T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

I²C-BUS SERİ İLETİŞİM PROTOKOLÜ İÇİN VERİ İZLEME SİSTEMİ

YÜKSEK LİSANS TEZİ

Sedat ATMACA

Enstitü Anabilim Dalı : ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ

Bu tez 21 / 08 2002 tarihinde aşağıdaki jüri tarafından Oybirliği/Oyçokluğu ile kabul edilmiştir.

Yrd. Doç. Dr. Ahmet Turan ÖZCERİT	Doç. Dr. Hüseyin EKİZ	Yrd. Doç. Dr. Cemil ÖZ
Jüri Başkanı	**Jüri Üyesi**	**Jüri Üyesi**

ÖNSÖZ

Bu tezin hazırlanmasında değerli görüş ve düşünceleriyle yardımcı olan sayın hocam Yrd.Doç.Dr. Ahmet Turan Özcerit'e içten teşekkürlerimi sunarım. Ayrıca tezin hazırlanması sırasında her türlü fedakarlık ve destekten çekinmeyen eşime, bu günlere gelmemi sağlayan anne ve babama teşekkürü bir borç bilirim.

Haziran 2002

Sedat ATMACA
Teknik Öğretmen

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: I^2C-Bus, I^2C-Bus denetleyicisi, Seri Bus'lar, Bus monitör

I^2C (Inter Integrated Circuit) Bus, mikroişlemciler, mikrodenetleyiciler ve mikroişlemci-tabanlı küçük zeki mikroçipler arasındaki seri haberleşmede yaygın olarak kullanılmaya başlayan bir seri iletişim protokolüdür.

Bu çalışmada, I^2C-Bus protokolü açıklanarak, I^2C-Bus monitör kavramı ele alınmış ve donanım tabanlı I^2C-Bus monitör elde edilmiştir. I^2C-Bus monitör teknikleri irdelenerek, donanım ve yazılım tabanlı I^2C-monitörler arasındaki farklar ve birbirlerine göre üstünlükleri açıklanmış, 8051 mikrodenetleyicisi ile PCF8584 denetleyicisi arasındaki gerekli bağlantılar yapılarak I^2C-Bus monitör'ün donanımsal yapısı elde edilmiştir. PCF8584'ün pasif I^2C-Bus monitör olarak kullanılmasını sağlayan programın yazılmasıyla birlikte sistem bütünüyle gerçekleştirilmiştir.

Yapılan mikrodenetleyici kontrollü sistem tasarımı ile I^2C-Bus monitör edilerek, Bus'taki verilerin gözlenmesi sağlanmış ve böylece elde edilen sistemin sayısal elektronik laboratuarlarında bir test cihazı olarak kullanılması mümkün hale getirilmiştir.

I^2C BUS MONITORING SYSTEM DESIGN IMPLEMENTATION

SUMMARY

Keywords: I^2C bus, I^2C bus controller, serial buses, bus monitoring,

In this thesis, hardware based, stand-alone 100Kbit/s speed I^2C bus monitor is carried out. I^2C traffic is logged to the local on-board memory. With the help of hard and software filter the stored messages can be limited to the interesting ones.

The I^2C-bus is a protocol which supports the communication of the various chips in embedded systems or portable devices.

All bus activity including start/stop events, slave addresses, read/write requests, acknowledgments, and data are displayed on computer.

This thesis explains software and hardware specification for monitoring an I^2C-bus with the standard of 80c51 microcontroller. The PCF8584 is a controller which can listen to I^2C-bus. This chip can be used to listen and monitor the actual data on the I^2C-bus. It provides what is going on the I^2C-bus. It has several internal register to tell it what to do and how to act upon the I^2C-bus line. The 80c51 sends control and data bytes to control the PCF8584 in monitor mode.

I^2C-bus monitör system can be used for testing available signals on the I^2C-bus and error conditions. Because of this, it can be used as a test equipment in a digital laboratory. I^2C-bus actions are logged to the on-board memory and the system designed filters the stored messages.

BÖLÜM 1. GİRİŞ

Son yıllarda, mikroişlemciler, mikrodenetleyiciler ve mikroişlemci-tabanlı küçük zeki mikroçipler arasındaki seri haberleşmede I^2C (Inter Integrated Circuit) Bus, yaygın olarak kullanılmaya başlanmış ve özellikleri (hız, adresleme kapasitesi) gün geçtikçe arttırılmıştır.

Bu çalışmada, I^2C-Bus'ın monitör edilmesi amaçlanmıştır. Yapılan mikrodenetleyici kontrollü sistem tasarımı ile I^2C-Bus monitör edilerek, Bus'taki verilerin gözlenmesi sağlanmış ve böylece elde edilen sistemin sayısal elektronik laboratuarlarında bir test cihazı olarak kullanılması mümkün hale getirilmiştir.

I^2C-Bus, Seri-Data (SDA) ve Seri Saat (SCL) olmak üzere yalnıza iki hattan meydana gelir ve Bus elemanları arasındaki seri iletişim bu iki hat üzerinden yapılır. Bus'taki iletişimin gerçekleştirilmesi ya donanım-tabanlı yada yazılım-tabanlı olarak gerçekleştirilir. Donanım-tabanlı I^2C-Bus'larda I^2C-Bus protokolünü bit yada bayt-tabanlı olarak destekleyen tümdevre elemanları vardır. Bu elemanların, mikroişlemcili sistemlerle gerekli bağlantısı yapılarak iletişim sağlanır. Yazılım-tabanlı I^2C-Bus'lar da ise çok az sayıda eleman kullanarak I^2C-Bus protokolü bit-bit, bit-tabanlı olarak gerçekleştirilir. Bu çalışmada I^2C-Bus ve I^2C-Bus monitor, donanım tabanlı I^2C-Bus elemanları kullanılarak gerçekleştirilmiştir. I^2C-Bus, 80c652 mikrodenetleyicisi, I^2C-Bus monitor ise PCF8584 mikrodenetleyicisi yardımı ile gerçekleştirilmiştir. 80c652 mikrodenetleyicisi ile PCF8583 gerçek zaman saati kullanılarak I^2C-Bus destekli saat yapılmış ve PCF8584 I^2C-Bus denetleyicisi de monitör modunda çalıştırılarak, I^2C-Bus'ın monitör edilmesi sağlanmıştır.

Tezin ikinci bölümünde, I^2C-Bus'ın tanımı, genel karakteristikleri, kapasitesi gibi temel kavramlar açıklanmıştır.

Üçüncü bölümde, I^2C-Bus'ı donanımsal olarak bayt tabanlı olarak destekleyen PCF8584 denetleyicisi açıklanmıştır. PCF8584'ün pin tanımlamaları, fonksiyon tanımları ve özellikleriyle birlikte monitor modunda nasıl kullanılacağı bu bölümde izah edilmiştir.

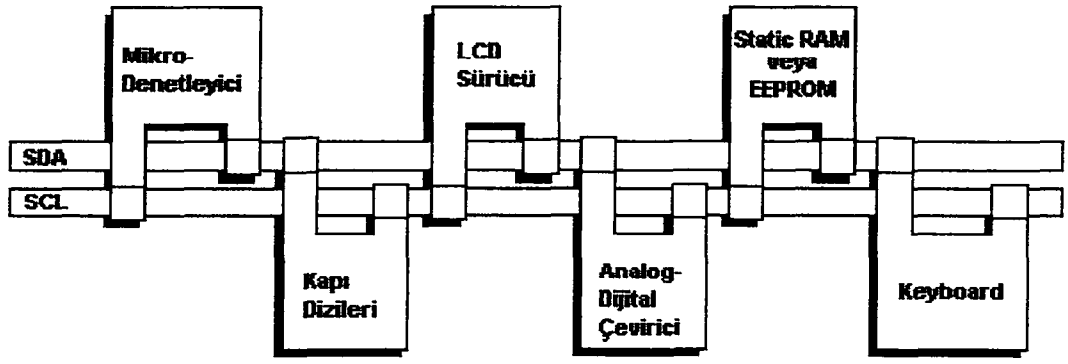
Dördüncü bölümde, I^2C-Bus monitör teknikleri irdelenerek, donanım ve yazılım tabanlı I^2C-monitörler arasındaki farklar ve birbirlerine göre üstünlükleri açıklanmış, 8051 mikrodenetleyicisi ile PCF8584 denetleyicisi arasındaki gerekli bağlantılar yapılarak I^2C-Bus monitör'ün donanımsal yapısı gerçekleştirilmiştir.

Sonuç bölümünde, yapılan I^2C-Bus monitör devresinden elde edilen sonuçlar, ve sistemin genel özellikleriyle birlikte iyileştirilebilmesi için nelerin yapılabileceği açıklanmıştır.

BÖLÜM 2. I^2C-BUS ve ÖZELLİKLERİ

2.1. I^2C-Bus Kavramı

I^2C (Inter Integrated Circuit) Bus, Seri Data(SDA) ve Seri Saat (SCL) olmak üzere iki hattan oluşur. Bus'a bağlı elemanlar arasındaki iletişim, bu iki hat üzerinden yapılır. Bus'taki her bir eleman yalnız bir adresle tanınır (Mikrodenetleyici, LCD sürücü, Hafiza, Klavye Arabirimi) ve özelliğine göre Gönderici (Transmitter), Alıcı (Receiver) veya hem Gönderici hem de Alıcı (Bus'a veri gönderebilir, alabilir veya her ikisini birden yapabilir) olarak çalışır. Örneğin Bus'taki eleman bir LCD sürücü ise sadece Bus'tan gelen verileri alır, mikrodenetleyici ise Bus'a veri gönderebilir, alabilir ve Bus'ı kontrol edebilir. Şekil 2.1'de, benzer bir I^2C-Bus sistem blok diyagramı gösterilmiştir. Burada Bus'a bağlı Gönderici veya Alıcı' lardan her biri uygun veri iletim özelliğine göre Yönetici(Master) veya Yönetilen(Slave) olarak seçilir. Yönetici eleman, veri iletimini başlatır, durdurur ve iletim için gerekli saat sinyallerini üretir. Bus'taki Yönetici tarafından adreslenmiş her bir eleman ise, Yönetilen olarak adlandırılır.



Şekil 2.1 Örnek I^2C-Bus Sistem

I^2C-Bus, Çok-Yöneticili-Bus'tır. Yani Bus'a bağlı birden fazla elemanın Bus'ı kontrol etme kabiliyeti vardır. Birden fazla yönetici'ın kullanıldığı Bus'larda Yönetici'ler genellikle mikrodenetleyiciler'dir ve Bus'ın kontrolu bunlar tarafından yapılır.

Çok-Yöneticili I^2C-Bus'larda Yönetici'ler aynı anda data iletimini başlatmaya çalışabilirler, bu durumda oluşan kaos'u önlemek için arbitrasyon(arbitration) işlemi kullanılır. Buna göre Yönetici'ler tarafından üretilen saat sinyalleri, arbitrasyon işlemi boyunca SCL hattının VE-bağlantılı lojik hattı (giriş hatlardan herhangi biri Düşük(LOW) seviyeye çekildiği anda çıkışı düşük seviyeye çekme) ile senkron hale getirilir.

I^2C-Bus'taki saat sinyallerinin üretiminden daima Yönetici eleman sorumludur. Her bir Yönetici, veri göndereceği zaman kendi saat sinyallerini üretir. Herhangi bir Yönetici'den Bus'a uygulanan saat sinyal genişliği, sadece saat hattını düşük seviyeye çekmeye çalışan bir yavaş Yönetilen veya Bus'ı ele geçirmeye çalışan bir başka Yönetici tarafından uzatılabilir.

2.2 I^2C-Bus'ta Kullanılan Terimler ve Tanımları

Tablo 2.1'de I^2C-Bus'ta kullanılan terimler ve terimlerin ne anlama geldikleri açıklanmıştır.

Tablo 2.1 I^2C-Bus'ta kullanılan terimler ve tanımları

TERİM	AÇIKLAMA
Gönderici (Transmitter)	I^2C-Bus'a veri gönderen elemana denir.
Alıcı (Receiver)	I^2C-Bus'dan veri alan elemana denir.
Yönetici (Master)	I^2C-Bus'taki veri transferini başlatan, durduran Ve veri transferi için gerekli saat sinyallerini Üreten elemana denir.

Tablo 2.1 (Devam) I^2C-Bus'ta kullanılan terimler ve tanımları

TERİM	AÇIKLAMA
Yönetilen (Slave)	Yönetici tarafından adreslenen elemana denir.
Multi-Master (Çoklu-Master)	Birden fazla yöneticini aynın zamanda Bus'taki veriyi bozmadan Bus'ın kontrolünü ele alamaya çalışmasıdır.
Arbitrasyon (Arbitration)	Eğer birden fazla yönetici aynı zamanda Bus'ın Kontrolünü ele almaya çalışırsa, bunlardan sadece birinin Bus'ı kontrol etmesi sağlanır ve veri kesilmeden iletilir.
Senkronizasyon (Synchronization)	İki veya daha fazla saat sinyalinin senkron hale getirilmesidir.

2.3. I^2C-Bus ve Diğer Standart Seri İletişim Protokolleri

Son yıllarda, elektronik ve bilgisayar mühendisliğinde en çok kullanılan seri iletişim protokolleri RS-232, SPI(Serial Peripheral Interface), Microwire, CAN(Controller Area Network) olarak sayılabilir. Bunun yanısıra USB(Universal Serial Bus), IEEE-1394 ve IrDA(Infrared Data Association)'da bilgisayar ve tüketici elektroniğinde kullanılmaya başlayan yeni seri iletişim protokolleridir. Son 3 protokol, bilgisayar ile sayısal kamera, video-kamera, tarayıcı ve yazıcı gibi çeşitli elektronik birimler arasında arabirim sağlamak için kullanılır.

Aşağıda I^2C-Bus'ın seçilme nedenleri verilmiştir:

- Sadece 2 hatta sahip olması.
- Eş zamanlı iletişimi
- TTL gerilim uyumluluğu
- Sadelik ve basitliği
- Çok-yönetici ve arbitrasyon (arbitration) desteği

- Yüksek hızı
- Mimari yapısı mikrodenetleyicili ve zeki sistemler için uygundur.

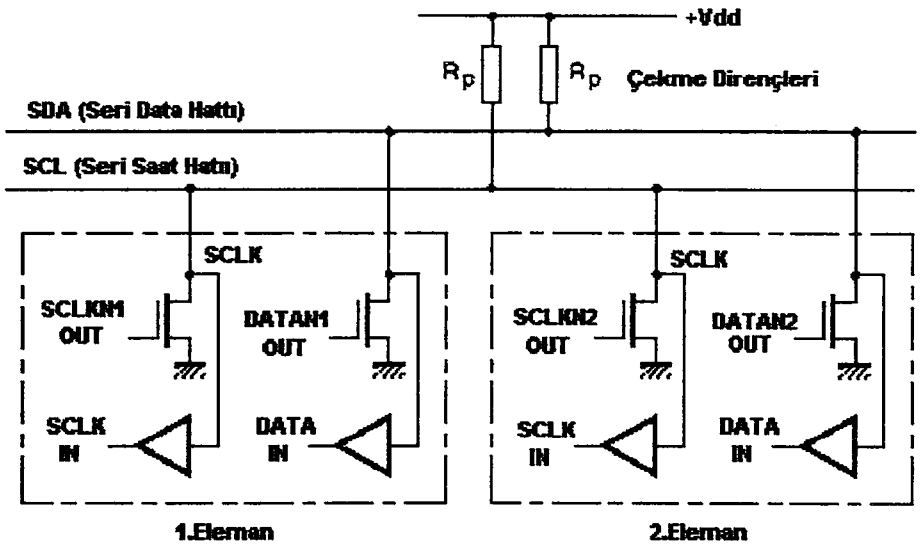
Tablo 2.2 Bazı standart seri iletişim protokolleri ve I^2C

Seri Arabirim	**Seri Saat**	**En fazla ünite sayısı**	**En fazla hat uzunluğu(m)**	**En fazla hız(bit/sn)**	**Hat sayısı**
RS-232	Yok	2	15-30	115K	2
IEEE-1394	Yok	64	5-6	400M	4
USB	Yok	127	5-6	12M	2
Microwire	Var	8	3-4	2M	3
SPI	Var	8	3-4	2.1M	3
CAN	Yok	127	40	1M	2
I^2C	Var	40	5-6	400K	2

Tablo 2.2'de seri iletişim protokolleri değişik kriterlere göre değerlendirilmiştir. Bunlardan en önemlileri destekleyebildikleri eleman sayılarıdır. SPI ve Microwire, 8 eleman'dan fazlasını desteleyememektedir. Bu yüzden bazı sistem tasarımları için uygun değildirler.

Seri iletişim arabirimleri için diğer bir kriter, Çok-Yöneticili-Bus fonksiyonlarını desteklemesidir. RS-232 ve Microwire Çok-Yöneticili-Bus fonksiyonlarını desteklememektedir. CAN Bus, orta çaplı sistemler için arabirim sağlayan seri iletişim protokolü olarak düşünülebilir. CAN, TTL gerilim seviyelerini desteklememektedir. CAN Bus'ın düşük seviyesi 1,75Volt, yüksek seviyesi ise 3,25Volt'tur.

2.4. I²C-Bus'ın Genel Karakteristikleri



Şekil 2.2. Standart ve hızlı moddaki I²C elemanlarının Bus'a bağlantısı

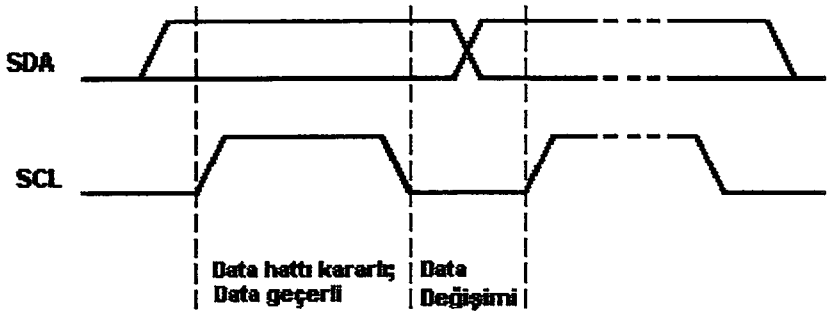
I²C-Bus'taki Seri-Veri(SDA) ve Seri-Saat(SCL) hatlarının her ikisi de iki yönlü çalışan hatlardır ve Şekil 2.'de de gösterildiği gibi çekme dirençleri veya bir akım kaynağı üzerinden pozitif gerilim kaynağına göre bağlanırlar. Bus'ta iletim olmadığında, yani Bus boş olduğunda, her iki hatta yüksek(HIGH) seviyededir. Bus'a bağlanacak elemanların çıkış uçları, Seri-Veri ve Seri-Saat' in wired-AND fonksiyonunu gerçekleştirebilmek için açık-oluk(open-drain) veya açık-kollektör(open-collector) olması gerekmektedir. I²C-bus'taki veriler, standart modda 100kbit/sn, hızlı modda 400kbit/sn, yüksek hızlı hodda ise 3,4Mbit/sn hızlarında iletişim kurulabilir. Bus'a bağlanacak eleman sayısı bus'ın kapasitesi ile sınırlıdır (400pf).

2.5. Bit İletimi

I²C bus'a bağlı elemanlar CMOS, NMOS veya bipolar gibi değişik teknolojilere sahip olabileceğinden bus'taki lojik-0 (LOW) ve lojik-1 (HIGH) sinyal seviyeleri sabit değildir ve Vdd seviyesine bağlı olarak değişir. Her bir veri bitinin iletimi için bir saat palsi üretilir.

2.5.1. Veri'nin Geçerliliği

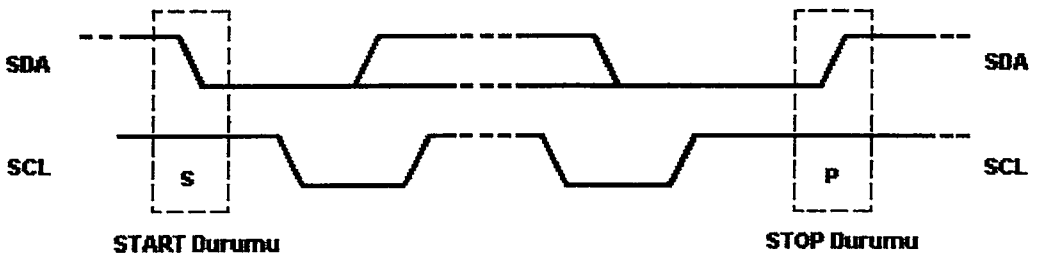
Şekil 2.3'de görüldüğü gibi saat palsinin yüksek peryodu boyunca, Seri-Data hattındaki Veri, kararlı durumda olmalıdır. Saat sinyali'nin düşük olduğu süre içerisinde SDA hattı, düşük veya yüksek konumlarını değiştirebilir.



Şekil 2.3. I^2C-Bus'taki bit iletimi

2.5.2. Start ve Stop Durumları

I^2C-Bus'ın bu özel durumları Şekil 2.4'de gösterilmiştir. Seri-veri hattı yüksek'ten düşüğe geçerken, saat sinyali yüksek seviyesinde olursa, bu durum tek uygun START durumudur.



Şekil 2.4. I^2C-Bus START ve STOP durumları

Seri-veri hattı düşük'ten yükseğe geçerken, saat sinyali yüksek seviyede kalırsa bu durumda STOP durumu olarak tanımlanır.

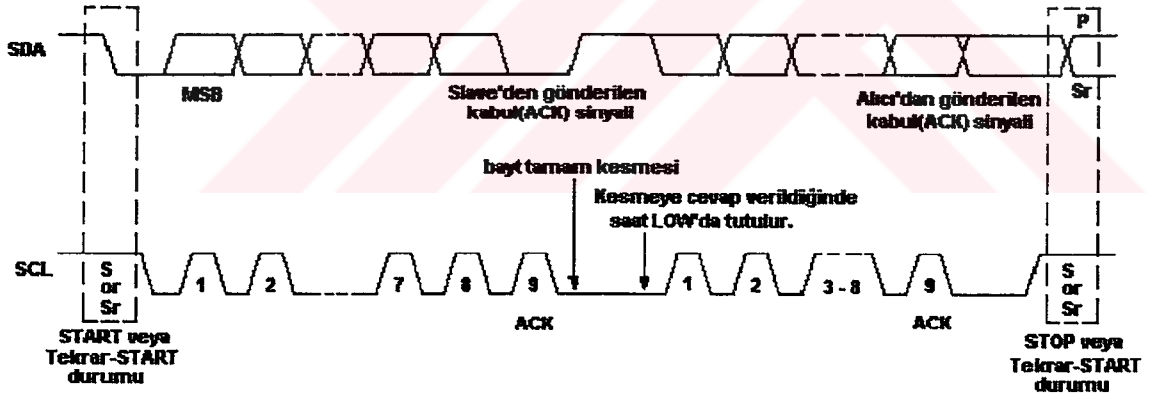
START ve STOP durumları daima yönetici tarafından üretilir. START durumundan sonra Bus'ın kullanımda olduğu, STOP durumundan sonra ise Bus'ın boş, yani kullanılabilir olduğu düşünülür.

Bus'ta STOP durumu yerine tekrar START durumu gönderilebilir. Bu durum Şekil 2.5'de START(S) ve Tekrarlanmış-START (SR) olarak gösterilmiştir.

Gerekli arayüz donanıma sahip Bus'a bağlı elemanların, START ve STOP durumlarını algılaması kolaydır. Ancak, böyle bir arayüzü olmayan mikrodenetleyiciler, geçişi hissedebilmeleri için Seri-veri hattını her saat periyodunda en az iki kere örneklemek zorundadır.

2.6. Veri İletimi

2.6.1. Bayt formatı



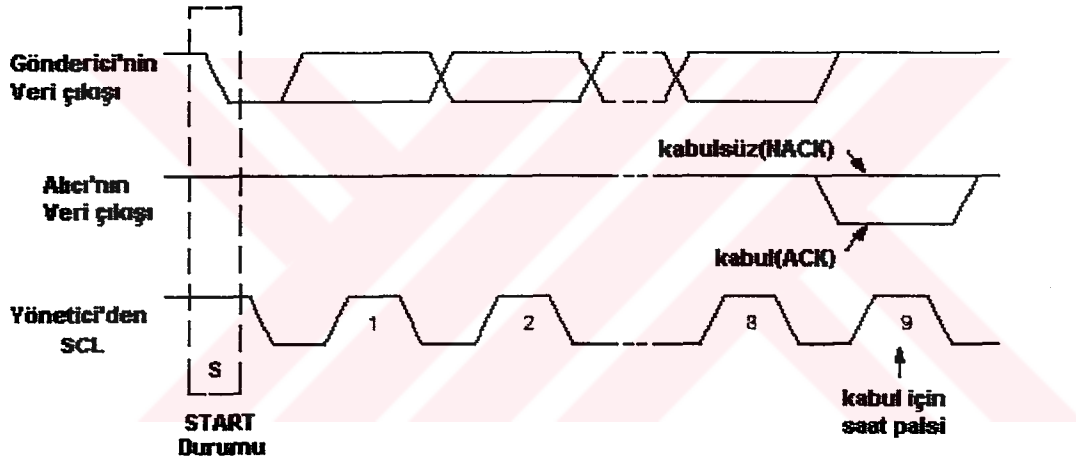
Şekil 2.5. I²C-Bus'ta veri iletimi

I²C-bus'ta iletilecek her veri 8-bit uzunluğunda olmalıdır. İletilecek bayt sayısında bir sınırlama yoktur. İletilen her bir bayt'ı kabul biti takip eder. Veri iletimine MSB bitinden başlanır. Eğer yönetilen başka bir işten dolayı (harici kesme cevabı) bayt'ların tamamını alamıyor veya gönderemiyorsa bu durumda saat, Yönetici tarafından LOW seviyesinde tutulur ve Yönetilen hazır hale geldikten sonra iletilir.
Bazı durumlarda, I²C-bus formatından başka formatların da kullanılmasına müsaade edilir Bu durumda her bir bayt iletim boyunca (CBUS uyumlu elemanlar için)

START bitiyle başlar ve STOP bitiyle son bulur. Burada non-acknowledge (NACK) biti üretilir.

2.6.2. Kabul (Acknowledge) sinyali

I^2C-Bus'ta veri iletimi için kabul sinyali kullanmak zorunludur. Kabul için gerekli saat sinyali yönetici tarafından üretilir. Gönderici, kabul saat palsi boyunca Seri Veri hattını yükseğe çekerek Bus'ı serbest bırakır.
Alıcı kabul saat palsi boyunca Seri Veri hattını düşük seviyeye çeker. Bu durum Şekil 2.6'da gösterilmektedir. Kabul saat palsinin yüksek periyodu boyunca SDA hattı kararlı düşük seviyededir.



Şekil 2.6. I^2C-Bus'ta kabul işlemi

Bus'taki adreslenmiş Alıcı, her bir baytı aldıktan sonra kabul sinyali üretmek zorundadır.

Yönetilen eleman Yönetilen-adres baytından sonra kabul sinyali göndermezse, SDA hattı Yönetilen tarafından yüksek seviyeye çekilir ve veri iletimi gerçekleşmez. Bu durumda yönetici, iletimi bitirmek için ya STOP durumu, yada yeni iletim için tekrarlı-START durumu üretecektir.

Eğer Yönetilen-Alıcı, Yönetici'in ürettiği Yönetilen-adrese kabul sinyali gönderir, fakat iletimden bir süre sonra veri iletimine cevap vermezse (kabul sinyali üretmezse) bu durumda yönetici iletimi kesecektir. Bu Yönetilen'in ilk bayt'tan sonra NACK sinyali ürettiğini gösterir. Bu durumda Yönetilen, data hattını yüksek seviyeye çeker ve Yönetici de, STOP veya tekrarlı-START üretir.

Bir Yönetici-alıcı bilgi iletimine katılmışsa, Yönetilen-Göndericiye gelen bilginin sonunda bir kabul sinyali göndererek işaretlemelidir.

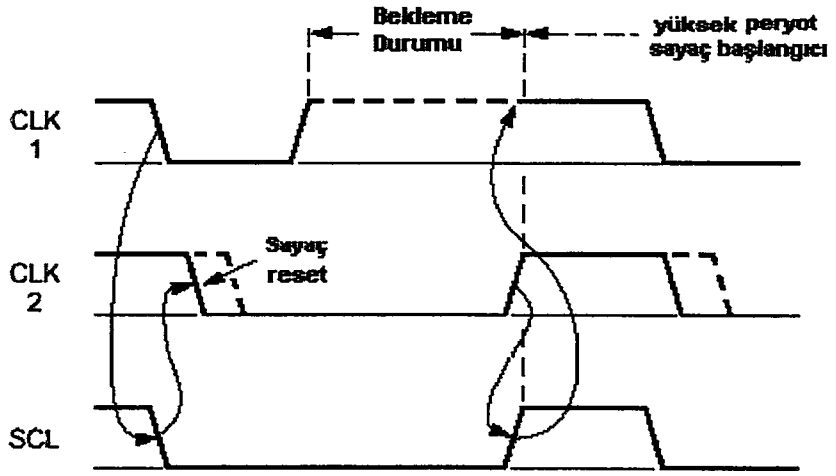
2.7. Arbitrasyon(Arbitration) ve Saat Üretimi

2.7.1. Senkronizasyon

Bütün Yönetici'ler I^2C-Bus'ta veri iletimi için SCL hattında kendi saat sinyallerini üretirler. Veri sadece saat sinyalinin yüksek peryodu boyunca geçerlidir. Bus'ta bit-bit arbitrasyon işlemi'nin meydana gelebilmesi için belirlenmiş bir saat sinyaline ihtiyaç duyulur.

Saat senkronizasyonu SCL hattında VE-bağlantılı lojik sayesinde gerçekleştirilir. Bu işlem şu şekilde yapılır: SCL hattının yüksekten düşüğe geçişi, ilgili I^2C-Bus elemalarının düşük-peryot sayaçlarının sıfırlanmasına neden olur. Bir I^2C-Bus elemanının saat'i düşüğe gittiğinde, SCL hattı yüksek seviyeye erişinceye kadar bus'ı bu seviyede tutar. Bununla birlikte saat'in düşük peryodu boyunca bir başka Saat'in düşükten yükseğe geçişi SCL hattının durumunu değiştirmeyecektir. Saat hattı, en uzun düşük peryoda sahip elemanın düşük peryodu boyunca düşük seviyede kalacaktır. Bu esnada kısa düşük peryoda sahip bus elemanları yüksek bekleme durumunda kalacaktır. İlgili bütün I^2C-Bus elemanları düşük-peryot sayaçlarını sıfırladıktan sonra, saat hattı serbest bırakılacak ve yüksek seviyeye geçecektir. Bu durumdan sonra I^2C-Bus elemanlarının saat'leri ile SCL hattı arasında bir fark olmayacak ve bütün I^2C-Bus elemanları kendi yüksek-peryot sayaçlarını sıfırlayacaktır. Kendi yüksek peryodunu tamamlayan ilk I^2C-Bus elemanı tekrar SCL hattını düşük seviyeye çekecektir.

Senkronizeli saat sinyali en uzun düşük peryoda sahip elemanın düşük peryodu ile en kısa yüksek peryoda sahip elemanın yüksek peryodu tarafından belirlenmiş olacaktır.



Şekil 2.7. Arbitrasyon işlemi boyunca saat senkronizasyonu

2.7.2. Arbitrasyon (Arbitration)

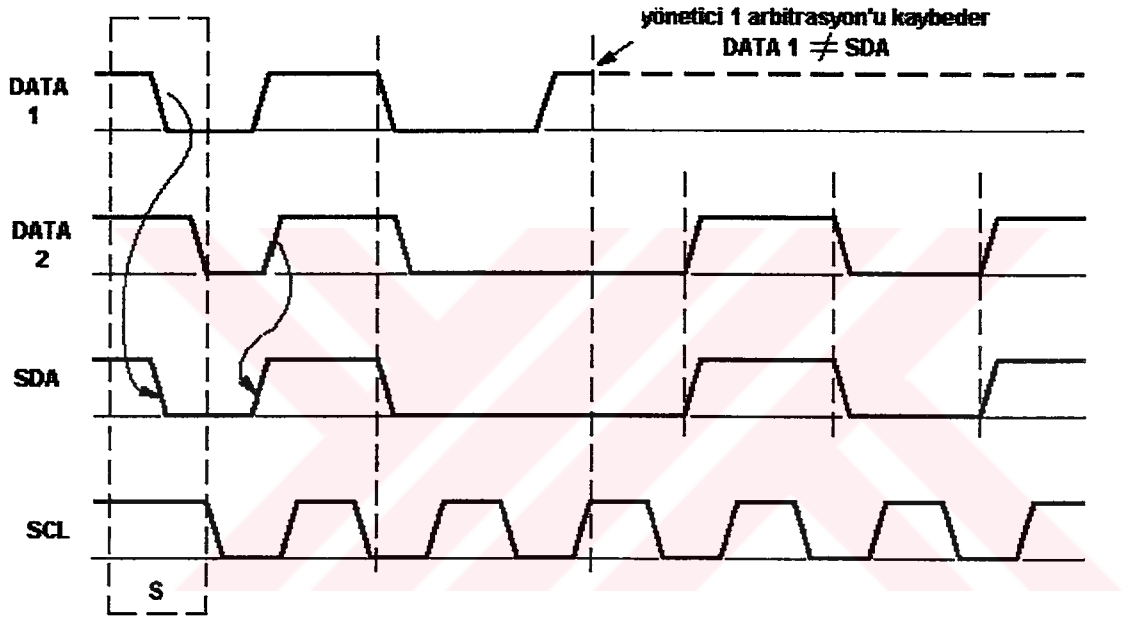
Bir Yönetici veri iletimini sadece Bus boş olduğunda, yani Bus'ta iletim olmadığında başlatabilir. İki yada daha fazla yönetici, START durumunun minimum tutma süresi içerisinde START durumu üretebilir ve bus'ta belirlenmiş bir START durumunun oluşmasına neden olur.

Arbitrasyon, SCL hattı yüksek seviyede iken, SDA hattında meydana gelir. Bir yönetici Bus'ı düşük seviye çekmeye çalışırken, bir başka yönetici Bus'ı yüksek seviyeye çekmeye çalışırsa, bu yönetici data çıkışını kesecektir. Çünkü Bus'taki SDA seviyesi ile yönetici'nin kendi data çıkış seviyesi aynı değildir.

Arbitrasyon, birçok bit boyunca devam edebilir. Bunun ilk aşaması adres bitlerinin karşılaştırılmasıdır. Eğer Yönetici'ler aynı adrese sahip elemanı adreslerse ve bunlar yönetici-gönderici iseler, arbitrasyon işlemi, veri bitlerinin karşılaştırılmasıyla devam eder. Yönetici'ler yönetici-gönderici değilde, yönetici-alıcı ise bu sefer de arbitrasyon işlemi, kabul bitinin karşılaştırılmasıyla devam eder. Çünkü I^2C-Bus'taki

adres ve veri bitleri arbitrasyon'u kazanan yönetici tarafından belirlenir. Arbitrasyon işlemi boyunca bus'taki veriler kaybolmaz.

Şekil 2.8'de İki yönetici için arbitrasyon işlemi gösterilmektedir. Şekilde görüldüğü gibi gerçek SDA hattı ile yönetici'lerin biri tarafından üretilen SDA 1'in dahili veri seviyeleri arasında fark vardır. Bundan dolayı bu fark başladığı anda DATA 1 çıkışı durdurulur ve çıkış seviyesi yüksek konuma getirilir. Bu durum arbitrasyon'u kazanan yönetici tarafından başlatılan data iletimini etkilemeyecektir.



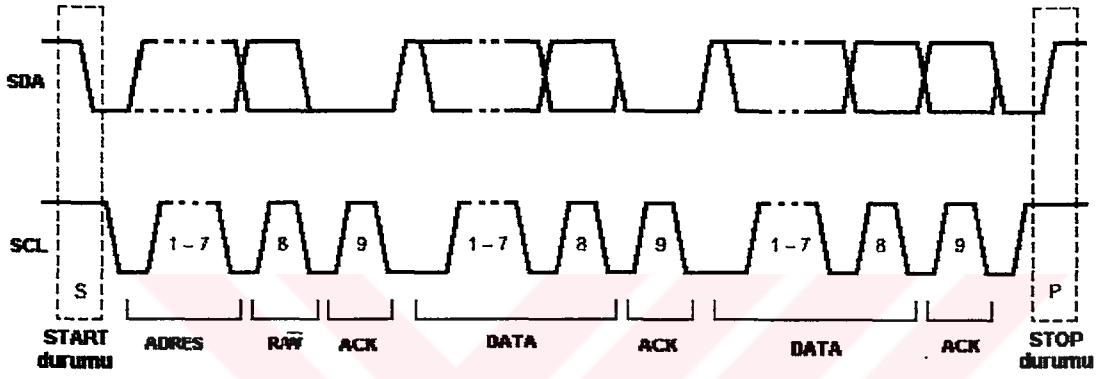
Şekil 2.8. İki yönetici arasındaki arbitrasyon işlemi

I^2C-Bus'ın kontrolüne karar verilinceye kadar, kontrolü elde etmeye çalışan yönetici'ler tarafından yalnızca adres veya yönetici kodu ve data gönderilir. Bus'ta merkezi veya öncelikli yönetici durumu yoktur. Bütün yönetici'ler eşit önceliklidir.
Seri veri iletimi boyunca, arbitrasyon işlemi esnasında bus'a tekrarlı-START veya STOP durumu iletilirse, bu duruma dikkat edilmelidir. Çünkü bu durumda arbitrasyon'a izin verilmez. Aşağıda hangi durumlar arasında arbitrasyon'a izin verilmeyeceği gösterilmiştir:

- Tekrarlı-START durumu ile Veri biti
- STOP biti ile Veri biti
- Tekrarlı-START durumu ile STOP biti

Yönetilen arbitrasyon işlemine'ne karışmaz.

2.8. 7-Bit Adresleme ile Veri Formatları



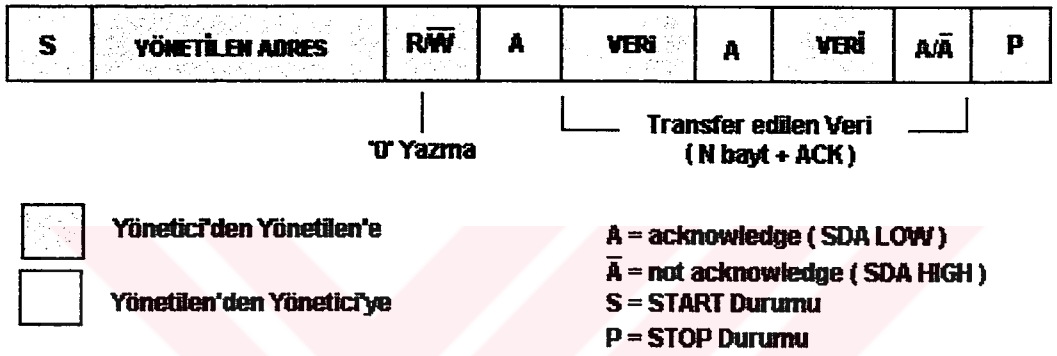
Şekil 2.9. I^2C-Bus'ta komple bir veri iletimi

I^2C-Bus'taki veri iletimi Şekil 2.9'da gösterilmektedir. START (S) durumundan sonra, bir Yönetilen adresi gönderilir. 7-bit uzunluğundaki bu bitleri veri yön biti olan R/W biti takip eder. Bu bitin '0' olması yazma işleminin yapılacağını '1' olması ise okuma işleminin yapılacağını gösterir. Veri iletimi daima yönetici tarafından üretilen STOP (P) durumu ile sonlandırılır. Bununla birlikte eğer Yönetici, iletimi devam ettirmek isterse tekrarlı-START durumu üretebilir ve STOP durumu üretmeden bir başka Yönetilen'i adresleyebilir.

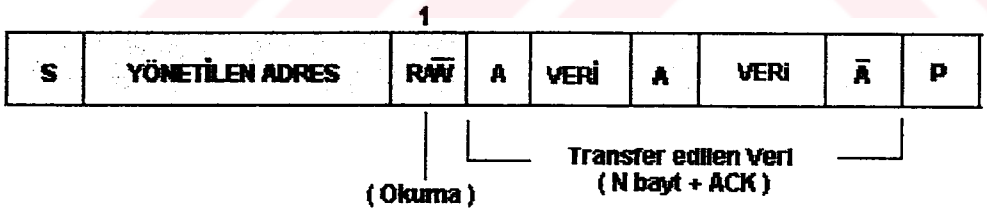
İletim esnasında R/W formatlarının değişik kombinasyonları vardır. Mümkün olan veri iletim formatları şunlardır:

- Yönetici-gönderici, Yönetilen-alıcı'ya gönderir bu sırada iletimim'in yönü değiştirilmez.

- Yönetici ilk bayttan sonra Yönetilen'i okur. İlk ACK bitinden sonra, yönetici-gönderici, yönetici-alıcı ve Yönetilen-alıcı, Yönetilen-gönderici olur. İlk ACK biti sadece Yönetilen tarafından üretilir. STOP durumu, NACK bitinden sonra yönetici tarafından üretilir.
- Şekil 2.11'de kombine veri formatı görülmektedir. İletim yönünün değişiminde START durumu ve Yönetilen adreslenmesi tekrarlanır, R/W biti terslenir. Eğer yönetici-alıcı tekrarlı-START durumu gönderirse öncesinde NACK biti gönderir.



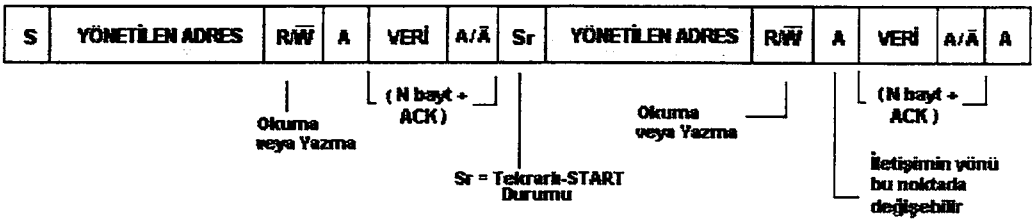
Şekil 2.10. Yöneten-Gönderici'nin Yönetilen-Alıcı'yı adreslemesi



Şekil 2.11. Yönetici'nin ilk bayttan sonra Yönetilen'i okuması

- Birleşik format seri hafızayı kontrol etmek için kullanılabilir. Bu durumda ilk data baytında dahili hafıza adresi yazılmalıdır. START durumu ve Yönetilen adresten sonra veri iletimi yapılabilir.
- Hafıza ile iletişimde hafıza adresleri otomatik olarak bir arttırılabilmekte veya azaltılabilmektedir. Bu işlem ilgili I^2C-Bus elemanını üreten firma tarafından belirlenir.

- Her biti bayt A veya \A ile gösterilen ACK biti takip eder.



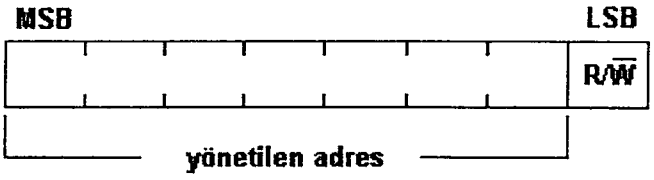
Şekil 2.12. Birleşik format

- START durumundan hemen sonra STOP durumun gelmesi geçersiz I^2C-bus formatıdır.

2.9. 7-Bit Adresleme

I^2C-Bus'taki adresleme işlemi genellikle, START durumundan sonra gönderilen bayt ile belirlenir. Yönetici tarafından gönderilen bu bayt ile hangi Yönetilen'in seçileceğini belirler. Genel Çağlı(General Call) adres bunların dışındadır ve bütün Bus elemanlarını adresleyebilir. Bu adres kullanıldığında teoride, bütün elemanlar ACK biti ile cevap verirler. Bununla birlikte bus elemanları bu adresi görmezden gelebilir. Genel Çağrı'nın ikinci baytı yapılacak işlemi belirler.

2.9.1. İlk Bayt'taki Bitlerin Tanımlanması



Şekil 2.13. START durumundan sonraki ilk bayt

Şekil 2.13'de görüldüğü gibi ilk baytın ilk 7 biti Yönetilen adresini oluşturur. 8.bit LSB bitidir ve aktarılacak mesajın yönünü belirler. Bu bitin '0' olması, Yönetici'nin

ilgili Yönetilen'e yazma işlemi yapacağını, '1' olması ise ilgili Yönetilen'den okuyacağını gösterir.

Adres gönderildiğinde sistemdeki bütün elemanlar START-durumundan sonraki 7 bit ile kendi adreslerini karşılaştırırlar. Eğerler adresler birbirlerine eşitse, bu eleman Yönetici tarafından R/W bitine bağlı olarak Yönetici-Alıcı veya Yönetilen-Gönderici olacak şekilde seçilir.

Yönetilen-Adres, sabit veya programlanabilir kısımlardan oluşturulabilir, bundan dolayı sistemde birden fazla eleman bulunabilir. Yönetilen-Adresin programlanabilir kısmı bus'a bağlanabilecek maximum eleman sayısını belirler. Bir elemanın programlanabilen adres bitlerinin sayısı, aynı elemandan Bus'a kaç adet bağlanacağını belirler. Örneğin bir eleman, 4 sabit ve 3 programlanabilen adres bitine sahipse bus'a bu elamandan toplam 8 tane bağlanabilir.

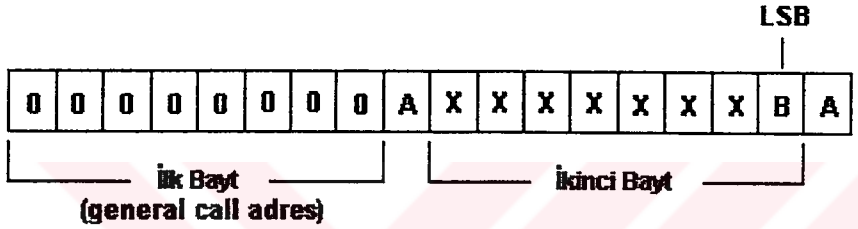
I^2C-Bus kurulu I^2C-adres yerleşim tablosunu koordine etmiştir. Tablo 2.3'de görüldüğü gibi 8 adres'in iki grubu ilerki işlemler için ayrılmıştır. 111110XX bit kombinasyonu ise 10-bit adresleme için kullanılır.

Tablo 2.3 İlk bayttaki bitlerin tanımlanması

Yönetilen Adres	**R/W Biti**	**Açıklama**
0000 000	0	Genel Çağrı Adresi
0000 000	1	START bayt
0000 001	X	CBUS adres
0000 010	X	Diğer bus formatları için ayrılmıştır.
0000 011	X	İlerki işlemler için ayrılmıştır.
0000 1XX	X	Hs-Mode Yönetici Kod
1111 1XX	X	İlerki işlemler için ayrılmıştır.
1111 0XX	X	10-bit Yönetilen Adresleme

2.9.9.1. Genel Çağrı Adresi (General Call Adres)

Genel Çağrı Adresi, I^2C-Bus'a bağlı bütün elemanların adreslenmesinde kullanılır. Bununla birlikte Genel Çağrı Adresi ile adreslenmiş bir elemanın verilerine ihtiyaç yoksa, bu eleman NACK biti ile göz ardı edilebilir. Eğer Genel Çağrı Adresi ile adreslenmiş bir elemanın verilerine ihtiyaç varsa, bu eleman bu adres ACK biti ile karşılık verir ve Yönetilen-Alıcı gibi davranır. İkinci ve izleyen baytlara bütün Yönetilen-Alıcı'ler tarafından ACK biti ile karşılık verilir. Yönetilen işleyemeyeceği baytları NACK biti ile ihmal etmelidir. Genel Çağrı Adresi'nin anlamı daima ikinci bit ile belirlenir. Bu durum Şekil 2.14'de gösterilmiştir.



Şekil 2.14. Genel Çağrı Adres formatı

Burada düşünülecek iki durum vardır:

- LSB biti B'nin '0' olma durumu.
- LBS biti B'nin '1' olma durumu.

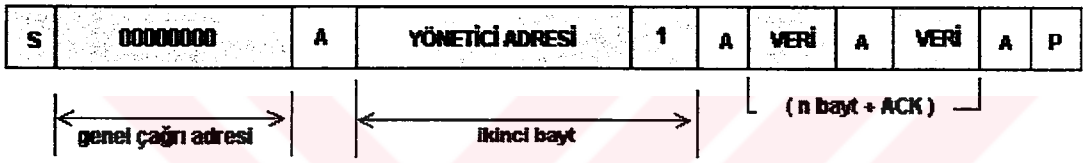
B biti '0' olduğunda ikinci bayt şu şekilde tanımlanır:

- 00000110 (06H) : Donanım ile resetleme yapılır. Yönetilen-Adresin programlanabilir parçası yazılır. I^2C-Bus'taki elemanlar sıralı gelen iki baytlık General Call Adrese cevap verecek şekilde dizayn edilmiştir ve bununla kendilerini resetleyip programlanabilir parçalarını alacaklardır.
- 00000100 (04H) : Yönetilen-Adresin programlanabilir parçasına donanım ile yazılır. Donanım ile adreslerinin programlanabilir kısımları tanımlanmış bütün elemanlar bu programlanabilir parçayı tutacaktır. Elemanlar resetlenmeyecetir.
- 00000000 (00H) : Bu kodun 2.bayt olarak kullanılmasına izin verilmez.

Geri kalan kodlar tanımlanmamıştır ve elemanlar tarafından ihmal edilmek zorundadır.

B biti '1' olduğunda ikinci bayt şu şekilde tanımlanır:

Bu durumda sıralı iki bayt donanım-genel çağrı adresidir.. Bu durum, sıralı datanın klavye tarayıcısı gibi donanım yönetici'si tarafından iletileceğini gösterir. Yani Yönetilen adrese iletmek için programlanmaz. Yönetici hangi elemana mesajın iletileceğini bilmediğinden sadece donanım-genel çağrı adresi ve kendi adresini üretebilir. Bu durum Şekil 2.15'de gösterilmiştir.



Şekil 2.15. Donanım Yönetici-Göndericisi tarafından veri iletimi

İkinci bayttaki kalan 7-bit donanım-yönetici'sinin adresini içerir. Bu adres mikrodenetleyici gibi bus'a bağlı zeki elemanlar tarafından tanınabilir. Eğer donanım-yöneticisi, Yönetilen gibi davranırsa, Yönetilen-adres aynı yönetici adresi gibi olur.

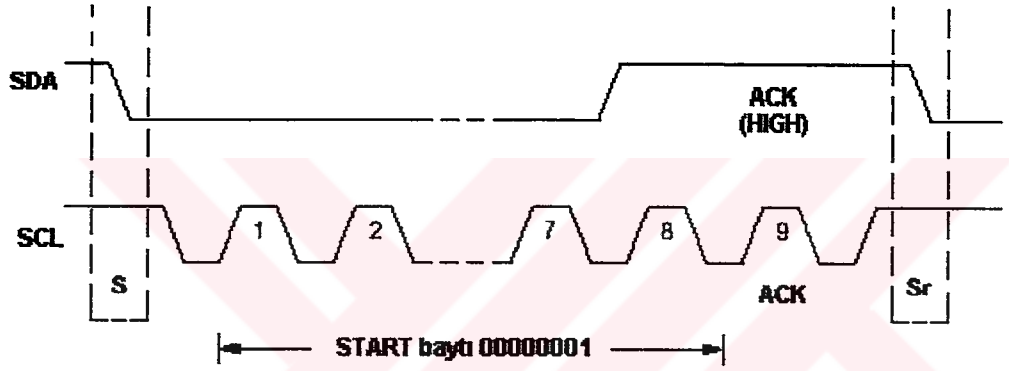
2.10. START Bayt

Mikrodenetleyiciler I^2C-Bus'a iki şekilde bağlanabilir. Eğer bir mikrodenetleyici kendi tümdevresi üzerinde I^2C donanımına sahipse, denetleyici, bus'tan gelen kesme isteğine cevap verecek şekilde programlanmalıdır. Eğer Bus'a bağlanan eleman, böyle bir donanımı desteklemiyorsa, Bus'la iletişimi tamamen yazılım ile yapılmalıdır. Bu yüzden mikrodenetleyici zamanının çoğunu Bus'u taramakla ve geri kalan zamanını da tasarlanan fonksiyonları yerine getirmekle geçirmektedir. Burada hızlı I^2C elemanı ile yavaş bir mikrodenetleyici arasında hız farkı oluşacaktır.

Bu durumda data transferi START prosedürü ile gerçekleştirilir. Bu prosedür aşağıdaki durumları içerir:

- A START durumu (S)
- A START baytı (00000001)
- Kabul sinyali (ACK)
- Tekrarlı-START durumu

START prosedürü Şekil 2.16'da gösterilmiştir.



Şekil 2.16. START bayt prosedürü

BÖLÜM 3. PCF8584 I^2C-BUS DENETLEYİCİSİ

3.1. Giriş

PCF8584, birçok standart paralel mikroişlemci ve mikrodenetleyici ile seri I^2C-Bus arasında arayüz sağlayan, CMOS teknolojisi ile imal edilmiş bir tüm devredir. Bu tüm devre ile yönetici ve yönetilen fonksiyonlarının her ikisi de gerçekleştirilebilmektedir.

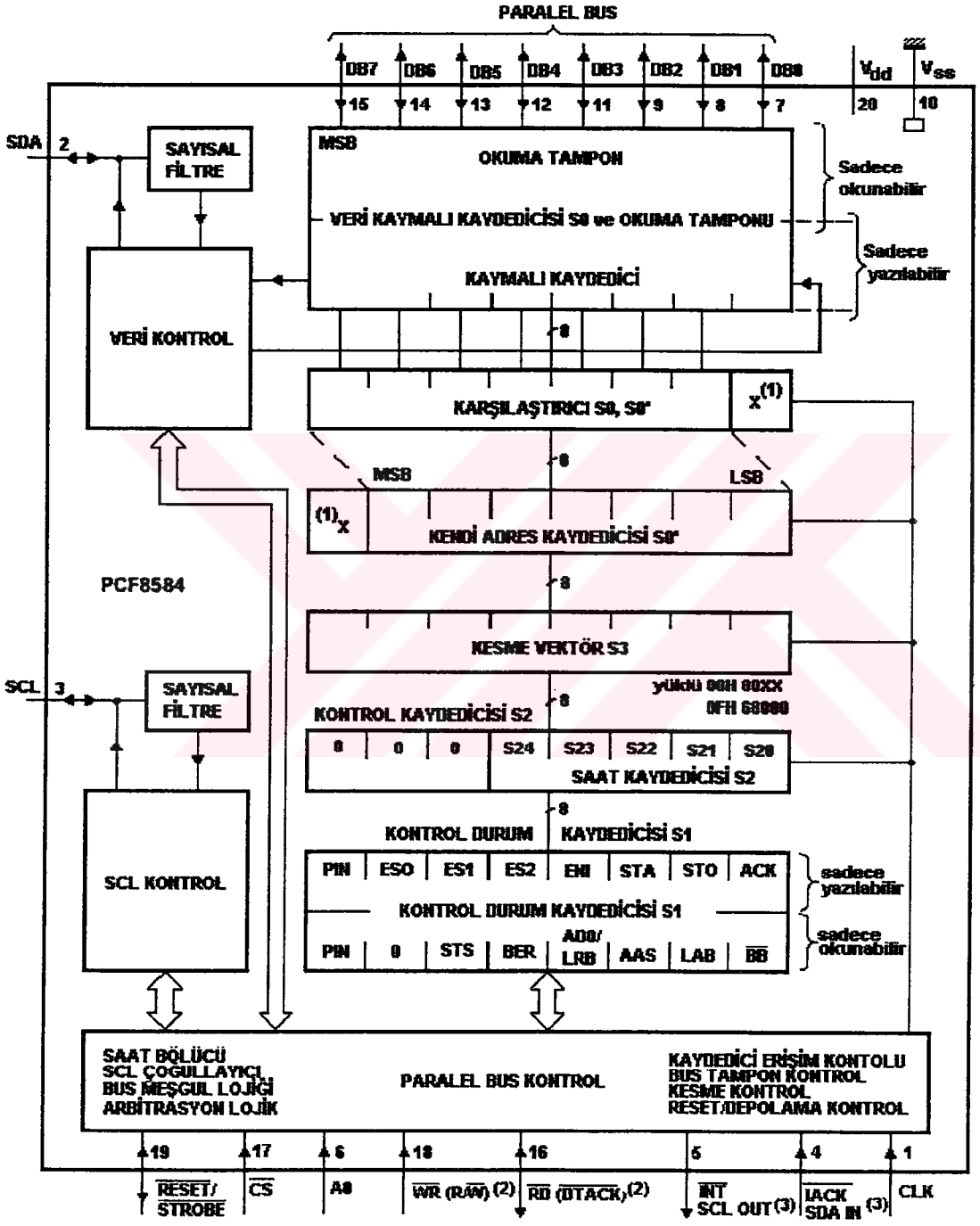
I^2C-Bus ile iletişim, kesme(interrupt) veya yoklamalı(polled) modları kullanılarak bayt-tabanlı olarak yapılır. PCF8584, arbitrasyon(arbitration) ve zamanlama ile I^2C-Bus'ın bütün özel durumlarını kontrol eder. PCF8584, paralel–bus sistemler ile çift yönlü I^2C-Bus sistemlerin haberleşmesinde kullanılır.

3.2. Özellikleri

- Paralel-Bus'ı I^2C-Bus protokolüne dönüştürür ve arabirim sağlar.
- 8049,8051,6800,68000,Z80 gibi birçok mikroişlemci ve mikrodenetleyici ile uyumludur.
- Yönetici(Master) ve Yönetilen(Slave) fonksiyonlarını içerir.
- Programlanabilen kesme vektörüne sahiptir.
- Çok-Yöneticilidir.
- I^2C-Bus monitor moduna sahiptir.
- Uzun-mesafe moduna sahiptir.

3.3. Blok Diyagramı

I^2C-Bus mikrodenetleyicisi PCF8584'ün blok yapısı Şekil 3.1'de gösterilmiştir.

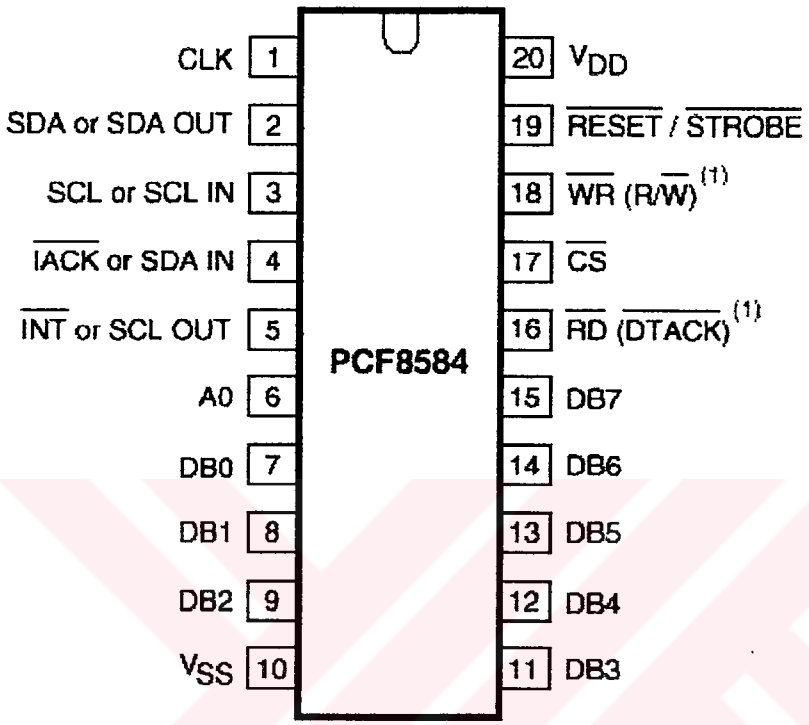


(1) X = dikkate alınmaz.
(2) 68000 pin tanımlamalarını gösterir.
(3) uzun mesafe pin isimleri.

Şekil 3.1. PCF8584 blok diyagramı

3.4. Pin Tanımlamaları

PCF8584'ün pinleri ve pinler hakkındaki ayrıntılı açıklamalar Tablo 3.1'de gösterilmiştir.



Şekil 3.2. PCF8584 Pin tanımlamaları

Tablo 3.1. PCF8584 Pinleri ve görevleri

ADI	PIN NO	I/O	AÇIKLAMA
CLK	1	I	Mikrodenetleyicinin saat üretecinden alınan CLK girişi
SDA	2	I/O	I^2C-bus seri data girişi
SCL	3	I/O	I^2C-bus seri saat girişi
IACK or SDA IN	4	I	Kesme bilgilendirme girişi; Uzun mesafe modunda seri data girişi
INT or SCL OUT	5	O	Kesme çıkışı; Uzun mesafe modunda saat çıkışı

Tablo 3.1.(Devam) PCF8584 Pinleri ve görevleri

ADI	PIN NO	I/O	AÇIKLAMA
A0	6	I	Kaydedici seçici girişi
DB0	7	I/O	İki yönlü veri giriş/çıkış ucu
DB1	8	I/O	İki yönlü veri giriş/çıkış ucu
DB2	9	I/O	İki yönlü veri giriş/çıkış ucu
Vss	10	-	Toprak
DB3	11	I/O	İki yönlü veri giriş/çıkış ucu
DB4	12	I/O	İki yönlü veri giriş/çıkış ucu
DB5	13	I/O	İki yönlü veri giriş/çıkış ucu
DB6	14	I/O	İki yönlü veri giriş/çıkış ucu
DB7	15	I/O	İki yönlü veri giriş/çıkış ucu
RD(DTACK)	16	I/(O)	Mikrodenetleyici için okuma kontrol biti
CS	17	I	Entegre seçme ucu.
WR(R/W)	18	I	Mikrodenetleyici için yazma kontrol biti
RESET/ STROBE	19	I/O	Reset girişi
VDD	20	-	Kaynak voltajı

3.5. FONKSİYON TANIMLARI

3.5.1 Giriş

PCF8584, standart paralel-bus ile seri I^2C-bus arasında arabirim sağlayan bir tüm devredir. I^2C-bus'da yönetici veya yönetilen fonksiyonlarının her ikisini de gerçekleştirebilir. Bu denetleyici ile, I^2C-bus'daki çift yönlü veri transferi, kesmeli veya el-sıkışmalı mod ile bayt tabanlı olarak gerçekleştirilir. 80XX ve 6800 işlemcileri ile arabirimi mümkündür. Burada yol seçimi otomatik olarak yapılır.

PCF8584 beş adet dahili kaydediciye sahiptir. Bunlardan üçü (Kendi adres kaydedicisi S0', saat kaydedicisi S2 ve kesme vektörü S3) başlangıç aşamasında PCF8584'ü kurmak etmek için kullanılır. Normalde sıfırlama işleminden sonra bu kaydedicilere bir kez yazılır ve sistem ilk başta kurulduğu şekilde çalışmaya devam eder.

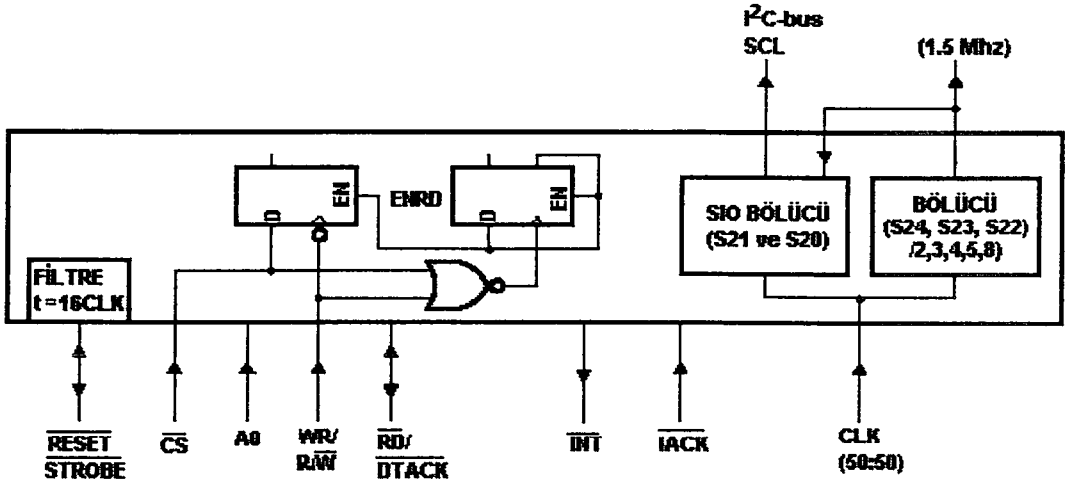
Geriye kalan iki kaydedici (veri tampon/kaymalı kaydedici S0, kontrol/durum kaydedicisi S1) veri iletimi sırasında çift fonksiyona sahiptir. S0 kaydedicisi yazma ve okuma için iki ayrı kaydediciye sahiptir ve her biri için ayrı kaydediciye erişilir. S0 kaydedicisi, kaymalı kaydedici ve veri tamponu olarak kullanılır. Bu kaydedici I^2C-Bus'taki bütün seri'den paralel'e arabirim işlemini yerine getirir. S1 kaydedicisi ise I^2C-bus'a erişim için gerekli durum bilgilerini içerir.

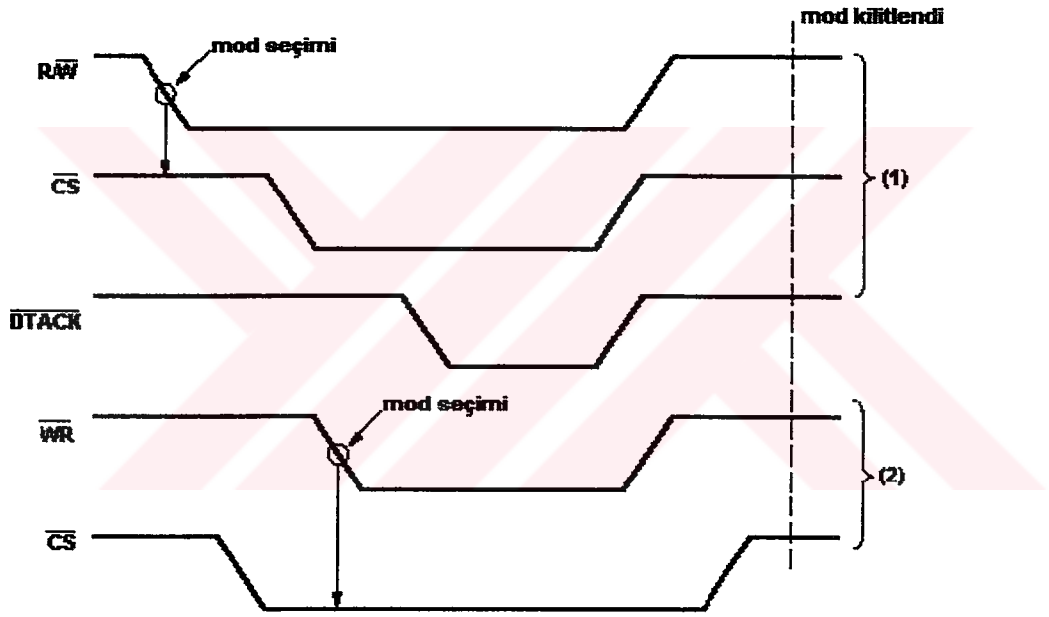
3.5.2. Mod Kontrol Arabirimi

80XX veya 6800 mod arabirimi WR ve CS sinyal sırasının sezilmesiyle yapılır. Burada WR kontrol ucunun her iki arabirim için de ortak oluşu bir avantajdır. Sistemde 80XX tipi aktif olarak seçilmiştir. Eğer /CS HIGH'da iken WR'ın HIGH'dan LOW'a geçişi sezilirse 6800 moduna geçilir ve /DTACK çıkışı aktif yapılır. /WR ve /CS reset işleminden sonra kararlı durumdadır. Şekil 3.3'te mod seçim işlemi gösterilmiştir.

3.5.3. Kurma Kaydedicileri S0', S2 ve S3

S0', S2 ve S3 kaydedicileri PCF8584'ü başlangıç aşamasında kurmak için kullanılır. Bu kaydediciler, PCF8584'ün çalışması için gerekli olan saat frekansını, seri saat frekansını(SCL), kesme vektör modunu ve PCF8584 yönetilen olarak kullanılacaksa cevap vereceği 7-bitlik adresi saklamak için kullanılırlar. PCF8584'ün ne şekilde çalışacağını belirleyen bu kaydediciler aşağıdaki bölümlerde tek tek ele alınarak ayrıntılı olarak açıklanmıştır.





Şekil 3.3. 68000/80XX zamanlama sinyalleri

3.5.4. Kendi Adres Kaydedicisi S0'

PCF8584 yönetilen olarak adresleneceği zaman bu kaydedici, PCF8584'ün cevap verebileceği 7-bit I^2C-Bus adresi ile yüklenmelidir. S0' kaydedicisi başlangıç aşamasında, ilerideki kullanılma ihtimaline karşın bu adres ile yüklenmelidir. I^2C-Bus'tan bu adres alındığında, S1 Durum kaydedicisi'nin **Addressed As Slave** (yönetilen olarak adreslen) biti kurulur. (S0 kaydedicisi S0' ile karşılaştılır.) S0

kaydedicisi S0' kaydedicisinin 1-bit sola kaymış halidir. Yani S0' 55H ile yüklendiğinde PCF8584 yönetilen adresi AAH olarak tanıtılır.

S0' kaydedicisinin programlanması paralel-bus'taki A0 hattı LOW seviyede iken uygun bit kombinasyonlarının S1 Kontrol Durum Kaydedicisine yazılmasıyla yapılır. PCF8584'ün sıfırlanmasından sonra S0' kaydedicisine otomatik olarak 00H adres değeri yüklenir.

3.5.5. Saat Kaydedicisi S2

S2 kaydedicisi, denetleyicinin çalışması gerekli olan saat frekansını ve SCL saat frekansının kontrol edilmesini sağlar. S20 ve S21 Tablo 3.2'de görüldüğü gibi 4 farklı I^2C-Bus SCL frekansının elde edilmesini sağlar. Burada dikkat edilmesi gereken bir nokta vardır. SCL saat frekansının elde edilebilmesi için S24, S23, S22 kaydedicileri giriş saat frenkansı (f_{clk}) ile uygun şekilde programlanmalıdır.

Tablo 3.2. I^2C-Bus SCL frekansının belirlenmesi

BIT		**Yaklaşık SCL frekansı F_{SCL} (kHz)**
S21	**S20**	
0	0	90
0	1	45
1	0	11
1	1	0.1

S22, S23, S24 kaydedicileri dahili saat bölücünün kontrolü için kullanılır. Değişik mikrodenetleyici saat sinyalleri için bu kaydediciler 5 farklı saat sinyaline programlanabilirler. Bu, sabit dahili saat frekansını sağlar. Kararlı SCL sinyali üretmek için bu durum gereklidir.

S2 kaydedicisi'nin programlanması paralel-bus'daki A0 hattı LOW iken, S1 kontrol/durum kaydedicisi'ne uygun bit kombinasyonlarının yazılmasıyla yapılır. (S1'e A0 HIGH durumda iken yazılır.) Dahili saat frekansı için uygun bit kombinasyonları Tablo 3.3'de verilmiştir.

Tablo 3.3. Dahili saat frekansının ayarlanması

Dahili Saat Frekansı			
S24	**S23**	**S22**	**fclk (Mhz)**
0	$X^{(1)}$	$X^{(1)}$	3
1	0	0	4.43
1	0	1	6
1	1	0	8
1	1	1	12

Tablo3.3'ki $X^{(1)}$ durumu dikkate alınmaz.

3.5.6. Kesme Vektor S3

Kesme vektör kaydedicisi, kesme-vektörlü mikrodenetleyiciler için 8-bit kullanıcı tarafından programlanabilen vektör sağlar. Bu vektör, Kesme kabul sinyali alındığında ve ENI bayrağı kurulduğunda DB0-DB7 portlarına gönderilir. Varsayılan vektör değerleri:

- 80XX için vektör değeri 00H
- 68000 için vektör değeri 0FH'dir.

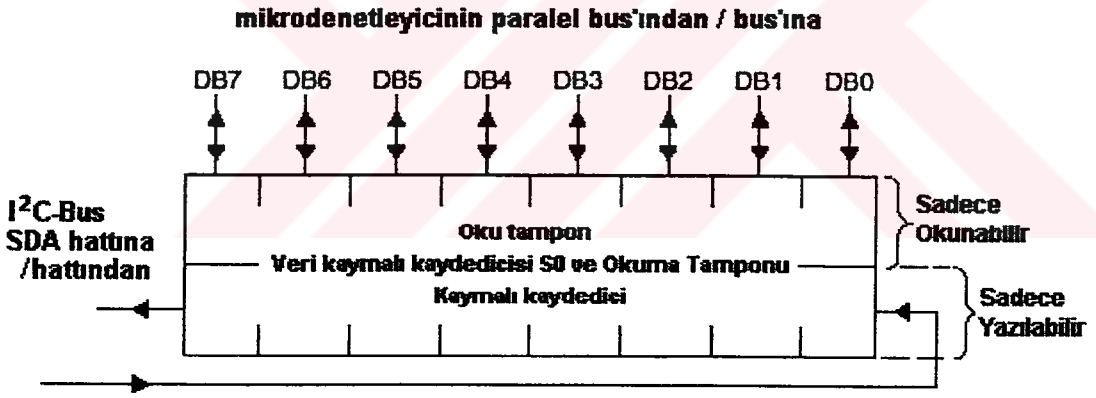
PCF8584 sıfırlandığında varsayılan kesme vektör modu 80XX için seçilmiştir ve 00H'dir. Eğer 68000 tabanlı işlemciler kullanılırsa S3 kesme vektör değeri 0FH olacaktır.

3.5.7. Veri Kaymalı Kaydedicisi/Okuma Tamponu S0

S0 kaydedicisi, I^2C-Bus arabiriminde seri kaymalı kaydedici ve okuma tamponu gibi davranır. I^2C-Bus'taki bütün yazma ve okuma işlemleri bu kaydedici üzerinden yapılır. S0 kaydedicisi, kaymalı kaydedici ve veri tamponunun birleşimidir. Paralel veri daima bu kaydediciye yazılır ve veri tamponundan okunur. I^2C-Bus'taki veri S0 kaydedicisinde yazılır ve bu kaydedicide kaydırılarak Bus'a gönderilir.

Alıcı modunda, kaymalı kaydedicideki veri, okuma tamponuna ACK sinyali boyunca kopyalanır. Daha sonra alınan veriler S0 veri tamponundaki veriler okununcaya kadar bekletilir. (SCL hattı LOW seviyede tutulur.)

Gönderici modunda, ESO biti kurulduğunda, S0 kaymalı kaydedicisine veri yazıldığı anda bu veri I^2C-Bus'a transfer edilir.



Şekil 3.4. Veri kaymalı kaydedici/Bus tampon S0

3.5.8. Kontrol / Durum Kaydedicisi S1

S1 kaydedicisi I^2C-Bus işlemlerini kontrol eder ve Bus durum bilgilerini içerir. S1 kaydedicisine A0 kaydedici seçici ucun HIGH seviyede olduğu durumda erişilir. Mikroişlemcili veya mikrodenetleyicili sistemler ile I^2C-bus arasındaki iletişim sırasında S1 kaydedicisi okuma ve yazma fonksiyonları için için ayrı ayrı bit kombinasyonlarına sahiptir. Sadece yazılabilen kısım, kaydedici erişim kontrolü ve

I^2C-bus 'daki sinyallerin kontrolünü sağlarken, sadece okunabilir kısım ise I^2C-Bus durum bilgilerini sağlar. Kontrol/Durum Kaydedici bit yapısı Tablo 3.4'de gösterilmiştir.

Tablo 3.4. S1 kaydedicisi ve bitleri

KONTROL/DURUM	BITLER								MOD
Kontrol	PIN	ES0	ES1	ES2	ENI	STA	STO	ACK	Sadece yazılabilen
Durum	PIN	0	STS	BER	AD0-LRB	AAS	LAB	BB	Sadece okunabilen

3.5.8.1. S1 Kaydedicisi Kontrol Bölümü

S1 kaydedicisinin sadece yazılabilen bu kısmı S0, S0', S1, S2 ve S3 kaydedicilerine erişimi aktif hale getirir ve I^2C-Bus işlemlerini kontrol eder.

3.5.8.1.1. PIN (Pending Interrupt Not)

PIN bitine lojik-1 yazıldığında bütün durum bitleri lojik-0 olur. Bu işlem, yazılım sıfırlama vazifesi görür. PIN biti S1 kaydedicisindeki hem okunabilen hem de yazılabilen tek bittir. PIN biti çoğu zaman seri iletişimin senkronizasyonu için kullanılır.

3.5.8.1.2. ESO (Enable Serial Output)

ESO biti seri I^2C-Bus giriş-çıkış'ını aktif veya pasif hale getirir. Bu bit LOW seviyede iken başlangıç için kaydedicilere erişim mümkündür. Bu bit HIGH seviyede iken ise, I^2C-Bus iletişimi aktiftir ve seri kaymalı kaydedici S0 iletişim için hazırdır. S1 Bus durum kaydedici bitleri okunabilir.

3.5.8.1.3. ES1 ve ES2

ES1 ve ES2 başlangıç aşaması için kaydedici seçiminin kontrolünde ve normal işlemlerin kontrolünde kullanılır. Bu bitler istenen kaydediciye erişim için programlandıktan sonra kaydedici, A0 kaydedici seçici girişinin lojik-LOW yapılmasıyla seçilir. Tablo 3.5 Kaydedici erişim kontrolünü göstermektedir.

Tablo 3.5 Kaydedici erişim kontrolü

DAHİLİ KAYDEDİCİ ADRESLEME				
A0	**ES1**	**ES2**	**$\overline{\text{IACK}}$**	**Fonksiyon**
ESO = 0 Seri iletişim pasif				
1	0	X	1	R/W S1:Kontrol
0	0	0	1	R/W S0':(Own Address)
0	0	1	1	R/W S3: (interrupt vektor)
0	1	0	1	R/W S2:(clock register)
ESO = 1 Seri iletişim aktif				
1	0	X	1	W S1: Kontrol
1	0	X	1	R S1:Durum
0	0	0	1	R/W S0:(data)
0	0	1	1	R/W S3(interrupt vektor)
X	0	X	0	R S3(interrupt vektor ACK)

3.5.8.1.4. ENI (Enable Interrupt)

ENI biti PIN biti aktif olduğunda üretilen harici kesme çıkışı INT'i aktif hale getirmek için kullanılır.

Uzun mesafe modu kullanılacaksa, bu moddan önce lojik-0 seviyesine kurulmalı ve bu işlem boyunca lojik-0'da kalmalıdır.

3.5.8.1.5. STA (Start) ve STO(Stop)

Bu bitler I^2C-Bus START durumu üretiminin kontrolünde, yönetilen adres ile R/W bitinin iletiminde, tekrarlı-START durumunun üretilmesinde ve STOP durumunun üretiminde kullanılır. Tablo 3.6'de bu durum gösterilmiştir.

Tablo 3.6. I^2C-Bus'taki özel durumların gösterilişi

STA	STO	MOD	Fonksiyon	İşlem
1	0	SLV/REC	START	START+adres gönder, Eğer R/W=0 ise MST/TRM durumunda kal; R/W=1 ise MST/REC moduna geç.
1	0	MST/TRM	Tekrarlı START	SLV/REC ile aynı
0	1	MST/REC; MST/TRM	STOP READ; STOP WRITE	STOP durumu gönder ve SLV/REC moduna geç
1	1	MST	DATA ZINCIRI	STOP'suz son yönetici çevriminden sonra START, STOP ve Adres gönderir.
0	0	HİÇBİRİ	İşlem yok	İşlem yok.

3.5.8.1.6. ACK (Acknowledge)

Bu bit normalde lojik-1 seviyesine kurulmalıdır. Bu durum, I^2C-Bus denetleyicisinin gönderilen her bayttan sonra otomatik olarak kabul sinyali göndermesini sağlar. Eğer I^2C-Bus denetleyicisi Yönetici/Alıcı modunda çalışarak Yönetilen/Gönderici'nin gönderdiği verilere ihtiyaç duymuyorsa , bu bit lojik-0 seviyesinde olmalıdır. Bu I^2C Bus üzerinde NACK sinyalinin oluşmasına neden olur ve yönetilen eleman tarafından iletişim kesilir.

3.5.8.2. S1 Kaydedicisi Durum Bölümü

S1 kaydedicisinin sadece okunabilen bu kısmı I^2C-Bus durum bilgisine erişimi aktif hale getirir.

3.5.8.2.1. PIN (Pending Interrupt Not) Biti

S1 kaydedicisinin MSB biti olan PIN (Pending Interrupt Not) seri iletişimi eş zamanlı hale getirmek için kullanılan durum bayrağıdır ve PCF8584 iletişim yaptığında lojik-0'a getirilir. Bu bit yoklamalı (polled) mod uygulamalarında, I^2C-Bus'ın bayt gönderimini/alımını tamamladığını belirtmek için sürekli okunur.

Seri iletişimin başladığı her seferinde PIN biti otomatik olarak lojik-1 seviyesine kurulacaktır(pasif). PCF8584 gönderici olarak kullanıldığında, S0 kaydedicisine her veri yazılışında PIN biti otomatik olarak lojik-1 seviyesine getirilir. Alıcı modunda ise, veri kaydedicisi S0'ın her okunuşunda PIN biti otomatik olarak lojik-1 seviyesine getirilir.

I^2C-Bus'daki her bir baytın gönderilmesi veya alınmasından sonra PIN biti gönderme veya alma işleminin tamamlandığını göstermek amacıyla otomatik olarak lojik-0 seviyesine çekilir. PIN biti sonradan takip eden baytlarda lojik-1 seviyesine getirilirse, bütün durum bitleri lojik-0 seviyesine gelir. PIN biti lojik-0 seviyesine çekilerek, I^2C-Bus'daki hata durumunu göstermek için de kullanılır.
Yoklamalı mod uygulmalarında PIN biti, seri gönderme veya alma işleminin tamamlandığını belirlemek için sürekli okunur. ENI biti aktif olduğunda yani lojik-1 seviyesine kurulduğunda ise, donanım kesmesi aktif hale gelir. Bu durumda PIN biti INT çıkışı üzerinden harici kesmeyi tetikler ve her seferinde lojik-0 seviyesine getirilir.

Yönetilen-Gönderici veya Yönetilen-Alıcı modunda PIN biti lojik-0 olduğunda, PCF8584 I^2C-Bus iletişimini SCL hattı LOW olduğu sürece geçici olarak askıya alacaktır. Bu, S0'daki geçerli veri baytının okunulmadan (Yönetilen-Alıcı) veya

sonraki veri baytının S0'a yazılmadan (Yönetilen-Gönderici) daha sonraki verilerin gönderilmesini veya alınmasını önler.

PIN biti özet olarak:

- Yoklamalı mod uygulamalarında seri iletişimin tamamlandığını göstermek için kullanılabilir. ENI biti aktif olduğunda ise harici kesmeyi INT üzerinden aktif hale getirir.
- STA bitinin set edilmesi PIN bitinin lojik-1 yapılarak pasif hale getirilmesini sağlar.
- Gönderici modunda, I^2C-bus'a bir baytın gönderilmesinden sonra, iletişimin tamamlandığını göstermek amacıyla lojik-0 seviyesine sıfırlanır.
- Gönderici modunda, S0 kaydedicisine her veri yazılışında otomatik olarak lojik-1 seviyesine kurulur.
- Alıcı modunda, PIN biti alının baytın tamamlandığını belirtmek için lojik-0 seviyesine getirilir ve SCL hattı PIN biti lojik-1 oluncaya kadar LOW seviyede tutulur.
- Alıcı modunda, S0 kaydedicisi her okunduğunda lojik-1 seviyesine getirilerek pasif yapılır.
- Yönetilen-Alıcı modunda I^2C-Bus STOP durumu PIN bitini lojik-0 yaparak aktif hale getirecektir.
- I^2C-Bus'da hata meydena gelirse PIN biti otomatik olarak lojik-0 seviyesine getirilir.

3.5.8.2.2. STS (Stop Slave) biti

Bu bit, Yönetilen-Alıcı modunda harici üretilen STOP durumu sezildiğinde aktif yapılır (Sadece Yönetilen-Alıcı modunda kullanılır).

3.5.8.2.3. BER (Bus Error) biti

Bus hatası anlamına gelmektedir. Yanlış START ve STOP durumları sezildiğinde BB bitini lojik-1 seviyesine, PIN bitini lojik-0 seviyesine çeker.

3.5.8.2.4. LRB/AD0 (Last Received Bit/Adres-0) biti

Bu durum bitinin çift fonksiyonu vardır ve sadece PIN biti lojik-0 olduğunda geçerlidir.

1. LRB(En son alınan bit), AAS biti lojik-0 olduğunda (Yönetilen olarak adreslenmediğinde) I^2C-Bus'tan alınan en son biti tutar. Normalde bu bit, yönetilen kabul bitidir ve Yönetilen kabul kontrolü bu bit üzerinden yapılır.

2. AD0; AAS lojik-1 olduğunda (Yönetilen olarak adreslendiğinde) I^2C-Bus denetleyicisi Yönetilen olarak adreslenmiştir. Bu bit genel çağrı adresi alındığında lojik-1, I^2C-Bus denetleyicisinin kendi yönetilen-adresi alındığında ise lojik-0 seviyesine getirilir.

3.5.8.2.5. AAS (Addressed As Slave) Biti

AAS(Yönetilen olarak adreslen) biti sadece PIN bitinin lojik-0 olduğu durumda geçerlidir. Yönetilen-Alıcı modunda I^2C-Bus'dan alınan adres ile S0' adres kaydedicisindeki adres birbirine eşit olduğunda veya general-call adresi alındığında bu bit set edilir.

3.5.8.2.6. LAB (Lost Arbitration Bit) Biti

LAB biti, Çok-Yöneticili I^2C-bus'larda arbitrasyon'un bus'taki başka bir yöneticiye kaybedildiğinde kurulur.

3.5.8.2.7. BB (Bus Busy) Biti

BB biti bus'ın kullanımda olduğunu gösteren sadece okunabilen bittir. Bu bitin lojik-0 olması, bus'ın kullanımda olduğunu ve bus'a erişimin mümkün olmadığını gösterir. Bu bit STOP ve START durumları tarafından lojik-0 veya lojik-1 seviyesine getirilebilir.

3.6. Özel Fonksiyon Modları

3.6.1. Strobe Modu

I^2C-Bus denetleyicisi kendi adresini veya '00H' genel çağrı adresini aldıktan sonra STOP durumu alırsa, /RESET/STROBE ucunda strobe çıkış sinyali üretilir. /STROBE sinyali 8 saat darbesi boyunca aktif LOW seviye de olan tek kararlı çıkış darbesidir. Bu sinyal STOP durumu alındıktan sonra üretilir ve Çok-Yöneticili paralel-bus sistemlerde bus erişim kontrollerinde kullanılır.

3.6.2. Uzun Mesafe Modu

Uzun mesafe modu, paralel-işlemci üzerinden iki I^2C-Bus denetleyicisini uzun mesafeli seri haberleştirmek için kullanılır. Bu mod Seri arabirim aktif olduğunda (ESO = 1) ES1 biti set edilerek yapılır.

Bu modda, I^2C-Bus protokolü tek yünlü 4 hat üzerinden yapılır. Bu hatlar SDA OUT, SCL IN, SDA IN ve SCL IN'dir. Bu iletişim hatları, uzun mesafe uygulamaları için hat sürücülerine bağlanmalıdır. Uzun mesafe modunda donanım karakteristikleri standart olarak verilir. Veri iletişiminin kontrolü normal I^2C-Bus modu ile aynıdır. Kaymalı kaydedici S0'daki veri okunduktan veya S0'a veri yazıldıktan sonra, uzun mesafe modu, ES0 ve ES1 lojik-1'e set edilerek başlatılmalıdır. Gönderilen veya alınan veri senkronizasyonu PIN biti üzerinden yoklamalı mod kullanılarak yapılmalıdır, çünkü kesme çıkışı /INT bu modda aktif değildir.

PCF8584 uzun mesafe moduna girmeden önce ENI biti lojik-0 seviyesine getirilmelidir.

PCF8584 uzun mesafe moduna girmeden önce hazırlık aşamasında, 4-hatlı bus'tan, 3 durumlu hat sürücüleri/alıcıları tarafından yalıtılmalıdır. Bu önlem sistemin düzgün ve verimli bir şekilde çalışmasını sağlayacaktır.

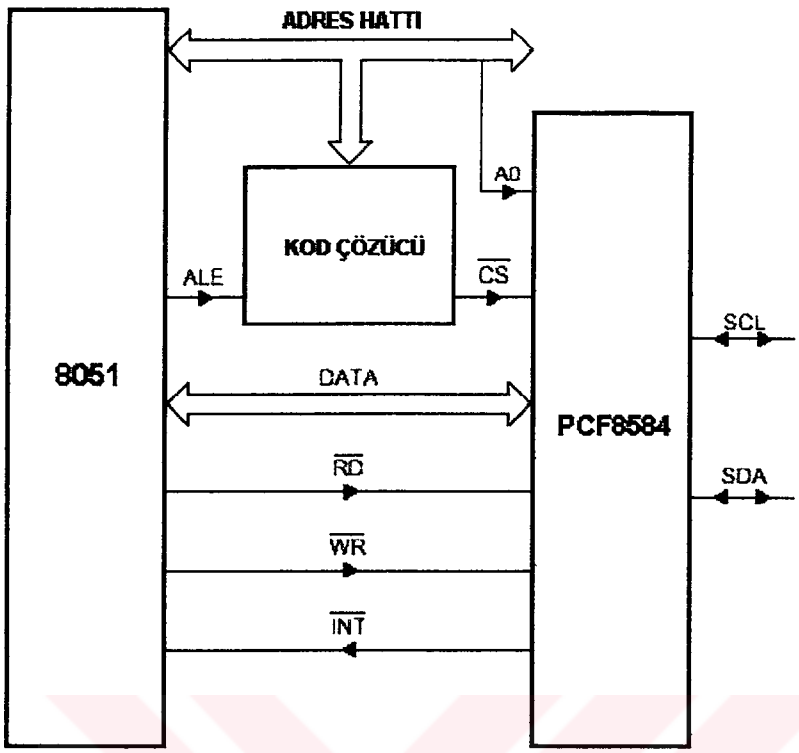
3.6.3. Monitor Modu

S0' Own address kaydedicisi 'nin bütün bitleri sıfır ile yüklendiğinde I^2C-Bus denetleyicisi, pasif I^2C-Bus monitor olarak çalışır, yani Bus'ta akan veriye cevap vermeyip sadece Bus'ı dinlemektedir. Monitor modunun temel özellikleri şunlardır:

- I^2C-Bus denetleyicisi daima seçilidir.
- I^2C-Bus denetleyicisi daima slave-receiver modunda çalışır.
- I^2C-Bus denetleyicisi kabul sinyali üretmez.
- I^2C-Bus denetleyicisi kesme isteği üretmez.
- START durumu sezildiğinde BB_ lojik-0'a set edilir, STOP durumu sezildiğinde ise lojik-1'e set edilir.
- Alınan veri otomatik olarak okuma tamponuna iletilir.
- I^2C-Bus, PIN ile monitor edilir. PIN biti alınan baytın kabul bitinden sonra lojik-0'a çekilir ve alınan ilk bit ile tekrar lojik-1 olur. S0 veri tamponunun okunması PIN bitini lojik-1 seviyesine getirir. Veri tamponundaki veri PIN biti lojik-0 olduğu sürece hazırdır ve 8 saat darbesi boyunca hazır olarak bekler.
- AAS her START durumundan sonra lojik-1'e set edilir ve 9. saat darbesinde lojik-0'a getilir.

3.7 PCF8584 ile 8051 Mikrodenetleyicisinin Birlikte Kullanımı

Şekil 3.5'te PCF8584 ve 8051 mikrodenetleyicisinin birlikte kullanıldığı uygulama devresi gösterilmektedir. Gerekli bağlantılar yapıldığında PCF8584, 8051 mikrodenetleyicisi ile birlikte kullanılarak I^2C-Bus oluşturulmuş olur. Burada isteğe bağlı olarak sistem kesmeli veya taramalı modda çalıştırılabilir. Eğer sistem kesmeli modda çalıştırılmayacaksa INT ucu kullanılmayabilir. Kesmeli mod kullanılacaksa PCF8584'ün INT ucu 8051'in kesme girişlerinden birisine bağlanmalıdır.



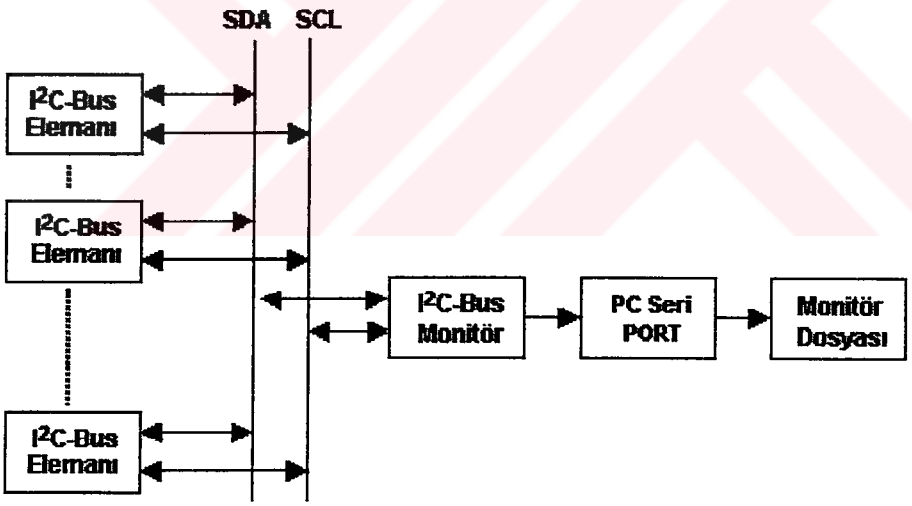
Şekil 3.5 8051 ile PCF8584 ün birlikte kullanımı

BÖLÜM 4. SİSTEM TASARIMI

4.1. Giriş

Bu bölümde, I^2C-Bus monitör ve I^2C-Bus monitör yöntemleri incelenmiştir. I^2C-Bus monitör açıklanarak, I^2C-Bus monitör yöntemlerinin birbirlerine göre avantaj ve dezavantajları belirtilmiştir. Ayrıca uygulaması yapılan I^2C-Bus monitör blok şemaları da bu bölümde ele alınmıştır.

4.2. I^2C-Bus Monitör



Şekil 4.1. I^2C-Bus monitör temel blok diyagramı

I^2C-Bus monitör, I^2C-Bus'ta akan verilerin gözlenmesini sağlayan, yazılım veya donanım tabanlı bir sistemdir. Bu sistem ile I^2C-Bus'da adreslenmiş bütün elemanlara hangi verilerin yazıldığı veya okunduğu gözlenebilmektedir. Bu şekilde, Bus'tan elde edilen veri ile Bus'ta olması gereken veri karşılaştırılarak I^2C-Bus

tabanlı bir sistemde hata kontrolü yapılabilir. Yani elde edilen sistemin bir test cihazı gibi çalıştırılması mümkün olur.

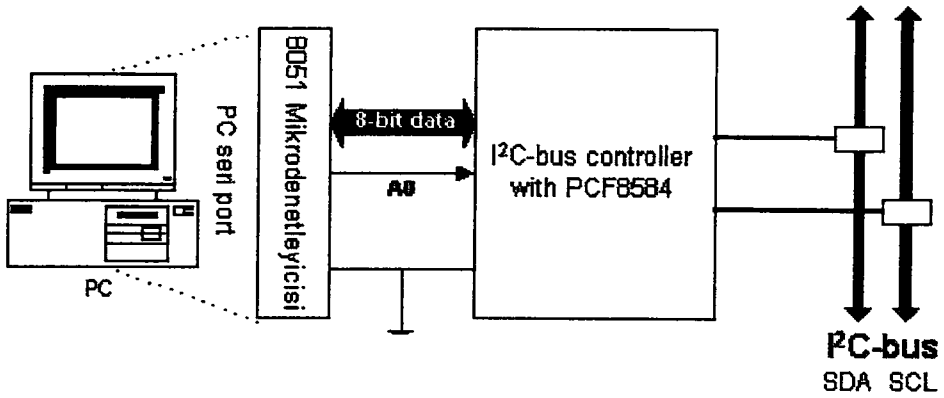
4.3. I^2C-Bus Monitor Yöntemleri

I^2C-Bus monitor işlemi, yazılım ve donanım tabanlı olarak iki şekilde yapılabilmektedir. Birinci yöntemde Bus, çok az devre elemanı kullanılarak, tamamen yazılım sayesinde gözlenebilmektedir. Yazılım ile Bus'ta akan veriler bit-bit kontrol edilerek ilgili Bus elemanın adresi ve veriler gözlenir. Bu yöntem ayrıca ek bir I^2C-Bus denetleyicisi gerektirmediğinden donanım tabanlı I^2C-Bus monitor'e göre daha ekonomiktir. Bu yöntemin en büyük dezavantajı ise, programlama kısmının zor olmasıdır.

Donanım tabanlı Bus'lar da ise, bir mikrodenetleyici ile birlikte bir I^2C-Bus denetleyicisi kullanılarak bus'ın gözlenmesi sağlanabilmektedir. Bu yöntemin avantajı programlama kısmının kolay olmasıdır. Bus'taki veriler bir I2C-Bus denetleyicisi tarafından yakalanır ve ilgili kaydedicide saklanır. I2C-Bus denetleyicisinin ilgili kaydedicileri başlangıç aşamasında monitör moduna kurularak sistemin bir Bus monitör olarak kullanılması sağlanır. I2C-Bus denetleyicisi Bus'tan aldığı her bir adres ve veri için mikrodenetleyiciyi bilgilendirmekte ve mikrodenetleyici de bu bilgileri, hafızanın ilgili yerlerine monitör etmek için saklamaktadır.

4.2. PCF8584 Denetleyicisi ile I^2C-Bus Monitör

I^2C-Bus denetleyicisi olan PCF8584, başlangıç aşamasında monitor moduna kurularak I^2C-Bus dinleyicisi gibi çalıştırılabilir. Bu aşamada PCF8584'ün kendi adres kaydedicisi olan S0' 0H sayısı ile yüklenir. Bu işlem PCF8584'ü monitör moduna sokar.

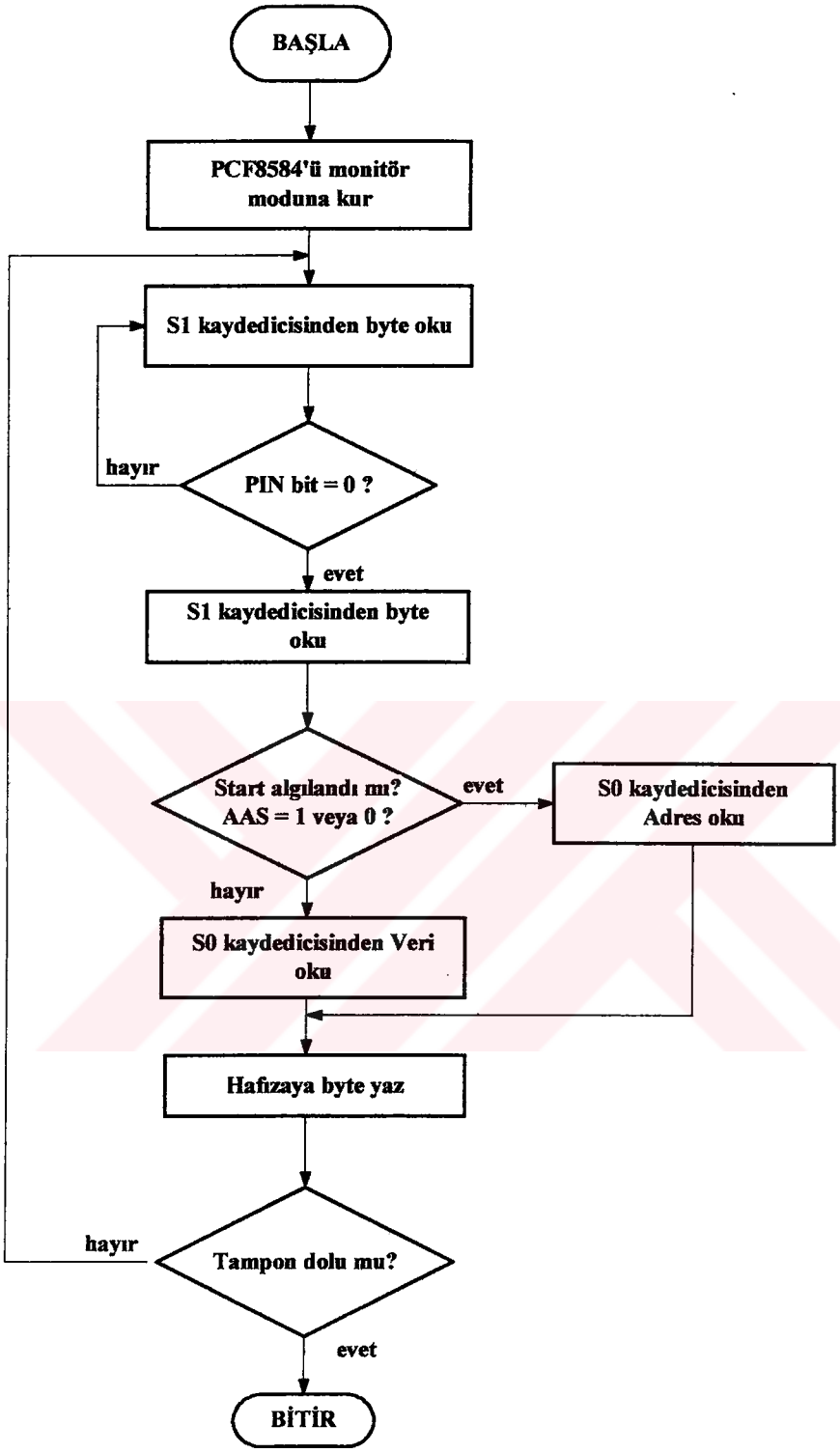


Şekil 4.2. PCF8584 ile I²C-Bus monitör blok diyagramı

Monitör modunda denetleyicini genel özellikleri şunlardır:

- Bütün adreslerde aktif durumdadır.
- Daima Yönetilen-Alıcı modunda çalıştırılır.
- ACK sinyali üretmez.
- Kesme istek sinyali üretmez.
- /BB lojik-0 olduğunda START durumu sezilir ve STOP durumundan sonra /BB tekrar lojik-1 durumuna döner.
- Bus'tan alınan data otomatik olarak okuma tamponuna yerleştirilir.
- Bustaki verilerin izlenmesi PIN biti sayesinde gerçekleştirilir. PIN biti, alınan baytların sonunda gönderilen ACK sinyali ile lojik-0'a getirilir ve ilk gelen baytın ilk bitiyle tekrar lojik-1 seviyesine döner. Veri tamponu S0'ın okunması, PIN bitini lojik-0 seviyesine çeker. Okuma tamponundaki veri PIN biti lojik-0 olduğu sürece yani 8 saat darbesi boyunca geçerlidir.
- AAS (Addressed As Slave) biti alınan her START durumundan sonra lojik-1 durumuna kurulur ve 9. saat darbesinde tekrar lojik-0 durumuna getirilir.

Şekil 4.3'de I²C-Bus monitör programı için gerekli akış diyagramı verilmiştir.



Şekil 4.3 I^2C-Bus monitor programı için akış diyagramı

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bu çalışmada, I^2C-Bus ve I^2C-Bus monitör sistem tasarımı, donanım tabanlı olarak gerçekleştirilmiştir.

Donanım ve yazılım tabanlı I^2C-Bus monitör teknikleri irdelenerek, aralarındaki farklar açıklanmıştır. Yapılan çalışmada yazılım tabanlı I^2C-Bus monitörün program kısmının zor olduğu görülmüş ve bu yöntemden vazgeçilerek donanım tabanlı I^2C-Bus monitör yapılmıştır, Donanım tabanlı I^2C-Bus monitör'de, I^2C-Bus'ı bayt tabanlı destekleyen PCF8584 kullanılmıştır. PCF8584 pasif dinleyici durumunda çalıştırılarak, Bus'dan okunan ve Bus'a yazılan verilerin dahili bir hafıza hücresine yazılması gerçekleştirilmiştir. Daha sonra bu veriler bilgisayara aktarılarak kullanıcının Bus'ta akan verileri gözlemesi sağlanmış olur. Dolayısıyla sistem Bus'taki hata durumlarını ve verileri gösterdiğinden, sayısal elektronik laboratuarlarına test cihazı olarak kullanılabilir.

Çalışmada gerçekleştirilen I^2C-Bus monitör, standart 100Khz. saat hızına saat sahip Bus'ları monitör edebilmektedir. Daha yüksek saat hızlara sahip Bus'larda ise yavaş kalmaktadır. Çünkü 8051 mikrodenetleyicisinin ortalama bir komutu işlemesi 12 saat darbesi gerektirmektedir. 8051 işlemcisi 11.0592MHz hızında çalıştırıldığında ortalama olarak bir komut 1μSn'de işletilir. Dolayısıyla 100KHZ'lik I^2C-Bus'ın monitör edilmesi için yaklaşık 10 komut kullanılmaktadır. 11.0592MHZ hızındaki mikrodenetleyicinin 100KHz hızındaki Bus için uygun olduğu fakat 400KHz ve daha yüksek hızlı Bus'larda yavaş kaldığı gözlenmiştir. Yüksek hızlı Bus'larda, Bus'ın monitör edilebilmesi için 8051 ailesinin yüksek hızlı (40MHz veya daha yüksek) ürünlerinden bir tanesinin kullanılması gerektiği gözlenmiştir.

Yapılan sistemde verilerin gözlenmesi için Windows-98 işletim sisteminin standart Hyper-Terminal isimli seri iletişim programı kullanılmıştır. Sistem, Bus'taki bütün verileri bu programa göndererek gözlenmesini sağlamıştır. Yapılacak bir Windows tabanlı arayüz programı ile bütün verilerin gözlenmesi yerine sadece istenen adresindeki veya istenen verilerin gözlenmesi sağlanabilir.

KAYNAKLAR

[1] James, M.Flynn, "Understanding and Using the I^2C-Bus", Embedded System Programming, www.embedded.com, 2001

[2] Philips Semiconductors, "The I^2C-Bus Specification", Version 2.1, 2000

[3] Koetsier Hilbert, "Personal Computer interface to I^2C bus via parallel printer port using PCF8584 bus controller", System basics and Specification, 1999

[4] Philips Semiconductors, "PCF8584 I^2C-Bus Controller", 1997

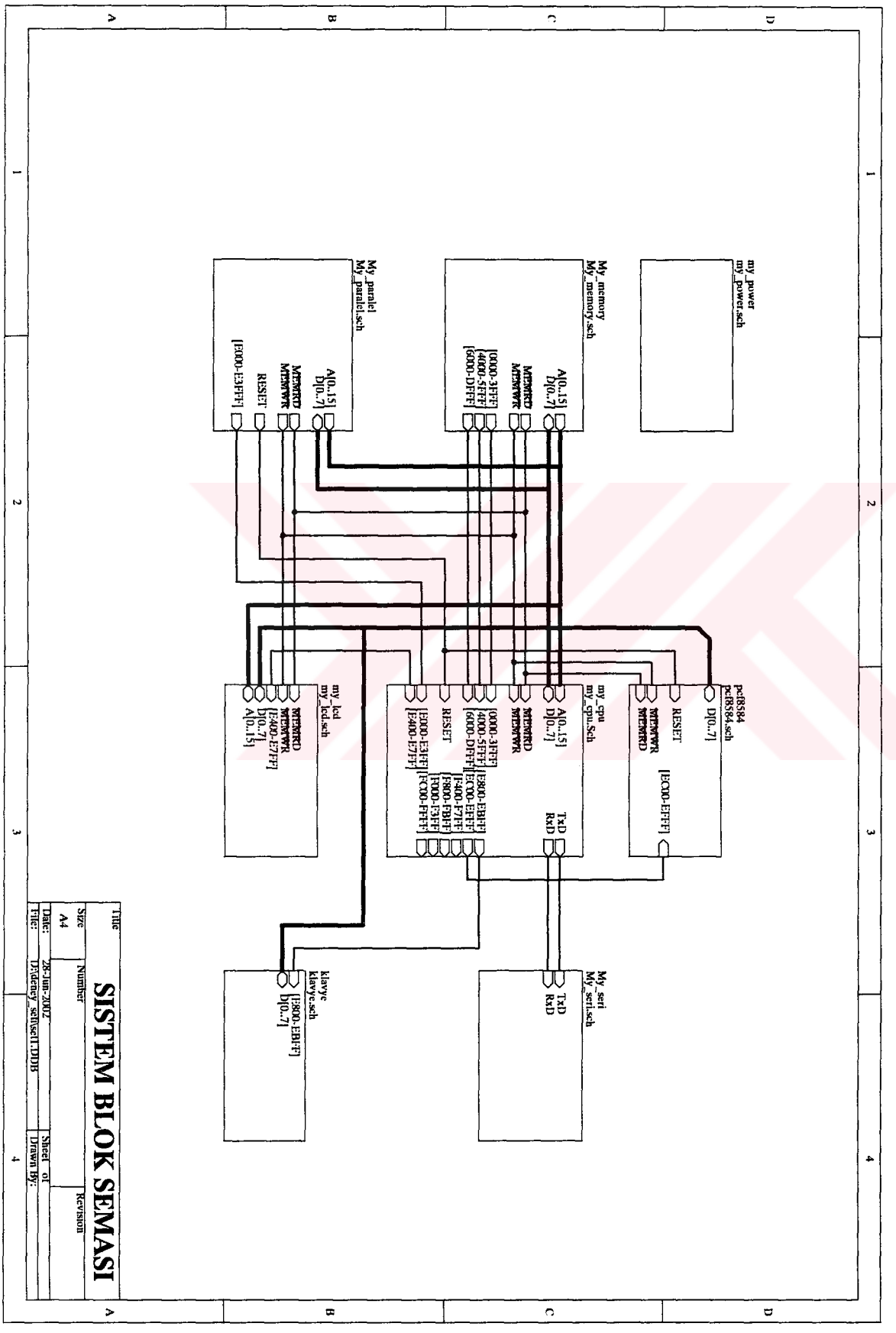
[5] Philips Semiconductors, "Interfacing the PCF8584 I2C-Bus controller to 80c51 family microcontrollers", 1994

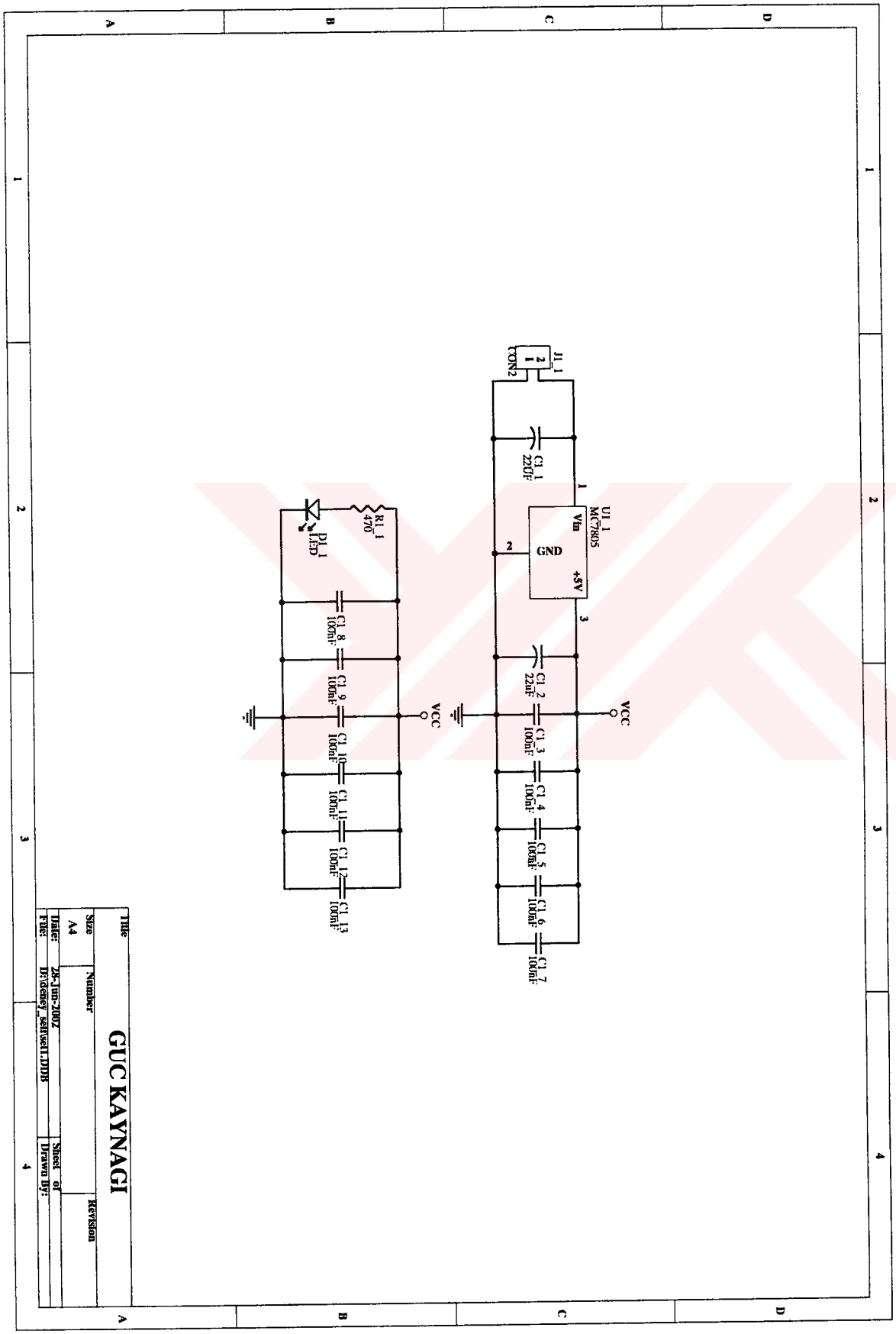
[6] A.T.Özcerit, PHD Tezi "Fault Tolerant embedded multi processing system with bus switching",Sussex University, 1999

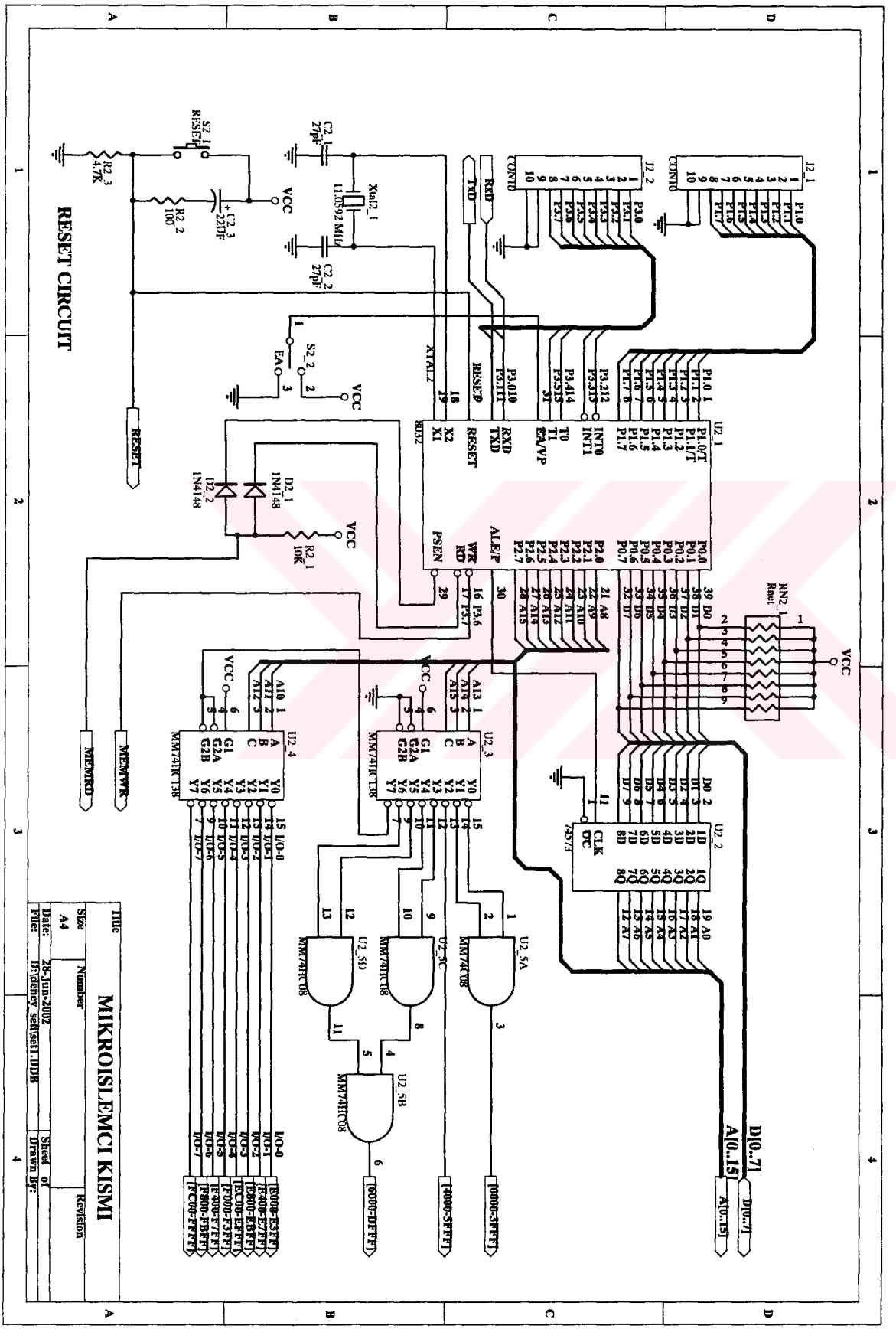
[7] Philips Semiconductor "www.semiconductor.philips.com"

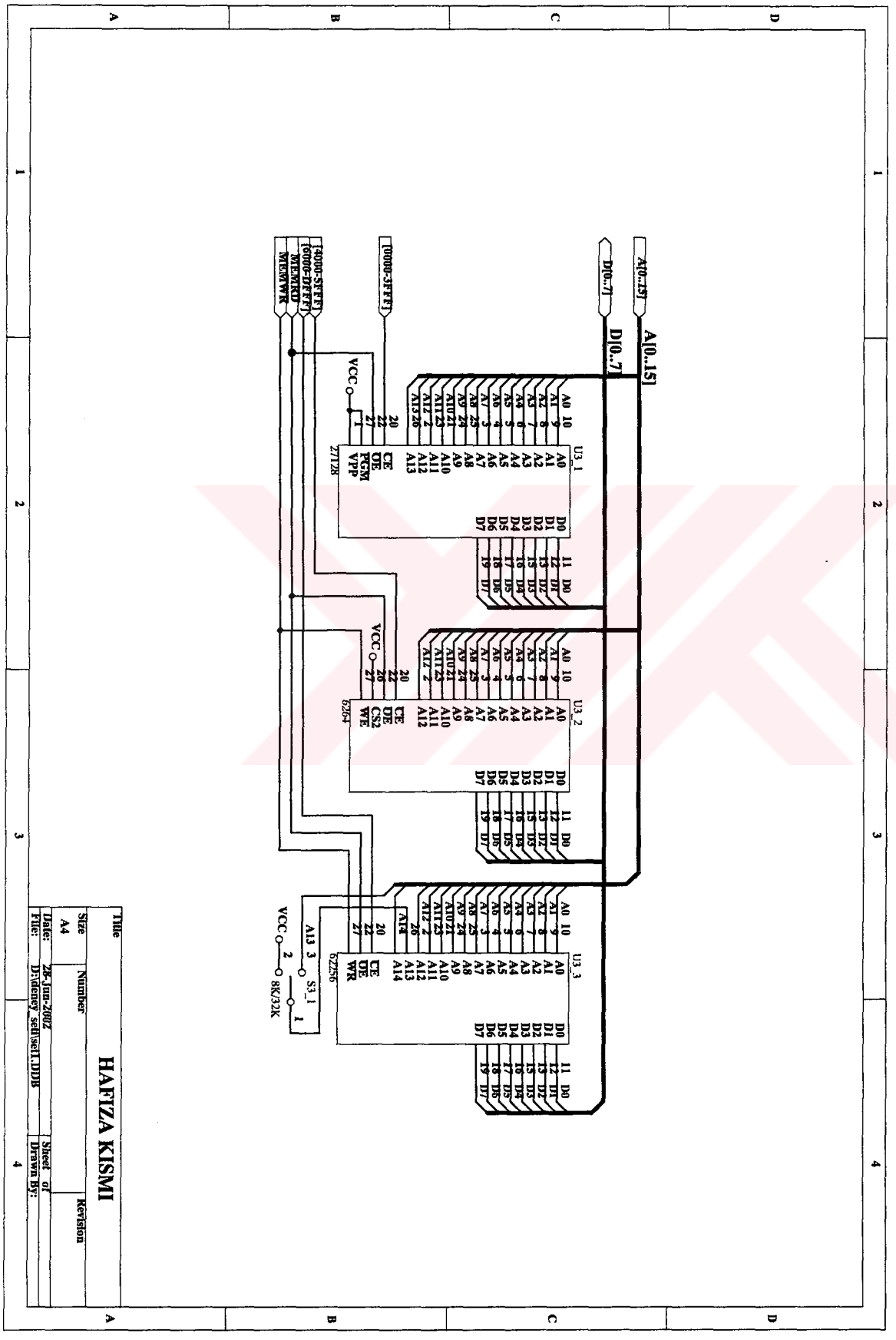
EKLER

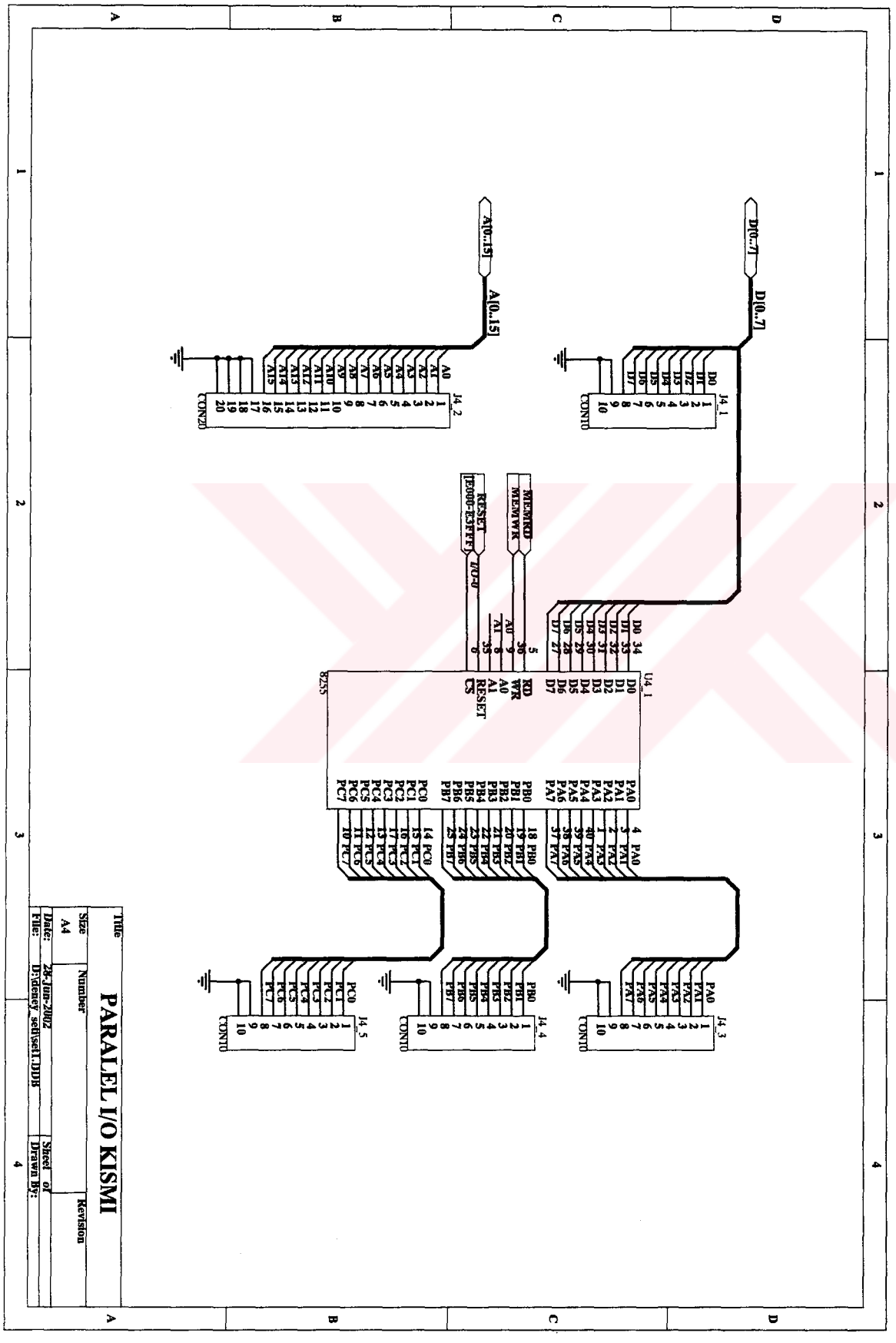
EK-A DEVRE ŞEMALARI

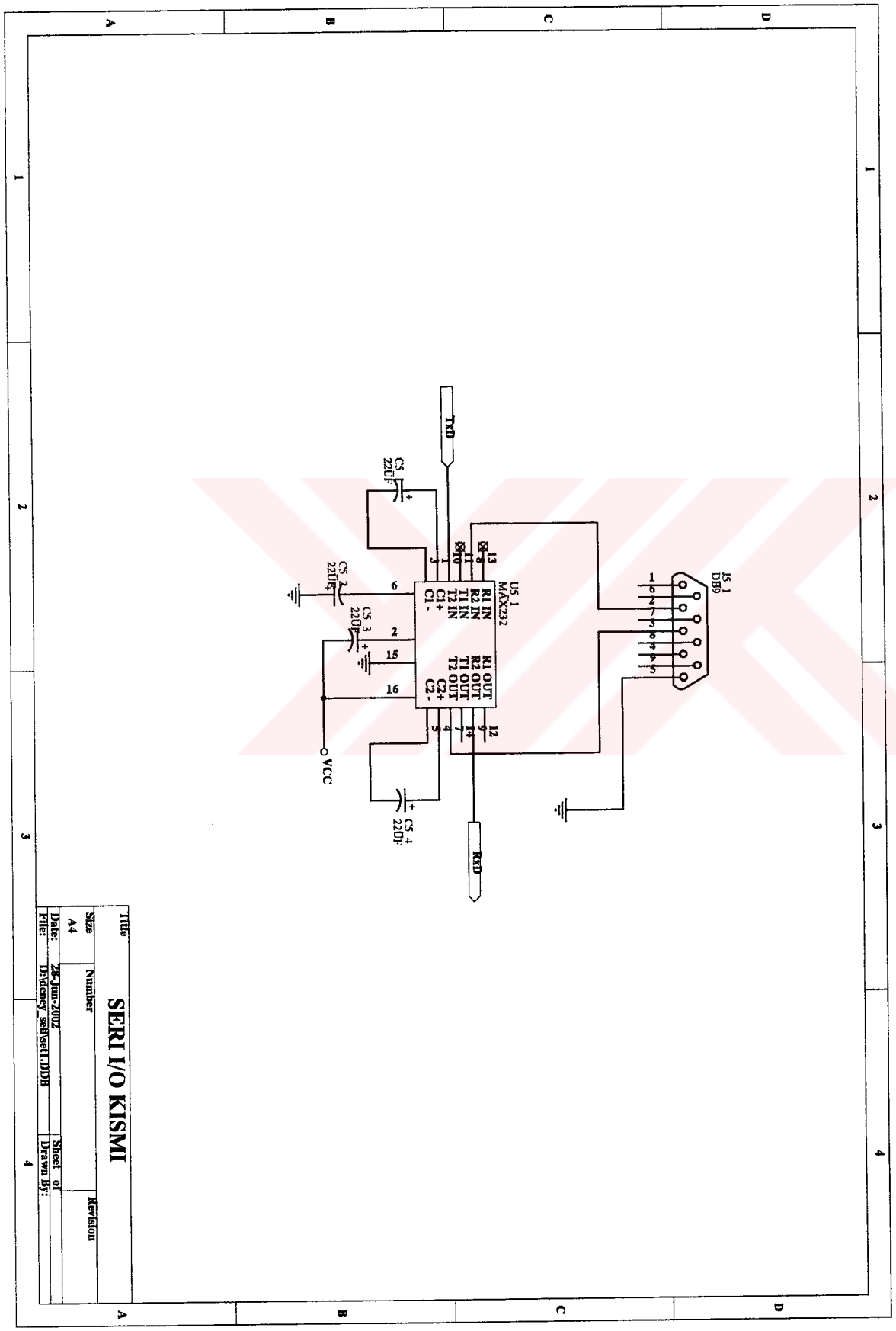

my_power
my_power.sch
My_memory
My_memory.sch
A[0..15]
D[0..7]
[0000-3FFF]
[4000-5FFF]
[6000-DFFF]
MEMRD
MEMWR
My_paralel
My_paralel.sch
A[0..15]
D[0..7]
RESET
[E000-E3FF]
pc8584
pc8584.sch
D[0..7]
RESET
[E000-E3FF]
my_cpu
my_cpu.Sch
A[0..15]
D[0..7]
TxD
RxD
RESET
[E000-E3FF]
[E400-E7FF]
[E800-EBFF]
[EC00-EFFF]
my_lcd
my_lcd.sch
A[0..15]
D[0..7]
[E400-E7FF]
klavye
klavye.sch
D[0..7]
[E800-EBFF]
My_seri
My_seri.sch
TxD
RxD
Title
SISTEM BLOK SEMASI
Size
A4
Number
Revision
Date:
28-Jun-2002
File:
D:\deney_set\set1.DDB
Sheet of
Drawn By:

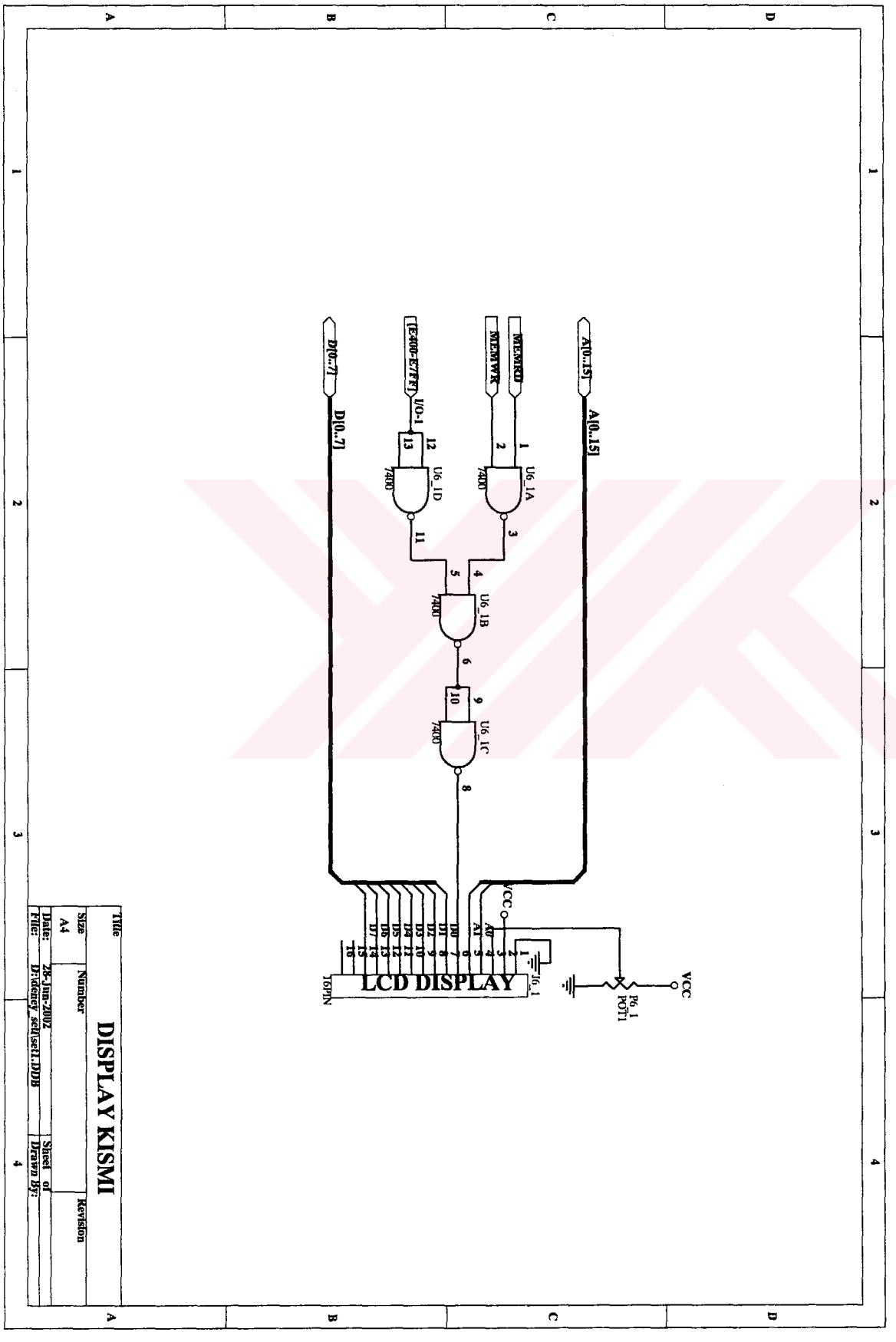

J1_1
CON2
C1_1
220uF
U1_1
MC7805
Vin
GND
+5V
C1_2
22uF
C1_3
100nF
C1_4
100nF
C1_5
100nF
C1_6
100nF
C1_7
100nF
VCC
VCC
R1_1
470
D1_1
LED
C1_8
100nF
C1_9
100nF
C1_10
100nF
C1_11
100nF
C1_12
100nF
C1_13
100nF
GUC KAYNAGI
Title
Size
A4
Number
Revision
Date:
25-Jun-2007
Sheet of
File:
D:\Desktop_sem\sem11\DDB
Drawn By:

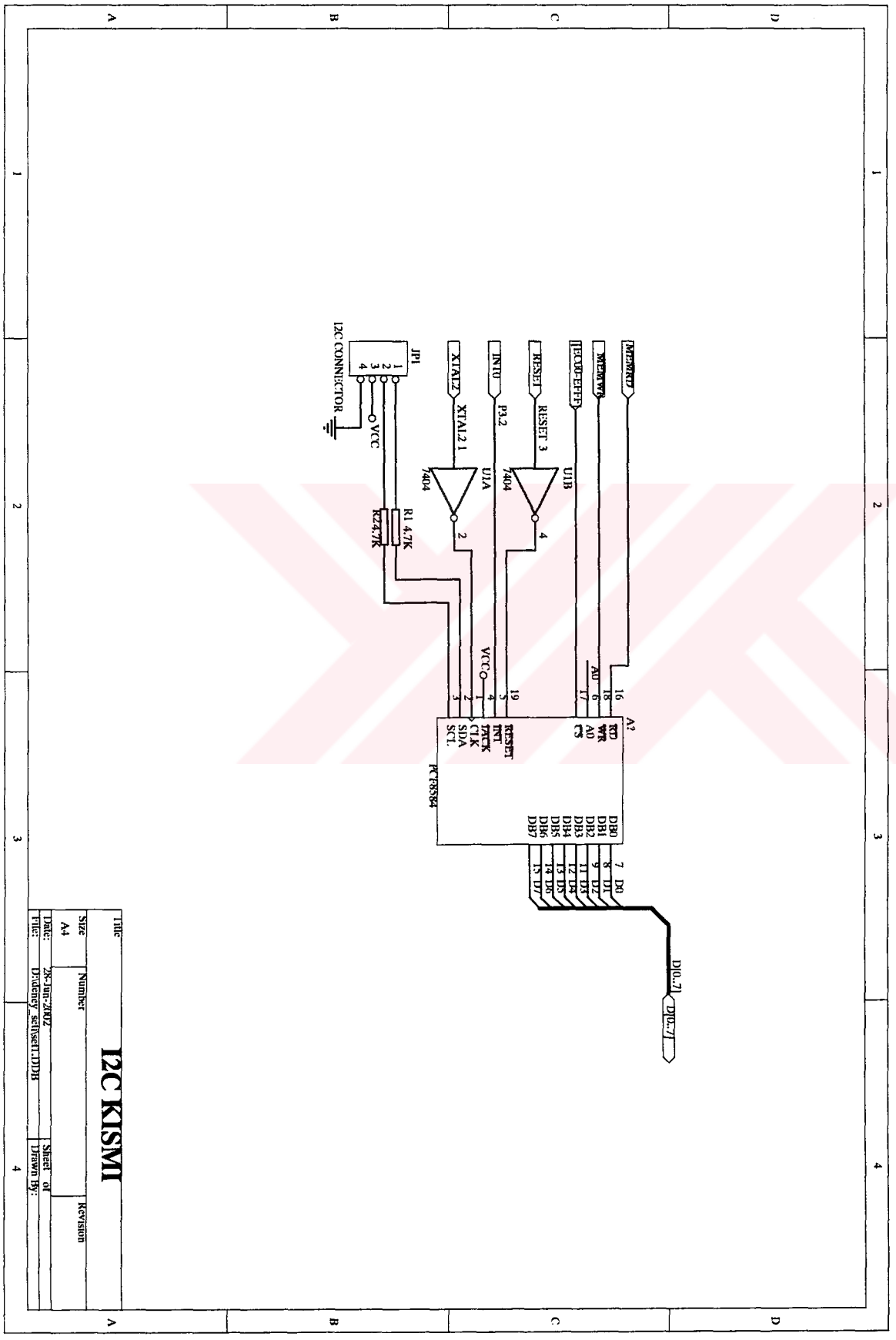

RESET CIRCUIT
MIKROISLEMCI KISMI
D[0..7]
A[0..15]
MEMRD
MEMWR
RESET
VCC
U2_1
U2_2
U2_3
U2_4
U2_5A
U2_5B
U2_5C
U2_5D
74573
MM74HC138
MM74HC08
[0000-1FFF]
[2000-3FFF]
[4000-5FFF]
[6000-7FFF]
[8000-9FFF]
[A000-BFFF]
[C000-DFFF]
[E000-FFFF]

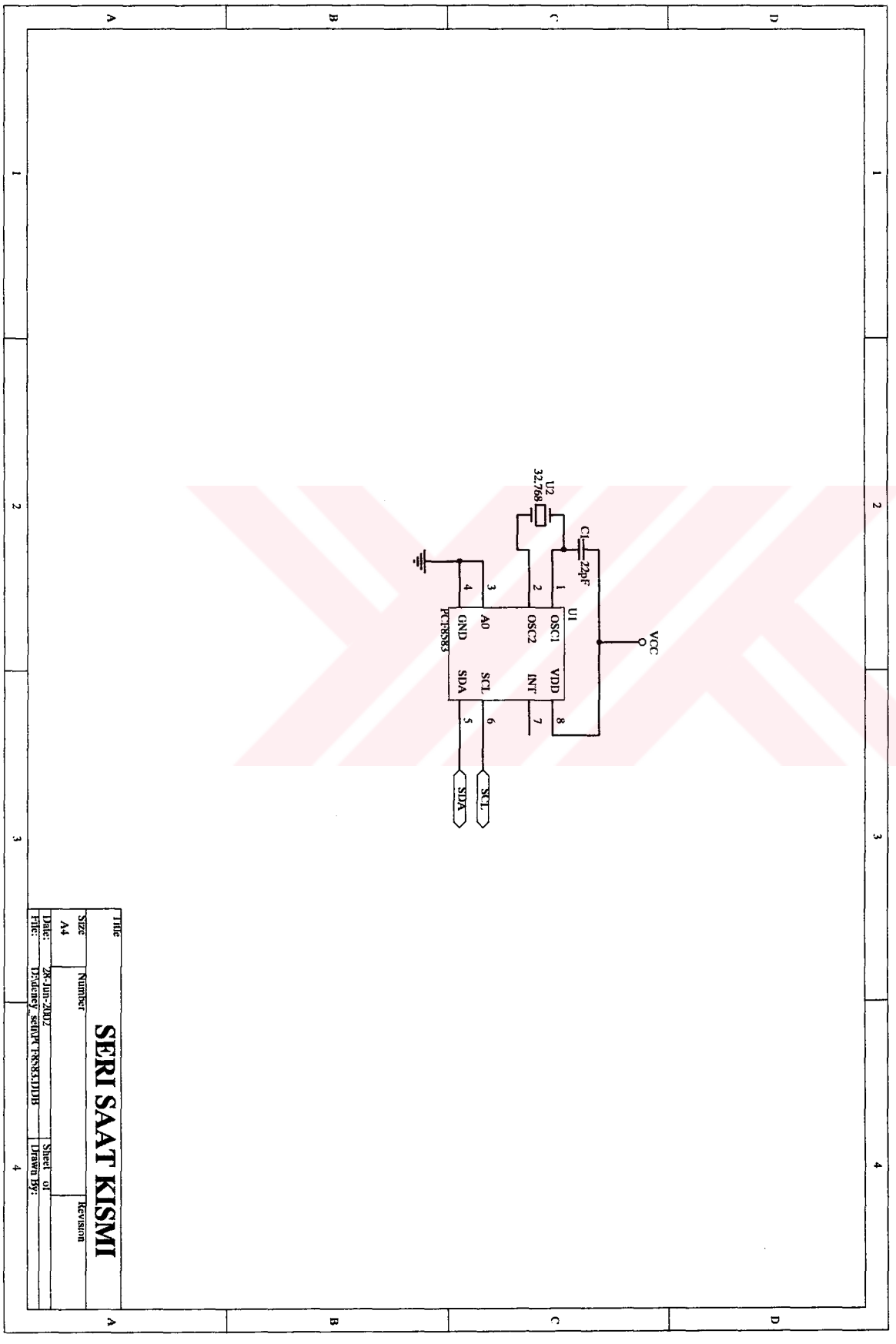

A[0..15]
D[0..7]
[0000-3FFF]
[4000-5FFF]
[6000-DFFF]
MEMRD
MEMWR
U3_1
27128
U3_2
6264
U3_3
62256
VCC
S3_1
8K/32K
Title
HAFIZA KISMI
Size
A4
Number
Revision
Date:
28-Jun-2002
File:
Drivdemy_sdlset1.DDB
Sheet of
Drawn By:


D[0..7]
A[0..15]
RESET
MEMRD
MEMWR
8255
PARALEL I/O KISMI
Size
A4
Number
Revision
Date:
28-Jun-2002
File:
D:\deney_set\set1.DDB
Sheet of
Drawn By:


TxD
RxD
VCC
U5_1
MAX232
R1 IN
R2 IN
T1 IN
T2 IN
C1+
C1 -
R1 OUT
R2 OUT
T1 OUT
T2 OUT
C2+
C2 -
J5_1
DB9
C5_1
22UF
C5_2
22UF
C5_3
22UF
C5_4
22UF
Title
SERI I/O KISMI
Size
A4
Number
Revision
Date:
28-Jun-2002
Sheet of
File:
D:\dersler_seri\seri1.DDB
Drawn By:


A[0..15]
MEMRD
MEMWR
E400-E7FF
I/O-1
D[0..7]
U6_1A
U6_1B
U6_1C
U6_1D
7400
J6_1
16PIN
LCD DISPLAY
VCC
R6_1
POT1
Title
DISPLAY KISMI
Size
A4
Number
Revision
Date:
28-Jun-2002
File:
Drivdgney_sesi(sesi1.DDB
Sheet of
Drawn By:


I2C KISMI
Title
Size
A4
Number
Revision
Date:
28-Jun-2002
File:
D:\Udemy_sel\sel1.DDB
Sheet of
Drawn By:
I2C CONNECTOR
JP1
VCC
XTAL2
XTAL2 1
P3.2
INTO
RESET
RESET 3
U1A
U1B
7404
R1 4.7K
R2 4.7K
A?
PCF8584
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
D0
D1
D2
D3
D4
D5
D6
D7
D[0..7]
MEMRD
MEMWR
ECUD-EFFE


U2
32.768
C1
22pF
VCC
U1
OSC1
OSC2
VDD
INT
A0
GND
SDA
SCL
PCF8583
SDA
SCL
Title
SERI SAAT KISMI
Size
A4
Number
Revision
Date:
28-Jun-2007
File:
D:\deney_set\PCF8583.DDB
Sheet of
Drawn By:

EK-B SİSTEM TASARIMINDA KULLANILAN PROGRAMLAR

I2CRTC.C

```
#include "reg591.h"
#include "i2crtc.h"
#define RTC_ADDRESS 0xA0
#define TIMEOUT 0xAA0F

void init_i2c(void);
void rtc_write_byte(unsigned char address, unsigned char byte);
unsigned char rtc_read_byte(unsigned char address );
unsigned char bcdtochar(unsigned char bcdnum);
unsigned char chartobcd(unsigned char n);
void i2c_isr(void);
void start_timer(void);
void stop_timer(void);
enum i2c_states { DATA_BYTE, FINISHED, DUMMY_WRITE };
enum i2c_operations { READ, WRITE };
unsigned char i2c_storage_address;
unsigned char i2c_byte;
unsigned char i2c_state;
unsigned char i2c_status;
unsigned char i2c_operation;

void init_i2c(void)
{
  SDA |= 1;
  SCL |= 1;
  TMOD |= 0x01;
  ES1 = 1;
  EA = 1;
}
```

```
unsigned char chartobcd(unsigned char n)
{
  return ((n / 10) << 4) | (n % 10);
}
unsigned char bcdtochar(unsigned char bcdnum)
{
  return (bcdnum & 0x0f) + (bcdnum >> 4) * 10;
}
void rtc_write_byte(unsigned char address, unsigned char byte)
{
  i2c_storage_address = address;
  i2c_byte = chartobcd(byte);
  i2c_status = I2C_BUSY;
  i2c_operation = WRITE;
  S1CON = 0x40;
  STA = 1;
  while(i2c_status == I2C_BUSY);
}
unsigned char rtc_read_byte(unsigned char address )
{
  i2c_storage_address = address;
  i2c_status = I2C_BUSY;
  i2c_operation = READ;
  i2c_state = DUMMY_WRITE;
  S1CON = 0x40;
  STA = 1;
  while(i2c_status == I2C_BUSY);
  return i2c_byte ;
}
void start_timer(void)
{
```

```
  TL0 = TIMEOUT & 0xFF;
  TH0 = (TIMEOUT >> 8) & 0xFF;
  TR0 = 1;
}
void stop_timer(void)
{
  TR0 = 0;
  TF0 = 0;
}
void i2c_isr(void) interrupt 5 using 1
{
  switch(S1STA)
  {
    // START CONDITION TRANSMITTED
    case 0x08:
      STA = 0;
      start_timer();
      S1DAT = RTC_ADDRESS & 0xFE;     // transmit Slave Address
                                         +Write command (SLA + W)
      break;
      // REPEATED START CONDITION TRANSMITTED
     case 0x10:
      STA = 0;
      if (i2c_operation == READ && i2c_state != DUMMY_WRITE)
        {
          S1DAT = RTC_ADDRESS | 0x01;  // transmit Slave Address +
                                            Read command (SLA + R)
        }
        else
        {
        S1DAT = RTC_ADDRESS & 0xFE;  // transmit Slave Address +
```

```
                                        Write command (SLA + W)
  }
  break;
  // SLAVE ADDRESS + WRITE TRANSMITTED - ACK RECEIVED
case 0x18:
  stop_timer();
  S1DAT = i2c_storage_address;
  i2c_state = DATA_BYTE;
  break;
  // SLAVE ADDRESS + WRITE TRANSMITTED - NO ACK RECEIVED
case 0x20:
          if (TF0)
          {
           STO = 1;
           stop_timer();                  // stop timer 0
           i2c_status = I2C_ERROR;
           }
          else
          {
           STA = 1;
          }
  break;
  // DATA BYTE OR REGISTER ADDRESS TRANSMITTED - ACK RECEIVED
case 0x28:
  switch(i2c_state)
  {
    case DATA_BYTE:
                    if (i2c_operation == READ)
          {
            STA = 1;
          }
```

```
            else
            {
             S1DAT = i2c_byte;          // transmit data byte
             i2c_state = FINISHED;
            }
         break;
       case FINISHED:
        STO = 1;
        i2c_status = I2C_OK;
        break;
     }
       break;
     // DATA BYTE OR REGISTER ADDRESS TRANSMITTED - NO ACK RECEIVED
     case 0x30:
       STO = 1;
       i2c_status = I2C_ERROR;       // status of operation is ERROR
     break;
     // SLAVE ADDRESS + READ TRANSMITTED - ACK RECEIVED
     case 0x40:
       break;
     // SLAVE ADDRESS + READ TRANSMITTED - NO ACK RECEIVED
     case 0x48:
       STO = 1;
       i2c_status = I2C_ERROR;      // status of operation is ERROR
     break;
     // DATA BYTE RECEIVED - NO ACK RETURNED
   case 0x58:
     i2c_byte = S1DAT;
     i2c_status = I2C_OK;           // status of operation is OK
     STO = 1;
     break;
```

```
      // UNKNOWN STATE
    default:
        STO = 1;
        i2c_status = I2C_ERROR;
        ES1 = 0;                          // disable I2C interrupts
        break;
  }
  SI = 0;
}
```

I2crtc.h

```
#define I2C_ERROR 1
#define I2C_OK 2
#define I2C_BUSY 0
#define SECONDS 0x02
#define MINUTE 0x03
#define HOUR 0x04
#define YEAR_DAY 0x05
#define WEEKDAY_MONTH 0x06

extern void init_i2c(void);
extern void rtc_write_byte ( unsigned char address, unsigned char
byte );
extern unsigned char rtc_read_byte ( unsigned char address );
extern unsigned char i2c_status;
```

Main.c

```
#include "reg652.h"
#include "i2crtc.h"
#include <string.h>
#define clrDisp         0x01
#define offCur          0x0C
code unsigned char *months[] = { "OCAK",
                                 "SUBAT",
                                 "MART",
                                 "NISAN",
                                 "MAYIS",
                                 "HAZIRAN",
                                 "TEMMUZ",
                                 "AGUSTOS",
                                 "EYLUL",
                                 "EKIM",
                                 "KASIM",
                                 "ARALIK"
                                };

enum weekdays { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
                FRIDAY, SATURDAY };
enum months { JANUARY = 1, FEBRUARY, MARCH, APRIL, MAY, JUNE,
              JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER
};

struct Time
{
 unsigned char seconds;
 unsigned char minutes;
```

```
 unsigned char hours;
};

struct Date
{
 unsigned char days;
 unsigned char weekdays;
 unsigned char months;
 unsigned char years;
};

struct Time current_time = { 40 , 59 , 23 };
struct Date current_date = { 31 , WEDNESDAY , DECEMBER , 01 };
unsigned char old_month;
unsigned char store_time ( struct Time *t , struct Date *d );
unsigned char read_time ( struct Time *t , struct Date *d);
void display_time ( struct Time *t , struct Date *d );

extern unsigned char bcdtochar(unsigned char bcdnum);
extern unsigned char chartobcd(unsigned char n);
extern void resetLCD ( void );
extern void initLCD ( void );
extern void printLCDbyte (unsigned char LCD_X , unsigned char LCD_Y,
unsigned char byteDATA);
extern void prnLCDstrXY ( unsigned char C_line, unsigned char C_pos,
unsigned char *data_STR );
extern void delay ( int msec );
extern void wrLCDcom ( unsigned char LCD_command );


unsigned char store_time(struct Time *t , struct Date *d )
```

```
{
  rtc_write_byte ( SECONDS , t->seconds);
  if (i2c_status == I2C_ERROR) return 0;
  rtc_write_byte ( MINUTE , t->minutes);
  if (i2c_status == I2C_ERROR) return 0;
  rtc_write_byte ( HOUR , t->hours);
  if (i2c_status == I2C_ERROR) return 0;

  rtc_write_byte ( YEAR_DAY , bcdtochar ( ( d->years << 6 ) |
chartobcd ( d->days ) ) );
  if (i2c_status == I2C_ERROR) return 0;

  rtc_write_byte ( WEEKDAY_MONTH ,  bcdtochar ( ( d->weekdays  << 5
) |  chartobcd ( d->months ) ) );
  if (i2c_status == I2C_ERROR) return 0;

  return 1;
}

unsigned char read_time ( struct Time *t , struct Date *d)
{
  unsigned char day;
  unsigned char month;

  t->seconds = bcdtochar ( rtc_read_byte ( SECONDS ) );
  if (i2c_status == I2C_ERROR) return 0;
  t->minutes = bcdtochar ( rtc_read_byte ( MINUTE ) );
  if (i2c_status == I2C_ERROR) return 0;
  t->hours = bcdtochar ( rtc_read_byte ( HOUR ) );
  if (i2c_status == I2C_ERROR) return 0;
```

```
  day = rtc_read_byte ( YEAR_DAY );
  if (i2c_status == I2C_ERROR) return 0;
  d->days = bcdtochar ( 0x3F & day );
  d->years = bcdtochar ( day >> 6 );

  old_month = d->months;

  month = rtc_read_byte ( WEEKDAY_MONTH );

  if (i2c_status == I2C_ERROR) return 0;
  d->weekdays = bcdtochar ( month >> 5 );
  d->months = bcdtochar ( month & 0x1F );

  if ( old_month != d->months ) wrLCDcom ( clrDisp );

  return 1;
}

void display_time ( struct Time *t , struct Date *d)
{
  printLCDbyte ( 0 , 4 , chartobcd ( t->hours ) );
  prnLCDstrXY ( 0 , 6 , ":" );
  printLCDbyte ( 0 , 7 , chartobcd ( t->minutes ) );
  prnLCDstrXY ( 0 , 9 , ":" );
  printLCDbyte ( 0 , 0x0A , chartobcd ( t->seconds ) );

  printLCDbyte ( 1 , ( 16 - strlen ( months [ d->months - 1] ) ) / 2
- 3 , chartobcd ( d->days ) );
  prnLCDstrXY ( 1 , ( 16 - strlen ( months [ d->months - 1] ) ) / 2
- 1 , "-" );
  //prnLCDstrXY ( 1 , 3 , months [ d->months - 1] );
```

```
  prnLCDstrXY ( 1, ( 16 - strlen ( months [ d->months - 1] ) ) / 2 ,
months [ d->months - 1] );
  //printLCDbyte ( 1 , 7 , chartobcd ( d->months ) );
  //prnLCDstrXY ( 1 , 9 , "-" );
  prnLCDstrXY ( 1, ( 16 - strlen ( months [ d->months - 1] ) ) / 2 +
strlen ( months [ d->months - 1] ), "-" );

  printLCDbyte ( 1 , ( 16 - strlen ( months [ d->months - 1] ) ) / 2
+ strlen ( months [ d->months - 1] ) + 1 , chartobcd ( d->years ) );

}
void main(void)
{
  resetLCD ( );
  initLCD ( );
  init_i2c();
   test_bit |= 1;
   wrLCDcom ( offCur );
   /*if ( !store_time ( &current_time , &current_date ) )
         prnLCDstrXY ( 0 , 0 ," ! writing error " );  */
  while(1)
  {
    while ( test_bit == 1);
      if ( read_time ( &current_time , &current_date ) )
       display_time(&current_time , &current_date);
      else
        prnLCDstrXY ( 0 , 0 ," ! reading error " );
    while ( test_bit == 0 );
  }
}
```

I2cmonitor.c

```
#include <stdio.h>
#include <reg51.h>
#include "pcf_8584.h"

#define BUFFER_START_ADDRESS  0x6000
#define BUFFER_LENGTH           0x50

xdata unsigned char PCF8584_DATA    _at_ 0xE800;
xdata unsigned char PCF8584_CONTROL _at_ 0xE801;

static unsigned char hex2ascii ( unsigned char HexNUMBER );
static void sendcharPC ( unsigned char sendBYTE );
static void send_PC_BYTE ( unsigned char byteDATA );
extern void initilationUART ( void );
//xdata unsigned char dene _at_ 0x6000;

unsigned char xdata *dene;
//unsigned char xdata *deneme;

bit READ_ERROR;
 /*****************************************************************/
 /***                                                           ***/
 /***                   Control BITS                            ***/
 /***                   ----------                              ***/
 /***       | PIN | ESO | ES1 | ES2 | ENI | STA | STO | ACK |    ***/
 /***       -----------------------------------------------     ***/
 /***                                                           ***/
 /*****************************************************************/
```

```
void main ( void )
{
 unsigned int buffer;
 unsigned int i;
 unsigned char I2C_DATA;

 dene = BUFFER_START_ADDRESS ;
 //deneme = 0x7000;

 initilationUART ( );
 init_PCF8584( );
 buffer = 0;
 printf (" \n\r I2C bus is listening.....\n" );
  do
  {
   while ( PCF8584_CONTROL & I2C_PCF_PIN ) ;
   if ( PCF8584_CONTROL & I2C_PCF_AAS ) *dene = 0xFF;
       else *dene = 00;

    dene++;
    *dene = PCF8584_DATA ;
    dene++;
    buffer++;
  }while( buffer < BUFFER_LENGTH );

  dene = BUFFER_START_ADDRESS ;
   for ( i = 0; i < BUFFER_LENGTH ; i++ )
       {
            //printf( "%BX" , I2C_DATA ); putchar ('\n' );
            I2C_DATA = *dene ;
        if ( I2C_DATA == 0XFF )
```

```
            {
                 sendcharPC ( 0x0D );
                  dene++;
                 printf( "[START] " ); send_PC_BYTE ( *dene );
                 sendcharPC ( 0x20 );
                 if ( *dene & 0x01 ) printf( "[RD] " );
                  else printf( "[WR] " );
            }
             else {
               dene++;
                   send_PC_BYTE ( *dene ); sendcharPC ( 0x20 );
                    }
          dene++;
        }
    //
   //sendcharPC ( I2C_DATA );
   while ( 1 );
 }

void init_PCF8584 ( void )
 {
  PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_S0_REG ;  // S0 register
                                                    sets to Slave address
  PCF8584_DATA = 0 ;
  PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_S2_REG ;
  // sets Clock Register fCLK = 12MHz and fSCK = 45KHz.
  PCF8584_DATA = I2C_PCF_CLK12 | I2C_PCF_TRANS_45 ;

  PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_ESO | I2C_PCF_S0_REG |
  I2C_PCF_ACK ;  // sets I2C bus IDLE MODE
 }
```

```
/*****************************************************************/
/***                          Control BITS                     ***/
/***       ------------------------------------------------    ***/
/***       | PIN | ESO | ES1 | ES2 | ENI | STA | STO | ACK |    ***/
/***       ------------------------------------------------    ***/
/*****************************************************************/

unsigned char start_I2C_Send_Address ( char Slave_Address )
{
 PCF8584_CONTROL = I2C_PCF_ESO ;       // Enable I2C bus
 if ( wait_I2C_busy_BB () != OK ) return ERROR;
 PCF8584_DATA = Slave_Address ;
 PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_ESO | I2C_PCF_STA |
 I2C_PCF_ACK ;  // 0XC5
 if ( wait_I2C_PIN ( ) != OK ) return ERROR;
 if ( test_I2C_LRB ( ) == 1 ) return ERROR;
 return OK;
}

unsigned char I2C_Send_Data ( unsigned char I2C_Data ,unsigned char
last_flag )
{
 PCF8584_DATA = I2C_Data;
 if ( wait_I2C_PIN ( ) != OK ) return ERROR;
 if ( test_I2C_LRB ( ) == 1 ) return ERROR;
 if ( last_flag == LAST ) stop_I2C ( );
 return OK ;
}

unsigned char I2C_Read_Data ( unsigned char I2C_READ_ADDRESS )
{
```

```
   unsigned char read_BYTE ;

   PCF8584_CONTROL =  I2C_PCF_ESO | I2C_PCF_STA | I2C_PCF_ACK ;
   PCF8584_DATA = I2C_READ_ADDRESS ;
   if ( wait_I2C_PIN ( ) != OK ) { READ_ERROR = ERROR ;return ERROR;
}
   if ( test_I2C_LRB ( ) == 1 )  { READ_ERROR = ERROR ; return
ERROR;}
   read_BYTE = PCF8584_DATA ;
   if ( wait_I2C_PIN ( ) != OK )
   { READ_ERROR = ERROR ;return ERROR; }
   if ( test_I2C_LRB ( ) == 1 )
   {
    READ_ERROR = ERROR ;return ERROR; }
    PCF8584_CONTROL = I2C_PCF_ESO ; //0x40
    read_BYTE = PCF8584_DATA ;
    if ( wait_I2C_PIN ( ) != OK )
       {
        READ_ERROR = ERROR ;return ERROR;
        }

   PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_ESO | I2C_PCF_STO |
                     I2C_PCF_ACK ; //0xC3
   return read_BYTE;
  }

/*****************************************************************/
/***                          Control BITS                     ***/
/***       ---------------------------------------------       ***/
/***       | PIN | ESO | ES1 | ES2 | ENI | STA | STO | ACK |    ***/
/***       ---------------------------------------------       ***/
/*****************************************************************/
```

```
void stop_I2C ( void )
{
 PCF8584_CONTROL = I2C_PCF_PIN | I2C_PCF_ESO | I2C_PCF_STO |
                   I2C_PCF_ACK ;  // 0xC3
}

//*****************************************************************//
unsigned char wait_I2C_busy_BB ( void )
{
 unsigned int wait;
 for ( wait = 0; wait < I2C_TIMEOUT ; wait++)
      {
        if ( PCF8584_CONTROL & I2C_PCF_BB ) return OK;
       }
 return ERROR;
}
//*****************************************************************/

unsigned char wait_I2C_PIN ( void)
{
 unsigned int wait;
 for ( wait = 0; wait < I2C_TIMEOUT ; wait++)
      {
           if ( !( PCF8584_CONTROL & I2C_PCF_PIN ) ) return OK;
       }
 return ERROR;
}

//*****************************************************************/
```

```
unsigned char test_I2C_LRB ( void )
{
 unsigned char I2C_status;
 I2C_status = PCF8584_CONTROL ;
 if ( I2C_status & I2C_PCF_LBR ) return 1;
 return 0;
}

static void send_PC_BYTE ( unsigned char byteDATA )
 {
  unsigned char highNIBBLE;
  unsigned char lowNIBBLE;
  highNIBBLE =  ( byteDATA & 0xF0 ) >> 4;
  lowNIBBLE = byteDATA & 0x0F ;
  sendcharPC ( hex2ascii ( highNIBBLE ) ) ;
  sendcharPC ( hex2ascii ( lowNIBBLE ) ) ;
 }

 static void sendcharPC ( unsigned char sendBYTE )
 {
  while ( !TI );
  TI = 0;
  ACC = sendBYTE ;
  SBUF = ACC ;
  //SBUF = sendBYTE ;
 }

 static unsigned char hex2ascii ( unsigned char HexNUMBER )
 {
  unsigned char asciiNUMBER;
```

```
 if ( HexNUMBER < 10 ) asciiNUMBER = HexNUMBER + 0x30;
 else asciiNUMBER = HexNUMBER + 0x37 ;

 return asciiNUMBER;
}
```

Serial.c

```
/*-------------------------------------------------------------------
It was written by Sedat ATMACA
-------------------------------------------------------------------*/

 #include <stdio.h>                /* prototype declarations for
                                          I/O functions */
 #include <reg51.h>                /* special function register
                                     declarations   */
 void initilationUART ( void );
 /*void main (void)
 {
  initilationUART ();
  while (1) {
  //P1 ^= 0x01;             /* Toggle P1.0 each time we print */
  //printf ("Hello World\n");   /* Print "Hello World" */
 // }
 //}

  void initilationUART ( void )
  {

   SCON  = 0x50 ;          //** SCON: mode 1, 8-bit UART, enable rcvr
   TMOD  = 0x20 ;         //** SCON: mode 1, 8-bit UART, enable rcvr
   PCON &= 0x7F ;         //** TMOD: timer 1, mode 2, 8-bit reload
   //PCON  |= 0x80 ;
   TH1   = 0xFD ;        //** SCON: mode 1, 8-bit UART, enable rcvr
   TL1   = 0xFD ;       //** TH1:  reload value for 1200 baud @ 16MHz
   TR1   = 1;            //** TR1:  timer 1 run
   //RI    = 0;
```

```
 RI     = 1;
 REN    = 1;
 //TI     = 1;
 TI     = 0;
}
```

Pch8584.h

```
/*****************************************************************/
/***              Name  :Sedat ATMACA                          ***/
 /***             Date  :19.11.2001                            ***/
 /***             Time  :22.05.00                              ***/
/*****************************************************************/

/*****************************************************************/
/***                    Control BITS                           ***/
/***       ------------------------------------------------    ***/
/***       | PIN | ESO | ES1 | ES2 | ENI | STA | STO | ACK |    ***/
/***       ------------------------------------------------    ***/
/*****************************************************************/

 #define I2C_PCF_PIN    0x80  // define PIN     Pending Interrupt Not
 #define I2C_PCF_ESO    0x40  // define Enable Serial Output
 #define I2C_PCF_ES1    0x20  //
 #define I2C_PCF_ES2    0x10  //
 #define I2C_PCF_ENI    0x08  // define Enable External Interrupt
                                 output
 #define I2C_PCF_STA          0x04  // define START
 #define I2C_PCF_STO          0x02  // define STOP
 #define I2C_PCF_ACK          0x01  // define ACK



/*****************************************************************/
/***                         Status BITS                       ***/
/***    ------------------------------------------------------   ***/
```

```
/***    | PIN | 0(x) | STS | BER | AD0/LBR | ASS | LAB | BB |    ***/
/***    -------------------------------------------------------    ***/
/***                                                               ***/
/*****************************************************************/

 #define I2C_PCF_PIN    0x80  // define PIN     Pending Interrupt Not
 #define XXX            0x40  // define XXX     not define
 #define I2C_PCF_STS    0x20  // define STS     Slave Stop Detected
 #define I2C_PCF_BER    0x10  // define BER     Bus Error
 #define I2C_PCF_AD0    0x08  // define AD0     Own Address Detected
 #define I2C_PCF_LBR    0x08  // define LBR     Own Address Detected
 #define I2C_PCF_AAS    0x04  // define ASS     Address As Slave
 #define I2C_PCF_LAB    0x02  // define LAB     Lost Arbitration Bit
 #define I2C_PCF_BB     0x01  // define _BB     Bus Busy
 #define I2C_PCF_S0_REG 0x00  // define S0_REG | S0' Register
 #define I2C_PCF_S2_REG 0x20  // define S2_REG | S2 Clock Register
 #define I2C_PCF_S3_REG 0x10  // define S3_REG | S3 Interrupt
                                 Register

/*****************************************************************/
/***                    Transmission Frequencies                 ***/
/*****************************************************************/

 #define I2C_PCF_TRANS_90      0x00 // define FSCLK90  | fSCL 90KHz
 #define I2C_PCF_TRANS_45      0x01 // define FSCLK45  | fSCL 45KHz
 #define I2C_PCF_TRANS_11      0x02 // define FSCLK11  | fSCL 11KHz
 #define I2C_PCF_TRANS_15      0x03 // define FSCLK01  | fSCL 1.5KHz

/*****************************************************************/
/***              Chip Clock Frequencies                         ***/
/*****************************************************************/
```

```
#define I2C_PCF_CLK3          0x00 // define FSCLK03   | fSCL 3MHz
#define I2C_PCF_CLK443        0x10 // define FSCLK04   | fSCL 4.43MHz
#define I2C_PCF_CLK6          0x14 // define FSCLK06   | fSCL 6MHz
#define I2C_PCF_CLK8          0x18 // define FSCLK08   | fSCL 8MHz
#define I2C_PCF_CLK12         0x1C // define FSCLK12   | fSCL 12MHz
#define I2C_TIMEOUT          0x5000
#define OK        0
#define ERROR     1
#define NOT_LAST  0
#define LAST      1

extern void init_PCF8584 ( void );
extern unsigned char start_I2C_Send_Address ( unsigned char
Slave_Address );
extern unsigned char I2C_Send_Data ( unsigned char I2C_Data,
unsigned char last_flag );
extern unsigned char I2C_Read_Data ( unsigned char I2C_READ_ADDRESS
 );
extern unsigned char wait_I2C_busy_BB ( void );
extern unsigned char wait_I2C_PIN ( void);
extern unsigned char test_I2C_LRB ( void );
extern void stop_I2C ( void );
```

ÖZGEÇMİŞ

1974, Ankara doğumlu olan Sedat Atmaca, ilk ve orta öğrenimini Ankara'da tamamladı. Lise öğrenimini Ankara Yenimahalle Anadolu Teknik-Teknik ve Endüstri Meslek Lisesi Elektronik Bölümünde yaptı. Kazanmış olduğu Gazi Üniversitesi Teknik Eğitim Fakültesi Elektronik Bölümünden 26.09.1996 yılında mezun oldu. 1998 yılında askerlik görevine başladı ve 1999 yılında askerlik görevini tamamladı. 2000 yılında Sakarya Üniversitesi Teknik Eğitim Fakültesinde Yüksek Lisansa Başladı. 16.09.1996 tarihinde M.E.B Körfez Anadolu Teknik-Teknik ve Endüstri Meslek Lisesinde göreve başladı ve halen aynı göreve devam etmektedir.