5. bitler kullanılmamaktadır. Hesaplamalarda bu bitler sıfır olarak değerlendirilir.

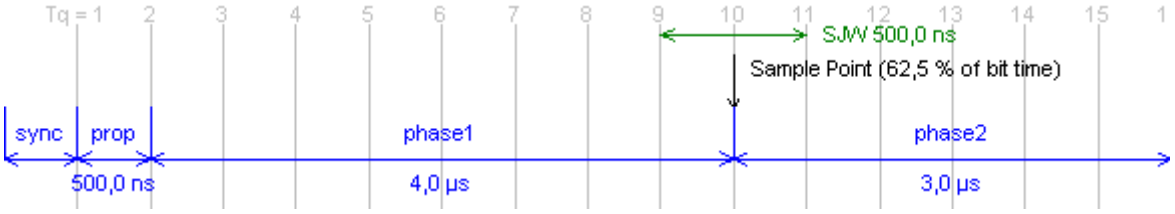
Çizelge 4.9. BRGCON3 Kayıtçısının Bitlerinin Açıklaması (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

Bit7	Kullanılmamaktadır. 0
Bit6	WAKFIL: Uyanma biti için hat filtresi
	1 = Uyanma bitiiçin hat filtresi kullan
	0 = Uyanma biti için hat filtresi kullanma
Bit5-3	Kullanılmamaktadır. 0
	SEG2PH2:SEG2PH0: 2.Faz Bölümünün zaman seçim bitleri
	111 = 2.Faz Bölümünün Zamanı = 8 x TQ
	110 = 2.Faz Bölümünün Zamanı = 7 x TQ
	101 = 2.Faz Bölümünün Zamanı = 6 x TQ
	100 = 2.Faz Bölümünün Zamanı = 5 x TQ
	011 = 2.Faz Bölümünün Zamanı = 4 x TQ
	010 = 2.Faz Bölümünün Zamanı = 3 x TQ
	001 = 2.Faz Bölümünün Zamanı = 2 x TQ
	000 = 2.Faz Bölümünün Zamanı = 1 x TQ

Mikrodenetleyicinin haberleşme hızının ayarlanmasında denklem (4.1.) kullanılır.

$$T_Q \ \mu s \ = \ 2\times \ BRP+1 \ \ / F_{OSC} \ \ MHz \qquad (4.1.)$$

(4.1.) denklemindeki değişkenlerle sistemin haberleşme hızı ayarlanır. BRGCON kayıtçıları bu değişkenlerin aldıkları değerleri kodlar. Mikrodenetleyicinin CAN modülünün 8 Mhz, 125 kHz, 16 Tq zamanı için BRGCON1=01h, BRGCON2=B8h, BRGCON3=05h değerlerini alır. Bu değerlerin temsil ettiği bit zamanı Şekil 4.4.'de verilmiştir.



Şekil 4.4. Bit Zamanı Bölümleri (Microchip Can Bit Timing Calculator programından alınmıştır.)

4.2.2. Kapalı Modu

Kapalı modda, CAN modülünde mesaj iletimi ve alınması gerçekleşmeyecektir. CANCON kayıtçısı ile kapalı moda geçilir. Kapalı modda PIR3 kayıtçısının 6.Biti olan WAKIF (Wake-up Interrupt Flag) ile veri yolunda bekleyen mesaj olup olmadığı belirlenebilir. WAKIF biti sadece kapalı modda geçerlidir. Çizelge 4.10.'da PIR3 kayıtçısının bitleri verilmiştir. CAN uygulamamızda kullandığımız diğer kesme kayıtçısı da INTCON kayıtçısıdır. Çizelge 4.11.'de INTCON kayıtçısının bitleri verilmiştir.

Çizelge 4.10. PIR3 Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Çizelge 4.11. INTCON Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

INTCON kayıtçısının 6. biti olan PEIE/GIEL biti de "0" olur. Bu bit reset denetim kayıtçısı RCON'un 7. biti olan kesme öncelik biti IPEN "1" iken mikrodenetleyiciye bağlı çevresel cihazlarda kesme oluşmamasını sağlar. INTCON kayıtçısının 7. biti GIE/GIEH, RCON'un 7. biti "1" durumunda iken yüksek öncelikli kesmeleri etkinleştirir. Mikrodenetleyicinin bir diğer önemli modülü ise karşılaştırıcı modülüdür. Mikrodenetleyicinin RD0, RD1, RD2, RD3 uçlarındaki analog girişlerin karşılaştırılmasını sağlar. Karşılaştırıcı modülünün ayarlanması CMCON kayıtçısı ile yapılır. Çizelge 4.12.'de CMCON kayıtçısının bitleri verilmiştir.

Çizelge 4.12. CMCON Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

CMCON kayıtçısında CM2:CM0 bitleri "1" değerini aldığında mikrodenetleyicinin karşılaştırıcı modülü devre dışı kalmaktadır. PIC18F458 mikrodenetleyicisinin analog dijital dönüştürücü modülünün 8 girişi vardır. ADC modülünün ADCON0 ve ADCON1 kayıtçıları analog dijital dönüştürücünün ayarlanmasını sağlar. Karşılaştırıcı modülü kayıtçısının CMCON ayarlanmasında ADCON1 kayıtçısının da ayarlanması gerekir. Çizelge 4.13.'de ADCON1 kayıtçısının bitleri verilmiştir. PCFG port ayar denetim bitleri analog dijital dönüştürücü modülünü ayarlar.

Çizelge 4.13. ADCON1 Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

ADFM	ADCS2	---	---	PCFG3	PCFG2	PCFG1	PCFG0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

4.2.3. Normal Mod

Mikrodenetleyicinin standart işlem modudur. Bu modda cihaz veri yolu üzerindeki tüm mesajları gözlemler. ACK bitleri, hata çerçeveleri gibi çerçeveler bu modda oluşturulur.

4.2.4. Dinleme Modu

Dinleme modunda PIC18F458 veri yolu üzerindeki tüm mesajları alır Mesajlar hatalı olsa bile dinleme modunda mikrodenetleyici tarafından alınır. Dinleme modu, veri yolu üzerindeki trafiği gözlemek için kullanılır. Otomatik haberleşeme baudunun belirlenmesi için en az iki düğümün haberleşmesi gerekir. Baud-rate oranı geçerli mesaj iletildiğinde basit deneysel yöntemlerle hesaplanabilir. Dinleme modu sessiz moddur. Yani bu modda mesaj iletimi olmaz. Buna hata işaretçileri ve ACK sinyalleri dâhildir. Filtreler ve filtre maskeleri alıcı kayıtçılarına yalnızca parça mesajların yüklenmesine izin verir veya filtre mesajları bir mesajın belirleyicisinin iletilmesi için hepsi sıfıra set edilir. Hata sayaçları resetlenir ve bu modda aktif olmaz.

4.2.5. Geri Çevrim Modu

Geri çevrim modu veri yolu üzerinde mesaj iletimi olmadan iletim tamponları ile alım tamponları arasında iç mesajın iletilmesini içeren moddur. Bu mod sistem geliştirme ve sistemin testinde kullanılır. Bu moda ACK biti ihmal edilir ve cihaz kendi kendine iletim yapar. Geri çevrim modu sessiz mod olarak adlandırılır. Veri yolu üzerinde mesaj iletilmez. TXCAN ucu I/O girişini bu moda eski haline getirecektir.

4.2.6. Hata Tanıma Modu

Hata tanıma modu veri yolu üzerinde oluşan hataları ihmal eder ve tüm mesajların alınmasını sağlar. Hata tanıma modu RXBnCON kayıtçısında RXM<1:0> bitleri olunca set edilir. Çizelge 4.14.'de RXB0CON kayıtçısının bitleri Çizelge 4.15.'de RXM0 bitlerinin açıklaması verilmiştir.

Çizelge 4.14. RXB0CON Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

RXFUL	RXM1	RXM0	---	RXRTRRO	RXB0DBEN	JTOFF	FILHIT
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bu modda, veri yolu üzerindeki mesajlar geçerli veya geçersiz olsun alıcı tamponuna kopyalanır. Alarm sistemi tasarımında RXM0 bitleri hatalı mesaj iletimi yapan mikrodenetleyicilerin mesajlarının alınmasında kullanılır. Standart veya genişletilmiş CAN mesajları alıcı tamponları tarafından alınmasını sağlar.

Çizelge 4.15. RXM0 Bitlerinin Açıklaması (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

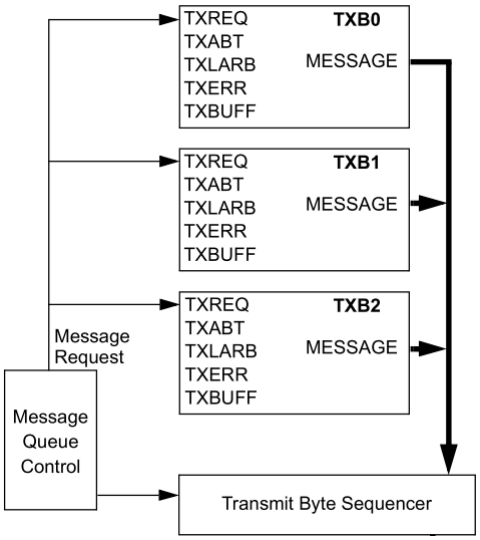
RXM1:RXM0	Alıcı Tamponunun Mod Bitleri
11	Tüm mesajları al (Hatalar dahil)
10	Sadece geçerli genişletilmiş belirleyicisi olan mesajları al
01	Sadece geçerli standart belirleyicisi olan mesajları al
00	Tüm mesajları al

4.3.CAN Protokolünde Mesaj İletimi

CAN protokolünde iletim tamponu mesajın iletilmesi ve denetlenmesini sağlar. Mesajların iletimde sıralanması TXB0CON kayıtçısının TXREQ bitiyle ayarlanır.

4.3.1. İletim Tamponu

PIC18F458 mikrodenetleyicisi, 3 iletim tamponu içerir. Şekil 4.5.'de İletim tamponlarının blok diyagramı gösterilmiştir. Bu tamponlar SRAM'da 14 bayt yer kaplar ve entegrenin hafıza haritasına işlenir. CAN modülünde veri iletiminin denetlendiği TXBnCON kayıtçısı önemlidir. Çizelge 4.16.'de TXBnCON kayıtçısının bitleri gösterilmiştir. Mikrodenetleyicinin mesaj tamponuna yazmak için TXREQ biti resetlenmelidir. İletim için en azından TXBnSIDH, TXBnSIDL ve TXBnDLC kayıtçılarının yüklenmesi gereklidir. Eğer veri baytları mesajda bulunursa, TXBnDm kayıtçıları da yüklenmelidir.



Şekil 4.5. İletim Tamponlarının Blok Diyagramı (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

4.3.2. İletim Önceliği

CAN protokolünde mesajların iletiminde öncelik kavramı tanımlanmıştır. Yüksek önceliğine sahip mesajlar önce iletilir. TXBnCON kayıtçısının TXPRI bitleri öncelik değerlerinin ayarlanmasını sağlar. Her tampon için 4 öncelik seviyesi vardır. Programlama da kolaylık olması amacıyla TXPRI bitlerinin 11 değerini taşıması yüksek öncelik olarak tanımlanmıştır.

Çizelge 4.16. TXBnCON Kayıtçısının Bitleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

---	TXABT	TXLARB	TXERR	TXREQ	---	TXPRI1	TXPRI0
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

TXPRI bitlerinin alacağı değerlere göre öncelik seviyeleri Çizelge 4.17.'de belirtilmiştir.

Çizelge 4.17. İletim Öncelik Seviyeleri (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

TXPRI1:TXPRI0	İletim Önceliği Bitleri
11	Öncelik seviyesi 3 (Yüksek Öncelik)
10	Öncelik seviyesi 2
01	Öncelik seviyesi 1
00	Öncelik seviyesi 0 (Düşük Öncelik)

TXABT biti set edilince mesajın iptal edildiği anlamına gelmektedir.

4.3.3. İletimin Başlaması

Mesaj iletiminin başlamasında, TXREQ biti iletilecek mesajda her tampon için set edilmelidir. TXREQ biti set edildiğinde TXABT, TXLARB ve TXERR bitleri clear durumunda olacaktır. İletim başarılı şekilde sonuçlandığında TXREQ biti clear edilecektir. TXBnIF biti set olup eğer TXBnIE biti de set durumundaysa kesme oluşacaktır. Eğer mesaj iletimi başarısız olursa, TXREQ set durumunda kalacaktır. Mesaj daha sonra iletilmek üzere tamponda kalacaktır. Eğer mesaj iletimi başlar ve hatalar devam ederse TXERR ve IRXIF bitleri set olacak ve kesme oluşacaktır. Eğer mesaj tahkim esnasında kaybedilirse TXLARB biti set edilecektir.

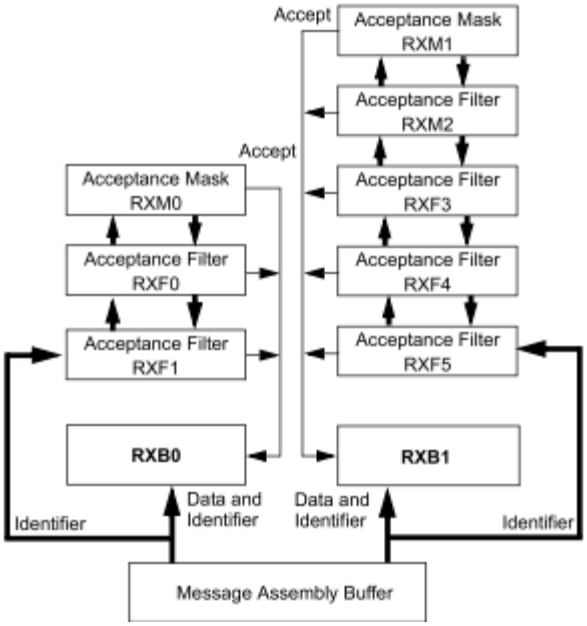
4.3.4. İletimin Durdurulması

Mikrodenetleyici bir mesajın iptal isteğini TXREQ bitine göre siler. CANCON kayıtçısının 4. biti ABAT bekleyen tüm mesajların iptali için istek gönderir. Eğer mesaj henüz iletime başlamamış veya mesaj başlamış fakat tahkimde veya bir hatadan dolayı kesilmişse, iptal işlemi devam edecektir. İptal, modülün ilgili tamponunun TXABT bitlerinin set edilmesiyle işaretlenir. Entegre TXREQ bitini reset ederek mesaj iletimini durdurur. ABAT

bitinin set edilmesi bekleyen tüm mesajlar için iptal isteği gönderilir. Eğer mesaj iletimi henüz başlamamış veya mesaj başlamış fakat tahkim esnasında oluşan kayıptan dolayı iletim durmuş olsa bile, iptal işlemi devam eder. İptal işlemi TXBnCON entegresinde TXABT biti ile yapılır.

4.4. Mesaj Alımı

PIC18F458'te 2 tane alıcı tamponu vardır. Buna ilaveten Mesaj Birleştirme Tamponu da vardır. Bu tamponlar için onay filtreleri tanımlanmıştır. Şekil 4.6.'da alıcı tamponlarının blok diyagramı gösterilmiştir.



Şekil 4.6. Alıcı Tamponlarının Blok Diyagramı (PIC18F458 Data Sheet [23]'ten değiştirilerek alınmıştır.)

Mesaj Birleştirme Tamponu, her zaman veri yolu üzerindeki son mesajı alır. RXB0 ve RXB1 tamponları, CAN modülünde protokolden tam mesajı alır.

5. CAN MODÜLÜNÜN ŞARTLANMASI

Microchip firması, 18XXX8 ailesi mikrodenetleyicilerin CAN modülünün programlanması için özel fonksiyonlar tanımlamıştır [24]. Çizelge 5.1.'de bu fonksiyonlar listesi bulunmaktadır. Fonksiyonlar ayar, modül ve durum denetim işlemleri olarak sınıflandırılmıştır.

Çizelge 5.1. CAN Fonksiyonları (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

FONKSİYON ADI	TÜR
CANInitialize	Ayarlama/Silme İşlemi
CANSetOperationMode	Ayarlama/Silme İşlemi
CANSetOperationModeNoWait	Ayarlama/Silme İşlemi
CANSetBaudRate	Ayarlama/Silme İşlemi
CANSetReg	Ayarlama/Silme İşlemi
CANSendMessage	Modül İşlemi
CANReadMessage	Modül İşlemi
CANAbortAll	Modül İşlemi
CANGetTxErrorCount	Durum Denetim İşlemi
CANGetRxErrorCount	Durum Denetim İşlemi
CANIsBusOff	Durum Denetim İşlemi
CANIsTxPassive	Durum Denetim İşlemi
CANIsRxPassive	Durum Denetim İşlemi
CANIsRxReady	Durum Denetim İşlemi
CANIsTxReady	Durum Denetim İşlemi

5.1. Ayar Fonksiyonları

5.1.1. CANInitialize Fonksiyonu

Mikrodenetleyicinin haberleşme hızını, iletilecek mesajın türünü, hat filtresinin durumunu belirleyen fonksiyondur. Çizelge 5.2.'de CANInitialize fonksiyonunun içerdiği değerler gösterilmiştir. CANIinitialize fonksiyonu mikrodenetleyicinin CAN modülü ayar

modunda iken gerçekleşir. Ayar modu haricindeki modlarda CANInitialize fonksiyonu ile mikrodenetleyici iletişim için şartlanamaz.

Çizelge 5.2. CANInitialize Fonksiyonu (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

DEĞER	ANLAMI	BİT
CAN_CONFIG_DEFAULTS	Standart ayarlar	1
CAN_CONFIG_PHSEG2_PRG_ON	PHSEG2 değeri kullan	1
CAN_CONFIG_PHSEG2_PRG_OFF	Maksimum PHSEG1 veya Bilgi İşlem Zamanı değerinden hangisi daha büyükse onu kullan	1
CAN_CONFIG_LINE_FILTER_ON	Hat filtresi kullan	1
CAN_CONFIG_LINE_FILTER_OFF	Hat filtresini kullanma	1
CAN_CONFIG_SAMPLE_ONCE	Veri yolu üzerinde bir kez örnekleme kullan	1
CAN_CONFIG_SAMPLE_THRICE	Veri yolu üzerinde üç kez örnekleme kullan	1
CAN_CONFIG_ALL_MSG	Geçersizde olsa tüm mesajları al	2
CAN_CONFIG_VALID_XTD_MSG	Sadece geçerli genişletilmiş mesajları al	2
CAN_CONFIG_VALID_STD_MSG	Sadece geçerli standart mesajları al	2
CAN_CONFIG_ALL_VALID_MSG	Tüm geçerli mesajları al	2

CANInitialize fonksiyonunun kullanımı şöyledir.

CANInitialize SJW, BRP, PHSEG1, PHSEG2, PROPSEG, İşaretçiler

CANInitialze fonksiyonunda işaretçilerin değeri Çizelge 5.2.'de verilen ifadelerdir. CAN modülün ayarlamasında fonksiyonun işaretçilerinin arasına "&" konarak birden fazla işaretçi eklenir. 20 Mhz, 125 Kbps hızında çalışan ve geçerli tüm mesajların alındığı CAN modülü için fonksiyonun alacağı değeri şu şekilde yazarız.

CANInitialize 1,5,6,7,2, CAN_CONFIG_ALL_VALID_MSG

Örnekleme sayısının üç kez kullanıldığı ve sadece standart mesajların alınmasının istenmesinde CANInitialize fonksiyonu şu şekilde yazılır. CANInitialize 1,5,6,7,2, CAN_CONFIG_VALID_STD_MSG & CAN_CONFIG_SAMPLE_THRICE

5.1.2. CANSetOperationMode Fonksiyonu

Entegrenin çalıştığı modu değiştirmeye yarayan fonksiyondur. Entegre aynı anda birden fazla modda çalışamaz. Çizelge 5.3.'de CANSetOperationMode fonksiyonunun aldığı değerler verilmiştir.

Çizelge 5.3. CANSetOperationMode Fonksiyonu Açıklaması (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

DEĞER	ANLAM
CAN_OP_MODE_NORMAL	Normal İşlem Modu
CAN_OP_MODE_SLEEP	Uyku Modu
CAN_OP_MODE_LOOP	Geri Çevrim Modu
CAN_OP_MODE_LISTEN	Dinleme Modu
CAN_OP_MODE_CONFIG	Ayar Modu

Mikrobasic programında bu fonksiyonun kullanımı şu şekildedir.

```
CANSetOperationMode  CAN_OP_MODE_NORMAL
```

5.1.3. CANSetBaudRate Fonksiyonu

CAN modülünün istenilen hızda iletim yapmasını sağlamaya yarar. Kullanımı CANSetBaudRate SJW, BRP, PHSEG1, PHSEG2, PROPSEG, Flags şeklindedir. 20 Mhz, 125 kbps hızında hat filtresi olan, faz 2 kısmı kapalı bir kez örneklemeli iletişim için gereken değer şu şekildedir.

```
CANSetBaudRate 1,5,7,6,2,
                    CAN_CONFIG_SAMPLE_ONCE&
                    CAN_CONFIG_PHSEG2_PRG_OFF&
                    CAN_CONFIG_LINE_FILTER_ON
```

5.1.4. CANSetReg Fonksiyonu

CAN modülünde maske ve filtre değerlerinin ayarlanmasını sağlar. Bu fonksiyon mikrodenetleyici yazılımı geliştiren bazı firmalar tarafından farklı isimle de tanımlanmıştır. Tezde kullandığımız MikroBasic programlama dilinde CANSetReg fonksiyonunun içerdiği değerler CANSetMask ve CANSetFiltre olarak kullanılmaktadır.

Microchip tarafından tanımlanan CANSetReg fonksiyonundaki değerler ve anlamları Çizelge 5.4.'de belirtilmiştir.

Çizelge 5.4.CANSetReg Fonksiyonunun Değerleri (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

DEĞER	ANLAMI
CAN_MASK_B1	1. Alıcı Tamponunun Maske Değeri
CAN_MASK_B2	2. Alıcı Tamponunun Maske Değeri
CAN_FILTER_B1_F1	1. Alıcı Tamponunun, 1. Filtre Değeri
CAN_FILTER_B1_F2	1. Alıcı Tamponunun, 2. Filtre Değeri
CAN_FILTER_B2_F1	2. Alıcı Tamponunun, 1. Filtre Değeri
CAN_FILTER_B2_F2	2. Alıcı Tamponunun, 2. Filtre Değeri
CAN_FILTER_B2_F3	2. Alıcı Tamponunun, 3. Filtre Değeri
CAN_FILTER_B2_F4	2. Alıcı Tamponunun, 4. Filtre Değeri

1. Alıcı tamponunun genişletilmiş mesajları almak için ayarlanması için CANSetMask(CAN_MASK_B1,-1,CAN_CONFIG_XTD_MSG) ifadesi yazılır. 1. Alıcı tamponunun 1.filtre değerinin genişletilmiş mesajlar için 1000 filtre değerini alması için CANSetFilter(CAN_FILTER_B1_F1,1000,CAN_CONFIG_XTD_MSG) ifadesi kullanılır.

5.2. Modül İşlemleri Fonksiyonları

Mikrodenetleyicinin CAN modülünde, mesaj iletim ve okuma modül işlem fonksiyonları olarak adlandırılır.

5.2.1. CANSendMessage Fonksiyonu

Mesajı boş durumda olan iletim tamponuna kopyalayarak mesaj iletimini sağlar. CANSendMessage fonksiyonun içerdiği değerler Çizelge 5.5.'de gösterilmiştir. İletilmek istenen mesajın biçimi bu fonksiyon aracılığıyla ayarlanır. Mesajın uzak çerçeve biti içerip içermediği CANSendMessage fonksiyonunda tanımlanır. Fonksiyonun kullanımı şöyledir. CANSendMessage Belirleyici, Veri, Veri uzunluğu, İşaretçiler

Fonksiyonun işaretçiler kısmında taşıdığı değerler Çizelge 5.5.'deki değerlerdir. Bu değerler ile mesajın özellikleri belirtilir ve mesaj iletilir.

Çizelge 5.5. CANSendMessage Fonksiyonu (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

DEĞER	ANLAMI
CAN_TX_STD_FRAME	Standart belirleyiciye sahip mesaj
CAN_TX_XTD_FRAME	Genişletilmiş belirleyiciye sahip mesaj
CAN_TX_NO_RTR_FRAME	RTR biti olmayan mesaj
CAN_TX_RTR_FRAME	Uzak çerçeve içeren mesaj

CANSendMessage kullanımında 0x20h belirleyiciye (ID) sahip 2 bayt uzunluğunda standart çerçeve ve uzak çerçeve biti içermeyen mesaj gönderimi için CANSendMessage 0x20, MessageData, 2, CAN_TX_STD_FRAME & CAN_TX_NO_RTR_FRAME yazılır.

5.2.2. CANReadMessage Fonksiyonu

Veri yolu üzerindeki mesajın kullanıcı tarafından uygun tampona kopyalanmasını sağlar. CANReadMessage fonksiyonunun içerdiği değerler ve anlamları Çizelge 5.6.'da gösterilmiştir.

Çizelge 5.6. CANReadMessage Fonksiyonu (PIC18XXX8 CAN Driver with Prioritized Transmit Buffer Data Sheet [24]'den değiştirilerek alınmıştır.)

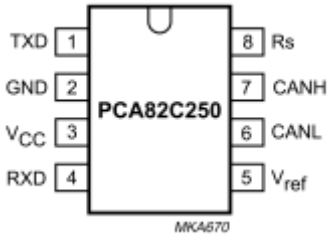
DEĞER	ANLAMI
CAN_RX_FILTER_1, CAN_RX_FILTER_2, CAN_RX_FILTER_3, CAN_RX_FILTER_4, CAN_RX_FILTER_5, CAN_RX_FILTER_6	Mesajın hangi filtre değerine göre alınacağını belirler.
CAN_RX_OVERFLOW	Alıcı tamponunda taşma durumu
CAN_RX_INVALID_MSG	Geçersiz mesaj
CAN_RX_XTD_FRAME	Genişletilmiş çerçeve
CAN_TX_RTR_FRAME	Uzak çerçeve içeren mesaj
CAN_RX_DBL_BUFFERED	İki tampon kullanılarak yazılan mesaj

6. ALARM SİSTEMİNDE KULLANILAN ENTEGRE ve DÖNÜŞTÜRÜCÜ MODÜLÜ

Alarm sisteminde kullandığımız CAN denetleyici ara yüz entegreleri PCA82C250 ile MCP2551 entegreleridir.

6.1. PCA82C250 CAN Denetleyici Ara Yüz Entegresi

Philips firması tarafından üretilen PCA82C250 entegresi CAN uygulamalarında sıklıkla kullanılan ara yüz entegresidir. Veri yolu üzerindeki sinyalleri yükselterek CAN denetleyicisine iletir. Elektromanyetik etkileşimi az olduğundan otomotiv sistemlerinde tercih edilir. 1 Mbps iletişimi desteklemesi ve radyo frekanslarının etkileşiminin denetlenmesi önemli özelliklerindendir.

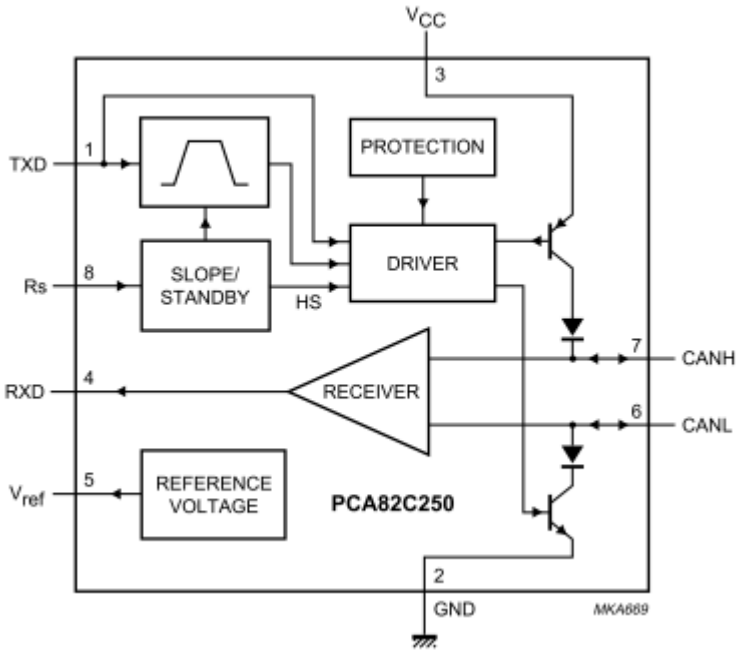


Şekil 6.1 PCA82C250 Entegre Uç Diyagramı [25]

Çizelge 6.1.'de 82C250 entegresinin uçlarının açıklaması, Şekil 6.2.'de ise blok diyagramı verilmiştir.

Çizelge 6.1. PCA82C250 Uç Diyagramı (PCA82C250CAN Controller Interface Data Sheet [25]'ten değiştirilerek alınmıştır.)

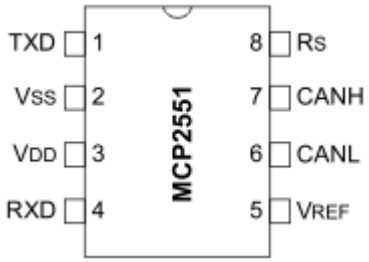
SEMBOL	UÇ	AÇIKLAMA
TXD	1	İletim veri girişi
GND	2	Toprak
Vcc	3	Besleme gerilimi
RXD	4	Alıcı veri çıkışı
Vref	5	Çıkış referans gerilimi
CANL	6	Düşük seviye CAN gerilimi giriş/çıkış
CANH	7	Yüksek seviye CAN gerilimi giriş/çıkış
Rs	8	Eğim direnci



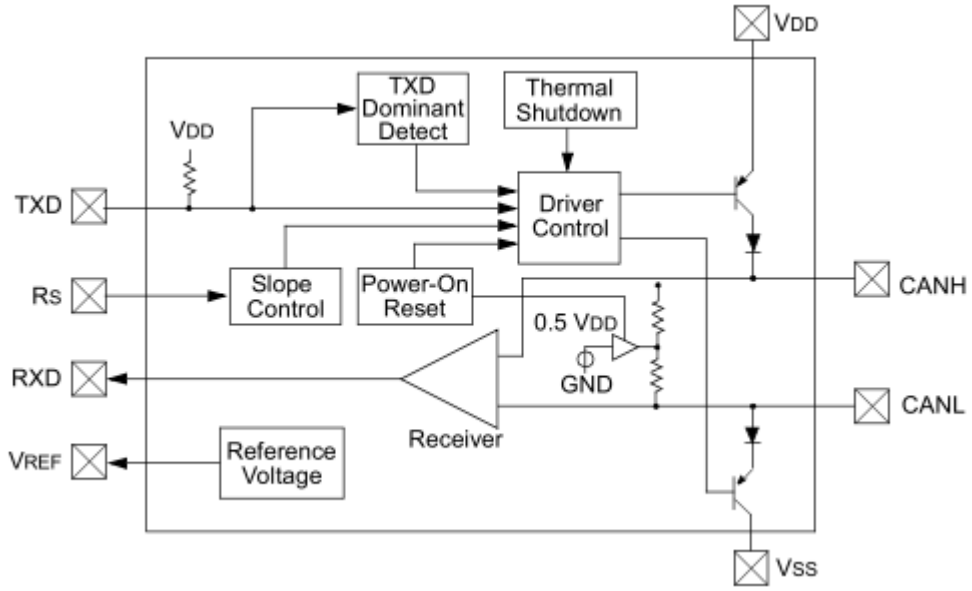
Şekil 6.2. PCA82C250 Blok Diyagramı [25]

6.2. MCP2551 Yüksek Hızlı CAN Ara Yüz Entegresi

Microchip firmasının MCP2551 entegresi 82C250 entegresi ile aynı özelliklere sahiptir. 12 ve 24 Voltluk sistemlerle uyumlu çalışma özelliği vardır. MCP 2551'e 112 düğüm bağlanabilir. Şekil 6.3.'de uç diyagramı, Şekil 6.4'de ise blok diyagramı gösterilmiştir. Entegrenin uçlarının sıralaması ve açıklaması 82C250 entegresi ile aynıdır ve Çizelge 6.1.'de verilmiştir.



Şekil 6.3. MCP2551 Uç Diyagramı [26]



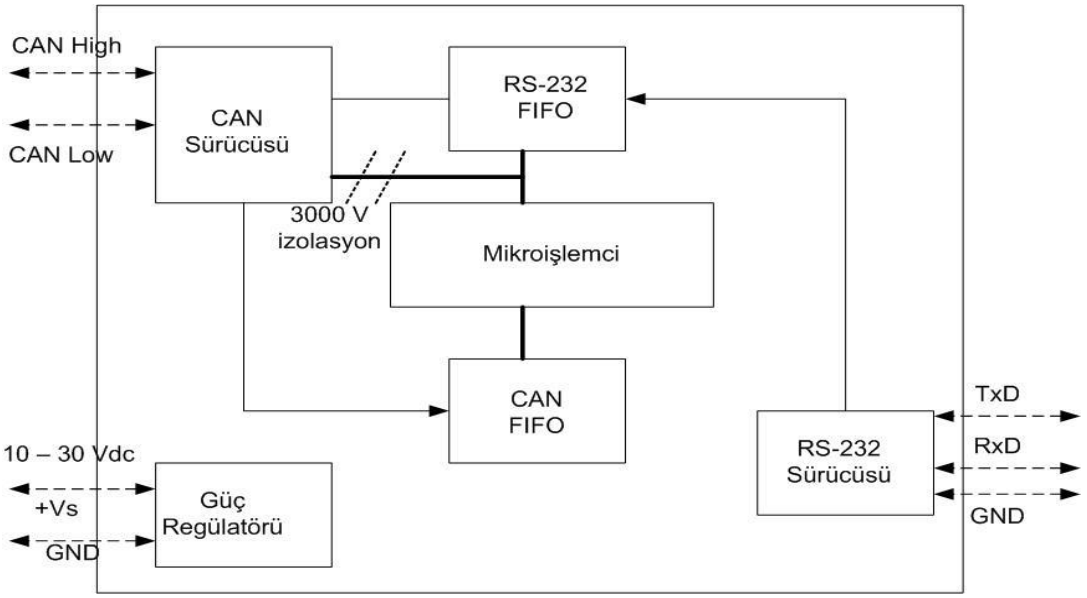
Şekil 6.4. MCP 2551 Blok Diyagramı [26]

6.3. RS232-CAN Dönüştücü Modülü

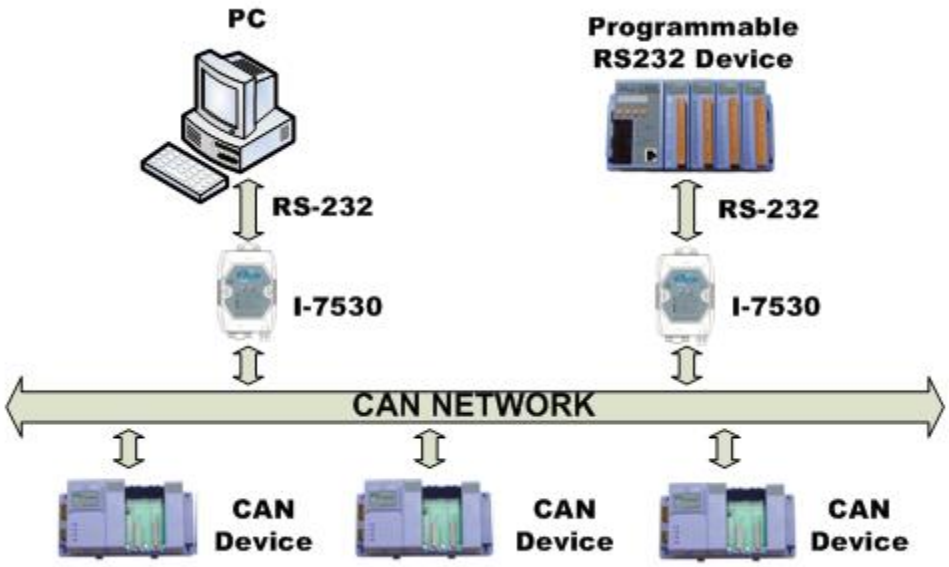
Tezde kullandığımız ICPDAS 7530G RS232-CAN dönüştürücüsü Şekil 6.5.'de gösterilmiştir. Maksimum 1 Mbps iletim hızında olan CAN dönüştürücüsü +10V - +30 V gerilimleri arasında çalışabilen içerisinde SJA1000 CAN mikrodenetleyicisi ile 82C250 ara yüz entegresi olan modüldür. Modül en fazla 1000 metreye kadar veri iletişimini desteklemektedir.

Şekil 6.5. ICPDAS I-7530G RS232-CAN Dönüştürücü Modülü [27]

RS232-CAN modülünün blok diyagramı Şekil 6.6.'da gösterilmiştir [27]. RS232 sürücü olarak gösterilen blok içerisinde MAX232C entegresi bulunmaktadır. Mikrodenetleyici SJA1000 entegresi RS232 biçiminde gelen sinyalleri CAN High ve CAN Low olarak dönüştürür ve modülün CAN ucundan bu sinyalleri veri yoluna iletir. Şekil 6.7.'de ICPDAS I-7530G modülünün CAN ağında örnek kullanımı verilmiştir [27]. PC, CAN ağının kontrolü, gözlemi veya CAN işlemcilerine erişimi için kullanılır. Programlanabilir RS-232 cihazları, CAN modülü sayesinde CAN ağını kontrol eder.



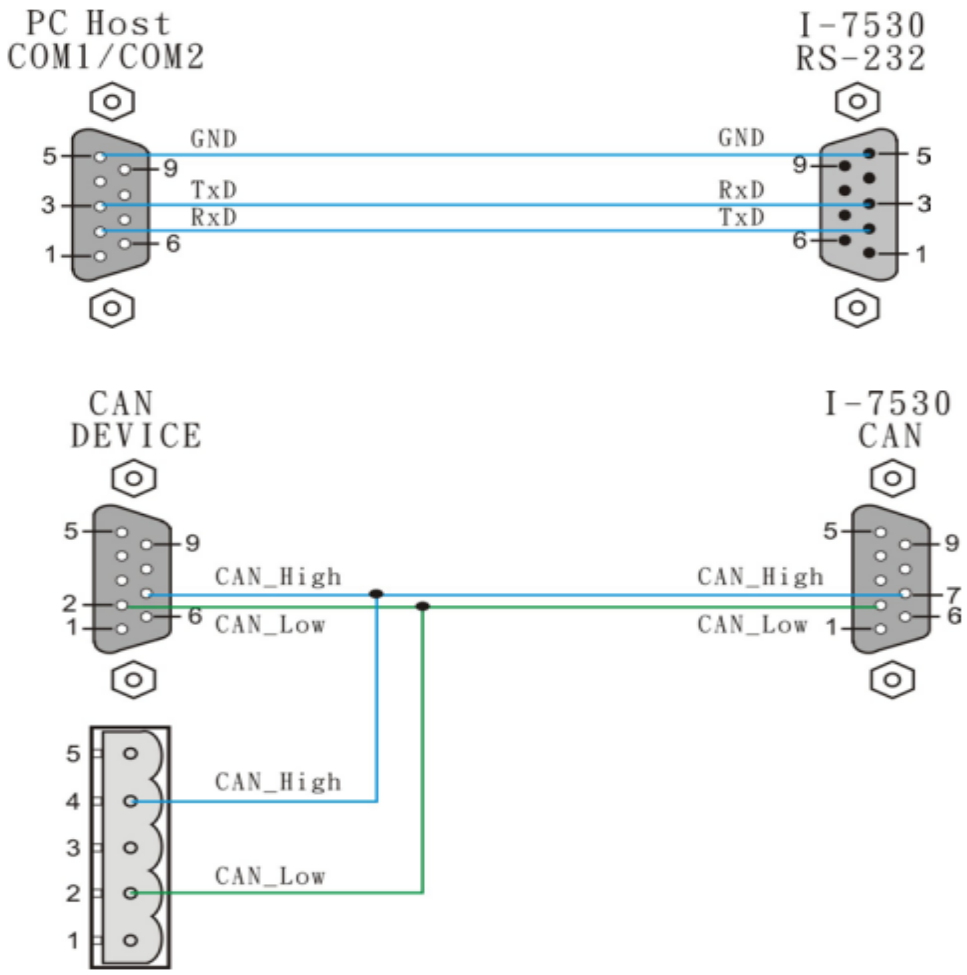
Şekil 6.6. RS232-CAN Modülünün Blok Diyagramı [27]



Şekil 6.7. RS232-CAN Dönüştürücüsünün Tipik Uygulaması [27]

6.3.1. CAN Modülü Bağlantısı

CAN sinyalleri DB9 konektörleri ile taşınabilir. DB9 konektörlerinde PC tarafında 2. ve 3. uçlar CAN cihazları arasında ise 2. ve 7. uçlar Rx ve Tx uçlar olarak kullanılır. Şekil 6.8.'de bağlantılar gösterilmiştir.



Şekil 6.8. RS232-CAN Dönüştürücü Modülünün Bağlantıları [27]

6.4.RS232-CAN Dönüştürücü Modülünün Kullanımı

RS232-CAN dönüştürücüsünde PC tarafından CAN mesajı göndermek için modül için tasarlanmış kod makroları bulunmaktadır. Bu kod makroları yüksek seviyeli programlama dillerinde kullanılarak ara yüz programlarının geliştirilmesinde kolaylık sağlamaktadır.

6.4.1. CAN Komut Listesi

RS232-CAN dönüştürücüsünde mesaj gönderimi ve modülün ayarları için kullanılan komutların listesi Çizelge 6.2.'de verilmiştir.

Çizelge 6.2. RS232-CAN Dönüştürücüsünün Komut Listesi (I-7530 RS-232/CAN Converter Manual [27]'den değiştirilerek alınmıştır.)

KOMUT	AÇIKLAMASI
tIIILDD…[CHK]<CR>	Standart veri çerçevesi gönder veya al
TIIIL[CHK]<CR>	Standart uzak çerçevesi gönder veya al
eIIIIIIIILDD…[CHK]<CR>	Genişletilmiş veri çerçevesi gönder veya al
EIIIIIIIIL[CHK]<CR>	Genişletilmiş uzak çerçevesi gönder veya al
S[CHK]<CR>	I-7530 modülünün durumu göster
C[CHK]<CR>	CAN/RS232 FIFO taşma hata işaretçisini sil
P0BBDSPAE[CHK]<CR>	Modülün RS-232 ayarını değiştir.
P1B [CHK]<CR>	Modülün CAN ayarını değiştir.
RA[CHK]<CR>	I-7530 modülünü yeniden başlat.

6.4.2. Standart Mesaj Gönderimi

Dönüştürücü modülünde standart çerçeve göndermek için tIIILDD…[CHK]<CR> biçimi kullanılır. Mesaj biçimindeki karakterlerin açılımı Çizelge 6.3.'de gösterilmiştir. Mesajın sonunda bulunan <CR> değeri ASCII kodlarında bulunan Carrige Return (<CR>) değeridir.

Çizelge 6.3. Standart Mesaj Biçiminin Açıklaması (I-7530 RS-232/CAN Converter Manual [27]'den değiştirilerek alınmıştır.)

KARAKTER	AÇIKLAMA
t	Standart (2.0A) Çerçevesi
III	11 bit belirleyici (000 ~ 7FF)
L	Veri uzunluğu (0 ~ 8)
DD…	Veri uzunluğuna bağlı veri çerçevesi

Standart mesaj olarak gönderilen t03F6112233445566<CR> ifadesinde, t standart mesajı, "03F" belirleyiciyi, "6" veri uzunluğunu, "112233445566" ise 5 veri uzunluğunda bulunan mesajın içerdiği veriyi göstermektedir.

6.4.3. Genişletilmiş Mesaj Gönderimi

Genişletilmiş mesaj gönderim biçimi de standart mesaj gönderim biçimi gibidir. eIIIIIIIILDD…[CHK]<CR>. Tek fark mesajın başında bulunan “e” karakteridir. Extended (Genişletilmiş) sözcüğünün ilk harfini ifade etmektedir. 12345678 ID içeren genişletilmiş mesajın gönderiminde e12345678511223344555<CR> biçiminde mesaj gönderilir. Burada “5” veri uzunluğunu, “1122334455” ise veriyi ifade etmektedir.

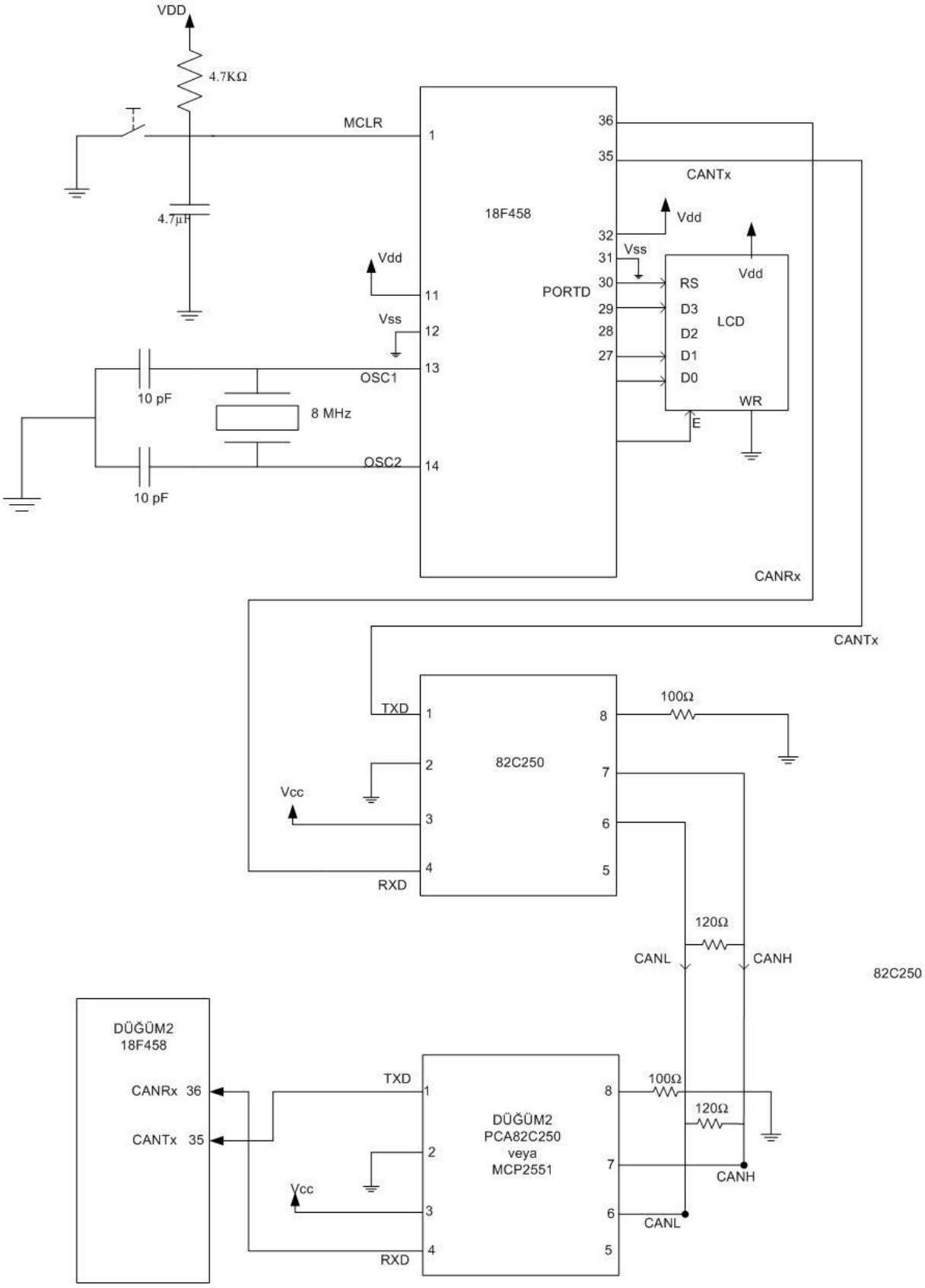
7. GERÇEKLEŞTİRİLEN CAN SİSTEMİ UYGULAMARI

Tezde alarm belirleme sistemi ile analog dijital dönüştürücü uygulamaları CAN protokolü kullanılarak gerçekleştirildi.

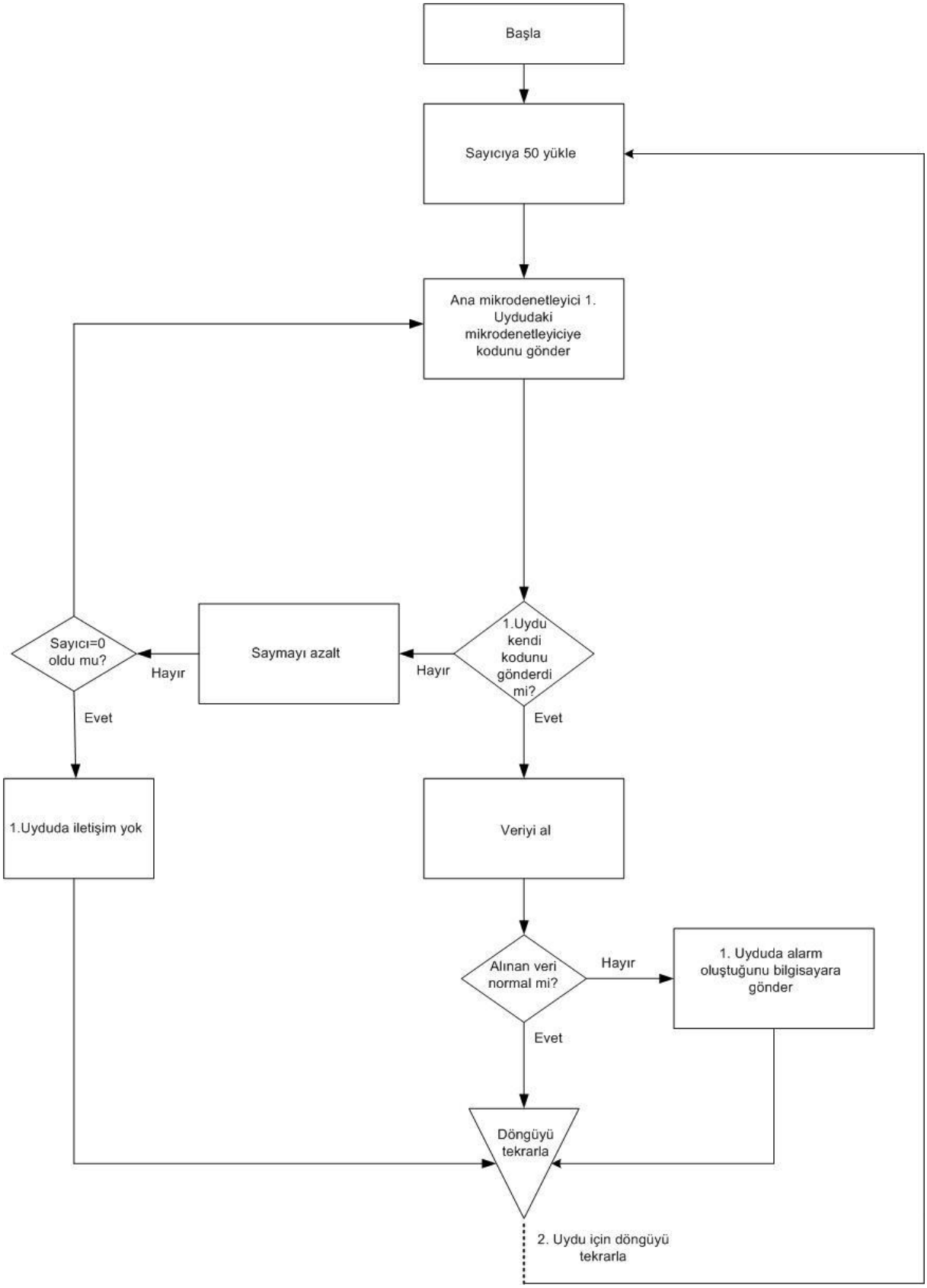
7.1. CAN Alarm Sistemi

CAN sistemiyle uygulaması yapılan alarm sisteminin devresi Şekil 7.1.'de gösterilmiştir. Şekil 7.3.'de n tane CAN düğümü kullanılabilir. Buradaki n sayısının değeri hattaki mesaj yüklemeleri ve gürültü gibi nedenlerden dolayı en fazla 112 olabilir. Uygulama devremizde Düğüm1 ve Düğüm2 kullanıldı. CAN ara yüz denetleyici entegreleri PCA82C250 ile MCP2551 entegreleri ile CAN veri yoluna bağlıdır. Düğüm1 ve Düğüm2'deki CAN mikrodenetleyicilerin osilatör yapısında 8 MHz'lik kristaller kullanılmıştır. Mikrodenetleyicilerin reset devresinde 4,7 kΩ direnç ve 4,7 µF'lık kondansatörler kullanıldı. Düğüm1 olarak adlandırdığımız PIC18F458 entegresine bağlı 2x16 4 bitlik LCD ile veri yolu üzerindeki mesajların belirleyicilerinin ve veri paketlerinin gösterimi sağlandı. Düğüm1'deki CAN mikrodenetleyicisi 3E8h belirleyici ve C8h veri içeren mesajı veri yoluna göndermektedir. Düğüm2'deki CAN mikrodenetleyicisi ise 1F4h belirleyici ve 64h veriyi iletmektedir.

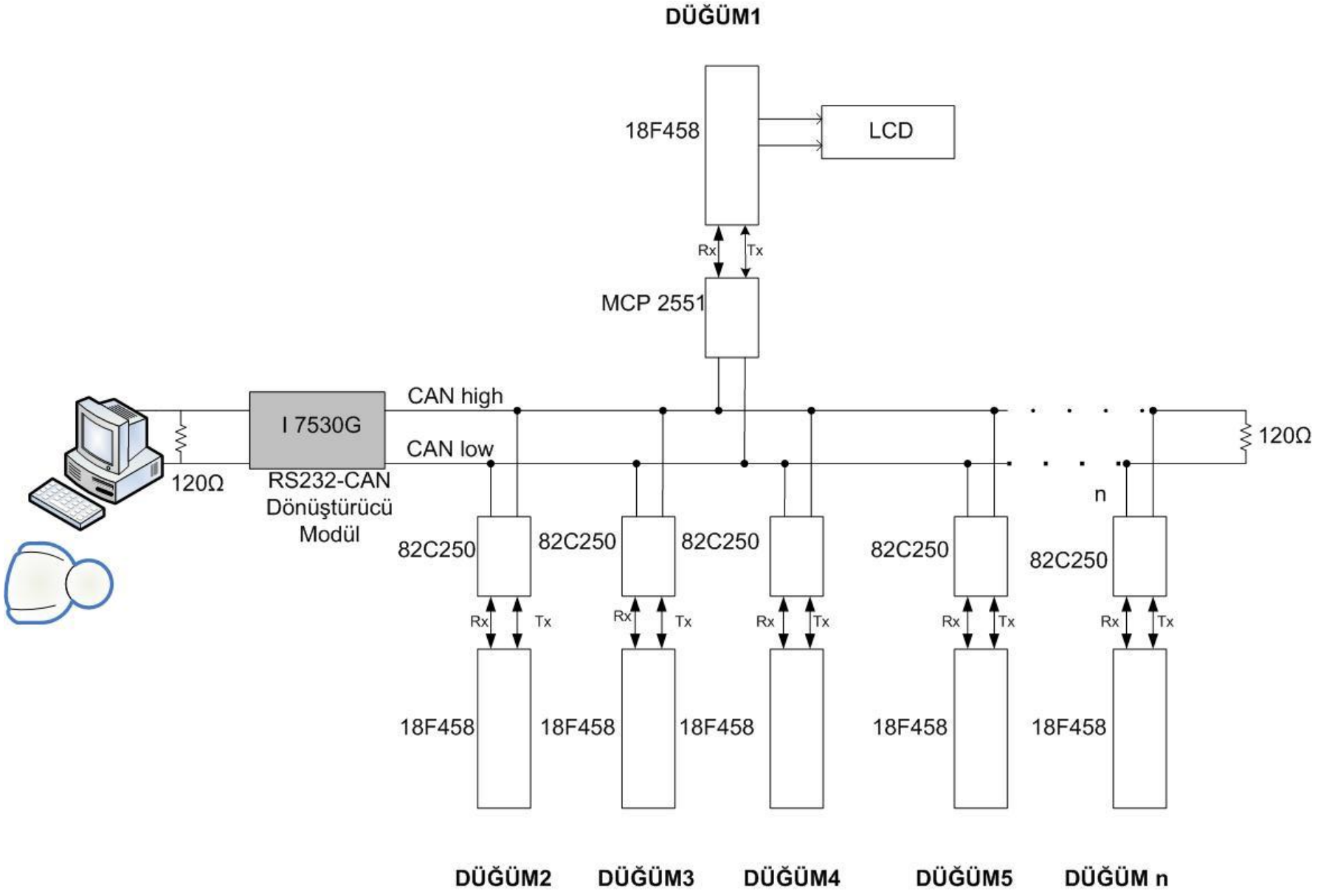
Düğümlerin belirleyicileri ve taşıdıkları veriler sistemin yapısına göre kullanıcının seçimine göre değişir. 3E8h ve 1F4 belirleyicileri onluk düzende 1000 ve 500 olduğundan düğümlerden gelen mesajları anlamak için rasgele atanmıştır. Veri yolunda yansımaların olmasını engellemek için 120Ω'luk dirençler kullanıldı. Şekil 7.2.'de tezde kullandığımız devrenin akış diyagramı gösterilmiştir. Sistemin akış diyagramında program önceden belirlenen süre kadar saymaya başlar. Bu süre sonunda ana mikrodenetleyici uydu mikrodenetleyicileri denetlemek için kod göndermeye başlar. İlk önce ana mikrodenetleyici 1. Uydunun normal çalışıp çalışmadığını denetler. Ana mikrodenetleyiciden gönderilen kodun uydu mikrodenetleyicisinde doğru alınması gerekir. Gönderilen kod 1. Uyduda doğru alınmamışsa saymanın süresi azaltılır ve sayma değeri sıfır olduğunda ana mikrodenetleyici tarafından 1.uyduya kod gönderimi tekrar başlar. 1. Uydu yine mesajı doğru alamadığından 1. uydunun alıcı hata sayacı 1 artar. Hata sayacının değeri 96 olduğunda 1.Uydudan gelen verilerin güvensiz olduğu anlaşılır. Hata sayacı 255 değerine ulaştığında ise 1.Uydu mesaj alımını durduracaktır ve ana mikrodenetleyiciye iletimin durduğu bilgisini iletecektir. 1. Uydunun aldığı mesaj CAN mesaj biçiminde ve normal ise 1. Uydu bu mesajı işleyecek ve ana mikrodenetleyici tarafından mesaj almaya devam edecektir.



Şekil 7.1. Alarm Sisteminin Devre Bağlantı Şeması



Şekil 7.2. Alarm Sisteminin Akış Diyagramı

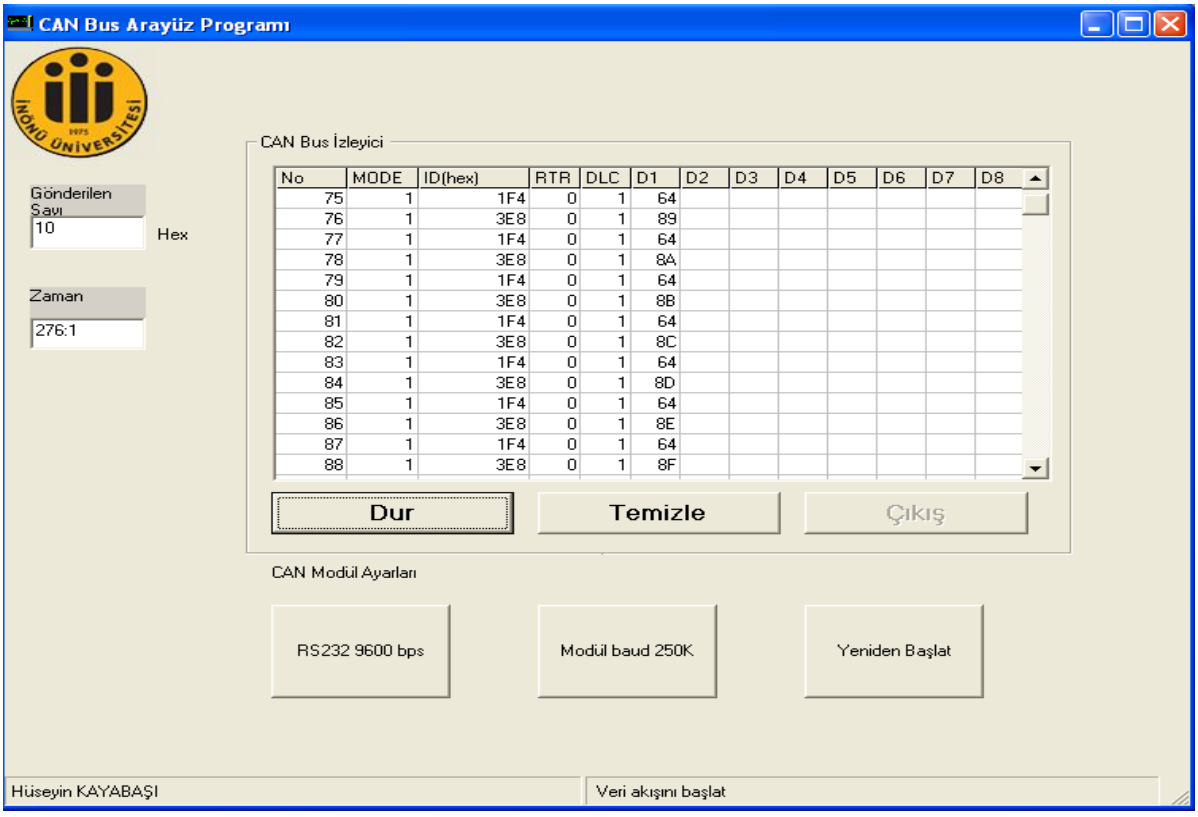

DÜĞÜM1
18F458
LCD
Rx
Tx
MCP 2551
CAN high
CAN low
I 7530G
RS232-CAN
Dönüştürücü
Modül
120Ω
120Ω
82C250
82C250
82C250
82C250
82C250
n
18F458
18F458
18F458
18F458
18F458
DÜĞÜM2
DÜĞÜM3
DÜĞÜM4
DÜĞÜM5
DÜĞÜM n

Şekil 7.3. Alarm Sistemi Düzeneği

Bilgisayarda oluşturulan ara yüz programında 3E8h ve 1F4h belirleyicili mesajlar ve taşıdıkları veriler Şekil 7.4.'de gösterilmiştir. İki düğümünde mesaj biçiminde uzak çerçeve biti kullanılmadığından ara yüz programında RTR sütununda "0" olarak gözükmektedir. Veri iletiminin başlayabilmesi için ara yüz programında başla butonuna tıklanması gerekmektedir. CAN haberleşeme protokolünün olay bazlı bir protokol olmasından dolayı bu işlem olmadan iletim gerçekleşmez.

Ara yüz programında Düğüm1'deki LCD'de gözükmesi için gönderilen onaltılık düzendeki sayı LCD'de onluk düzende PC karakterlerinden sonra gösterilmektedir. Gönderilen sayı, Düğüm1 tarafından doğru alınıp alınmadığını denetlemek için kullanılmıştır. İstenirse bu sayı değeri Düğüm1 entegresi için veri iletiminin durdurulması veya farklı bir belirleyiciye sahip bir mesajın gönderilmesi için de kullanılabilir.

Veri iletimi başlar başlamaz Düğüm2'den gelen mesajla birlikte, Düğüm1'deki veriyi 1 arttırarak veri yolu üzerinde gözlemliyoruz. Bu artma işlemi bilgisayar tarafında istenilen değere göre ayarlanabilir. Bizim devremizde 256'ya kadar sayma işlemi devam edip daha sonra sıfırlanıp tekrar sayma işlemine devam etmektedir. Sayma işleminin de önceden belirlenecek bir değer alarm seviyesi olarak ayarlanabilir. Bu sayma sonucunda düğümlere bağlı sensörler, röleler varsa bunlar harekete geçirilebilir.



Şekil 7.4. Ara Yüz Programının Görüntüsü

Alarm sistemimizin çalışma prensibinde, Düğüm1 devre dışı kaldığında bilgisayar tarafında veri akışının olmadığı görürüz. Düğüm1'in kapalı olması sonucunda iletimin olmaması Düğüm1'in yüksek önceliğe sahip olduğunu gösterir. Düğüm2'den gelen mesajın ara yüz programında bir değeri yoktur.

Düşük öncelikli olarak ayarladığımız Düğüm2'nin herhangi bir nedenle devre dışı kalması durumunda Düğüm2'den veri gelmeyeceği için Düğüm1'deki veri değerinin artışı otomatik olarak devam etmeyecektir. Fakat ara yüz programında başla butonuna sürekli tıklandığında sayma işlemi bir artarak gerçekleşecektir. Bunun nedeni CAN protokolünde iletim için en az iki düğümün devrede olması gerektir.[3] Bilgisayarda başla butonuna tıklanması sonucunda RS232-CAN dönüştürücüsüne gelen sinyal dönüştürücü modülünü CAN düğümü haline getirir. Bu sayede Düğüm1'de veri artışı olacaktır. Düğüm1'in belirleyicisi 1000, Düğüm2'den veri iletimi olmadığı için LCD'de gözükmeyecektir. Bilgisayardan gönderilen sayı bu durumdan etkilenmeyeceği için LCD'de gönderilen sayı onluk düzende gösterilir.

7.2. Analog Dijital Dönüştürücü Uygulaması

CAN protokolü kullanılarak uygulamasını gerçekleştirdiğimiz ADC uygulamasının akış diyagramı Şekil 7.5.'de gösterilmiştir. PIC18F458 mikrodenetleyicisinin portuna bağlı ayarlı direnç ile portun bacakları üzerindeki değişken gerilim değeri LCD ve bilgisayara aktarılmıştır. Mikrodenetleyicinin, ADC modülünün öncelikle şartlanması gerekir. Mikrobasic programında yapılan şartlama da mikrodenetleyicinin ADC modülünde bulunan CMCON bitine 07h değeri yüklenerek karşılaştırma girişinin olmaması sağlanır. ADCON1 kayıtçısına 07h değeri yüklenir. Mikrodenetleyicinin D portuna LCD bağlı olduğundan CMCON ve ADCON1 kayıtçılarına 07h değerleri yüklenmelidir. Bu değerler girilmediği takdirde LCD veri gözükmeyecektir. Bu kayıtçıların değerleri INTCON, (Interrupt Controller) kayıtçısına "0" değeri atanır. Böylece, mikrodenetleyicinin analog dijital dönüştürücüsünde kesme olmaması sağlanır.

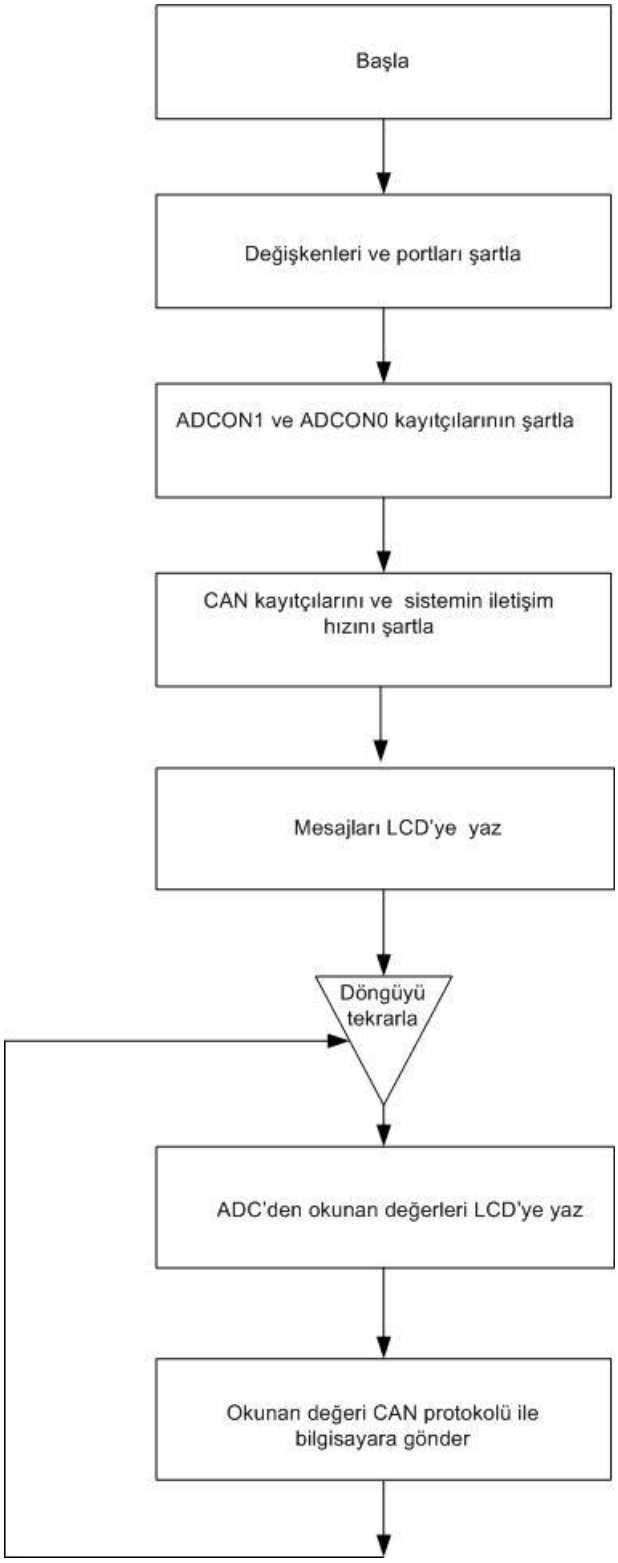
```
CMCON=$07
 LATB = 0
 PortD = 0
 PortB =0
 TRISB.3 = 1
```

```
TRISB.2 = 0
TRISC  = 0
INTCON = 0
ADCON1 = $07
TRISA  = $FF
TRISD=0
```

Analog dijital dönüştürücü uygulamasının akış diyagramı Şekil 7.5.'de gösterilmiştir. PIC18F458 mikrodenetleyicisinin analog dijital dönüştürücü modülünde değişken ve portların şartlanması yapıldıktan sonra CAN sisteminin iletişim hızı ayarlanır. 8 Mhz, 125 kbps ve veri yolunda 3 kez örnekleme yapılması için iletişim hızı için BRGCON kayıtçılarına değerleri yüklenir. BRGCON1 kayıtçısına 01h, BRGCON2 kayıtçısına B8h, BRGCON3 kayıtçısına ise 05h değerleri atandıktan sonra CAN modülünün iletişim hızının ayarları yapılmış olur. ADC modülünden gelen verilerin LCD'ye yazımı işlemi başlar. Uygulamamızda ADC modülüne bağlı ayarlı direnç gerilimin 0 ila 5 V arasında olmasını sağlar. Virgüllü gerilim değerlerinin LCD'ye yazımında Mikrobasic programında gerilim değeri string türüne dönüştürülür. Dönüştürme işlemi yapıldıktan sonra virgülden sonraki basamakların LCD'ye yazımı için ADC değeri mod 10'a göre ayarlanması gerekir. Bunun için okunan değer önce 1000'e bölünür kalan değer LCD'ye yazılır. Sonra 100'e daha sonra ise 10'a bölme işlemleri yapılarak bölme sonucunda kalan değerler LCD'ye yazılır. Bölme işleminin yapıldığı satırlar şöyledir.

```
ch     = tlong / 1000
LCD_Chr(2,9,48+ch)
LCD_Chr_CP(".")
ch    = (tlong / 100) mod 10
LCD_Chr_CP(48+ch)
ch    = (tlong / 10) mod 10
LCD_Chr_CP(48+ch)
```

Ayarlı direnç ile değiştirilen gerilim değeri için döngü tekrarlanır. Sürekli olarak LCD ekranında ve bilgisayarda gerilim değerleri gözlenir.



Şekil 7.5. Analog Dijital Dönüştürücü Akış Diyagramı

8. ARA YÜZ ve MİKRODENETLEYİCİ PROGRAMLARI

8.1. Mikrodenetleyici Programı

Tezde kullandığımız PIC18F458 mikrodenetleyicisinin programlamasında Mikrobasic programı kullanıldı.Mikrobasic programında ilk önce parametreler tanımlanmıştır. Can_Init_Flags, Can_Send_Flags, Can_Rcv_Flags değişkenleri CAN modülünün ayarlarını içermektedir. CMCON, karşılaştırıcı bitinin değeri 07h olarak girilip entegrenin analog dijital dönüştürücüsü devre bırakılmıştır. B ve D portları entegrenin çıkış değerlerini verdiği portlar olduğundan 0 değeri verilmiştir. B portunun 2.ve 3. biti CAN sinyallerinin alındığı ve iletildiği pinler olduğundan sırasıyla çıkış ve giriş değerleri olarak şartlanmışlardır.

2x16 karakterli 4 bitlik LCD'nin 1. satırının 1. sütununa veri yolunda gözlenen mesajın belirleyicisinin okunması için ID karakterleri yazılmıştır. Programda, veri iletimine başlamadan önce tüm ayar değişkenleri sıfırlanıp sonraki satırlarda iletilecek mesajın özellikleri bu değişkenlere atandı.

CANInitialize fonksiyonu içerisindeki 1,1,8,6,1 değerleri ile PIC18F458 mikrodenetleycisinin 8 Mhz 125 Kbps hızında haberleşmesi sağlandı. CANInitialize fonksiyonu ve CANSetMask ile CANSetFilter fonksiyonları mikrodenetleyicinin CAN modülü ayar modunda iken tanımlandı. Bu fonksiyonlara değişken ataması yapıldıktan sonra mikrodenetleyici normal moda çekilerek işleme devam edildi. CANSetFilter fonksiyonunda 1000 belirleyici kısmına (ID) sahip genişletilmiş çerçevelerin alınması sağlandı.

Programda 500 belirleyici kısmı (ID) için modül ayar moduna çekilip maske ve filtre işlemleri yapıldı. Ayar modundan normal moda geçiş ile entegre haberleşmek için hazır konuma gelir.

8.2.Ara Yüz Programı

Tezde kullandığımız ara yüz programın MS Visual Basic programındaki kodlar EK'de verilmiştir. Programın veri gönderme ve dönüştürme kısmı kısaca şöyledir. Seri porttan veri alışının başlaması için başla butonuna tıklanması gerekir. MSComm1.Output değişkeninde 3E8h belirleyicili genişletilmiş mesaj 1 veri uzunluğunda olan Text1.Text değişkeni ve Ascii kodu taşıma dönüşü (Carriage Return, vbCr) ile bilgisayarın seri portundan RS232-CAN modülüne gönderildi. Seri porta bilgi gelmesinde ise CAN mesajın belirleyicisine göre değerlendirme yapıldığı için RS232 veri biçimi içerisinde verinin bölümlemesi yapıldı. Databuf değişkeni döngü öncesinde sıfırlanıp seri porta bilgi geldiğinde databuf değişkeni

değer alacaktır. Databuf değişkenin 7.bitinden sonraki 3 bit alınarak belirleyici kısmı algılanır. 3E8h ve 1F4h .

```
Databuf = ""
    Do While (bStart.Caption = "Dur")
      Do While (1)

   MSComm1.Output = "e000003e81" + Text1.Text + vbCr

       If MSComm1.InBufferCount > 0 Then
        Buffer = Buffer + Trim(MSComm1.Input)
         Pos = InStr(Buffer, Chr(13))
   If Pos > 0 Then
         Databuf = Mid(Buffer, 1, Pos - 1)
         Buffer = Mid(Buffer, Pos + 1, Len(Buffer) - Pos)
 CanID = Hex("&h" + Mid(Trim(Databuf), 7, 3))
```

Belirleyiciden sonraki RTR, DLC ve veri ise daha sonraki satırlarda ekrana yazdırılır. RS232-CAN dönüştürücüsünde kullanılan “e” ve “t” durumlarına göre veri paketinde bölümleme yapılır. Databuf değişkenin ilk biti “e” veya “t” olacağından programlama Mid ve Trim fonksiyonları ile sadece 1. bit ele alındı ve bu bitin değerine göre belirleyici ve veri paketinin uzunluğu hesaplandı. Standart mesaj durumunda CanID 2.bitten sonra izleyen boşluksuz 3 bittir. Veri uzunluğunu gösteren bit ise 5.bitten sonraki ilk bittir. Genişletilmiş çerçevede ise CanID 2.bitten sonra izleyen boşluksuz 8 bittir. Veri uzunluğunu gösteren bit ise 10.bitten sonraki ilk bittir.

```
       Select Case (Mid(Trim(Databuf), 1, 1))
        Case "t"
          CanMode = 0
          CanRTR = 0
   CanID = Hex("&h" + Mid(Trim(Databuf), 2, 3))
  DLC = Mid(Trim(Databuf), 5, 1)
Case "e"
  CanMode = 1
```

```
CanRTR = 0
 CanID = Hex("&h" + Mid(Trim(Databuf), 2, 8))
   DLC = Mid(Trim(Databuf), 10, 1)
```

9.SONUÇ

Tezde uygulamasını yaptığımız CAN temelli alarm belirleme sisteminde iki farklı düğümün tam denetimi bilgisayar tarafından ve bir düğüm tarafından yarı denetimi sağlandı. CAN haberleşme protokolünde çok sayıda ana kullanıcı tanımını bu tezde gerçekleştirildi. Ara yüz programı iki düğümü de denetleyebildiği halde Düğüm1 sadece Düğüm2'deki CAN mikrodenetleyicisini denetlemek için programlandı. Çok sayıda düğümden oluşan sistemde öncelik seviyesi yüksek olan mesajların iletimi gerçekleşecektir. Tezde kullandığımız yapıda düğümlerin öncelik seviyeleri aynı olduğundan veri iletimini bilgisayardan anlık olarak denetlenebildi.

Analog dijital dönüştürücü uygulamasında gerilimin değişimi anlık olarak LCD ve bilgisayarda gözlendi. Gerilim değerinin gecikmeli olarak LCD'ye ve bilgisayara aktarımı için ara yüz programında gecikme süresi eklendi. Bu sayede gerilim değerinin bilgisayarda gözlenmesi kolaylaştırıdı.

Sistemin maliyeti benzer protokollerle gerçekleştirilen uygulamalara göre ucuzdur. Düğüm noktaları arasında mesafenin uzun olması maliyeti ve veri iletişimini etkilemektedir. Kablo mesafesinin kısa olduğu sistemlerde veri kaybının ve elektromanyetik etkileşimin az olması CAN protokolü için tercih nedenidir. Düğümler arası mesafenin kısa tutulması ile tezde kullanılan devrede maksimum performansın alınması sağlandı.

Mikrodenetleyicinin kodları Mikrobasic dilinde derlendi. Mikrobasic programın kullanımının 18XX8 ailesi için CAN kütüphanesinin olması yazılan programın satırlarının az olmasını sağladı.

Ara yüz programının kodları ise, MS Visual Basic programında yazıldı. Veri yolu üzerideki mesajların iletim hızının gözlenmesi ve takibi text kutusu içerisinde gösterildi. Mesajların ayrıca kaydedilebilir olması geriye dönük denetim işlemleri için yazılan programın büyük bir avantajıdır.

CAN protokolü ile farklı sektörlerde uygulama yapılabilir. Uygulamanın yapılacağı alana göre seçilecek sensör ve sensöre bağlı mekanizmalara göre sistemin yapısı farklılaşacaktır. Kullanılacak düğüm sayısı ve bu düğümlerin öncelik seviyeleri sistemin karakteristik özelliğini ortaya çıkaracaktır. CAN protokolü kablosuz haberleşme protokolleri olan ZigBee, Bluetooth gibi protokoller ile birleşerek kablo mesafesinden dolayı sinyal zayıflamasının olacağı düğümleri denetlemek için kullanılabilir. CAN protokolü için kullanıcı tarafında özel fonksiyonlar tanımlanabilir. Mevcut fonksiyonların haricinde olan bu

fonksiyonlar aracılığıyla tasarlanan sistem özgün bir yapıda ve güvenliği yüksek derece olacaktır.

Sonuç olarak CAN protokolü insan hayatı için önem arz eden ve gönderilen mesajların önemli olduğu her sektör için CAN haberleşme protokolü maliyeti ve yapısı ile tercih nedeni olacaktır.

Bu tezde CAN alarm sisteminin tüm altyapısı kullanılmış ve çalıştırılmıştır. Bu sistem sayesinde başka bir ara yüz programı geliştirilerek alarm sistemi için geliştirmelerin yapılabilmesi mümkündür.

KAYNAKLAR

[1] http://zone.ni.com/devzone/cda/tut/p/id/2732
[2] K. C. Lee H.H. Lee, "*Network-based fire-detection system via controller area network for smart home automatio",* **IEEE Transactions on Consumer Electronics**, 50:4 (2004) 1093-1100
[3] C. Bayılmış, "*IEEE 802.11B KLAN Kullanarak CAN Segmentleri Genişleten Arabağlaşım Birimi Tasarımı*",Doktora Tezi, Kocaeli Üniversitesi, 2006.
[4] C. Çeken, *"Real Time Data Transfer With Quality of Service Support Using Wireless ATM"* Ph.D. Thesis, University of Kocaeli, 2004.
[5] Y. Santur, *"Kontrol Alan Ağı Protokolü Tabanlı Bir Endüstriyel Ağın Kurulması ve Yönetilmesi"* Yüksek Lisans Tezi, Fırat Üniversitesi, 2006
[6] A.Özdemir, *"Industrial Networks (An Application Protocol On Canbus:Canup)"* M.S. Thesis, İstanbul Technical University, 2004
[7] S.Tuncel, *"Deneleyici Alan Ağı Endüstriyel İletişim Protokolünün Eğitim Amaçlı Benzetimi"*Yüksek Lisans Tezi, Sakarya Üniversitesi, 2002
[8] A.Karaca, "*Denetleyici Alan Ağı Kullanılarak (CAN)Bina Güvenlik Uygulaması"* Yüksek Lisans Tezi, Sakarya Üniversitesi, 2003
[9] D.Paret, "*Multiplexed Networks for Embedded Systems CAN, LIN, Flexray, Safe-by-Wire...,"* Wiley, 2007, 25-351
[10] M. Passemard, "Atmel Microcontrollers for Controller Area Network (CAN)", San Jose, 2004
[11] http://www.vector-worldwide.com/vi_osek_en.html
[12] http://www.exampleproblems.com/wiki/index.php/Power_line_communication
[13] http://www.vmars.tuwien.ac.at/projects/xbywire/projects/tuwien.html
[14] Anonymous, FlexRay Communications System Protocol Specification, V 2.1 Rev. A, 2005
[15] http://www.vmars.tuwien.ac.at/projects/ttp/ttpc.html
[16] H. Fuhrmann, Prof. Dr. R.V.Hanxleden, Model-Based Design and Distributed Real-Time Systems-Lecture 11, Mod. Based Design & Dist RT Systems, 2006
[17] http://www.embedded.com/story/OEG20010718S0073
[18] H. Boterenbrood, CANopen high-level protocol for CAN-bus NIKHEF,V3., Amsterdam, March , 2000
[19] Anonymus, CAN Specification V.2, 1991
[20] P.Richards, AN 228 A CAN Layer Discussion, Microchip, 2002
[21] G. Cena and A.Valenzano, *FastCAN: a high-performance enhanced CAN-like network*, Industrial Electronics, **IEEE Transactions on Industrial Electronics,** 47:4 (2000) 951-963
[22] G. Cena and A.Valenzano, *A Multistage Hierarchical Distributed Arbitration Technique for Priority-Based Real-Time Communication Systems*, **IEEE Transactions on Industrial Electronics,** 49:6 (2002) 1227-1239
[23] Anonymus, PIC18FXX8 Data Sheet, 2004
[24] G. Kavaiya, PIC18XXX8 CAN Driver with Prioritized Transmit Buffer, 2002
[25] Anonymus, PCA82C250 CAN Controller Interface Data Sheet, 2000
[26] Anonymus, MCP2551 High-Speed CAN Transceiver Data Sheet, 2003
[27] Anonymus, I-7530 RS-232/CAN Converter User's Manual, V. 2.2., 2008

EKLER

EK 1 (TEZDE KULLANILAN MIKROBASIC MİKRODENETLEYİCİ PROGRAMI)

DÜĞÜM 1 İÇİN YAZILAN KOD

```
program cantez
dim Can_Init_Flags, Can_Send_Flags, Can_Rcv_Flags, Rx_Data_Len  as byte
   RxTx_Data  as byte[8]
  RxTx_Data_G  as byte[8]
   Tx_ID, Rx_ID as longint
   Msg_Rcvd as byte
   dim text as char[20]

 main:
CMCON=$07
  LATB = 0
  PortD = 0
  PortB =0

  TRISB.3 = 1
  TRISB.2 = 0
  TRISD  = 0
  TRISC  = 0

 Lcd_Init(PORTD)
 Lcd_Cmd(LCD_CLEAR)
  Lcd_Cmd(LCD_CURSOR_OFF)
      Lcd_Out(1,1,"Id:")

  Can_Init_Flags   = 0
  Can_Send_Flags   = 0
  Can_Rcv_Flags    = 0

  Can_Send_Flags   = CAN_TX_PRIORITY_0 and
                CAN_TX_XTD_FRAME  and
                CAN_TX_NO_RTR_FRAME
```

```
Can_Init_Flags   = CAN_CONFIG_SAMPLE_THRICE and
                   CAN_CONFIG_PHSEG2_PRG_ON and
                  CAN_CONFIG_DBL_BUFFER_OFF and
                  CAN_CONFIG_ALL_MSG and
                   CAN_CONFIG_LINE_FILTER_OFF
      CANSetOperationMode(CAN_MODE_CONFIG,0xFF)
     CANInitialize(1,1,8,6,1,Can_Init_Flags)
      CANSetBaudrate(1,1,8,6,1,Can_Init_Flags)
      CANSetMask(CAN_MASK_B1,-1,CAN_CONFIG_XTD_MSG)
       CANSetMask(CAN_MASK_B2,-1,CAN_CONFIG_XTD_MSG)
      CANSetFilter(CAN_FILTER_B1_F1,1000,CAN_CONFIG_XTD_MSG)
       CANSetOperationMode(CAN_MODE_Normal,0xFF)
  RxTx_Data_G[0]=0
   Tx_ID = 1000
  RxTx_Data[0]=0
  while TRUE
     Msg_Rcvd = CANRead(Rx_ID,RxTx_Data,Rx_Data_Len,Can_Rcv_Flags)
    LongintToStr(Rx_ID,text)
    Lcd_Out(1,3,text)

     if (Rx_ID = 500) and Msg_Rcvd  then
      CANSetOperationMode(CAN_MODE_CONFIG,0xFF)
      CANSetMask(CAN_MASK_B1,-1,CAN_CONFIG_XTD_MSG)
      CANSetMask(CAN_MASK_B2,-1,CAN_CONFIG_XTD_MSG)
      CANSetFilter(CAN_FILTER_B1_F1,500,CAN_CONFIG_XTD_MSG)
      CANSetOperationMode(CAN_MODE_Normal,0xFF)
        Delay_ms(100)
       Msg_Rcvd = CANRead(Rx_ID,RxTx_Data,Rx_Data_Len,Can_Rcv_Flags)
      bytetostr(RxTx_Data[0],text)
     Lcd_Out(2,9,"PC:")
     Lcd_Out(2,12,text)
     Delay_ms(1)
        end if
```

```
      if (Rx_ID = 1000) and Msg_Rcvd  then

        Delay_ms(100)
      RxTx_Data_G[0]= RxTx_Data_G[0]+1

    CANWrite(Tx_ID, RxTx_Data_G, 1, Can_Send_Flags)
      bytetostr(RxTx_Data_G[0],text)

      Lcd_Out(2,1,"P1:")
      Lcd_Out(2,4,text)
      Delay_ms(50)
    end if
    wend
end.
```

DÜĞÜM 2 İÇİN YAZILAN KOD

```
program cantez
dim  Can_Init_Flags,  Can_Send_Flags,  Can_Rcv_Flags,  Rx_Data_Len,  Msg_Rcvd     as
byte
    RxTx_Data  as byte[8]
    RxTx_Data_G  as byte[8]
    Tx_ID, Rx_ID as longint
    dim text as char[20]
 main:
CMCON=$07
  LATB = 0
  PortD = 0
  PortB =0
  TRISB.3 = 1
  TRISB.2 = 0
  TRISD  = 0
  TRISC  = 0
Can_Init_Flags   = 0
```

```
  Can_Send_Flags   = 0
  Can_Rcv_Flags    = 0
  Can_Send_Flags   = CAN_TX_PRIORITY_0 and
                     CAN_TX_XTD_FRAME  and
                     CAN_TX_NO_RTR_FRAME
  Can_Init_Flags   = CAN_CONFIG_SAMPLE_THRICE and
                     CAN_CONFIG_PHSEG2_PRG_ON and
                     CAN_CONFIG_DBL_BUFFER_OFF and
                   CAN_CONFIG_ALL_MSG and
                     CAN_CONFIG_LINE_FILTER_OFF
       CANSetOperationMode(CAN_MODE_CONFIG,0xFF)
       CANInitialize(1,1,8,6,1,Can_Init_Flags)
       CANSetBaudrate(1,1,8,6,1,Can_Init_Flags)
      CANSetMask(CAN_MASK_B1,-1,CAN_CONFIG_XTD_MSG)
       CANSetMask(CAN_MASK_B2,-1,CAN_CONFIG_XTD_MSG)
       CANSetFilter(CAN_FILTER_B1_F1,500,CAN_CONFIG_XTD_MSG)
       CANSetOperationMode(CAN_MODE_Normal,0xFF)
   Tx_ID = 500
  RxTx_Data[0]=0
  while TRUE
     if (Rx_ID = 500) and Msg_Rcvd  then
       Delay_ms(100)
    CANWrite(Tx_ID, RxTx_Data[0], 1, Can_Send_Flags)
    end if
    wend
end.
```

EK 2 (CAN ARAYÜZ PROGRAMI)

```
Dim Databuf, CanID, Data(8) As String
Dim x, y
Dim MsgCount, CanMode, CanRTR, DLC  As Integer
Private Sub bClear_Click()
  MsgCount = 0
  Grid1.Clear
  With Grid1
        .TextMatrix(0, 0) = "No"
        .TextMatrix(0, 1) = "MODE"
        .TextMatrix(0, 2) = "ID(hex)"
        .TextMatrix(0, 3) = "RTR"
         .TextMatrix(0, 4) = "DLC"
         .TextMatrix(0, 5) = "D1"
         .TextMatrix(0, 6) = "D2"
         .TextMatrix(0, 7) = "D3"
         .TextMatrix(0, 8) = "D4"
         .TextMatrix(0, 9) = "D5"
         .TextMatrix(0, 10) = "D6"
         .TextMatrix(0, 11) = "D7"
         .TextMatrix(0, 12) = "D8"
         .TopRow = 1
  End With
End Sub
Private Sub bExit_Click()
  If MSComm1.PortOpen Then MSComm1.PortOpen = False
  End
End Sub
Private Sub bStart_Click()
 Dim Pos As Integer
 Dim Buffer As String

 If bStart.Caption = "Dur" Then   'stop
```

```
    bStart.Caption = "Başla"
    bExit.Enabled = True
    StatusBar1.Panels(2).Text = " Veri akışını durdur ! "
  Else
    bStart.Caption = "Dur"       'start
    bExit.Enabled = False
    StatusBar1.Panels(2).Text = " Veri akışını başlat "
 End If
Databuf = ""
 Do While (bStart.Caption = "Dur")
      Do While (1)

  MSComm1.Output = "e000003e81" + Text1.Text + vbCr

     If MSComm1.InBufferCount > 0 Then
          'MSComm1.InputLen = 0
        Buffer = Buffer + Trim(MSComm1.Input)



          Pos = InStr(Buffer, Chr(13))
         If Pos > 0 Then
          Databuf = Mid(Buffer, 1, Pos - 1)
           Buffer = Mid(Buffer, Pos + 1, Len(Buffer) - Pos)


  CanID = Hex("&h" + Mid(Trim(Databuf), 7, 3))

         Pos = 0
 If CanID = "3E8" Then
Exit Do
End If
   End If

      End If
```

```
        DoEvents
      Loop
Do While (1)

   MSComm1.Output = "e000001F41" + Text1.Text + vbCr

       If MSComm1.InBufferCount > 0 Then
          'MSComm1.InputLen = 0
          Buffer = Buffer + Trim(MSComm1.Input)



          Pos = InStr(Buffer, Chr(13))
          If Pos > 0 Then
            Databuf = Mid(Buffer, 1, Pos - 1)
            Buffer = Mid(Buffer, Pos + 1, Len(Buffer) - Pos)
  CanID = Hex("&h" + Mid(Trim(Databuf), 7, 3))

        Pos = 0

  If CanID = "1F4" Then

Exit Do
End If
          End If

        End If
        DoEvents
      Loop
 MsgCount = MsgCount + 1

        If MsgCount = 2048 Then
         MsgCount = 0
         Grid1.Clear
         With Grid1
```

```
        .TextMatrix(0, 0) = "No"
        .TextMatrix(0, 1) = "MODE"
       .TextMatrix(0, 2) = "ID(hex)"
       .TextMatrix(0, 3) = "RTR"
       .TextMatrix(0, 4) = "DLC"
       .TextMatrix(0, 5) = "D1"
       .TextMatrix(0, 6) = "D2"
       .TextMatrix(0, 7) = "D3"
       .TextMatrix(0, 8) = "D4"
       .TextMatrix(0, 9) = "D5"
       .TextMatrix(0, 10) = "D6"
       .TextMatrix(0, 11) = "D7"
       .TextMatrix(0, 12) = "D8"
        End With
       End If

    If MsgCount > 14 Then Grid1.TopRow = MsgCount - 14
Select Case (Mid(Trim(Databuf), 1, 1))
    Case "t"
      CanMode = 0
      CanRTR = 0
      CanID = Hex("&h" + Mid(Trim(Databuf), 2, 3))
      DLC = Mid(Trim(Databuf), 5, 1)

    Case "T"
      CanMode = 0
      CanRTR = 1
      CanID = Hex("&h" + Mid(Trim(Databuf), 2, 3))
      DLC = Mid(Trim(Databuf), 5, 1)
    Case "e"
      CanMode = 1
      CanRTR = 0
      CanID = Hex("&h" + Mid(Trim(Databuf), 2, 8))
      DLC = Mid(Trim(Databuf), 10, 1)
```

```
        Case "E"
          CanMode = 1
          CanRTR = 1
          CanID = Hex("&h" + Mid(Trim(Databuf), 2, 8))
          DLC = Mid(Trim(Databuf), 10, 1)
        End Select

        Call GetData(CanMode, CanRTR, DLC)

      DoEvents
    Loop
End Sub

Private Sub Form_Load()
On Error GoTo OpenError
  With MSComm1
   .InputLen = 0
   .CommPort = 1
   .PortOpen = True
  End With
```

EK 3 CAN ADC UYGULAMASI KODLARI

```
program ADC_on_Lcd_BL
dim ch as byte
    adc_rd as word
    text as string[16]
    tlong as LongInt
dim Can_Init_Flags, Can_Send_Flags, Can_Rcv_Flags, Rx_Data_Len  as byte
    RxTx_Data  as byte[8]
    Tx_ID, Rx_ID as longint
    Msg_Rcvd as byte
     yaz as string[3]
    dim textadc as char[3]
    dim cc as word
main:

CMCON=$07
  LATB = 0
  PortD = 0
  PortB =0

  TRISB.3 = 1
  TRISB.2 = 0

  TRISC  = 0
  INTCON = 0
  ADCON1 = $07
  TRISA  = $FF
  TRISD=0

  Lcd_Init(PORTD)
  LCD_Cmd(LCD_CURSOR_OFF)
  LCD_Cmd(LCD_CLEAR)
    Can_Init_Flags   = 0
```

```
  Can_Send_Flags   = 0
  Can_Rcv_Flags    = 0

  Can_Send_Flags   = CAN_TX_PRIORITY_0 and
                     CAN_TX_XTD_FRAME  and
                     CAN_TX_NO_RTR_FRAME

  Can_Init_Flags   = CAN_CONFIG_SAMPLE_THRICE and
                     CAN_CONFIG_PHSEG2_PRG_ON and
                     CAN_CONFIG_DBL_BUFFER_OFF and
                     CAN_CONFIG_VALID_XTD_MSG and
                     CAN_CONFIG_LINE_FILTER_OFF

CANSetOperationMode(CAN_MODE_CONFIG,0xFF)
CANInitialize(1,1,8,6,1,Can_Init_Flags)
CANSetBaudrate(1,1,8,6,1,Can_Init_Flags)
CANSetMask(CAN_MASK_B1,-1,CAN_CONFIG_XTD_MSG)
CANSetMask(CAN_MASK_B2,-1,CAN_CONFIG_XTD_MSG)
CANSetFilter(CAN_FILTER_B1_F1,1023,CAN_CONFIG_XTD_MSG)
  CANSetOperationMode(CAN_MODE_Normal,0xFF)
   Tx_ID = 1023

  text = "hex:"
  LCD_Out(1,1,text)
  text = "LCD ornek"
  LCD_Out(2,1,text)

  Delay_ms(2000)
  text  = "Gerilim: "
  while TRUE


    adc_rd  = ADC_read(2)
```

```
wordtostr(adc_rd,yaz)
 LCD_Out(1,8,yaz)

tlong = adc_rd
ch=tlong/256
if ( ch<10) then
ch=ch+48
else
ch=ch+55
end if
LCD_Chr(1,5,ch)

ch=(tlong/16) mod 16
if ( ch<10) then
ch=ch+48
else
ch=ch+55
end if
LCD_Chr(1,6,ch)

ch=tlong mod 16
if ( ch<10) then
ch=ch+48
else
ch=ch+55
end if
LCD_Chr(1,7,ch)

ch = adc_rd/256
RxTx_Data[0]=ch
ch=adc_rd mod 256
 RxTx_Data[1]=ch
CANWrite(Tx_ID, RxTx_Data, 2, Can_Send_Flags)
```

```
    LCD_Out(2,1,text)

    tlong = adc_rd * 5000
    tlong = tlong / 1023

    ch     = tlong / 1000
    LCD_Chr(2,9,48+ch)
    LCD_Chr_CP(".")


    ch    = (tlong / 100) mod 10
    LCD_Chr_CP(48+ch)
    ch    = (tlong / 10) mod 10
    LCD_Chr_CP(48+ch)
    ch    = tlong mod 10
    LCD_Chr_CP(48+ch)
    LCD_Chr_CP("V")

    Delay_ms(1000)
  wend
end.
```

ÖZGEÇMİŞ

1980 yılında Şanlıurfa'da doğdu. İlköğrenimini Samsun'da, orta öğrenimini Şanlıurfa'da tamamladı. 1998 yılında Süleyman Demirel Üniversitesi, Elektronik-Haberleşme Mühendisliğini kazandı. 2003 yılında bu bölümden mezun oldu. 2005 yılında İnönü Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Ana Bilim Dalında yüksek lisans eğitimine başladı. Halen, Şanlıurfa Türk Telekom İl Müdürlüğü, Telekomünikasyon Sistemleri Grup Müdürlüğünde mühendis olarak çalışmaktadır.