بسم الله الرحمن الرحیم

دانشگاه تربیت معلم سبزوار

پایان نامه کارشناسی

پیاده سازی شبکه ای از میکرو کنترلرها

جهت کنترل لوازم الکتریکی

ارائه دهنده : مهدی صالح پور

استاد راهنما : دکتر سید ابراهیم حسینی

شهریور ۸۶

چکیده

در دنیای صنعتی امروز، انتقال دیتا به عنوان یکی از مهمترین بخشهای پروسه های کنترلی شناخته شده است و طراحان در تلاش برای بوجود آمدن پروتکل های جدیدی با ایمنی، صحت و سرعت بالا در انتقال دیتا هستند. در این پایان نامه، در ابتدا بـه معرفـی تبادل دیتا (Data Communication) پرداخته و برخی از مباحث کلیـدی آن از جملـه ارتبـاط سـریال و مـوازی، فرسـتنده و گیرنده، اتصالات و تبادل شفاف، ساختار[1] Master-Slave ، سرعت انتقال، مدوله سازی،[2] Handshaking را مختصراً توضـیح می دهیم. سپس پروتکل های موجود را بررسی کرده و به قسمت اصلی پایان نامه، معرفـی پروتکـل I^2C مـی پـردازیم کـه شـامل سخت افزار آن، ساختار داخلی، شیوه های آدرس دهی، مدهای کاری است. سپس این پروتکل را با سـایر پروتکـل هـای معرفـی شده مقایسه کرده و کاربرد های این پروتکل را بررسی نموده و مثال هایی نیز ارائه می کنیم. و در نهایت هدف اصلی این پروژه که پیاده سازی عملی این پروتکلتوسط میکرو کنترلر avr می باشد. به این نحو که ایـن پـروژه متـشکل از یـک master و تعـدادی slave می باشد که هدف آن کنترل یک پروسه صنعتی است.

کلمات کلیدی

I^2C ، میکروکنترلر، SDA ، SCL ، Master ، Slave

[1] - ارباب و برده
[2] - تبادل دو طرفه اطلاعات (دست دهی)

تقدیم به

پدر و مادرم

که همواره مدیونشان هستم

فهرست

مقدمه ۷

فصل اول : کلیات انتقال دیتا

۱-۱- چگونه تبادل دیتا صورت می گیرد ۸

۱-۲- فرستنده و گیرنده ۹

۱-۳- اتصال صحیح ۱۰

۱-٤-تبادل شفاف ۱۰

۱-٥-ساختار Master و Slave ۱۰

۱-٦-سرعت انتقال ۱۱

۱-۷-Handshaking ۱۲

۱-۷-۱- Handshaking نرم افزاری ۱۲

۱-۷-۲- Handshaking سخت افزاری ۱۲

فصل دوم : بررسی پروتکل های موجود

۲-۱- پروتکل USART ۱٤

۲-۲- پروتکل SPI ۱٥

۲-۳- پروتکل CAN ۱۷

۲-۳-۱- مزایای باس CAN ۱۸

۲-٤- پروتکل USB ۱۸

۲-٤-۱- مزایای باس USB ۱۹

۲-٥- پروتکل IEEE 1394 ..... ۱۹

فصل سوم : I^2C

۳-۱- تاریخچه I^2C ۲۱

۳-۲- مزایای باس برای طراح ۲۱

۳-۳- سخت افزار باس I^2C ۲۲

۳-٤- مشخصات رابط سریال I^2C ۲٤
۳-٥- شرایط ارسال start و stop ۲٥
۳-٦- فرمت ارسال داده ۲٦
۳-۷- مسئله همزمان سازی پالس ساعت ۲۷
۳-۸- مسئله داوری و حاکمیت یک Master ۲۷
۳-۹- آدرس دهی ۲۸
۳-۹-۱- آدرس دهی ۷ بیتی ۲۸
۳-۹-۲- آدرس دهی ۱۰ بیتی ۳۲

فصل چهارم : ساختار داخلی I^2C

٤-۱- ساختار داخلی ماژول TWI ۳٤
٤-۱-۱- واحد تنظیم Bit Rate ۳٤
٤-۱-۲- واحد Bus Interface ۳٥
٤-۱-۳- واحد Address Match ۳٥
٤-۱-٤- واحد Control ۳٦
٤-۲- رجیستر های TWI ۳٦
٤-۳- روش استفاده از TWI ٤۰
٤-٤-۱- تحولات در Fast – Mode ٤۲
٤-٤-۲- تحولات در Hi Speed – Mode (HS-Mode) ٤۲
٤-٥- فرمت ارسال داده های سریال در HS-Mode ٤۳
٤-٦- مدهای کاری TWI ٤٥
٤-٦-۱- مد Master Transmitter ٤٥
٤-٦-۲- مد Master Reciver ٤۹
٤-٦-۳- مد Slave Reciver ٥۱
٤-٦-٤- مد Slave Transmitter ٥٤

فصل پنجم : مقایسه و کاربردها

۵-۱- مقایسه I^2C با سایر پروتکل ها ۵۷
۵-۲- کاربردهای I^2C ۵۹
۵-۳-۱- I^2C و RS232 ..... ۵۹
۵-۳-۲- ارتباط باس موازی با I^2C ۶۳

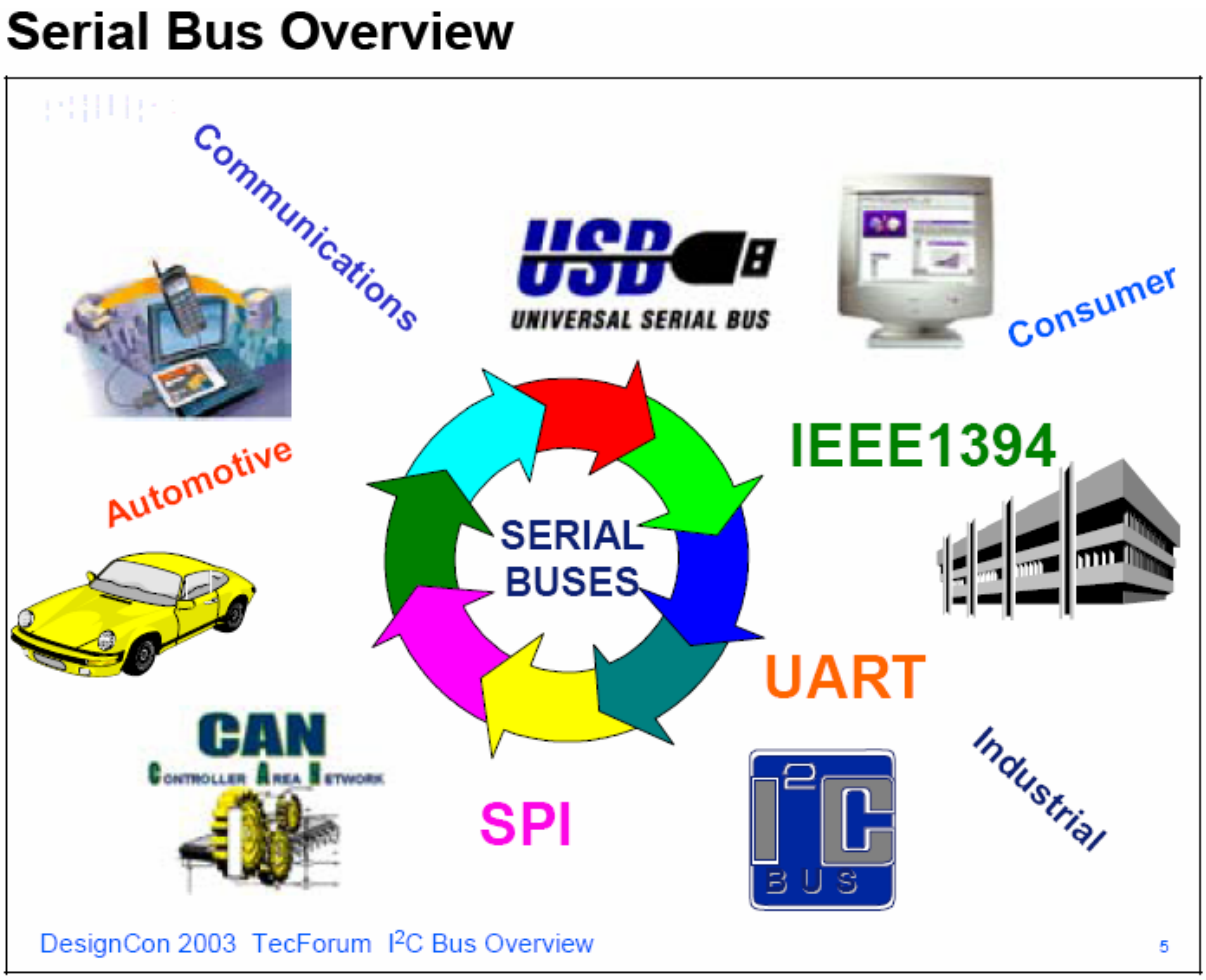
فصل ششم : نرم افزار

۶-۱- نرم افزار ۶۶
۶-۲- شماتیک مدار ۸۲

پیشنهاد ۸۴
نتیجه گیری ۸۵
منابع ۸۶

مقدمه

این پروتکل توسط شرکت فیلیپس در دهه ۱۹۸۰ جهت ارتباط دستگاههای TV با پردازشگر ابداع شده است. که بدلیل سادگی و سرعت مناسب آن مورد توجه دیگر سازندگان قطعات الکترونیک قرار گرفت و هم اکنون به عنوان یکی از پروتکل های کاربردی در صنعت شناخته شده است.



فصل اول

کلیات انتقال دیتا

۱-۱- چگونه تبادل دیتا انجام می پذیرد

هدف از تبادل دیتا، انتقال دیتا بین ۲ یا تعداد بیشتری واحد می باشد. به عنوان یک اصل، این کاراکترها می توانند دستورات باشند که نیاز به نمایش دارند.ساده ترین سطح زبان کامپیوتر، کاراکترهای باینری است که شامل ۷ یا ۸ عدد، صفر یا یک می باشد. اکثر کامپیوترها با این سطح کار می کنند.

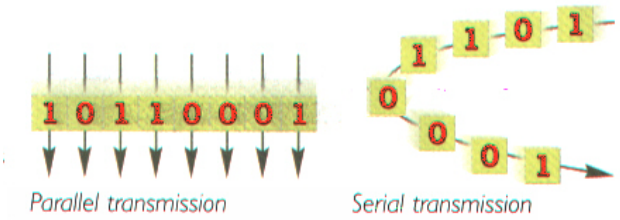
یکی از استانداردهای معمول در کامپیوترها، استاندارد ASCII[3] می باشد که شامل ۱۲۸ کاراکتر است که هر کدام از آنها از ۷ بیت تشکیل شده است. باید توجه داشت که ارتباطات در داخل کامپیوتر با سرعت زیادی انجام می شود و برای ارتباط با محیط خارج باید ارتباطات همزمان شوند و همچنین باید صحت تبادل دیتا، کنترل شود.

استانداردهای مختلفی از ASCII وجود دارد. به عنوان مثال Extended ASCII[4] که از هشتمین بیت نیز برای انتقال data[5] استفاده می کند.

دو روش برای انتقال دیتا وجود دارد

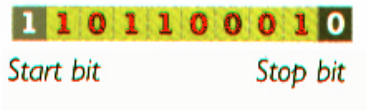
۱- سریال

۲- موازی



در انتقال موازی، برای هر بیت یک مسیر در نظر گرفته شده است. بنابراین کاراکترها می توانند بطور همزمان ارسال شوند. با توجه به این مزیت، که سرعت بالای انتقال است این روش در سیستمهای ارتباطی کوتاه مورد استفاده قرار می گیرد.

در مقابل، در روش سریال هر بیت در هر لحظه فرستاده می شود. بنابراین پروتکل ارتباطی، باید بتواند برای مقصد، ابتدا و انتها را مشخص کند. علاوه بر این، سرعت انتقال نیز با واحد bit/s[6] معرفی می شود.



شکل ۱- ۲

[3] - کد استاندارد آمریکایی
[4] - کد استاندارد آمریکایی گسترش یافته
[5] - داده
[6] - بیت بر ثانیه

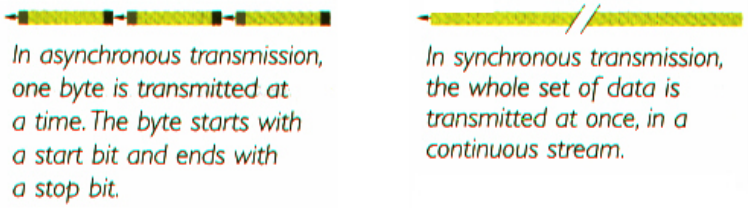
ما دو روش برای انتقال سریال داریم

۱- *انتقال غیر همزمان (Asynchronous)*

۲- *انتقال همزمان (synchronous)*

در انتقال غیر همزمان، فرستنده، کاراکترها را در یک لحظه با بیت شروع و توقف می فرستد. و گیرنده هر بیت شروع را که دریافت می کند، بقیه بیتها را به عنوان کاراکتر تفسیر می کند. و بیت توقف گیرنده را ریست می کند. در حدود ۹۰ تا ۹۵ درصد از انتقال نوع سریال داده بصورت غیر همزمان است.

در انتقال همزمان همه بیت های یک بایت در یک لحظه فرستاده می شود. سرعت انتقال توسط خط کلاک[۷] بر روی یک سیم جداگانه یا بصورت مدوله شده بر روی سیگنال دیتا، تعیین می شود. عیب روش غیر همزمان در مقابل روش همزمان این است که حدود ۲۰ الی ۲۵ درصد پیغام شامل بیتهای پریتی می باشد.



شکل ۱- ۳

۱-۲-فرستنده و گیرنده

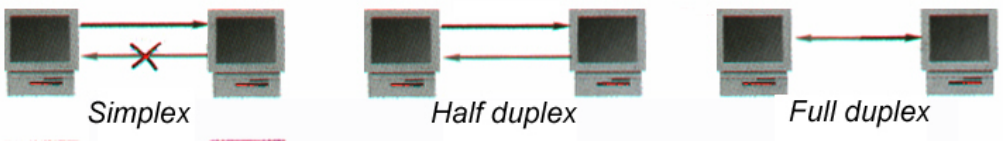
در مبحث تبادل دیتا، سخت افزارهایی با نام فرستنده و گیرنده وجود دارد. مانند PC و ربات که می توانند هم به عنوان گیرنده و هم به صورت فرستنده در یک زمان عمل کنند.

این انتقال به سه روش می تواند انجام شود

۱- simplex : انتقال دیتا تنها یک طرفه است و از جانب فرستنده به گیرنده، روی یک line می باشد.

۲- Half duplex : انتقال دیتا، به صورت دو طرفه می باشد ولی نه بصورت همزمان بلکه روی دو line جداگانه انجام می پذیرد.

۳- Full duplex : انتقال دیتا، به صورت دو طرفه، همزمان روی یک line انجام می پذیرد.(مانند انتقال دیتا در مکالمات تلفنی)



شکل ۱- ٤ : نمایی از تبادل دیتا به روش های مختلف

---

[7] - clock

۱-۳- اتصال صحیح

DTE(data terminal equipment) و DCE(data communication equipment) از جمله اصطلاحاتی است که در تبادل دیتا وجود دارد. کامپیوترها و ترمینالها معمولاً DTE هستند، مودم و سخت افزارهای ارتباطی معمولاً DCE هستند در حالی که تجهیزات دیگری نظیر مولتی پلکسرها و پرینترها می توانند هم DTE و هم DCE باشند. در DTE پینهای استفاده شده برای انتقال و دریافت دیتا متفاوت با پینهای کانکتور DCE می باشند. بدین ترتیب می توان DTE را مستقیماً به DCE متصل کرد. در صورتی که دو DCE را به هم متصل کنیم مجبوریم که فرمت اتصال را تغییر دهیم تا خط TD(Transmit Data) بر خط RD(receive data) منطبق شود.

شکل ۱- ۵

۱-٤- تبادل شفاف (transparent communication)

در سیستمهای کامپیوتری که بوسیله تعدادی مودم با هم شبکه شده اند از ارتباط شفاف استفاده می کند. شفافیت به معنای این است که همه واحدها همه پیغامها را می شنوند.

۱-۵- ساختار Master-Slave[۸]

بخش گسترده ای از شبکه های صنعتی از این ساختار استفاده می کنند بدین صورت که چندین Master پیغام ها را بطور متناوب به Slave هایی که پاسخ می دهند می فرستد. این توالی را [۹]پولینگ می نامند. در این سیستم هر Slave آدرس مخصوص به خود را دارد.

Master فرمان خود را به همراه آدرس Slave مورد نظر می فرستد. Slave مورد نظر پس از تشخیص آدرس، فرمان را انجام داده و در بعضی مواقع سیگنال تاییدی برای master می فرستد تا به کار خود ادامه دهد.

---

[8] - ارباب و برده

[9] - **polling**

ساختار و شکل آدرس و پیغام بستگی به نوع پروتکل ارتباطی که استفاده می شود، دارد. پیغامی که برای همه slave ها فرستاده می شود پیغام broadcast[10] نامیده می شود. این می تواند پیغامی باشد که توسط master به تمامی slave ها دستور داده می شود که آن وظیفه را انجام می دهند. به عنوان مثال می توان plc[11] های کنترل کننده آژیر را نام برد. درهنگام خطر همه آژیرها باید به صدا درآیند بنابراین یک پیغام broadcast باید فرستاده شود.

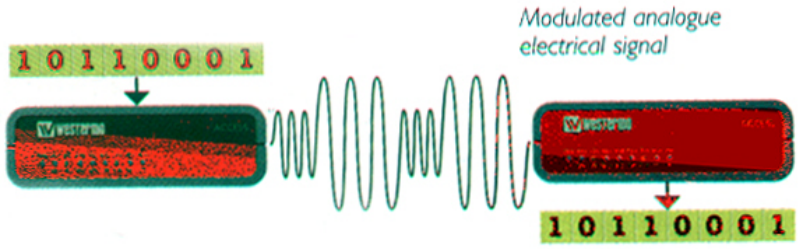
سرعت انتقال

همواره بهینه ترین سرعت، بیشترین سرعت نیست بلکه باید خطای انتقال و ارتباطات را نیز در نظر گرفت. نوع کابل و فاصله سرعت بهینه را تعیین می کند. در این صورت ما به امنیت بالا و قابل اطمینان در انتقال دیتا دست می یابیم. برای انتقال دیتای دیجیتال به وسیله سیم های مسی باید در ابتدا تغییر شکل پیدا کند.

کابل ارتباطی سبب تضعیف و متغیرشدن سیگنال می شوند که در سرعتهای بالا این اثرها می تواند بحرانی باشند.

دو اصطلاح که در این مبحث وجود دارد bit/s و سرعت انتقال می باشند. سرعت انتقال با bit/s اندازه گیری می شود. بطور تقریبی برای انتقال هر کاراکتر ۱۰ بیت نیاز است بنابراین می تواند با سرعت 9600 bit/s تقریباً ۹۶۰ کاراکتر را در ثانیه انتقال داد.

برای تغییر شکل سیگنال پیش از فرستادن به شبکه از مودم استفاده می شود. مودم، سیگنال و سرعت انتقال را تغییر می دهد. سرعت انتقال تعیین می کند سیگنال در هر ثانیه چند بار تغییر شکل پیدا می کند (مدوله می شود). هر تغییر شکل در سیگنال در واقع ایجاد بسته ای است که در طول خط به سوی مودم گیرنده فرستاده می شود و در آنجا کدگشایی شده و دوبار اطلاعات به دیجیتال تبدیل می شود.



شکل ۱-۶

در مودمهای short haul (برای مسیرهای کوتاه) سیگنال تغییر شکل پیدا نمی کند و همان چیزی که فرستاده می شود در مودم گیرنده دریافت می شود و به صورت Transparent[12] ارتباط برقرار می کنند.

[10] - انتشار
[11] - program logic control
[12] - شفاف

مودمهای PTT مانند مودمهای short-haul عمل می کنند با این تفاوت که بافری دارند که دیتا را قبل از فرستادن ذخیره می کند. با توجه به baud rate میزان سرعت انتقال دیتا مشخص می شود به عنوان مثال اگر مودم بتواند با 1400 baud کار کند و در هر انتقال ٤ بیت داشته باشیم باید سرعت انتقال 9600 bit/s باشد.

۱-۷- Handshaking[13]

Handshaking روشی برای تجهیزات ارتباط دیتا است تا بتوانند جریان دیتا بین دستگاه هایی که به شبکه متصل هستندرا کنترل کند. به خصوص در مواردی که یکی از دستگاه ها نسبت به بقیه کندتر باشند.

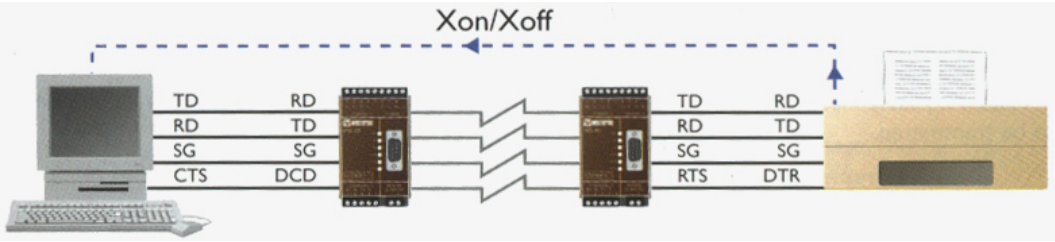
دونوع Handshaking وجوددارد: نرم افزاری و سخت افزاری.

زمانی را در نظر بگیرید که بین کامپیوتر و پرینتر ارتباط بر قرار کرده اید. حال اگر بخواهید اطلاعات را با سرعت بیشتری از آنچه پرینتر چاپ میکند، بفرستید. دستگاه پرینتر این قابلیت را دارد که اطلاعات اضافی را در یک بافر ذخیره کند. در اینجا هنگامی که بافر پر می شود با Handshaking نرم افزاری یا سخت افزاری، کامپیوتر را از این مساله آگاه کرد.

مثال دیگر در استفاده از مودم می باشد. سرعت دیتا بین کامپیوتر و مودم معمولا بیشتر از ان است که خطوط تلفن پشتیبانی می کند بنابراین مودم باید از این روش استفاده کند تا به کامپیوتر اطلاع دهد با سرعت کمتری دیتا را انتقال دهد.

۱-۷-۱- Handshaking نرم افزاری

درمثال پرینتر با این روش وقتی بافر پر می شود کاراکتری را برای کامپیوتر ارسال میکند(Xoff). وقتی که بافر خالی شد کاراکتری برای کامپیوتر ارسال می شود(Xon) تا انتقال دیتا ادامه پیدا کند.کاراکترهای معمولی که در این پروتکل استفاده می شوند شماره ۱۷ اسکی(Xon) وشماره ۱۹ اسکی (Xoff) می باشد.



شکل ۱- ۷

[13] - دست دهی

۱-۷-۲- Handshaking سخت افزاری

به جای استفاده از کارکترهای اضافی در جریان دیتا، پروتکل RS-232[14] خطوط سخت افزاری اضافی بدین منظور در نظر گرفته است. رایج ترین خطوط استفاده شده RTS(Requet to send) و CTS(Clear to Send) می باشند. به عنوان نمونه وقتی که کامپیوتر می خواهد با یک مودم ارتباط بر قرار کند:

۱- اگر کامپیوتر بخواهددیتا را انتقال دهد خط RTS را از ۳+ به ۱۵+ افزایش می دهد.(دیتا انتقال نیافته است)

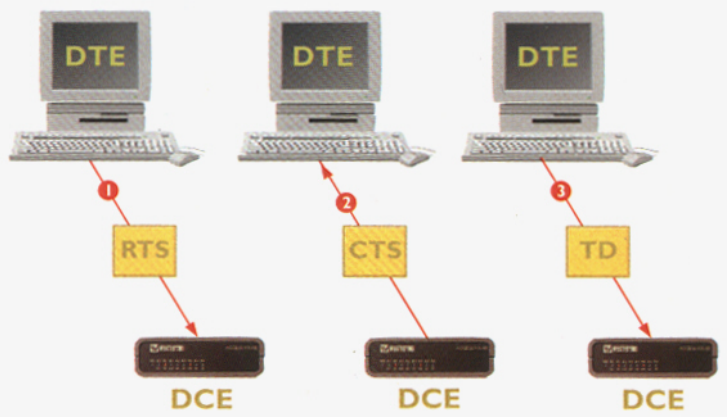
۲- مودم تغییردر خط RTS را تشخیص داده وهنگامی که آماده دریافت دیتا است خط CTS را تغییر می دهد.

۳- کامپیوترمنتظر میماندکه اگر خط CTS به سطح بالا تغییرکرد دیتا را انتقال دهد.

درهر نقطه ای که خط CTS افت کند کامپیوتر انتقال دیتا را متوقف می کند.

خط سخت افزاری که غالبا برای پرینترهای سریال فوری استفاده می شودخط DTR است که به کامپیوتر اطلاع می دهد انتقال دیتا را به دلیل نبودن کاغذ متوقف کند.

سیگنالهای سخت افزاری تنها برای روش Handshaking استفاده نمی شوند بلکه کاربردهای دیگری نیز می توانند داشته باشند. هنگامی که دوقطعه از یک دستگاه به طور ویژه ای ترکیب شده و سوئیچ خط انجام می پذیرد برای اطمینان از اینکه هر قطعه سیگنال صحیح را درزمان صحیح دریافت کرده استفاده از این روش کارگشا می باشد.



شکل ۱- ۸

[14] - استاندارد انتقال دیتا ی سریال

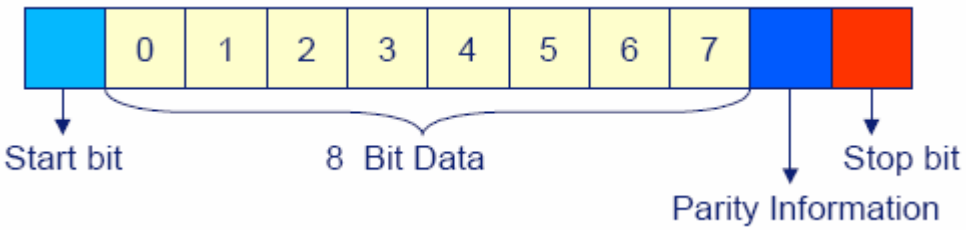
فصل دوم

معرفی مختصر پروتکل های دیگر

۲-۱- ارتباط USART

بخش ارتباط سریال USART در میکروهای AVR، قابلیت های متنوعی دارد که از جمله آنها می توان به موارد زیر اشاره کرد.

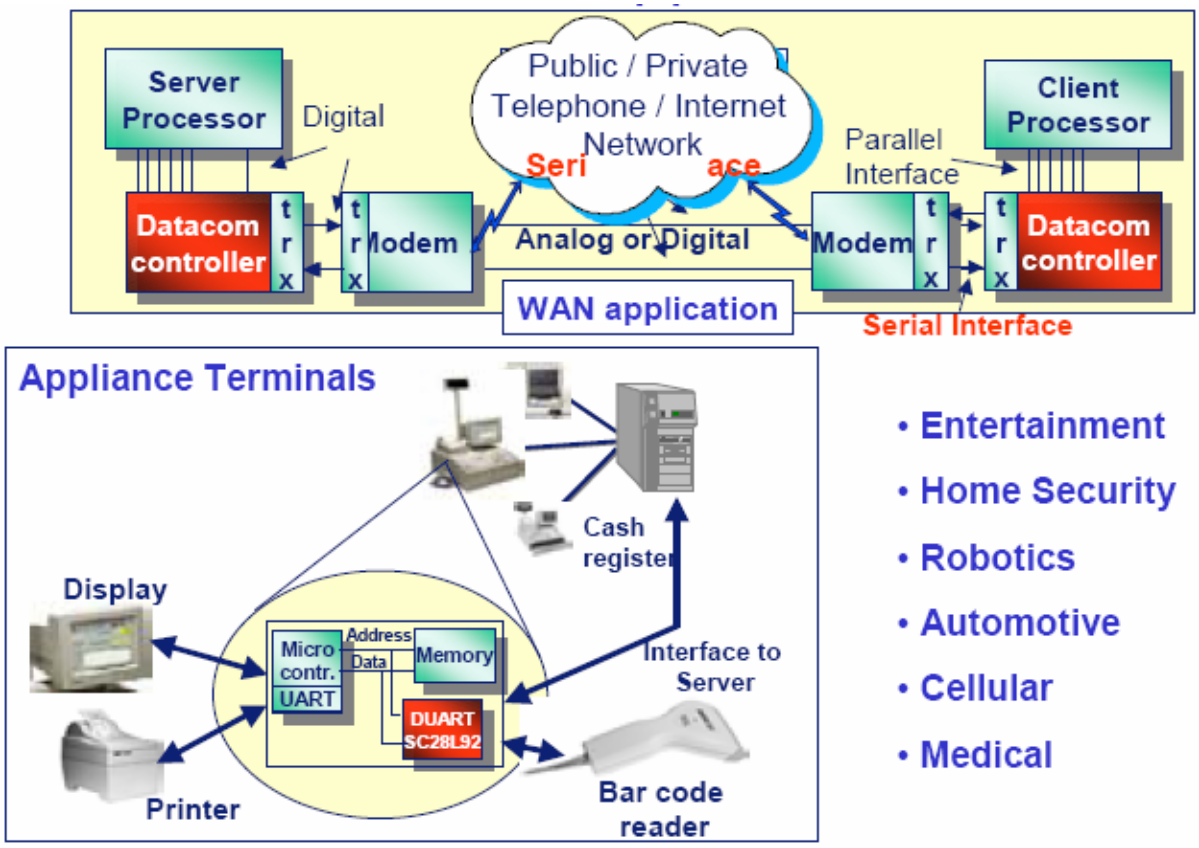
- عملکرد Full Duplex[15] (رجیسترهای سریال مستقل برای دریافت و ارسال).
- عملکرد سنکرون و آسنکرون.
- عمل به صورت Master و Slave در حالت سنکرون.
- تولید کننده نرخ ارسال (Baudrate) دقیق.
- حمایت از فریم های سریال با ۵ ، ۶ ، ۷ ، ۸ یا ۹ بیت داده و یک یا دو بیت توقف.
- تولید Parity[16] به صورت زوج یا فرد و امکان چک کردن سخت افزاری آن.
- تشخیص خطاهای سرریز و نوع فریم .
- فیلتر پایین گذر دیجیتال.
- تولید سه وقفه مجزا برای اتمام TX ، خالی شدن رجیستر داده TX و اتمام RX .
- کار در حالت ارتباط چند پردازنده.
- امکان دو برابر کردن سرعت در حالت آسنکرون.



شکل ۱- ۱۱ :فرمت ارسال دیتا در usart

همانطوری که از شکل مشخص است، در این پروتکل هر فریم با بیت شروع، آغاز می شود و بعد از آن بیت های داده، از کم ارزش ترین بیت ارسال می شوند. بعد از بیت های داده بیت Parity و بعد از آن بیت توقف فرستاده می شود.

[15] - انتقال دو طرفه همزمان
[16] - بیت توازن



شکل ۲- ۱ : ابزارهای متصل به usart و کاربردهای آن

در شکل ۲-۱ کاربردهای این پروتکل نشان داده شده است. همانطوری که در شکل نشان داده شده است، از این پروتکل می توان در انتقال دیتا از میکرو کنترلر به کامپیوتر یا پرینتر یا دستگاه تشخیص دهنده کد های میله ای (bar code reader) و برعکس یا دستگاه های پزشکی یا در روباتیک برای انتقال دیتا به پردازنده روبات و یا در انتقال دیتا بین مودمها از طریق خط تلفن و .... استفاده می شود.

۲-۲- پروتکل SPI

ارتباط سریال SPI (Serial Peripheral Interface) ، یک پروتکل ارتباط سریال سنکرون با سرعت بالا است که می تواند برای ارتباط بین میکروکنترلرهای AVR مختلف و وسایل جانبی که در آنها تعریف شده باشند به کار رود. خصوصیات این ارتباط به صورت زیر است:

- Full – Duplex ، ارسال داده به صورت سنکرون توسط سه سیم.
- عملکرد در حالت های Master و Slave .
- ارسال ابتدای بیت LSB یا MSB .

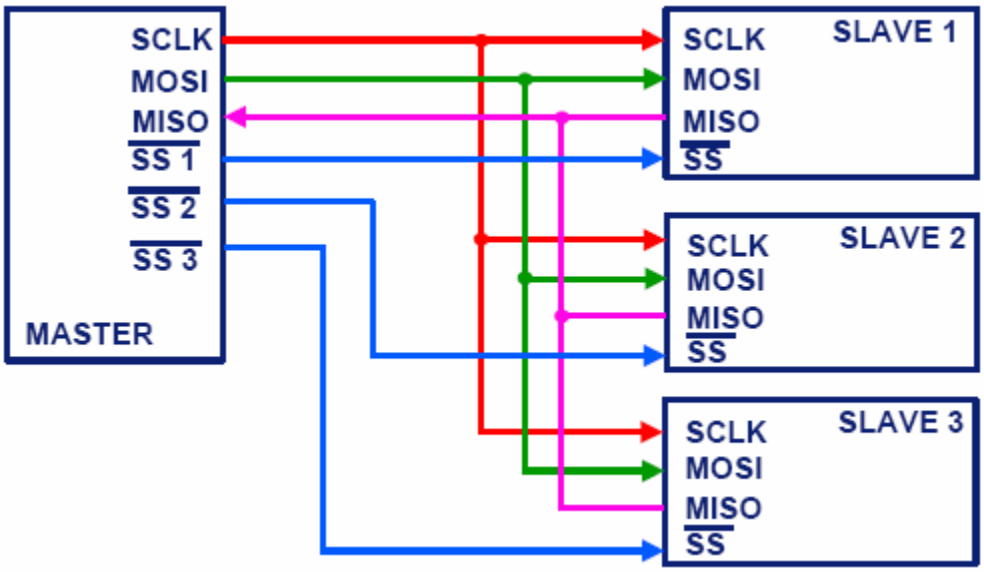
- هفت سرعت قابل برنامه ریزی.
- ایجاد وقفه در پایان ارسال.
- بیدار شدن در مد Idle.
- امکان دو برابر کردن سرعت ارسال.

در این پروتکل در هر زمان فقط یک Master فعال می باشد. و سرعت انتقال دیتا به سرعت Clock سیستم بستگی دارد.همچنین تعداد پین های میکروکنترولر که در این پروتکل استفاده می شود 3+n می باشد، که n تعداد Slave ها می باشد.

به طور کلی کار چهار پین اصلی به قرار زیر می باشد.

SCK مربوط به کلاک بوده و توسط Master کنترل می شود. MOSI[17] (Master Out Slave In) یعنی اینکه دیتا از Master خارج شده و به slave وارد شود. MISO[18] (Master In Slave Out) یعنی اینکه دیتا از Slave خارج شده و به Master وارد گردد.

زمانی که Master پایه SS[19] (Slave Select) مربوط به Slave را زمین کند، سیکل ارتباطی آغاز می گردد. در این صورت Master ، پالس کلاک لازم برای انتقال اطلاعات بین Master و Slave بر روی پایه SCK تولید می کند. داده ها همواره به کمک پایه های MOSI از Master به Slave و به کمک پایه MISO از Slave به Master شیفت پیدا می کند.در پایان هر بسته از داده ها نیز Master ، Slave را با یک کردن پایه SS با خود سنکرون می کند. SPI در حالت ارسال یک بافر و در حالت دریافت دو بافر دارد.

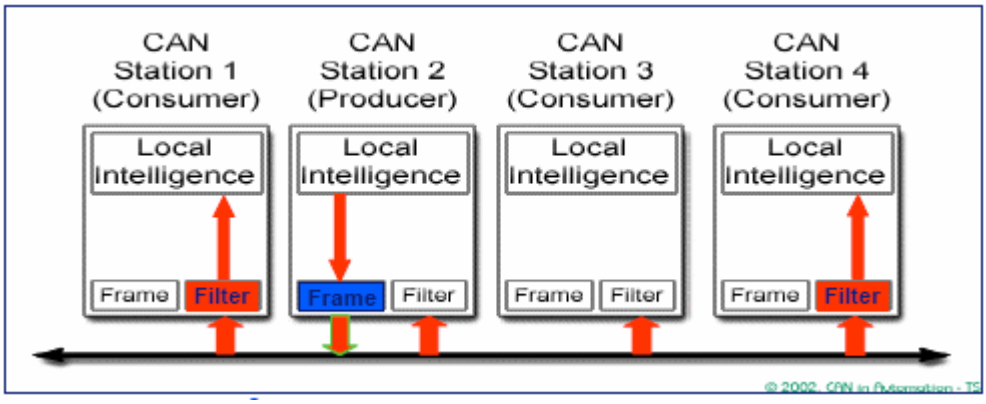


شکل ۲-۲: نمایی از طرح spi

[17] - از ارباب خارج و وارد برده شود.
[18] - از برده خارج و وارد ارباب شود.
[19] - انتخاب برده

۲-۳-۱- پروتکل CAN

- کد گزاری نسبتا پیچیده اطلاعات.
- مخابره صحیح و با زمان معین.
- تمام ماژول ها در هر ارتباطی دخالت دارند.



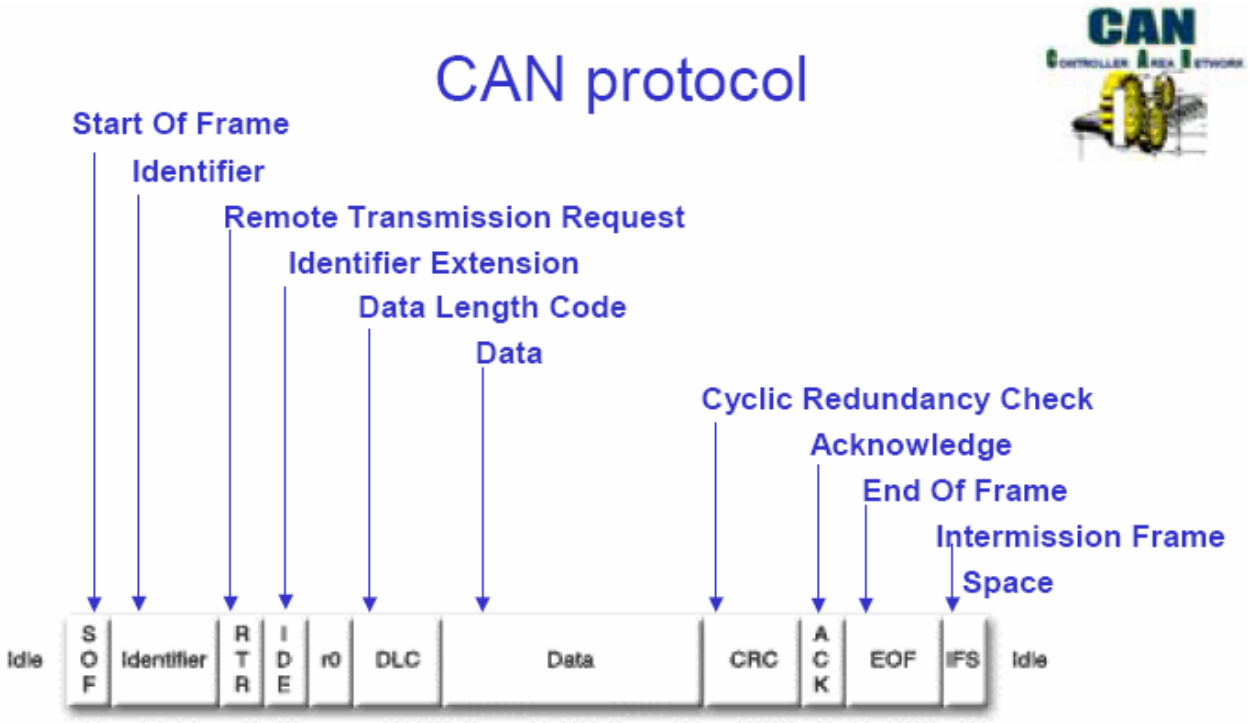
شکل ۲- ۳

هدف CAN دستیابی به اطلاعات معتبر در کاربرد سیستم های کنترل نسبتا بحرانی است. این پروتکل در محیط های نویزی استفاده می شود. وقتی اطلاعات مهمی نیاز است تا فرستاده شود، امکان تداخل نویز در باس به صورت تداخل الکتریکی و مکانیکی در سیم ها وجود دارد بنابراین چندین جنبه قابل دسترس معتبر برای باس وجود خواهد داشت.

برای تولید زمان بندی دقیق، نیاز به مقاومت سرامیکی و کریستال کوارتز خواهد بود. در اینجا کلاک دیتا با هم ترکیب شده و ٦ بیت hi به صورت پی در پی شبیه یک خطای باس تفسیر می شود. بنابراین کلاک و بیت زمان بندی مهم می باشند. تمام ماژول هایی که متصل شده اند باید در همان زمان بندی استفاده شوند. تمام ماژول ها هر نوع خطایی را در دیتا در هر نقطه از سیم بندی جستجو و خطا را گزارش می کنند، بنابراین دیتا می تواند چند بار فرستاده شود.

سیم های باس CAN باید توسط مقاومت $Pull-up$[20] شوند.

[20] - بالا کشنده



شکل ۲- ٤ : فرمت انتقال دیتا در can

۲-۳-۲- مزایای باس CAN

- ✓ برای انتخاب سخت افزار مناسب آزاد است.
  - باس تفاضلی یا یک سیمه.
- ✓ ارتباط منابع، سطح بالایی از شناخت خطا را ممکن می سازد.
  - گزارش/ قطع کردن.
  - ابزارهای خراب می توانند خودشان را از مجموعه جدا کنند.
  - زمان دیر کرد کم.
- ✓ درجه بالایی از ایمنی EMC (با استفاده از تکنولوژی ایزوله کننده Si – on).

۲-٤-۱- پروتکل USB

استانداردهای کامپیوترهای جدید بر مبنای کاهش اسلات ها وکاهش استفاده از پورت های سری و موازی قرار گرفته است. به همین جهت پورت های جدیدی به نام USB[21] (Universal Serial Bus) ساخته شده است. از مزایای این پورت عبارت اند از:

- اهداف استاندارد های جدید را کاملا در بر می گیرد.

[21] - باس سریال دو طرفه

- دستگاه های خارجی را وقتی که کامپیوتر یا دستگاه روشن هستند می توان به کامپیوتر وصل یا از آن قطع کرد.
- امکان استفاده از 127 دستگاه جانبی.
- عدم نیاز به تنظیم وقفه ها یا جامپرها، برای دستگاه های جانبی.
- سرعت این پورت 1.5 Mbit/s یا 12 Mbit/s است که سرعتهای بسیار بالایی است.
- حداکثر طول سیم برای 12 Mbit/s ، 3 متر است البته در حداکثر ارسال 1.5 Mbit/s

این پورت از جریان اطلاعات ترتیبی استفاده می کند. کامپیوتر مانند یک مدیر عامل عمل می کند و از دستگاه های متصل شده به کامپیوتر نمونه برداری می کند. این کار در زمان های منظم و تنظیم شده انجام می شود. هر پاسخ محیطی که دستگاه به کامپیوتر بدهد و اطلاعات را بخواهد ارسال کند، کامپیوتر آن را در کمتر از 1ms دریافت و شناسایی می کند. به همین دلیل است که دستگاه هایی که به USB متصل می شوند، بلافاصله شناسایی می شوند.کامپیوتر این کار را برای ماکزیمم 127 دستگاه می تواند انجام دهد.

در این پورت اطلاعات در بسته هایی ارسال و دریافت می شوند. این بسته ها Transaction[22] نامیده می شوند.

۲-٤-۲- مزایای باس USB

- شناسایی اتوماتیک.
- نیازی به به باز کردن دستگاه وجود ندارد.
- 127 وسیله می توانند با هم ارتباط داشته باشند.
- این پروتکل خیلی سریع است.
- واسطی برای کانالهای ارتباطی دیگر وجود دارد.
  - USB به پورت سریال.

۲-۵- پروتکل IEEE 1394

- یک ابزار استاندارد باس تا دیتا را از میان DVD و MPEG-2 جابجا کند.
- ویدئو نیاز به سرعت انتقال ثابت با پهنای باند ضمانت شده دارد.
- نرخ انتقال دیتا 400 , 200 , 100 مگا بیت بر ثانیه است.

[22] - انتقال

- به صورت اتوماتیک خودش را مثل هر ابزار اضافه شده بازیابی می کند.
- تعداد 63 وسیله و همچنین 4.5 متر کابل 'hops' با ماکزیمم 'hops' 16 را پشتیبانی می کند.
- پهنای باند تضمین شده.

1394 مانند باس هایی که به دستگاه هایی مثل DVD و تلویزیون های دیجیتالی متصل شده اند، توسط شرکت فیلیپس ساخته شده است. ورژن 'a' برای 1394 سرعتی معادل 400 Mb/s دارد و اخیرا ورژن 'b' برای سرعت های بالاتر و طول کابل بیشتر ساخته شده اند.

کلاک به راحتی از دیتا و سیگنال های خط strobe[23] بازیابی می شوند. همچنین در این پروتکل نیازی به هیچ گونه PLL[24] نیست.سیم های تغذیه می توانند تا 30 – 8 ولت و 1 آمپر (تقریبا) به بعضی از ابزارهای متصل شده برقرار کنند.

نرم افزار 1394 یا همان فرمت فرستادن دیتا شامل نوعی زمان بندی است که اطلاعات در بلوک یا کانال فرستاده شود. برای انتقال دیتا به فرم Real – Time[25] برای ضمانت قابل دسترس از یک کانال یا کانال های بیشتر، برای تضمین نرخ دیتای ویژه امکان آن وجود خواهد داشت. برای ویدئوها آن مهم است زیرا آن خوب نیست که یک بسته از دیتای تصحیح شده بعد از یک فضای خالی فرستاده شود.

1394 IEEE متمم USB می باشد، به طور خواص برای متصل کردن ابزار های چند کاره دیجیتالی و ابزارهای ذخیره سازی با سرعت بالا برای کامپیوتر، بهینه سازی شده است. با اتصال برای ذخیره سازی، اسکن کردن، چاپ کردن و .... ، 1394 IEEE به کاربران خود امکان اتصال Plag&Play[26] را می دهد.

---

[23] - سیگنال تائید
[24] - phase lack loop
[25] - زمان حقیقی
[26] - شناسایی اتوماتیک

فصل سوم

پروتکل I^2C

۳-۱- تاریخچه I^2C

پروتکل I^2C در اوایل دهه ۱۹۸۰ توسط شرکت Philips ابداع گردید که هدف ابتدایی آن فراهم کردن راهی ساده جهت ارتباط یک CPU با تراشه های جانبی در یک دستگاه تلویزیون بود زیرا باسهای سابق و موجود دارای تعداد خطوط زیاد بود که سبب ازدحام در PCB مربوطه می گردید.

I^2C طبق تعریف شرکت فیلیپس مخفف Inter-IC می باشد که بیانگر هدف آن یعنی فراهم آوردن یک لینک ارتباطی بین مدارات مجتمع می باشد.

امروزه این پروتکل به صورت عمومی در صنعت پذیرفته شده است و کاربرد آن از سطح تجهیزات صوتی و تصویری نیز فراتر رفته است. به گونه ای که امروزه در بیش از ۱۰۰۰نوع IC مختلف به کار گرفته شده است .

۳-۲- مزایای باس برای طراح

پروتکل I^2C سبب سهولت و سرعت در طراحی مدارات می گردد زیرا :

۱- بلوک دیاگرام عملیاتی کاملا با IC های واقعی مطابقت دارد و طراح به سرعت می تواند به شماتیک نهایی دست پیدا کند.

۲- نیاز به طراحی رابط(interface) اضافی ندارد زیرا به صورت on-chip[27] وجود دارد و محاسبات مربوط به تطبیق امپدانس و ... حذف می گردد.

۳- هم از لحاظ نرم افزاری و هم سخت افزاری قابل کنترل می باشد.

٤- ICهای مربوطه به راحتی قابل اضافه کردن و یا کم کردن می باشند.

۵- زمان طراحی نرم افزاری به دلیل وجود کتابخانه های آماده کاهش می یابد.

همچنین با توجه به تکنولوژی ساخت آنها وسیله های سازگار با i2c دارای ویژگیهای زیر می باشد:

۱- مصرف بی نهایت کم جریان

۲- امنیت در برابر نویز بسیار خوب

۳- محدوده ولتاژ تغذیه گسترده :

۲,۵ تا ۵ ولت یا ۲,۷ تا ۵,۵ ولت و در وسایل جدید ۲,۳ تا ۵,۵ ولت یا ۳ تا ۳,٦ ولت

[27] - بر روی بورد

٤- رنج گرمایی کاری گسترده :

از ٤٠- تا ٨٥ درجه سانتگراد و در بعضی موارد ٠ تا ٧٠ درجه و یا ٠ تا ١٢٠ درجه

مزایای باس I^2C برای تولید کنندگان :

این باس دارای مزایای زیر برای تولیدکنندگان می باشد .

- دو سیمه بودن آن سبب سادگی و کوچک شدن PCB ها شد.
- حذف decoder[28] های آدرس
- قابلیت ارائه در بسته های (Package) ریز و مناسب موجود می باشد.

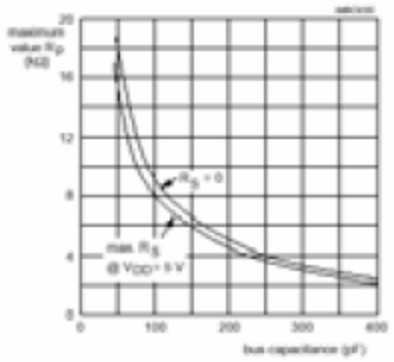
۳-۳- سخت افزار باس I^2C

این باس به طور فیزیکی شامل دو خط فعال می باشد:

- خط داده سریال (SDA) serial data line
- خط پالس ساعت سریال (SCL) serial clock line

این خطوط هر دو به صورت دو جهته عمل می کنند به همین دلیل می توان چند خدمات رسان Master داشت یا اینکه هر وسیله به عنوان فرستنده یا گیرنده عمل کند.

در ساخت این وسایل از تکنیک open-collector[29] استفاده شده است. در این صورت هرگاه چند خروجی به یک سطح متصل شوند نتیجه آن سطح AND شده این چند خروجی خواهد بود. همچنین در این باس خطوط SDA و SCL از طریق مقاومتهای pull-up و یا منابع جریان به یک منبع ولتاژ مثبت متصل هستند که نتیجه آن نگه داشتن خط در سطح HIGH است.



شکل۳- ۱

البته مقاومت pull-up که در شکل بالا نشان داده شده از طریق فرمول زیر بدست می آید.

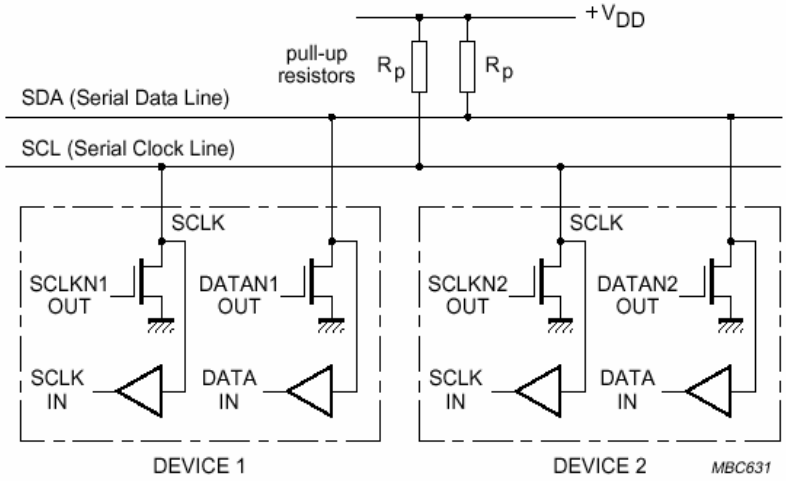
[28] - رمز گشا
[29] - کلکتور باز

$$R=\frac{Vdd(min)-0.4(v)}{0.3}$$

ماکزیمم جریان بار که خروجی ترانزیستور می تواند بدهد، که در اینجا 0.3 در نظر گرفته شده است.

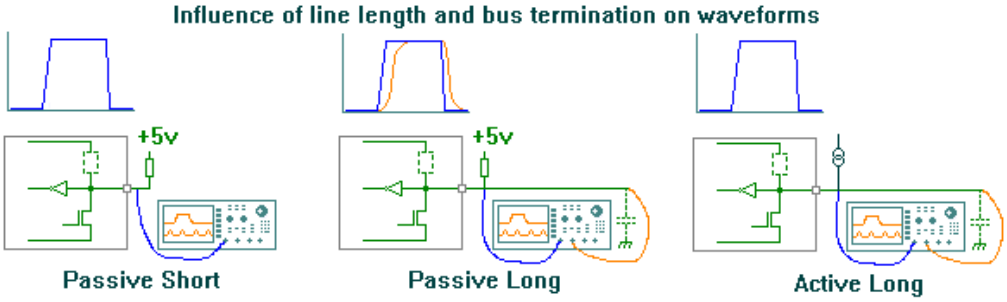
ماکزیمم میزان Rise Time[۳۰] برای Standard mode[31] برابر با 1µs و برای Fast mode[32] برابر با 0.3µs می باشد. و همچنین بار دینامیک به وسیله تعداد وسایل جانبی که به Bus متصل است تعریف می شود.

با وجود Vdd=5v آنگاه Rmin=1.3kΩ خواهد بود.



شکل ۳- ۲ : نمایی از مقاومت pull-up

هر چند تکنیک ارائه شده در مورد کلکتور باز بودن و مقاومتهای بالا کشنده دارای مزیت wired-and[33] می باشد ولی این موضوع در مورد خطوط طولانی که دارای یک ظرفیت خازنی می باشند ایجاد یک ثابت زمانی RC می گردد که برای رفع این موضوع به جای مقاومت از منابع جریان می توان استفاده کرد.



شکل ۳-۳: مقایسه مقاومت های بالا کشنده

[30] - زمان خیز
[31] - مد استاندارد
[32] - مد سرعت
[33] - and- سیمی

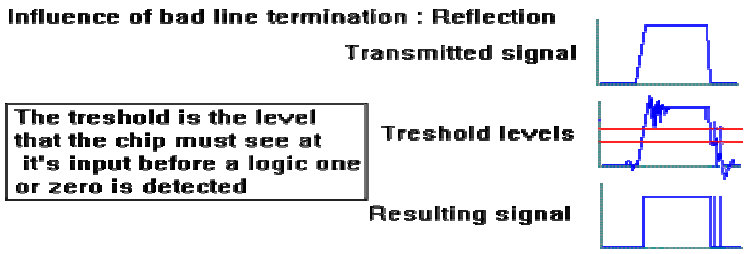
با توجه به تکنیکهای به کار گرفته شده و با توجه به امپدانس موجود در خط که باید محدود به 400pf گردد . این نوع باس در سه سرعت زیر قادر به جابجایی داده ها می باشد:

۱- حالت Standard-mode : تا حداکثر 100kbit/s

۲- حالت Fast-mode : تا حداکثر 400kbit/s

۳- حالت High Speed-mode(HS-mode)[۳۴] : تا حداکثر 3.4Mbit/s

همان طور که دیده می شود در حالاتی سرعت انتقال داده ها به قدری بالا می رود که ممکن است اثرات نا مطلوبی بر روی سیگنال ارسالی داشته باشد از جمله سوار شدن نویزهای سوزنی بر روی سیگنال که می تواند با توجه به منطق موجود ما را دچار مشکلات سازد.



شکل ۳ - ٤: حذف نویز سوزنی

برای رفع این مشکل در ابزارهایی که با سرعت بالا عمل می کنند از فیلترهای ویژه ای استفاده شده است.

همان طور که اشاره شد هر وسیله باید در این پروتکل به صورت منحصر بفرد آدرس دهی گردد که قسمتی از آدرس آن به صورت داخلی در IC موجود می باشد و قسمت دیگر آن توسط پینهای آدرس که بر روی این ICها در نظر گرفته شده تعیین می گردد.

فرضاً در ICهایی که بصورت ۷ بیتی آدرس دهی می شوند معمولاً سه پایه (A0,A1,A2) موجود می باشد که اجازه می دهد $2^3 = 8$ IC قابل شناسایی از یک مدل در این باس وجود داشته باشد.

۳-٤- مشخصات رابط سریال I2C

I2C ارتباط ساده، قوی و قابل انعطافی را با استفاده از دو باس فراهم کرده است که می تواند به عنوان فرستنده وگیرنده در دو حالت master و slave کار کند. این رابط با استفاده از فضای آدرس دهی هفت بیتی قابلیت اتصال ۱۲۸ وسیله را به یکدیگر فراهم می کند ودر هنگام وجود چندین master با روش خاصی یکی از آنها را به عنوان master نهایی انتخاب می کند. I2c

[34] - مد سرعت بالا

قابلیت آدرس دهی یک slave منحصر به فرد یا آدرس دهی تمامی slave ها در حالت General call[35] داراست. آدرس دهی یک slave منجر به wake up[36] شدن CPU از حالت sleep[37] می شود.

I2C روش مناسبی برای برقراری ارتباطات میکروکنترلرهاست. این پروتکل با استفاده از دو باس، پالس (SCL) و دیتا (SDA) امکان بر قراری ارتباط را فراهم می کند. تنها المان خارجی به کار برده شده در این روش ، دو مقاومت Pull up است که باسها را به VCC متصل می کند.

Master تجهیزی است که پس از آغاز به کار، شروع به ارسال اطلاعات کرده و پالس SCL را تولید می کند. slave نیز تجهیزی است که توسط master آدرس دهی می شود .درایور های باس در I2C به صورت درین باز یا کلکتور باز هستند.

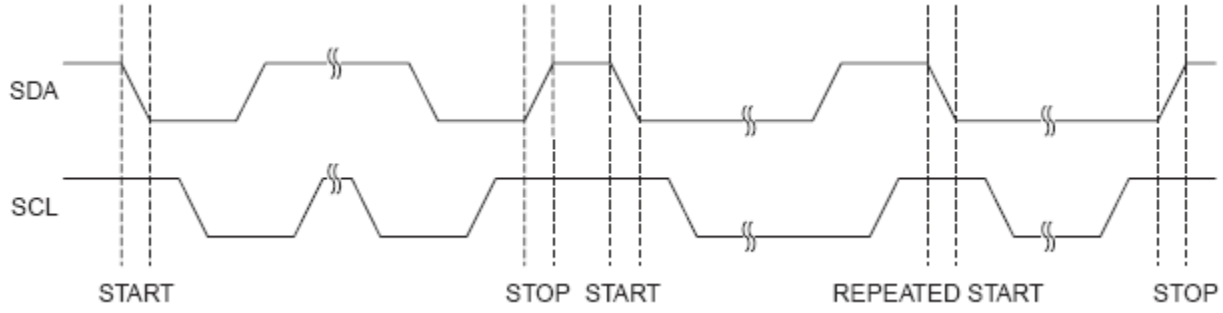
توجه شود که تمامی تجهیزات در هنگام اتصال به باس باید تغذیه شده باشند. تعداد تجهیزاتی که به باس وصل می شوند توسط محدودیت خازنی باس که حدود 400pf است و هفت بیت آدرس دهی ،محدود می شود.

۳-۵- شرایط ارسال STOP و START

Master وظیفه شروع و به پایان رساندن انتقال داده را به عهده دارد. انتقال اطلاعات زمانی آغاز می شود که Master شرط START[38] را بر روی باس قرار دهد و پایان انتقال اطلاعات با قرار دادن شرط STOP[39] اعلام می شود. بین شرایط STOP و STSRT باس به صورت Busy[40] در نظر گرفته می شود وMaster دیگری نباید سعی در اشغال باس داشته باشد. شرایط ویژه زمانی رخ می دهد که یک شرط STSRT جدید بین دو شرط STSRT و STOP ارسال شود. این شرط تحت عنوان شرط REPEATED[41] شناخته شده و زمانی است کهMaster قصد دارد یک انتقال جدید را بدون از دست دادن کنترل باس آغاز نماید. بین زمان ارسال REPEATED STSRT تا زمان ارسالSTOP بعدی، باس ممکن است در حال ارسال اطلاعات بوده و اصطلاحاً اشغال شده باشد. مطابق شکل زیر شرایط STSRT و STOP با تغییر سطح باس SDA زمانی که باSCL ، High **است**، ارسال می شوند.

---

[35] -فراخوانی عمومی
[36] - بیدار شدن
[37] - خواب
[38] -شروع
[39] - توقف
[40] - مشغول
[41] -تکرار شده



شکل ۳- ۵: وضعیت باس در هنگام انتقال دیتا و آدرس با هم

چنانچه در شکل بالا دیده می شود، در زمان High بودن SCL، تغییر سطح باس SDA از High به Low شرط STSRT و تغییر سطح باس SDA ازLow به High شرط STOP تلقی می شود. در نتیجه تغییر دیتا فقط در زمان Low بودن باس SCL صورت می گیرد.

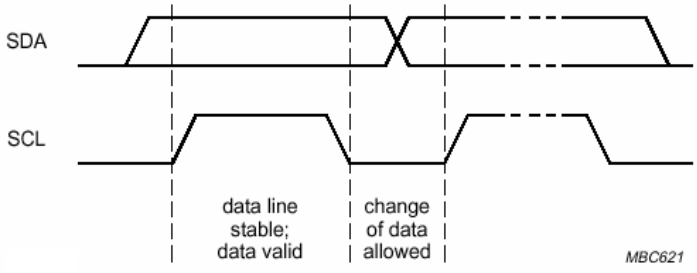
۳-٦- فرمت انتقال داده ها

هر بایت داده بر روی خط SDA باید ۸ بیت طول داشته باشد. و همچنین هیچ محدودیتی در مورد تعداد بایتهای ارسالی توسط یک فرستنده بر روی SDA وجود ندارد.

باید توجه داشت که در هنگام انتقال داده ها زمانی که clock در سطح high[42] است نباید خط SDA تغییر کند مگر در دو حالت که نشان دهنده حالت شروع و پایان می باشد.

بیت START : تغییر سطح منطقی SDA از high به low زمانی که clock در سطح high قرار دارد.

بیت STOP : تغییر سطح منطقی SDA از سطح low به high زمانی که clock در سطح high قرار دارد.



شکل ۳-٦ : فرمت انتقال دیتا

[42] -یک منطقی

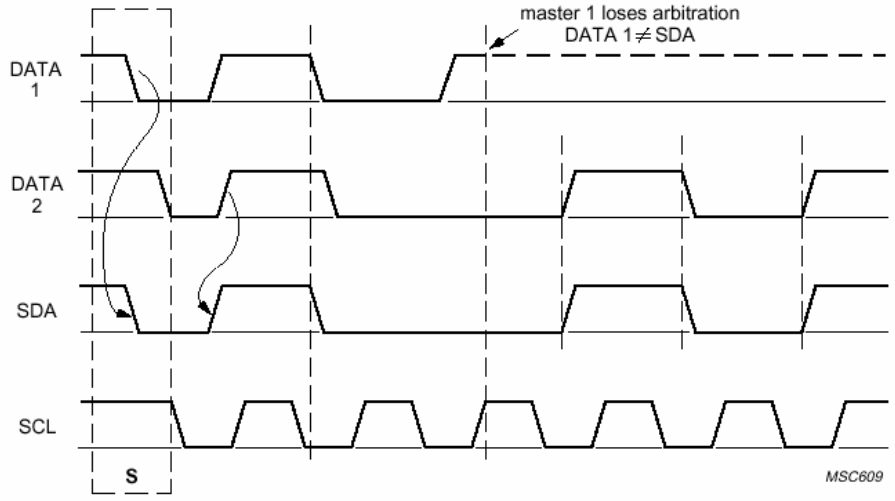
بعد از هر بایت ارسال شده جهت تعیین اینکه این بایت توسط گیرنده پذیرش شده است یا نه یک بیت دیگر در پالس بعدی ارسال می گردد که به Acknowledge bit[43] معروف است. اگر در این بیت SDA در سطح high قرار بگیرد به این معنی است که پذیرش صورت نگرفته است و اگر SDA به سطح low[44] برود به این معنی است که پذیرش صورت گرفته است.

۳-۷- مساله همزمان سازی پالس ساعت

پالس ساعت توسط Master ها تولید می گردد. هر Master پالس ساعت خود را بر روی SCL قرار می دهد و با توجه به خاصیت wired-AND در این گونه باس پالس ساعت ها با هم AND شده و باعث تولید یک پالس ساعت مشترک می گردد.

۳-۸- مساله داوری و حاکمیت یک Master

هر Master تنها در زمانی می تواند به باس دسترسی پیدا کند که خط SDA آزاد باشد. در اینجا نیز وجود خاصیت wired-AND باعث حل مشکل می گردد یعنی چند Master بطور همزمان داده هایشان را بر روی خط SDA به صورت سریال ارسال می دارند و با هم AND شده و بر روی باس یک دیتای واحد قرار می گیرد در اولین مکانی که خط SDA با خط داده مربوط به یک Master مطابقت نداشت آن Master خط داده سریال را در سطح یک منطقی رها می کند (حالت پیش فرض با توجه به وجود pull-up سطح high می باشد) تا بر روی کار دیگر Master ها تاثیر نداشته باشد.



شکل ۳-۷ : مشخص کردن حاکم

همان طور که دیده می شود مساله Arbitration[45] تنها در مورد حالتی معنی دارد که چند Master داشته باشیم زیرا

[43] - بیت تصدیق
[44] - صفر منطقی
[45] -حاکم شدن

۱- در مورد Slave ها با توجه به اینکه در هر زمان یک Slave آدرس دهی می شود و حق دسترسی به SDA را دارد معنی نخواهد داشت.

۲- یک Master دیگر رقیبی برای دسترسی به خط SDA ندارد.

همچنین باید توجه داشت که در موارد زیر Arbitration بکار گرفته نمی شود :

- بین وضعیت repeated Start (Sr) و بیت دیتا
- بین وضعیت Stop و بیت دیتا
- بین یک وضعیت repeated Start (Sr) و وضعیت Stop

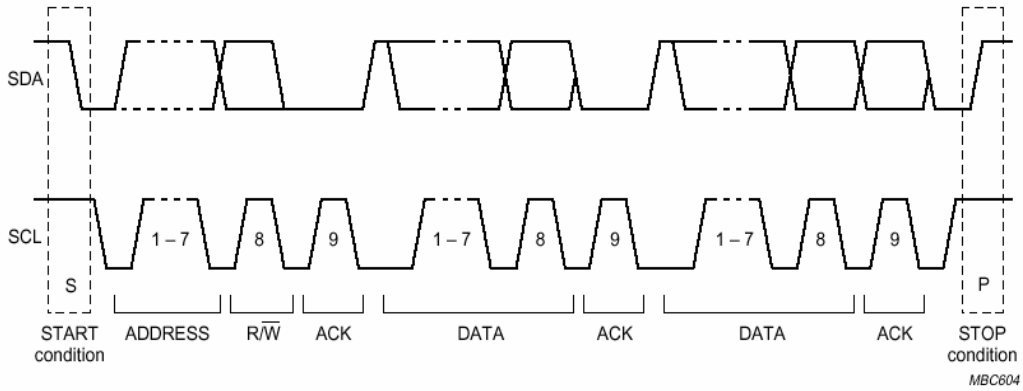
۳-۹- آدرس دهی

برای شناسایی هر وسیله در این پروتکل نیاز به یک آدرس داریم که در مورد هر وسیله تعدادی از بیتهای آن به صورت on-chip آدرس دهی می شود و مابقی از طریق پینهای آدرس سخت افزاری مشخص می گردد. در ابتدا آدرسی که به هر وسیله اختصاص می یافت یک آدرس ۷ بیتی بود ولی با گسترش این باس و تعداد وسیله های بکار رفته در آن نیاز به آدرس دهی با تعداد بیت بالاتر احساس شد و آدرس دهی به صورت ۱۰ بیتی بوجود آمد.

۳-۹-۱- آدرس دهی ۷ بیتی

در این حالت بعد از وضعیت START(S)، آدرس یک slave فرستاده می شود که ۷ بیت طول دارد. سپس در بیت ۸ جهت انتقال داده $(R/\overline{W})$ مشخص می گردد. صفر بیانگر فرستادن داده (WRITE)[46] و یک بیانگر درخواست دیتا (READ) می باشد. هر ارسال داده نیز با یک وضعیت STOP خاتمه می یابد. اگر master همچنان نیاز به دسترسی به باس داشته باشد از یک وضعیت شروع مجدد Repeated Start (Sr) استفاده می کند و سپس یک slave دیگر را آدرس دهی می کند، بدون اینکه ابتدا نیاز به وضعیت STOP داشته باشد.

[46] - نوشتن

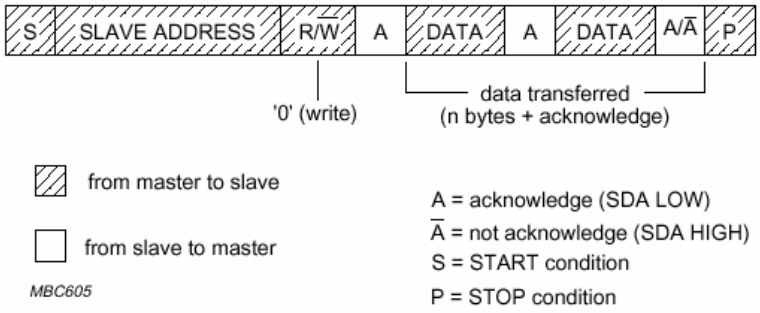


A complete data transfer.

شکل ۳-۸ : نمایی از انتقال دیتا

با توجه به این نوع آدرس دهی سه حالت زیر موجود می باشد:

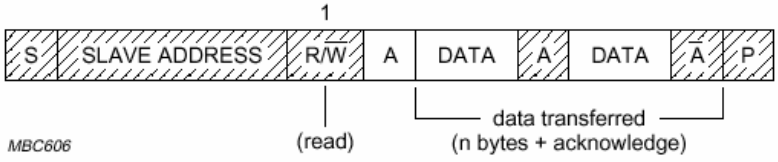
۱- یک master-transmitter داده هایش را به یک slave-receiver بدون اینکه جهت ارسال تغییر کند، منتقل می کند.



A master-transmitter addressing a slave receiver with a 7-bit address.
The transfer direction is not changed.

شکل ۳-۹

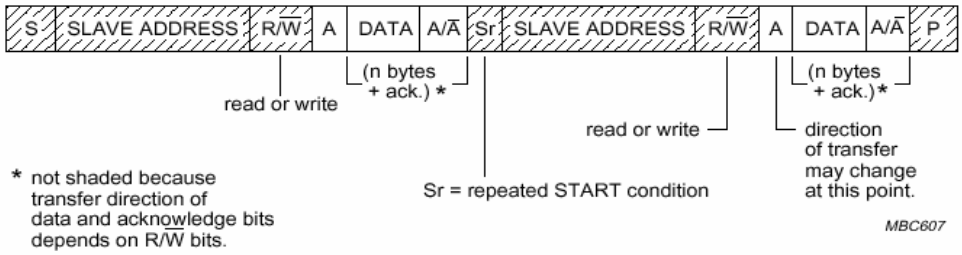
۲- یک master داده ها را بلافاصله بعد از اولین بایت از slave می خواند.



A master reads a slave immediately after the first byte.

شکل ۳-۱۰ : master دیتا را از slave می خواند

۳- حالت ترکیبی : ترکیبی از دو حالت قبل است یعنی جهت ارسال قابل تغییر است.



شکل ۳-۱۱ : ترکیبی از حالت های قبل

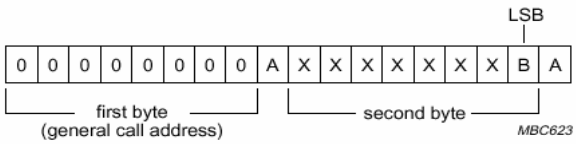
اما باید توجه داشت که در بایت اول تعدادی از آدرسها برای مقاصد خاصی ذخیره شده اند که در جدول زیر به آنها اشاره می گردد

Definition of bits in the first byte

SLAVE ADDRESS	R/W BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

شکل ۳-۱۲: ادرس های مخصوص

General call[47] : برای این است که پس از شروع کار باس و ارسال این بایت تمام وسیله ها آدرسهای مربوط به خود را تنظیم کنند.



General call address format.

شکل ۳-۱۳

[47] - فراخوانی عمومی

اگر وسیله ای نیاز به آدرس دهی داشته باشد بعد از این کد به عنوان slave-receiver[48] عمل می کند.

اما بایت دوم دو حالت دارد:

- هنگامی که بیت کم ارزش آن صفر باشد.
- هنگامی که یک باشد.

در حالتی که B صفر است چند حالت داریم :

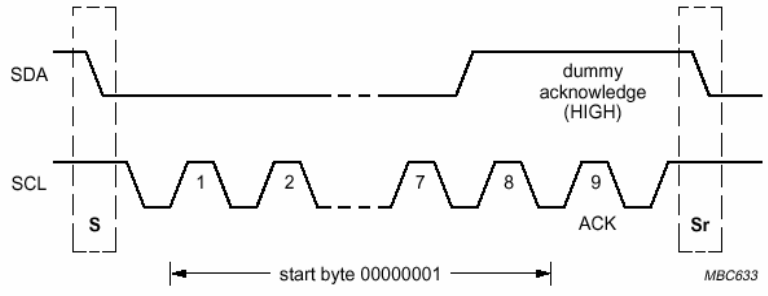
- 00000110(06H) : ریست کردن و نوشتن قسمت قابل برنامه ریزی آدرس slave بوسیله سخت افزار.
- 00000100(04H) : نوشتن قسمت قابل برنامه ریزی slave بوسیله سخت افزار. در این حالت دستگاه ریست نمی شود.
- 00000000(00H) : به عنوان بایت دوم نباید استفاده شود.

باید از باقی کدها صرف نظر کرد.

هنگامی که بیت B یک است به Hardware general call[49] معروف است و به این معناست که توالی توسط سخت افزار مانند keyboard scanner[50] که نمی تواند برای ارسال یک آدرس slave مورد نظر برنامه ریزی شوند ، ارسال می گردد.

هفت بیت دیگر شامل آدرس hardware master[51] است که بوسیله یک میکروکنترلر تشخیص داده می شود. در بعضی از سیستم ها این hardware master به صورت slave-receiver عمل می کند که توسط یک master پیکربندی کننده، آدرس slave که باید به آن داده ارسال گردد ، تعیین می شود.

Start byte : برای کاهش میزان دسترسی میکروها به این باس از این بایت استفاده می شود. ابتدا یک میکرو با سرعت کم از SDA نمونه برداری می کند تا یکی از این ۷ بیت را (که نسبت به حالت عادی زمان طولانی تری را می گیرند) تشخیص دهد. سپس سرعت نمونه برداری خود را افزایش می دهد تا بتواند داده های روی SDA را تشخیص دهد. در این حالت رویه شروع به صورت زیر است:



START byte procedure.

- وضعیت START (S)

[48] - برده دریافت کننده باشد
[49] - فراخوانی عمومی سخت افزاری
[50] - اسکن کننده صفحه کلید
[51] - سخت افزار ارباب

- بایت START (00000001)
- پالس ساعت مربوط به Acknowledge
- وضعیت شروع مجدد Repeated start (Sr)

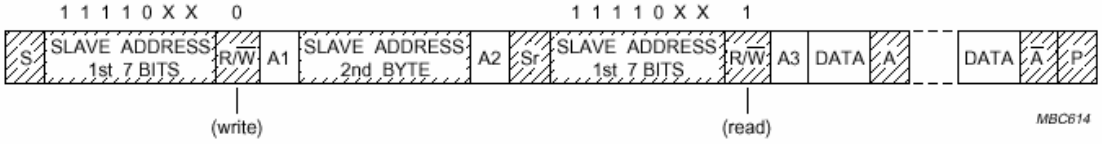
۳-۹-۲- آدرس دهی ۱۰ بیتی

همان طور که گفته شد حالت 1111XXX برای این مقصود نگه داشته شده است. اما باز هم تنها از حالت 11110XX برای این مقصود استفاده شده است و حالت 11111XX برای مقاصد آتی رزرو شده است. بیت هشتم نیز مانند حالت ۷ بیتی مشخص کننده Read/Write می باشد. بایت دوم پس از این بایت بیان گر ادامه آدرس Slave مورد نظر می باشد.

مانند حالت ۷ بیتی در این حالت نیز وضعیتهای مختلفی وجود دارد، که در شکل های زیر می توان به آنها اشاره کرد:

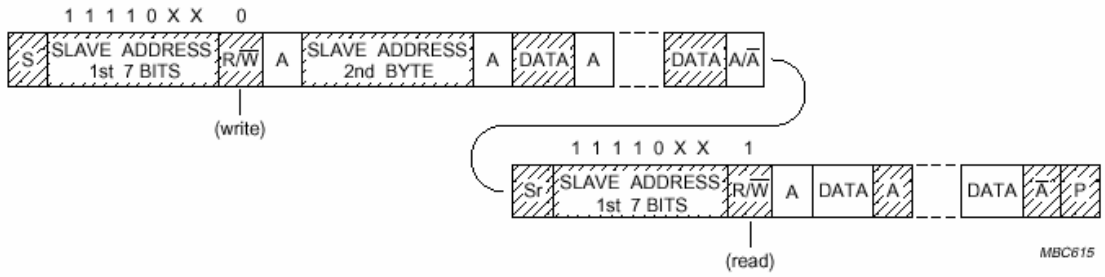
A master-transmitter addresses a slave-receiver with a 10-bit address.

شکل ۳-۱٤ : فرمت آدرس دهی۱۰ بیتی



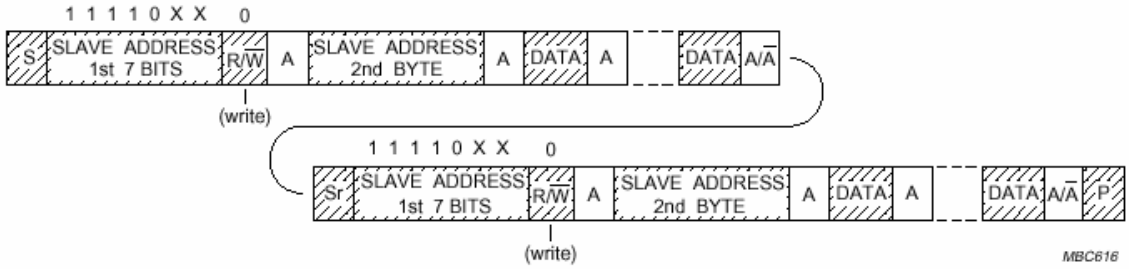
A master-receiver addresses a slave-transmitter with a 10-bit address.

شکل ۳-۱۵: فرمت انتقال دیتا از slave به master به فرم ۱۰ بیتی



Combined format. A master addresses a slave with a 10-bit address, then transmits data to this slave and reads data from this slave.

شکل ۳-۱۶



Combined format. A master transmits data to two slaves, both with 10-bit addresses.

شکل ۳-۱۷ : ترکیبی از انتقال دیتا بین master و slave

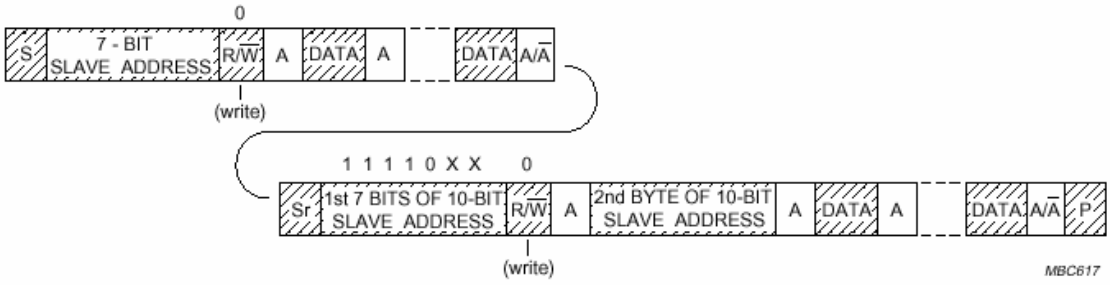


Fig.30 Combined format. A master transmits data to two slaves, one with a 7-bit address, and one with a 10-bit address.

شکل ۳-۱۸ : ترکیبی از آدرس دهی master و slave به فرم ۷ و ۱۰ بیتی

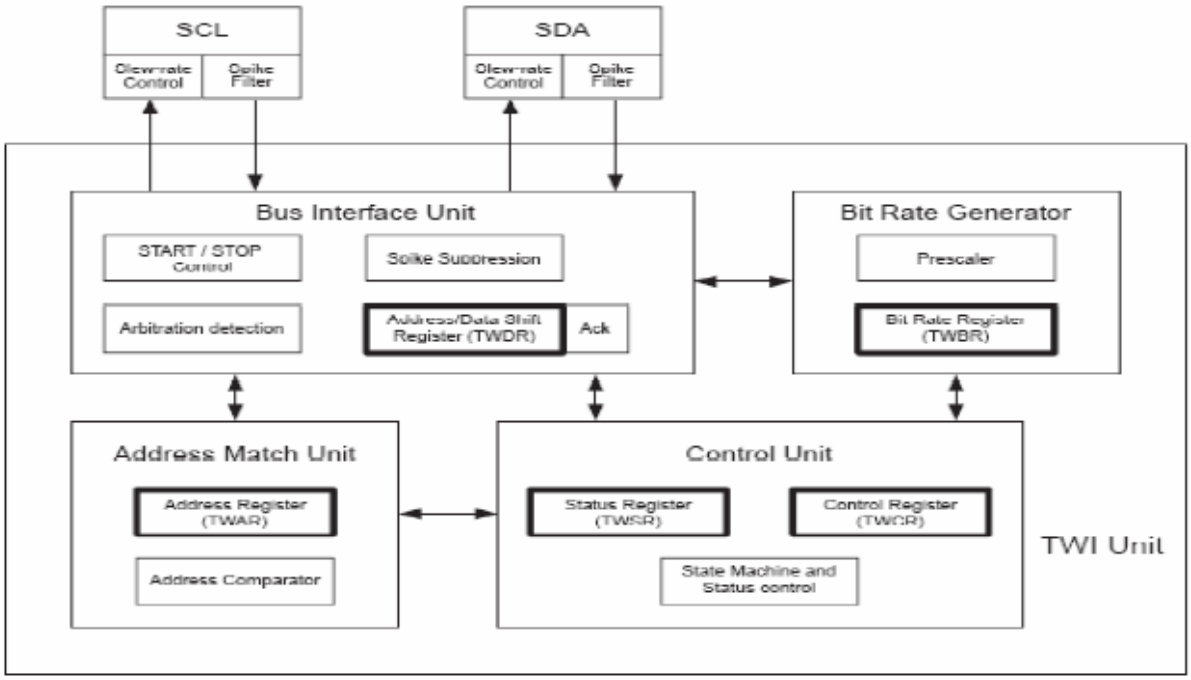
فصل چهارم

ساختار داخلی I2C

٤-١- ساختار داخلی ماژول TWI

مطابق شکل زیر، ماژول TWI از چند قسمت تشکیل شده است. رجیستر هایی که درون مستطیل پر رنگ نشان داده شده است توسط باس دیتای CPU قابل دسترسی هستند.

پایه های SCL و SDA ، رابط TWI با بقیه MCU ها می باشند. راه انداز های خروجی دارای محدود کننده Slew Rata بوده و واحد ورودی شامل حذف کننده نویز های باپریود کمتر از sn ٥٠ می باشند. مقاومت های Pull - Up برای پایه های SDA و SCL با تنظیم بیت های PORTx می توانند فعال شوند. در بعضی مواقع مقاومت های Pull - Up داخلی این پایه ها، نیاز به مقاومت های خروجی را حذف می کنند.



شکل ٤-١ : شمایی از ساختار داخلی twi

٤-١-١- واحد تنظیم Bit Rate[52]

[52] - سرعت انتقال بیت

این واحد فرکانسSCL را در مد Master کنترل می کند. فرکانس پایه SCL با تنظیم کردن بیت های رجیستر TWBR و بیت هایز در رجیستر TWBR کنترل می شود. عملکرد Slave هیچگونه وابستگی به این بیت ها ندارد اما فرکانس CPU درSlave باید ۱۶ برابر بیشتر از فرکانس SCL باشد. فرکانس SCL تولید شده توسط Master از رابطه زیر به دست می آید.

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

در رابطه بالا، TWBR مقدار رجیستر TWBR و TWPS مقدار بیت های ۰..TWPSI در رجیستر TWBR است. توجه به این نکته ضروری است که مقدار TWBR در مد Master باید ۱۰ یا بیشتر باشد.اگر در این مد، TWBR کمتر باشد، Master ممکن است خروجی های اشتباه بر روی باس های SDA و SCL ایجاد نماید. این مشکل زمانی که سیگنال SLA+R/W +START ارسال می شود شکل حادتری به خود می گیرد.

٤-١-٢- واحد[٥٣] Bus Interface

واحدBus Interface دارای شیفت رجیستر های دیتا و آدرس TWDR، کنترل کننده START/STOP و سخت افزار تشخیص دهنده Arbitration می باشد. رجیستر TWDR در حین انتقال اطلاعات ، شامل اطلاعات آدرس یا دیتا برای انتقال و در مواقع گیرنده بودن ، شامل اطلاعات آدرس یا دیتای دریافتی است. علاوه بر هشت بیت TWDR ، واحد فوق دارای رجیستری برای ذخیره بیت (ACK) N برای ارسال و دریافت نیز می باشد. این رجیستر به طور مستقیم توسط نرم افزار قابل دسترسی نیست. اما در هنگام دریافت این بیت میتواند توسط رجیسترTWDR نوشته یا پاک شود. در مد انتقال مقدار بیت (ACK) N با توجه به مقدار رجیستر TWDR مشخص میشود.

کنترل کننده START/STOP تولید کننده و تشخیص دهنده شرایط START ، REPEATED STSRT STOP

می باشد. این واحد شرط START/STOP راحتی در مد Sleep تشخیص داده و می تواند MCU را در صورتی که آدرس دهی شده باشد، به حالت Wake Up ببرد.

اگر TWI به عنوانMaster انتقال اطلاعات را آغاز نماید، سخت افزار تشخیص Arbitration به[٥٤] Monitor کردن انتقال برای تشخیص Arbitration می پردازد.

٤-١-٣- واحد[55] Address Match

[53] - رابط باس
[54] - نمایش
[55] -تطبیق آدرس

واحدAddress Match Unit ، وظیفه بررسی مطابقت داشتن آدرس دریافتی با مقدار رجیستر TWAR را به عهده دارد. اگر TWI حالت General Call را با فعال کردن بیت TWGCE در رجیستر TWAR ، فعال کرده باشد، تمام آدرس ها به جز آدرس General Call مقایسه می شود. در صورت تطابق یک آدرس عمل خواسته شده انجام می شود. TWI با توجه به تنظیمات رجیستر TWCR ممکن است یک آدرس را دریافت یا رد نماید. در حالت Sleep این واحد با بررسی کردن آدرس میتواند MCU را به حالت Wake Up ببرد.

٤-١-٤- واحد[٥٦] Control

واحد Control وظیفه Monitor کردن باس های TWI و تولید پاسخ مناسب بر اساس تنظیمات رجیستر TWCR را بر عهده دارد. وقتی رخدادی نیاز به انجام کار بر روی باس TWI را داشته باشد از پرچم TWINT استفاده می کند. در سیکل بعدی ، مقادیر جدید متناسب با حالت های جدبد باس در رجیستر TWSR نوشته می شود. این رجیستر هنگام یک بودن پرچم TWINTحالت های باس را بیان می کند و در بقیه موارد کاربردی ندارد.تا زمانی که پرچم TWINT فعال است باس SCL ، Low نگاه داشته شده و اجازه انجام وظایف مرتبط با وقفه را قبل از ارسال مجدد اطلاعات را به نرم افزار را میدهد. پرچم TWINT در شرایط زیر یک می شود

- پس از ارسال شرطSTART/ STOP
- هنگام از دست دادن SLA+R/W
- پس از ارسال هر بایت آدرس یا پس از ارسالArbitration
- پس از آدرس دهی به صورت General Call یا آدرس دهی Slave
- پس از دریافت دیتا یا پس از دریافت REPEATED STSRT وSTOP
- پس از وقوع خطای باس که در اثر ارسال شرطSTOP و START اشتباهی رخ داده است.

٤-٢- رجیستر های TWI

TWI Bit Rate Register – TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – TWI Bit Rate Register**

شکل ٤-٢ : رجیستر TWBR

[56] - کنترل

این بیت ها ضریب تقسیم فرکانس را برای مولد Bit- Rait مشخص می کنند. عدد مشخص شده در این رجیسستر برای تعیین فرکانسCLS در رابطه بالا و مد Master به کار می رود.

TWI Control Register – TWCR

این رجیستر برای کنترل عملکرد TWI به کار می رود. این رجیستر برای فعال کردن TWI ، برای شروع ازسال اطلاعات توسط Master با استفاده از شرط START ، برای دریافتACK گیرنده، ارسالSTOP وکنترل توقف پالس به هنگام نوشته شدن اطلاعات در رجیسترTWDR به کار می رود. همچنین شرایط Collision[57] را هنگام در دسترس نبودن TWDR بیان می کند.

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TWINT: TWI Interrupt Flag**

شکل ٤-٣ : رجیستر TWCR

زمانی کهTWI کار فعلی خود را به پایان می رساند و منتظر عکس العمل برنامه است، این بیت توسط سخت افزار یک می شود. اگر بیت I و بیت TWIE در رجیستر TWCR یک باشد، برنامه سرویس وقفه TWIاجرا می شود. تا زمانی کهTWINT یک باشد، SCL ، Low نگاه داشته می شود. پرچمTWINT توسط نرم افزار و با نوشتن یک در آن پاک می شود. توجه به این نکته الزامی است که این بیت پس از اجرای برنامه سرویس وقفه پاک نمی شود. باصفر کردن این بیتTWI کار خود را مجدداً آغاز می کند. بنابر این باید قبل از صفر کردن TWINT و شروع به کارTWI ، تنظیمات رجیستر های TWSR ، TWAR و TWDR انجام شده باشد.

- **Bit 6 – TWEA: TWI Enable Acknowledge Bit**

بیت TWEA تولید بیتACK را کنترل می کند. با یک شدن این بیت، ACK در زمان های خاصی نظیر، پس از دریافت آدرس توسط Slave ، پس از دریافت ادرس General Call هنگام یک بودن بیت TWCGE ، پس از دریافت بایت دیتا در مد های MR یا SR ارسال می شود. با صفر شدن بیتTWEA ، تجهیز موقتاً از TWI جدا می شود. تشخیص آدرس با نوشتن بیت TWEA مجدداً امکان پذیر است.

- **Bit 5 – TWSTA: TWI START Condition Bit**

[57] -تلاقی

این بیت توسط نرم افزار و زمانی که میکرو کنتر لر قصد Master شدن داشته باشد، یک می شود. سخت افزار TWI در صورت وجود باس آزاد، شرطSTART را ارسال می کند. در صورت آزاد نبودن باس، TWI منتظر شرایط STOP می ماند و سپس START جدیدی ارسال می کند. بیت TWSTA توسط نرم افزار و پس از انتقال START باید صفر شود.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

بایک شدن این بیت در مد Master ، شرط STOP بر روی باس ارسال می شود. پس از ارسال شرط STOP بر روی باس ، این بیت به طور خودکار پاک می شود. در مد Slave یک نمودن این بیت می تواندبرای تشخیص شرایط خطا به کار رود. در این حالت شرط STOP ایجاد نمی شود ولی باس هایSCL و SDA به حالت امپدانس بالا تغییر حالت می دهند.TWI نیز به مد Slave آدرس دهی نشده تغییر حالت می دهد.

- **Bit 3 – TWWC: TWI Write Collision Flag**

هنگامی که بیت TWINT صفر است، اگر کاربربخواهد مقداری را در رجیستر TWDR بنویسد، بیت TWWC یک می شود تا از بروز Collision جلوگیری نماید. این پرچم با نوشتن رجیسترTWDR ، هنگامی که پرچمTWINT یک است، پاک می شود.

- **Bit 2 – TWEN: TWI Enable Bit**

یک کردن این بیتTWI را فعال می کند. با یک شدن این بیت TWI کنترل پورت های SCL و SDA را در اختیار گرفته ، فیلتر، محدود کننده Slew Rate را فعال می کند. صفر شدن این بیت منجر به غیر فعال شدنTWI و متوقف شدن کلیه عملیات در حال انجام می گردد.

- **Bit 0 – TWIE: TWI Interrupt Enable**

با یک کردن بیتTWIE و بیت، وقفه مربوط بهTWI فعال شده، زمانی که بیت TWINT یک می شود، برنامه سرویس وقفه می تواند اجرا شود.

TWI Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: TWI Status**

شکل ٤-٤: رجیستر TWSR

این پنج بیت، وضعیت TWI را مشخص می کنند. کد های مختلف در ادامه فصل بررسی می شوند. مقدار خوانده شده از این رجیستر دارای پنج بیت حالت و دو بیت TWPSI..0 است. هنگام بررسی بیت های حالت، بیتهای TWPSI..0 به صفر Mask[58] می شوند، تا بررسی کردن بیت های حالت مستقل از مقدار بیت های TWPSI..0 باشد.

[58] - پوشش

مطابق جدول زیر، مقادیر مختلف بیت ها عددی را برای کنترلBit Rate یا به عبارت دیگر کنترل فرکانس باس SCL در مد Master، مشخص می کنند. این عدد در رابطه ۹-۱ مقدارTWPS را تعیین می کند.

Table 73. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWI Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

شکل٤-٥: رجیستر TWDR

در مد انتقال رجیستر TWDR شامل دیتایی است که قرار است فرستاده شود و در مد دریافت، شامل آخرین دیتایی است که دریافت شده است. در حالی که TWI در حال ارسال اطلاعات نباشد، پرچم TWINT یک بوده و مقدار دلخواه در این رجیستر نوشته می شود. به عبارت دیگر تا زمانی که پرچم TWINT یک نشود نمی توان مقداری در رجیسترTWDR نوشت یا مقدار آن را خواند. مقدار رجیستر TWDR تا زمانی که بیت TWINT یک باشد ثابت می ماند. وقتی دیتا از رجیسترTWDR به بیرون فرستاده می شود از طرف دیگر دیتای جدید به داخل شیفت داده می شود. به عبارت دیگر رجیسترTWDR حاوی آخرین اطلاعات باس است، مگر این که TWI از حالت Sleep بهWake Up تغییر حالات داده باشد. بیت ACK توسط CPU به طور مستقیم قابل دسترسی نیست و نوسط خود TWI تست می شود.

TWI (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

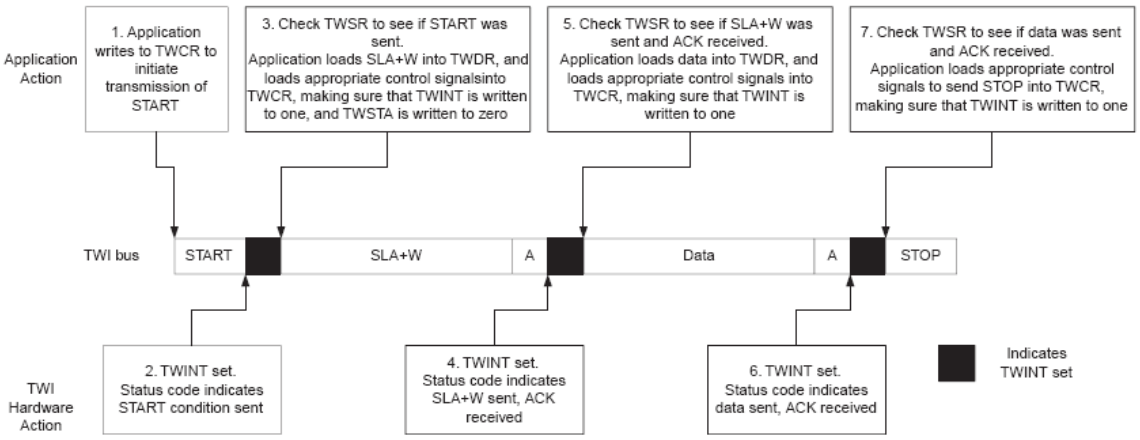
شکل 4-6: رجیستر TWAR

آدرس TWI پیکر بندی شده در یکی از مدهای SR یا ST توسط هفت بیت پرارزش رجیستر TWAE مشخص می شود. بیت TWGCE نیز برای تشخیص حالت General Call به کار می رود. هنگامی که Master اطلاعات آدرس را بر روی باس SDA

قرار می دهد، تمام Slave ها، اطلاعات باس را با محتویات رجیستر TWAR خود مقایسه می کنند. Slave ای که مقدار رجیستر TWAR آن با اطلاعات باس یکسان باشد. توسط Master انتخاب شده است.

٤-٣- روش استفاده از TWI

عملکرد TWI مبتنی برارسال و دریافت بایت و وقفه است. به این معنا که پس از هر کاری بر روی باس، نظیر دریافت یا انتقال بایت، بیت TWINT یک می شود. با توجه به این کهTWI مبتنی بر وقفه است برنامه نرم افزاری قادربه انجام بسیاری از کارهاست. در این شرایط اجرای برنامه سرویس وقفه به مقدار بیت TWIE و بیت I بستگی دارد.به عبارت دیگر در صورت صفر بودن TWIE ، بیتTWINT فقط برای کنترل و نحوه کار TWI به کار می رود. وقتی بیت TWINT یک می شود سیستم TWI کار خود را تمام کرده و منتظر پاسخ برنامه می ماند. در این حالت رجیستر TWSR دارای مقداری است که وضعیت باس را مشخص می کند. کاربر با توجه به مقدار رجیستر TWSR مقدار رجیستر های TWCR وTWDR را برای عملکردTWI ، در سیکل بعدی مشخص می کند. شکل 7-4 مثال ساده ای نحوه عملکرد متقابل رابط TWI و نرم افزار را نشان می دهد. مربع های سیاه رنگ در این شکل زمانی را نشان می دهد که بیت TWINT یک شده است. در این مثالMaster قصد ارسال یک بایت دیتا به Slave دارد. مراحل مختلف انجام این کار و عکس العملTWI به شرح زیر است.



شکل 7-4

١- مرحله اول ارسال شرطSTART است که این عمل با نوشتن مقدار مشخص در رجیسترTWCR انجام می شود. مقادیری که در هر مرحله مثال باید در رجیسترTWCR نوشته شود در ادامه بحث بررسی می شوند.

مقدار نوشته شده در رجیسترTWCR ، سخت افزارTWI را مجبور به ارسالSTART می کند. همچنین باید در مقدار نوشته شده،مقدار بیتTWINT یک باشد تا باعث صفر شدن آن در سیکل بعدی شود. زیرا تا زمانی کهTWINT یک باشدTWI هیچ عملی را انجام نمیدهد. بلافاصله پس از پاک شدن TWINT،TWI شروع به ارسال شرط START می کند.

۲-زمانی که شرطSTART فرستاده شد پرچمTWINT یک شده و مقدار رجیسترTWSR در این حالت نشان دهنده موفقیت یا عدم موفقیت در ارسال شرط START ، خواهد بود.

۳- در این مرحله برنامه، مقدار رجیستر TWSR را به منظور اطمینان از صحت ارسال شرط START بررسی می کند. در صورتی که رجیستر TWSR شرایط صحیح ارسال START را اعلام نکند برنامه ممکن است کار های مختلفی نظیر صدا زدن Error Routione[۵۹] را انجام دهد. اگر ارسال شرط START صحیح انجام شده باشد، در این صورت برنامهSLA+W را در رجیستر TWDR ، می نویسد. توجه شود که رجیستر TWDR برای ارسال آدرس و دیتا استفاده می شود. پس از نوشتنSLA+W در رجیسترTWDR ، مقدار مشخصی در رجیسترTWCR نوشته می شود تا سخت افزار را مجبور به ارسال SLA+W کند. در مقدار نوشته شده حتماً باید بیت TWINT را یک کرده باشد تا ان را در سیکل بعدی صفر کند ، در غیر این صورت TWI هیچ عملی را انجام نخواهد داد. بلا فاصله پس از پاک شدن این بیت، TWI اطلاعات آدرس را انتقال می دهد.

٤- پس از ارسال آدرس ، پرچم TWINT در رجیسترTWCR یک می شود و رجیسترTWSR به منظور نشان دادن موفقیت یا عدم موفقیت در ارسال صحیح آدرس تغییر می کند.این مقدار رجیستر TWSR ، ارسال شدن ACK یا ارسال نشدن آن را نیز مشخص می کند.

۵- در این مرحله مقدار TWSR به منظور حصول اطمینان از ارسال صحیح آدرس و مقدار بررسی می شود. در صورت عدم ارسال صحیح، کار خواص مورد نظر برنامه نویس اجرا می شود. اگر ارسال آدرس صحیح صورت گرفته باشد، برنامه دیتای مورد نظر برای ارسال را در رجیسترTWDR ، می نویسد وسپس مقدار مشخص ، در رجیسترTWCR نوشته می شود تا TWINT ارسال دیتا را آغاز نماید. در مقدار نوشته شده دررجیسترTWCR ، مقدارTWINT نیز یک میشود که برای پاک کردن آن به کار رفته است. بلا فاصله پس از پاک شدن این بیت انتقال دیتا آغاز می شود.

٦- پس از ارسال دیتا، پرچم TWINT در رجیسترTWCR یک می شود و مقدار رجیسترTWSR نشان دهنده صحت یا عدم صحت ارسال دیتا خواهد بود. رجیسترTWSR همچنین بیانگر ارسال یا عدم ارسال ACK نیز می باشد.

۷- برنامه کاربردی به منظور حصول اطمینان از صحت ارسال دیتا و همچنین ارسال یا عدم ارسال مقدار رجییستر TWSR را بررسی می کند. اگر ارسال درست انجام شده باشد برنامه با نوشتن مقدار مشخصی در رجیستر TWCR ، سخت افزار را ملزم به ارسال STOP می کند. همچنین به منظور انجام این عمل و پاک شدن TWINT ، مقدار این بیت باید یک شود. بلافاصله پس از پاک شدن بیتTWINT ،STOP ارسال می شود ولی در این حالت باس هیچ بررسی برای ارسال صحیح STOP انجام نمی دهد.

مراحل مختلف این مثال را می توان به صورت زیر خلاصه کرد.

- وقتیTWI کاری را به اتمام می رساند و منتظر پاسخ برنامه است، پرچم TWINT یک می شود. باسSCL تا زمانی کهTWINT پاک نشود TWI باقی می ماند وTWINT هیچ کاری انجام نمی دهد.

[59] -زیر برنامه خطا

– وقتی پرچم TWINT یک می شود، کاربر باید مقادیر تمامی رجیستر ها را برای انجام عمل بعدی TWI ، تنظیم نماید. به عنوان مثال نوشتن رجیستر TWDR با دیتایی که قرار است بعداً فرستاده شود.

– پس از تنظیم رجیستر ها ، مقدار مناسب در رجیستر TWCR نوشته شده و TWINT باید یک شود تا منجر به پاک شدن خودش در سیکل بعدی شود.

٤-٤-١- تحولات در Fast-mode

بعد از حالت استاندارد با توجه به نیاز به سرعت بالاتر حالت Fast-mode ابداع گردید که دارای ویژگی های زیر بود:

- حداکثر سرعت 400 kbit/s
- زمانبندی سیگنالهای SDA و SCL تطبیق داده شد.
- در نظر گرفتن سیستم حذف نویزهای سوزنی و Schmitt trigger[٦٠] در ورودی دستگاههای Fast-mode
- در بافرهای خروجی از کنترل شیب برای لبه پایین رونده استفاده شد.
- دستگاههای pull-up خارجی با زمان خیز کم برای این باس تطبیق داده شد.

برای کمتر از 200pf از pull-upهای مقاومتی و برای بیشتراز آن از منابع جریان (حداکثر 3mA) یا مدارات مقاومتی سوئیچینگ استفاده شد.

٤-٤-٢- تحولات در High Speed-mode (HS-mode)[61]

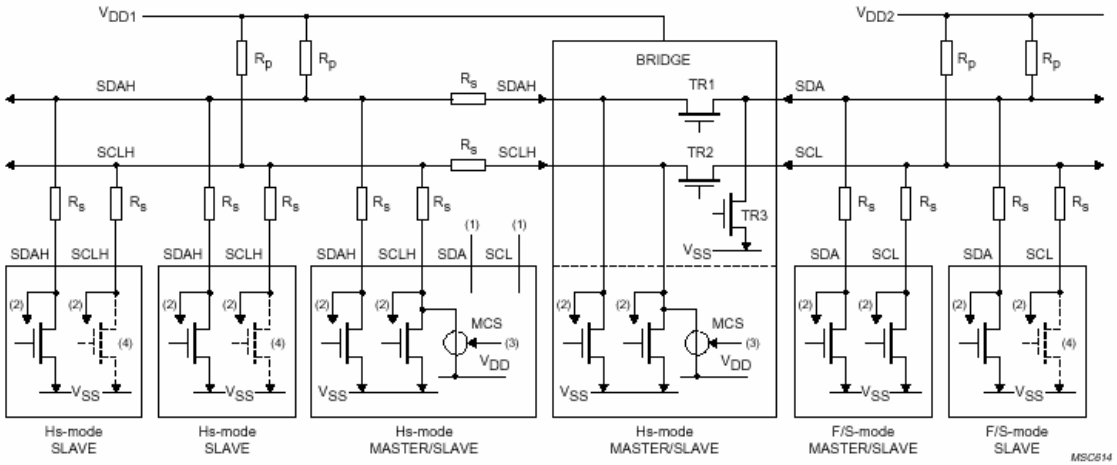
در این گونه وسایل :

- یک بافر خروجی Open-Drain برای سیگنال SDAH (یک ترکیب از مدارات open-drain pull down و current-source pull-up بر روی خروجی SCLH است که زمان خیز را برای SCLH کاهش می دهد.
- Arbitration و همزمان سازی clock وجود ندارد بلکه این عمل در زمانی که از حالت Fast به High تغییر وضعیت می دهیم صورت می گیرد.
- دارای پلهای داخلی بودند که برای اتصال SDAH و SCLH به SDA و SCL در حالت Fast-mode استفاده می شوند. که معمولاً در دو سطح ولتاژ متفاوت عمل می کنند.
- مقاومتهای (اختیاری) RS باعث حفاظت سطوح I/O از ضربه های سوزنی ولتاژ بالا و تداخل می شود.

---

[60] - راه انداز
[61] - مد سرعت بالا

- مقاومتهای پول آپ (RP) نیز وجود دارند ولی برای بالاتر از 100pf از منابع جریان پول آپ خارجی رسیدن به زمان خیز مورد استفاده می شود.



(1) Bridge not used. SDA and SCL may have an alternative function.
(2) To input filter.
(3) Only the active master can enable its current-source pull-up circuit.
(4) Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCL or SCLH.

Bus system with transfer at Hs- and F/S-mode speeds.

شکل ٤-٨: نمایی از باس در حالت HS-mode

٤-٥- فرمت ارسال داده های سریال در HS-mode

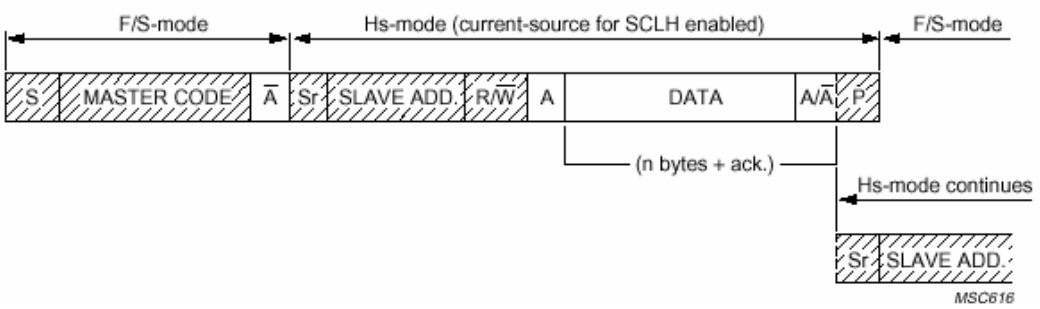
تمام وسایلی که در حالت HS-mode باید کار کنند در ابتدا در حالت Fast-mode قرار دارند و تنها بعد از وضعیتهای زیر وارد حالت HS-mode می گردند:

١- وضعیت Start (S)

٢- کد به صورت ٨ بیتی (00001XXX)

٣- بیت Not-Acknowledge ($\overline{A}$)

بعد از این وضعیتها وارد حالت HS-mode شده و با توجه به نوع آدرس دهی ٧ بیتی و یا ١٠بیتی آدرس دهی آغاز می گردد.



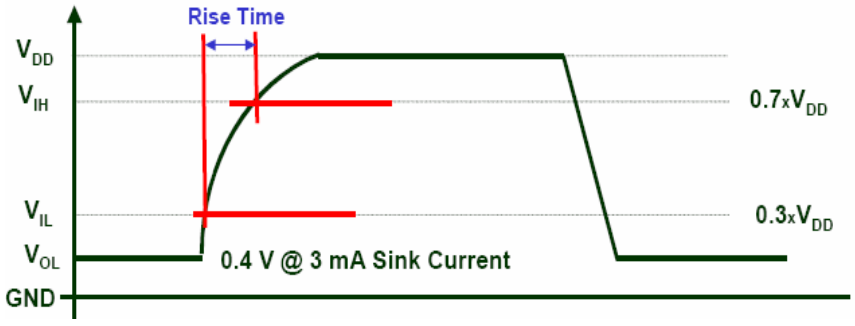
شکل ٤-٩ : فرمت انتقال دیتا در HS-mode

جدول زیر حداکثر سرعت انتقال داده و ماکزیمم ظرفیت خازنی و rise time[62] و تعداد بیت آدرس را در مورد مدهای مختلف I^2C مورد مقایسه قرار می دهد.

	Standard-Mode	Fast-Mode	High-Speed-Mode	
Bit Rate (kbits/s)	0 to 100	0 to 400	0 to 1700	0 to 3400
Max Cap Load (pF)	400	400	400	100
Rise time (ns)	1000	300	160	80
Spike Filtered (ns)	N/A	50	10	
Address Bits	7 and 10	7 and 10	7 and 10	

شکل ٤-١٠ : مقایسه مدهای TWI

در شکل زیر شمایی از Rise Time را مشاهده می کنید.



شکل ٤-١١: زمان خیز

[62] -زمان خیز

٤-٦- مد های کاریTWI

برای کار کردنTWI ، چند مد از مد های فوق باید همزمان کار کنند. به عنوان مثال TWI از مد های MR[63] و MT[٦٤] برای نوشتن و خواندن EEPROM استفاده می کند. در ادامه مد های مختلف بررسی و با شکل هایی حالتهای هر مد نشان داده می شود. در این شکلها از حروفی استفاده شده است که این حروف ابتدای کلمات مشخص شده در زیر هستند.

S:START RS:REPEATED START R:Read bit

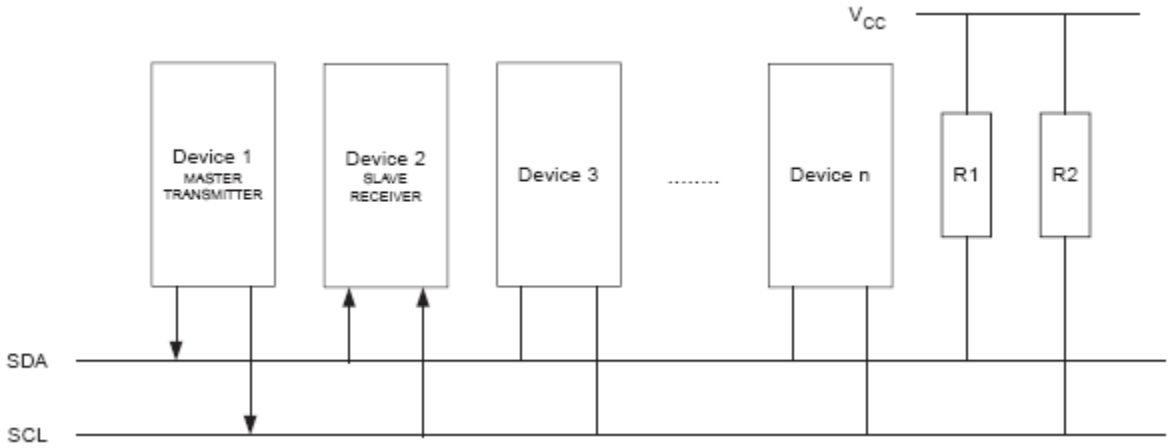
SLA:Slave Address $\overline{A}$: $NACK$ P:STOP

W:Write bit A:Acknowledge

در شکل های ادامه ، دایره ها به منظور نشان دادن یک شدن پرچم TWINT و اعداد نشان داده شده در آن ها ؛ مقدار رجیستر TWSRاست به شرط آن که بیت های TWPSI..0 صفر در نظر گرفته شوند.هنگامی که بیت TWINT یک است ، عدد نوشته شده در رجیستر TWSR مشخص کننده وضعیت TWI است. برای هر عدد مشخص شده در رجیستر TWSR ، عملی که هر مد باید انجام شود ، در جدول ها آورده شده است. برای مقادیر ارائه شده در تمام جدول ها فرض بر این است که بیت های TWPSI..0 یا صفر هستند یا به صفر Mask شده اند.

٤-٦-١- مد Master Transmitter[65]

در مدMT ، مطابق شکلزیر، اطلاعات از فرستنده Master به گیرنده Slave (مد SR) ارسال می شود.



شکل ٤-١٢

چنانچه در شکل زیر دیده می شود، در این حالت دو مد مختلف با یکدیگر کار می کنند.نحوه عملکرد TWI در مد SR در ادامه فصل بررسی شده است.به منظور وارد شدن به مدMT ، ابتدا شرطSTART فرستاده می شود. پس از ان فرمت آدرس ارسالی

[63] master receive-
[64] master transmit-
[65] - ارباب ارسال کند

مشخص کننده مدMT یا MR است. اگر سیگنالSLA+W (آدرس Slave و بیت کنترل عمل نوشتن) ارسال شود مدMT و اگرSLA+R ارسال شود مد MR فعال می شود . به عبارت دیگر عملR/W ، مد MR یاMT را مشخص می کند. ارسال START با نوشتن مقدارزیر در رجیستر TWCR انجام می شود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

در جدول بالا ، بیتTWEN برای فعال کردن TWI ، بیت TWSTA به منظور ارسالSTART و بیت TWINT به منظور پاک کردن خودش یک نوشته می شود. پس از این مرحله رابط TWI ، باس را تست کرده و بلافاصله پس از آزاد شدن باس ، شرط START را ارسال می کند. پس ازارسال START بیت TWINT توسط سخت افزار یک و مقدار رجیسترTWSR مطابق جدول زیر برابر۰۸$ می شود. به منظور وارد شدن به مدMT، باید SLA+W منتقل شود. این عمل با نوشتنSLA+W در رجیستر TWDR و مقدار زیر در رجیستر TWCR انجام می شود. پس از پاک شدن بیتTWINT در سیکل بعدی، SLA+W ارسال می شود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

پس از ارسال SLA+W و دریافت ACK از Slave آدرس دهی شده، بیت TWINT مجدداً یک می شود و مقدار رجیستر TWSR قابل دسترسی است. مقدار های ممکن برای این رجیستر در مد MT یکی از مقادیر ۱۸$ ، ۲۰$ و ۳۸$ است. کار مورد نیاز برای هر کدام از مقادیر فوق در جدول زیر بیان شده است.

پس از انتقال موفق SLA+W ، باید دیتا ارسال شود. این عمل با نوشتن دیتا در TWDR انجام می شود. نوشتن دیتا فقط زمانی امکان پذیر است که بیت TWINT یک باشد. در غیر این صورت نوشتن امکان پذیر نبوده و بیت TWWC یک می شود. پس از نوشتن اطلاعات مورد نیاز در رجیستر TWDR، بیت TWINT باید صفر شود تا انتقال اطلاعات آغاز شود. این کار با نوشتن مقدار زیر دررجیسترTWCR انجام می شود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	0	X	1	0	X

پس از انتقال کامل دیتا، مقدار زیر در رجیسترTWCR نوشته می شود تا شرط STOP ارسال شود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	0	1	X	1	0	X

پس از ارسال REPEATED STSRT ، TWCR می تواند همان Slave یا Slave دیگری را انتخاب نماید. به عبارت دیگر این سیگنال، Master را قادر به سوئیچ بین Slave ها کرده یا تغییر مد از MT به MR را ممکن می سازد.

شکل زیر حالت های مختلفی را که ممکن است در مد MT رخ دهد را بیان کرده است. عدد های نوشته شده در داخل بیضی ها بیانگر مقادیری است که رجیستر TWSR در هر مرحله دارا خواهد بود. دیتاهای که در شکل به صورت خاکستری نشان داده شده اند ، دیتایی است که توسط Master ارسال می شود و بقیه دیتا ها توسط Slave ارسال می شود.

چنانچه در جدول زیر دیده می شود ، پس از ارسال شرط START مقدار رجیستر TWSR برابر ۰۸$خواهد بود.پس از ارسال شرط START باید سیگنال SLA+W ارسال شود. برای انجام این کار باید مقدارSLA+W در رجیستر TWDR و مقداری که در جدول زیر نشان داده شده است، در رجیستر TWCR نوشته شود. پس از ارسال SLA+W ، چندین حالت ممکن است رخ دهد. حالت اول این که Slave آدرس را دریافت نموده و سیگنال ACK در پاسخ به دریافت آدرس ارسال شود.در این حالت مقدار رجیسترTWSR برابر۱۸$ و پس از ان در صورت نیاز ، دیتا ارسال میشود. در صورت دریافت سیگنال ACK دیتا، مقدار رجیستر TWSR برابر ۲۸$ و پس از ان شرط REPEATED STSRT یا STOP ارسال می شود یا Master دیگری باس را در اختیار می گیرد.

حالت دوم اینکه سیگنال ACK دریافت نشود ، در این حالت اگر مقدار رجیسترTWSR برابر ۲۰$ باشد، پس از آن شرط STOP ارسال می شود و اگر مقدار رجیستر TWSR برابر ۳۸$ باشد ، ممکن است Master دیگری باس را در اختیار بگیرد.

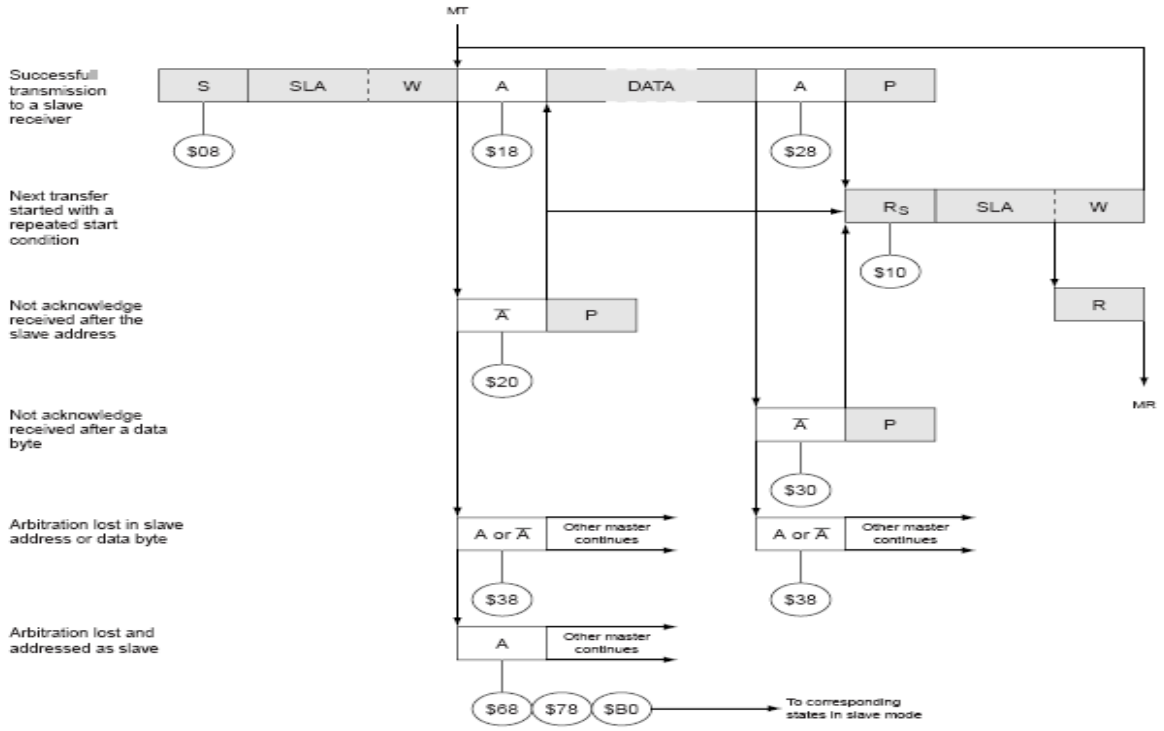
Table 74. Status Codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

جدول 4-1 : رجیستر TWSR در مد MT

حالت دیگری که پس از ارسال شرط START ممکن است رخ دهد، زمانی است که مقدار رجیستر TWSR برابر ۱۰$ باشد. در این حالت یک شرط REPEATED STSRT ارسال شده و بعد از ان بسته به نظر کاربر دو وضعیت مختلف ممکن است رخ دهد. اول آن که کاربر نخواهد مد کاری را عوض نماید. در این حالت با نوشتن مقدار مشخص در رجیسترTWCR ، مقدار SLA+W را برای باقی ماندن در مد MT ارسال می نماید. در صورت نیاز به تغییر مد کاری به مدSR ، با نوشتن SLA+R در رجیستر TWDR و مقدار مورد نیاز در رجیستر TWCR این مقدار ارسال می شود. پس از ارسالSLA+R سخت افزارTWI، مد کاری را به مد SR تغییر خواهد داد. همچنین زمانی که مقدار رجیستر TWSR برابر ۳۸$ است، زمانی است که باس Arbitration رااز دست داده است. در این شرایط اعمال عنوان شده در جدول بالا انجام می شود. حالت های مختلف در شکل زیر نشان داده شده است. با توجه به این که کلیه حالت ها با همین روش تحلیل می شود، بررسی بقیه حالت ها به خواننده واگذار می شود.

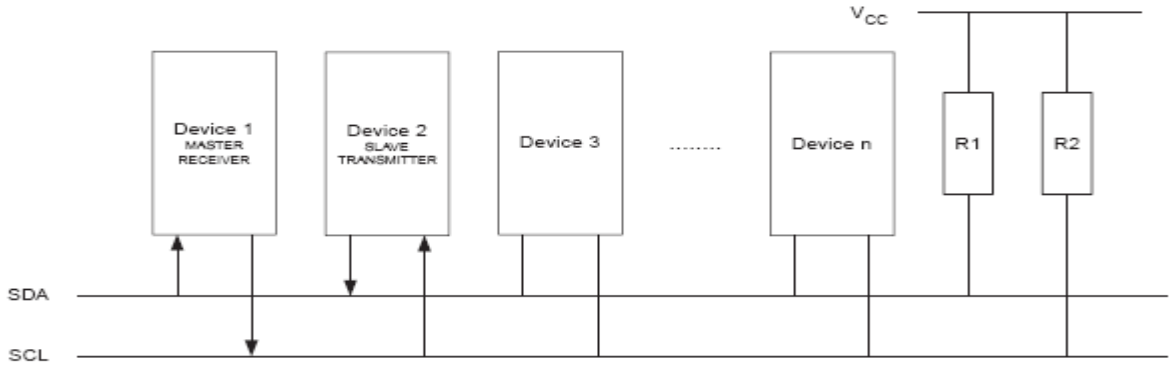
توجه به این نکته ضروری است که اطلاعات ارائه شده توسط جدول بالا و شکل زیر یکسان است. چنانچه در این شکل نشان داده شده است، پس از ارسال شرط START برای وارد شدن به مدMT باید SLA+W ارسال شود. حالت های مختلف ممکن پس از این مرحله به راحتی در شکل به فلش هایی قابل پیگیری است. اطلاعات ارسال شده توسط Master در شکل با زمینه خاکستری نمایش داده شده است.



شکل 4-13 : مربوط به مد master transmit

٤-٦-٢- مد Master Recive[66]

مطابق شکل زیر در مد MR ، گیرنده Master اطلاعاتی را از یک فرستنده Slave دریافت می کند. به منظور وارد شدن به مد MR ، پس از ارسال سیگنال START، مد آدرس دهی مشخص کننده مد MR یاMT است. اگر SLA+W ارسال شود مدMT و اگر SLA+R ارسال شود مد MR فعال می شود.



شکل ١٤-٤

در این مد مشابه حالت قبل ، ابتدا سیگنال START ارسال میشود.برای وارد شدن به مد MR باید SLA+R در رجیستر TWDR نوشته شود. به عبارت دیگر آدرس Slave ای که قرار است اطلاعات آن خوانده شودهٔ مشخص می شود. سپس برای ارسال اطلاعات آدرس مقدار زیر در رجیستر TWCR نوشته میشود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	1	X	1	0	X	1	0	X

پس از ارسال اطلاعات آدرس یا SLA+R ، بیت ACK از Slave آدرس دهی شده دریافت و پرچم TWINT یک می شود. مقادیر رجیستر TWSR مطابق جدول ١-٤ برابر یکی از مقادیر $٣٨، $٤٠ و $٤٨ خواهد بود. عملکرد متناظر با هر کدام از این مقادیر در جدول زیر ذکر شده است. پس از این مرحله بسته به نحوه بر نامه ریزی فرستنده Slave ، ممکن است دیتا ارسال شود. پس از دریافت دیتا توسط Master ، حالت های مختلفی ممکن است رخ دهد. مقدار دیتای دریافتی توسط رجیسترTWDR در زمانی که بیتTWINT یک است خوانده می شود. این عمل تا رسیدن آخرین بایت ادامه می یابد. پس از رسیدن آخرین بایت، MR با فرستادن NACK ، به مد ST تغییر حالت میدهد.

فرستنده با فرستادن STOP یا REPEATED STSRT به ارسال اطلاعات خاتمه می دهد. ارسال STOP و REPEATED STSRT مطابق حالت قبل (مدMT) صورت می گیرد.

[66] -ارباب دریافت کند

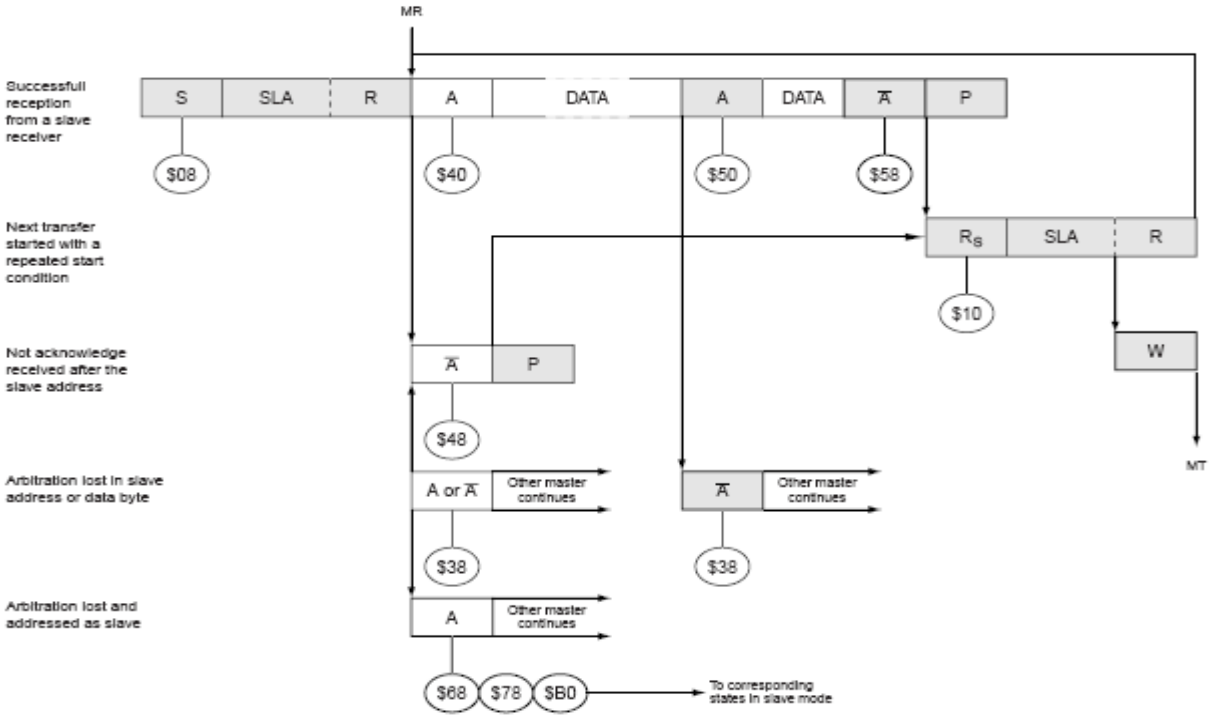
پس از ارسالREPEATED STSRT ،Master می تواند همان Slave یا Slave دیگری را بدون ارسالSTOP انتخاب نماید. به عبارت دیگر سیگنال REPEATED STSRT ، Master را قادر به سوئیچ بین Slave ها کردهن یا تغییر از مد MT به MR را ممکن می سازد.

شکل زیر حالت های مختلفی را که ممکن است در مد MR رخ دهد بیان کرده است. عدد های نوشته شده در داخل بیضی ها بیان گر مقادیری است که رجیسترTWSR در هر مرحله دارا خواهد بود.

Table 75. Status Codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to master Transmitter mode
$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
$58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

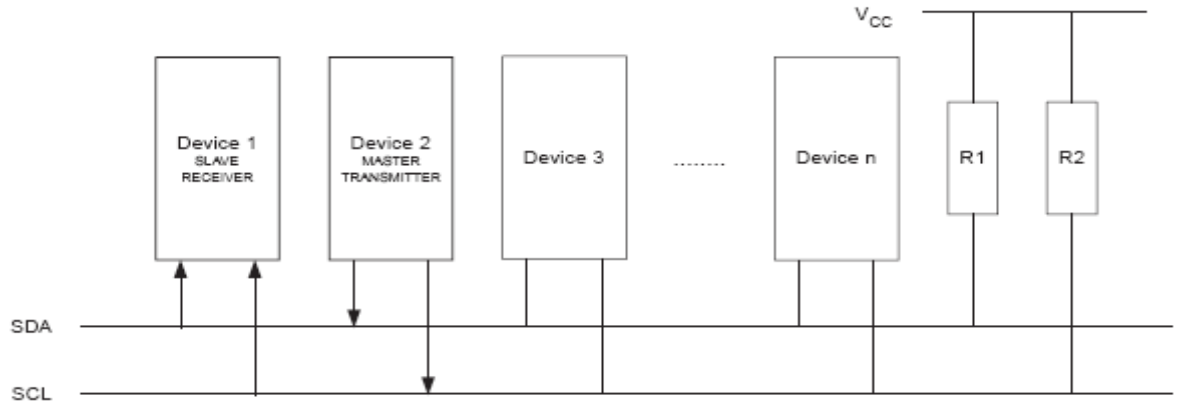
جدول ٤-٢: رجیستر TWSR در مد master receive



شکل ۱۵-٤ : مد MR

٤-٦-٣- مد Slave Reciver[67]

مطابق شکل زیر در مد SR ، اطلاعات از فرستنده Master به گیرنده Slave منتقل می شود.



چنانچه در مدMT بیان شد، فرستندهMaster پس از ارسالSTART ، آدرس گیرنده Slave را ارسال می کند. در نتیجه آدرس گیرنده Slave باید قبل از این مرحله تنظیم شده باشد تا Slave آدرس دهی شده تشخیص داده شود. در نتیجه در مد SR آدرس Slave به صورت زیر در رجیستر TWAR نوشته می شود.

[67] -برده دریافت کند

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Value	Device's Own Slave Address							

در مقدار فوق، اگر بیت صفر یک باشد فراخوانی عمومی صورت می گیرد. پس از تنظیم آدرس، مقدار رجیستر TWCR باید به صورت زیر تنظیم شود.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Value	0	1	0	0	0	1	0	X

پس از تنظیم رجیستر هایTWAR وTWCR ، Slave منتظر آدرس دهی شدن توسط Master و بیت های جهت(R/W) می شود. پس از تطابق آدرس دریافتی با مقدار رجیستر TWAR، با توجه به یک بودن بیت TWEA در رجیستر TWCR، سیگنال ACK توسط Slave ارسال می شود. در این حالت اگر مقدار بیت های R/W صفر باشد، TWI در مد SR و در غیر این صورت در مد ST کار خواهد کرد. پس از دریافت آدرس و بیت R/W ، بیت TWINT یک شده، مقدار رجیستر TWSR قابل خواندن خواهد بود. این مقدار رجیستر TWSR برای تصمیم گیری برنامه به کار میرود. مقادیر رجیسترTWSR در این مد در جدول ٤-٢ و عملکرد مورد نیاز بیان شده است. مد SR در هنگام از دست دادن نیز ممکن است رخ دهد.

اگر بیت TWEA در حین انتقال صفر شود، TWI بعد از دریافت دیتا، سیگنال NACK را ارسال میکند. در این حالت Slave قادر به دریافت اطلاعات قبلی نخواهد بود. همچنین هنگام صفر بودن بین TWEA ، TWI قادر به مقایسه اطلاعات آدرس قرار گرفته برروی باس با مقدار رجیستر TWAR خود نبوده ، تشخیص آدرس Slave ممکن نیست. با یک کردن بیت TWEA این مشکل رفع میشود.

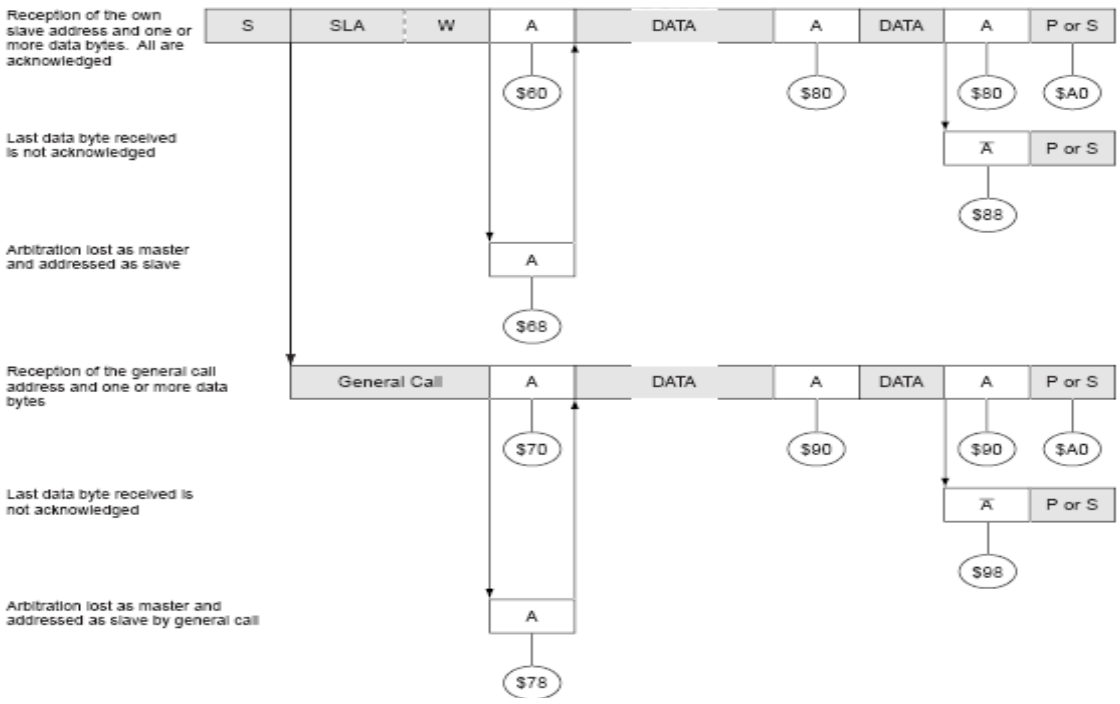
در تمام مد های Sleep به جز Idle ، پالس سیستم TWI غیر فعال می شود. در این مد ها که پالس سیستم TWI یا$CLK_{I/O}$ غیر فعال است، اگر بیت TWEA یک باشد، TWI می تواند از پالس SCL به عنوان منبع پالس استفاده کرده و در صورت آدرس دهی شدن توسط Master یا تشخیص حالت General Call ، میکرو کنترلر را به حالت Wake Up منتقل کند. هنگام Wake Up شدن باس SCL ، Low نگه داشته می شود تا بیت TWINT نیز پاک شود. پس از Wake Up شدن، پالس $CLK_{I/O}$ فعال و عملکرد نرمالTWI آغاز می شود. اگر در این حالت زمان START up طولانی باشد، به دلیلLow نگه داشتن SCL به مدت طولانی، مدتی انتقال اطلاعات دچار مشکل می شود. اطلاعات رجیستر TWDR نیز، پس از Wake Up شدن معتبر نیست.

Table 76. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$A0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

جدول ٤-٣ : رجیستر TWSR در مد slave receive

شکل زیر حالت های مختلفی را که ممکن است در مد MR رخ دهد را بیان کرده است. عدد های نوشته شده در داخل بیضی ها بیان گر مقادیری است که در رجیستر TWSR در هر مرحله دارا خواهد بود . دیتاهایی که در شکل به صورت خاکستری نشان داده شده اند، دیتایی است که توسط Master ارسال می شود.

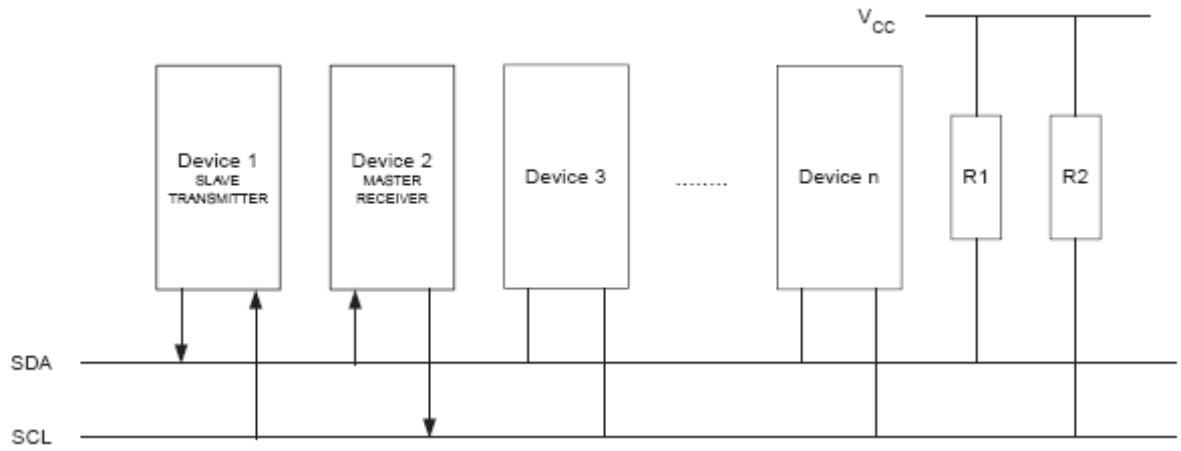


شکل ٤-١٦ : مد slave receive

٤-٦-٤- مد Slave Transmitter[68]

مطابق شکل زیر در مد ST یک گیرنده Master اطلاعاتی را از یک فرستنده Slave دریافت می کند. در این حالت برای مشخص کردن آدرس فرستنده Slave از رجیستر های TWAR و TWCR دقیقاً مشابه حالت قبل استفاده می شود. پس از تکمیل رجیستر های فوق، Slave منتظر آدرس دهی شدن توسط Master به همراه بیت های جهت می ماند. اگر مقدار بیت R/W یک باشد، TWI در مدST و در غیر این صورت در مد SR خواهد بود. پس از رسیدن آدرس و بیتR/W ، پرچم TWINT یک شده، رجیستر TWSR مبین وضعیت TWI ، مطابق جدول زیر خواهد بود. عملی که قرار است پس از این مرحله انجام شود با توجه به مقدار رجیسستر TWSR انجام می شود. عدم موفقیت در پروسه Arbitration نیز می تواند منجر به وارد شدن TWI به این مد شود.

[68] -برده ارسال کند



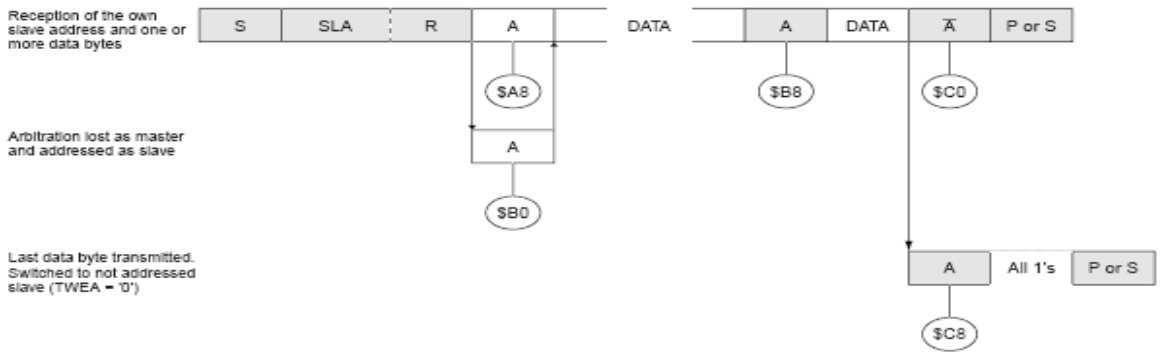
اگر بیتTWEA در حین انتقال صفر شود، TWI اطلاعات در حال انتقال را ارسال کرده و بسته به دریافت یا عدم دریافت سیگنال ACK بعد از آخرین بایت اطلاعات، وارد یکی از مراحل$C0 یا$C8 می شود. سپس TWI در مد Slave آدرس دهی نشده قرار گرفته و از اطلاعات Master در صورتی که در حال انتقال اطلاعات باشد، صرف نظر میکند. در این حالت Master تعدادی بیت ۱ به صورت سریال دریافت می کند. حالت $C8 نیز زمانی رخ میدهد که با صفر بودن TWEA ، Master انتظار دریافت دیتاهای دیگر دارد. همچنین هنگام صفر بودن بیت TWEA ،TWI قادر به مقایسه اطلاعات آدرس قرار گرفته برروی باس با مقدار رجیستر TWAR خود نبوده، تشخیص آدرس Slave ممکن نیست. با یک کردن بیت TWEA این مشکل رفع می شود. با توجه به این نکات می توان گفت بیت TWEA برای ایزوله کردن موقت TWI از باس ها به کار می رود.

Table 77. Status Codes for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$A8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$C0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$C8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

جدول ٤-٤: رجیستر TWSR در مد slave transmit

شکل زیر حالتهای مختلفی را که در مد ST رخ می دهد بیان کرده است. مراحل نهایی نشان داده شده در شکل مربوط به زمانی است که بیت TWEA در حین انتقال صفر شده است. در صورتی که در این حالت سیگنال ACK از Master دریافت نشود، باید شرط STOP/STRAT، بسته به نظر برنامه نویس و توسط Master ارسال شود.



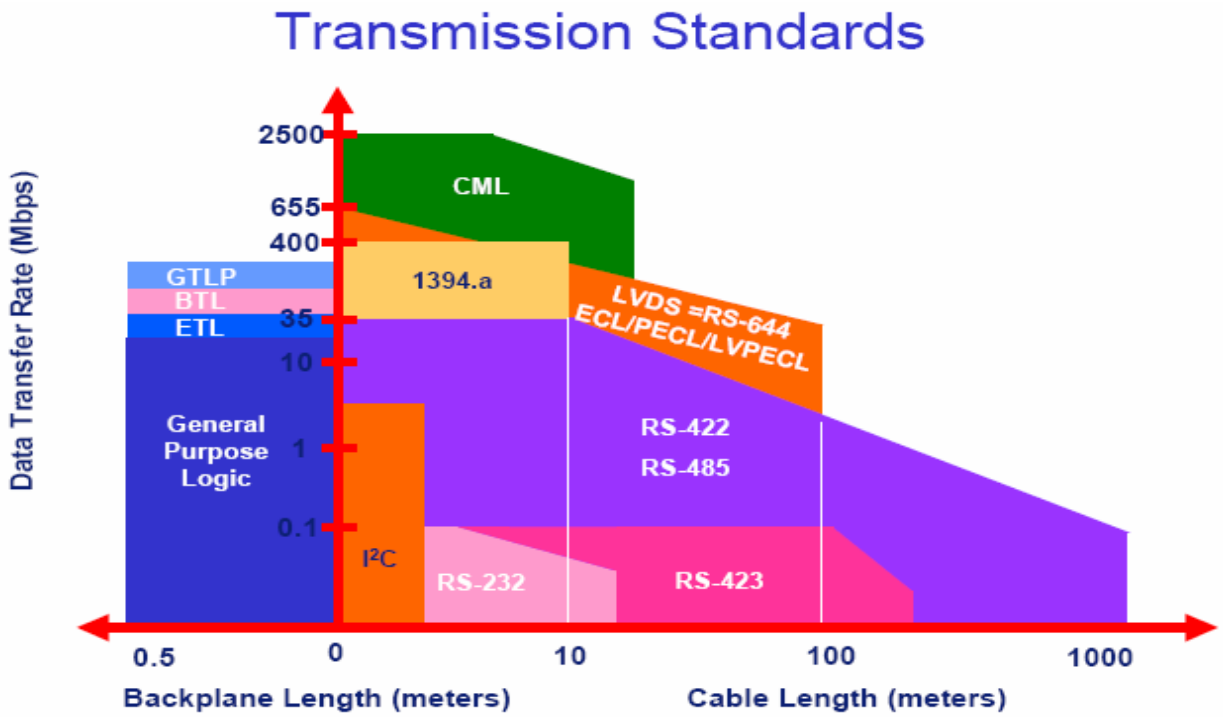
شکل ٤-١٧ : مربوط به مد slave transmite

فصل پنجم

مقایسه و کاربرد

۵-۱- مقایسه I^2C با سایر پروتکل ها

شکل پایین سرعت انتقال دیتا را بر حسب طول کابل در برخی از پروتکل ها نشان می دهد.



شکل ۵-۱ : سرعت انتقال دیتا بر حسب طول کابل در برخی از پروتکل ها

در جدول زیر سرعت انتقال دیتا را برای تمام پروتکل های معرفی شده با هم مقایسه می کنیم.

CAN (1 Wire)	33 kHz (typ)
I²C ('Industrial', and SMBus)	100 kHz
SPI	110 kHz (original speed)
CAN (fault tolerant)	125 kHz
I²C	400 kHz
CAN (high speed)	1 MHz
I²C 'High Speed mode'	3.4 MHz
USB (1.1)	1.5 MHz or 12 MHz
SCSI (parallel bus)	40 MHz
Fast SCSI	8-80 MHz
Ultra SCSI-3	18-160 MHz
Firewire / IEEE1394	400 MHz
Hi-Speed USB (2.0)	480 MHz

جدول۵-۱ : مقایسه سرعت انتقال در برخی از پروتکل ها

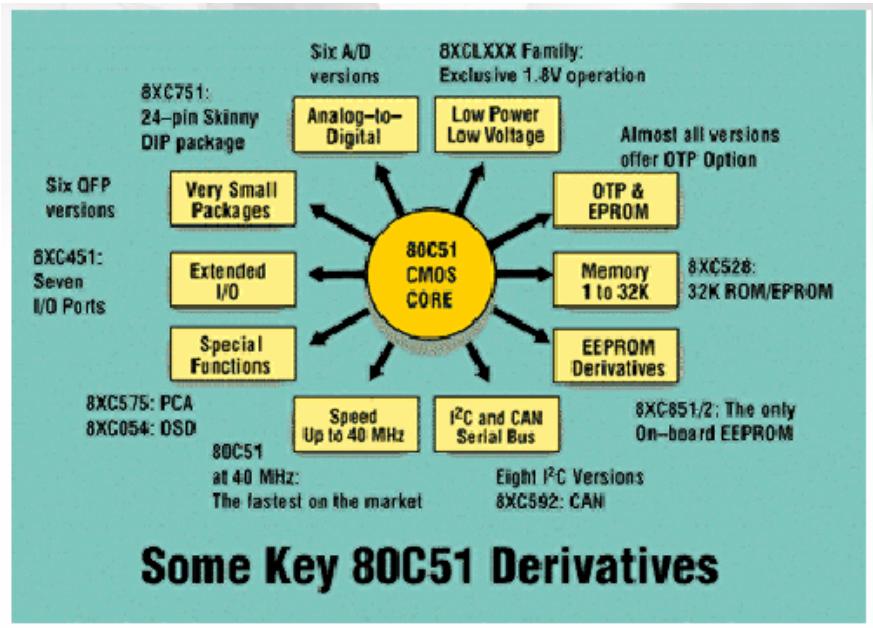
جدول زیر مقایسه برخی از پارامتر های این پروتکل ها را از قبیل سرعت انتقال، طول کابل استفاده شده و ماکزیمم ظرفیت خازنی و یک سری از محدودیت ها نشان می دهد.

Bus	Data rate (bits / sec)	Length (meters)	Length limiting factor	Nodes Typ.number	Node number limiting factor
I²C	400k	2	wiring capacitance	20	400pF max
I²C with buffer	400k	100	propagation delays	any	no limit
I²C high speed	3.4M	0.5	wiring capacitance	5	100pF max
CAN 1 wire	33k	100	total capacitance	32	load resistance and transceiver current drive
CAN differential	5k	10km	propagation delays	100	
	125k	500			
	1M	40			
USB (low-speed, 1.1)	1.5M	3	cable specs	2	bus specs
USB (full-speed, 1.1)	1.5/12M	25	5 cables linking 6 nodes (5m cable node to node)	127	bus and hub specs
Hi-Speed USB (2.0)	480M				
IEEE-1394	100 to 400M+	72	16 hops, 4.5M each	63	6-bit address

جدول ۵-۲ : مقایسه برخی از پارامترها

۵-۲- کاربردهای I^2C

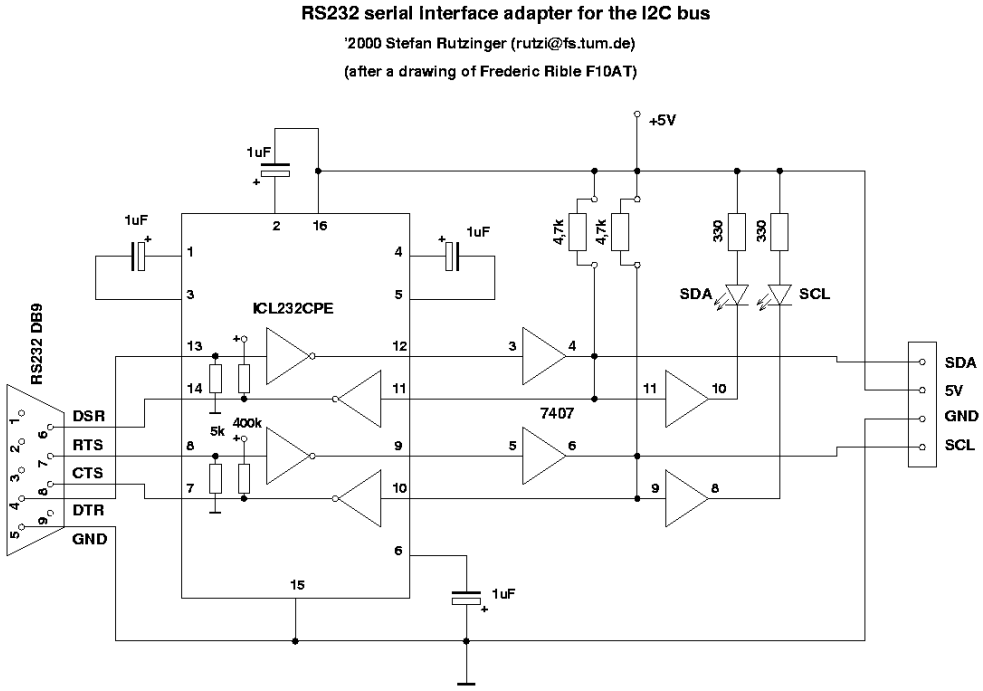
شرکت فیلیپس پایه و اساس میکرو های خود را بر معماری 80c51 بنا نهاده است. با توجه به ساختار کوچک و ارزان این تراشه می توان از آن در اکثر سیستم های کنترلی ارزان قیمت استفاده کرد.شرکت مذکور تراشه هایی در کاربردهای متنوع ارائه داده است. در مجموع این تراشه ها شامل ROM(OTP/Flash) و RAM قابل افزایش، پورت I^2C ، I/O گسترده ، ADC و... می باشند.



شکل ۵-۲ : نمایی از کاربردهای i2c

۵-۳-۱- I^2C و RS-232

با توجه به کاربرد گسترده RS-232 در ارتباط دستگاه های مختلف، وجود رابطی که RS-232 را به I^2C تبدیل کند،می تواندزمینه های جدیدی را پیش پای طراحان بگذارد. درزیرنمونه ای ازمدار آن را می بینید.

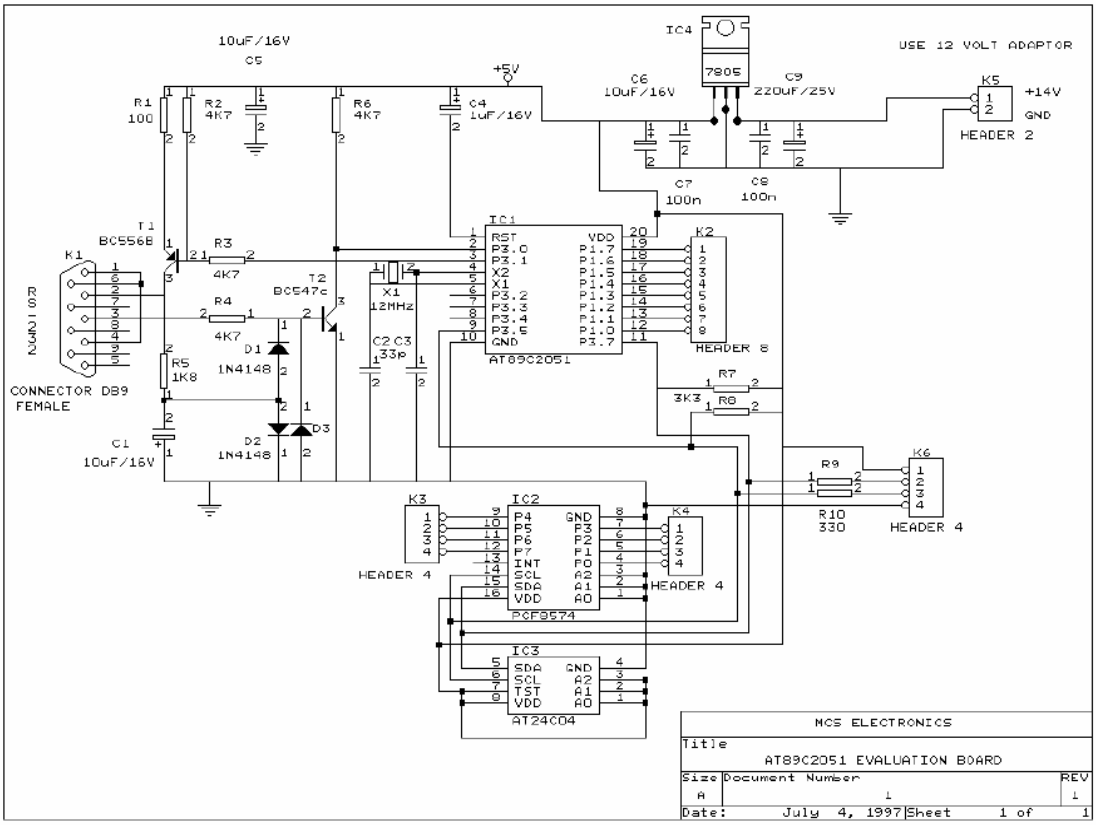


شکل ٥-٤ : ارتباط i2c با RS232

مثال١ : استفاده از میکروهای AT89C2051/AT89C4051 در ارتباط با قطعاتی با رابط I2C

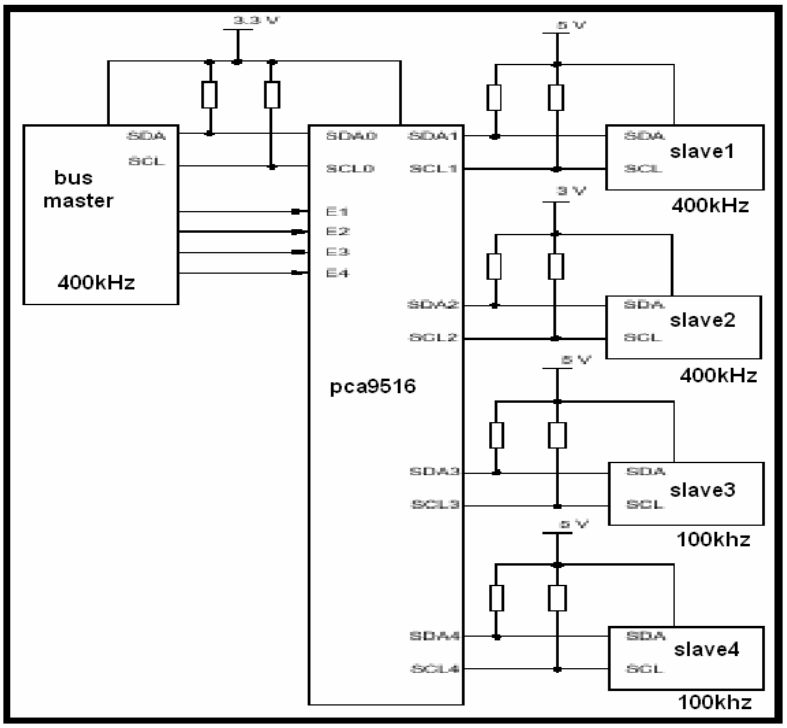
قطعـات مـورد نیـاز در ایـن مثـال : رابطهـای RS-232 وDB9 ،Header بـرای نمـایش LCD ، PCF8574 I/O Extender، AT24C04(I^2C EEPROM)

البته قابل ذکراست که این ارتباط با برنامه نویسی مناسب میکروکنترلر ٤٠٥١/٢٠٥١ تکمیل می شود.



شکل ۵-۵: نمایی از ارتباط RS232 با EEPROM با I2C

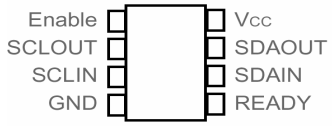
مثال۲: استفاده از HUB



شکل ۵-۸

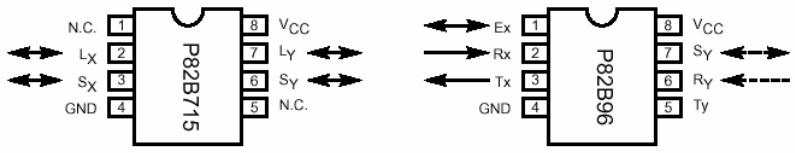
مثال۳: I^2C Hot swap Buffer[69]

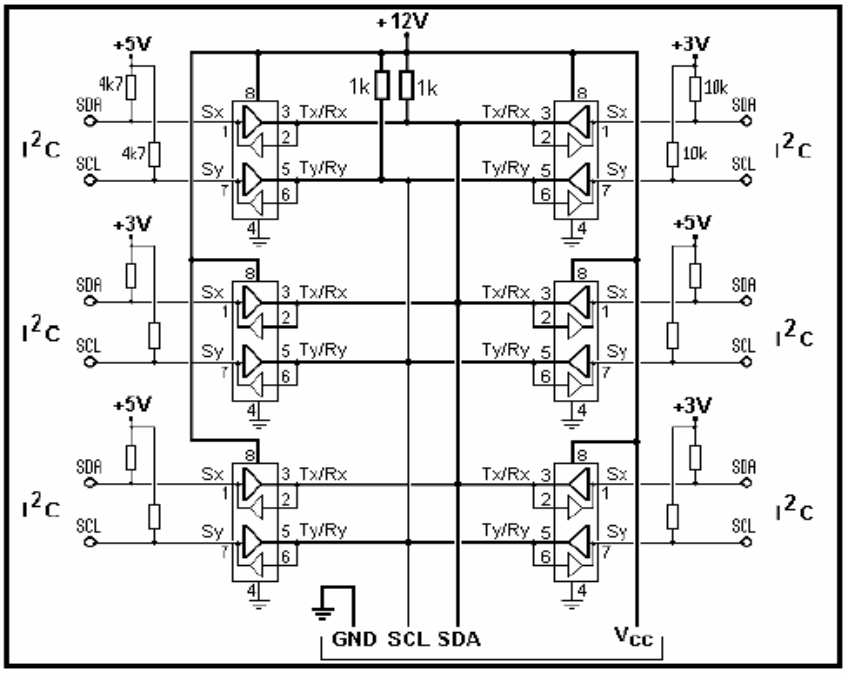
از این بافرها برای اضافه کردن وسایل جانبی به باس استفاده می شود. علاوه بـر ایزولاسـیون دو جهتـه، امکـان وجـود ظرفیـت حداکثرPF ٤٠٠ را در هر انشعاب می دهد. همچنین برای ایمنی در مقابل نویز سطح ولتاژخطوط SDA وSCL را تـا حـد ۱ ولـت کاهش داده است. خروجی درین باز با امپدانس خروجی بالا و جریان دهی بهتر از مزیت های این IC می باشد.



شکل ۵-۹ : نمایی از بافر برای اضافه کردن وسایل جانبی به باس

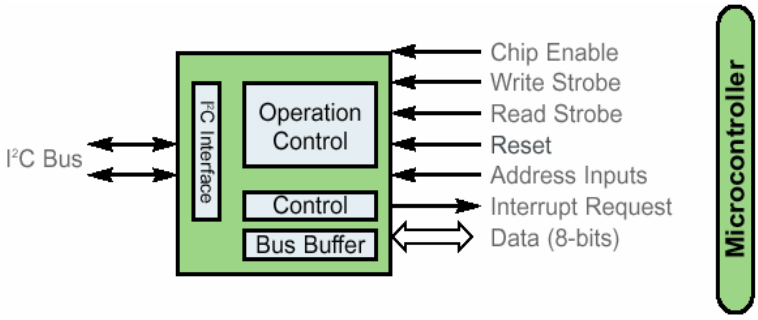
توجه: استفاده از I^2C Bus Extenders امکان وجود ظرفیت حداکثرPF ٤٠٠ را در طرف دیگر فراهم می کند.





[69] - بافر گسترش دهنده باس

۵-۳-۲- ارتباط باس موازی با I^2C Controller

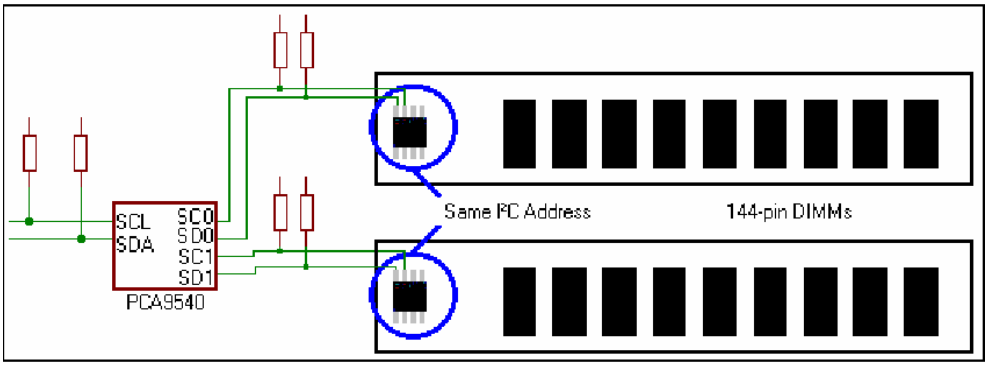


	Voltage range	Max I²C freq	Clock source	Parallel interface
PCA8584	4.5 - 5.5V	90 kHz	External	Slow
PCA9564	2.3 - 3.6V w/5V tolerance	360 kHz	Internal	Fast

شکل ۵-۱۱ : ارتباط باس موازی با I2C

مثال:کاربرد مالتی پلکسر

در اکثر RAM های کامپیوتری از این IC استفاده می شود.



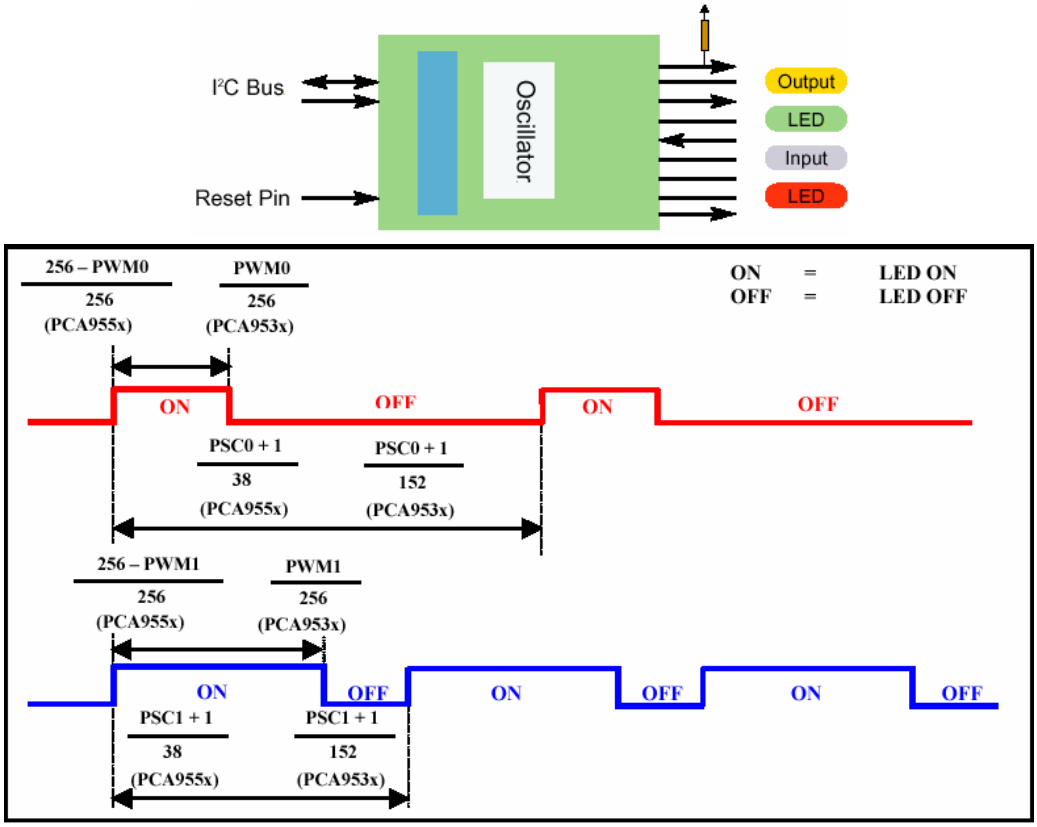
شکل ۵-۱۳ : نمایی از اسلات ram و نحوه اتصال I2C به آن

I^2C Device for LED Display Control[70]

ازجمله: PCA9530-33 , PCA9550-53 , SAA1064

- دارای کریستال و اسیلاتور داخلی می باشند.
- دارای ۲ حالت برای چشمک زدن (blinking) می باشند.
- جریان بالای خروجی ها (open drain)

[70] - ابزارهای i2c برای کنترل نور LED



شکل ۵-۱٤ : نمایی از کنترل نور LED توسط I2C

مبحث دیگر در مورد LED ها، مربوط به کنترل شدت نورمی باشند. با تغییر duty cycle[71] و همچنین تنظیم متوسط جریان که به LED ها می رسد می توان شدت نور آنها را کنترل کرد.

سایر کاربردها

این پروتکل ارتباطی در سنسورها نیز کاربرد گسترده ای دارد نظیر IC های LM75A(Philips),LM82(National semiconductor),AD7416(Analog Device) که برای اندازه گیری دما استفاده می شوند. همچنین در CMOS Image Sensors مانند MB86S02A(Fujitso) ، کارتهای Server Management ، Motorola Handheld Computer کاربرد دارد.

[71] -سیکل وظیفه

در ضمن اکثر IC های شرکت Xicor که برای ارتباط با کامپیوترو نرم افزار Labview می باشند، از پروتکل ارتباطی I^2C استفاده می کنند.

در یک دسته بندی خواص ابزار هایی که با I^2C می توانند ارتباط برقرار کنند عبارتند از:

- گیرنده های تلویزیونی : تولید وفق دهنده ها و دریافت کننده های تلویزیون
- گیرنده های رادیویی : تولید وفق دهنده ها و دریافت کننده های رادیویی
- پردازش صدا
- کنترل کننده مادون قرمز
- DTMF (Dual Tone Multiple Frequency).
- کنترل نمایش LCD : تولید تغذیه برای segment[72] های یک LCD که به وسیله باس I^2C کنترل می شوند.
- کنترل نمایش LED : تولید تغذیه برای segment های یک LED که به وسیله باس I^2C کنترل می شوند.
- اهداف کلی ورودی ها و خروجی های دیجیتالی : نمایش اطلاعات "yes"یا "no" شبیه به کنترل روشن کردن LED یا خاموش کردن رله، شروع یا متوقف کردن موتور یا خواندن شماره های دیجیتالی از یک پورت.
- مبدل های آنالوگ به دیجیتال : اندازگیری میزان کمیت فیزیکی مثل (دما و فشار و ...) ، کنترل خواص ، تبدیل داده های آنالوگ به مقادیر دیجیتالی برای محاسبه.
- مبدل های دیجیتال به آنالوگ : تولید ولتاژ کنترل مخصوص برای کنترل کردن موتور DC یا شدت نور LCD.
- RAM (Random Access Memory).
- EEPROM (Electrically Erasable Programmable Read Only Memory) .
- نمایش سخت افزاری : نمایش دما و ولتاژ سیستم ها.
- میکروپرسسور ها

[72] -بخش

فصل ششم

نرم افزار

٦-١- بخش اول نرم افزار

برنامه زیر مربوط به master می باشد.

ابتدا توابع میکرو کنترلر را به کامپایلر معرفی کرده سپس سر فایل Stdio که مربوط به توابع lcd است را تعریف نموده و سر فایل بعدی مربوط به تابع تاخیر می باشد.

```
#include <mega8.h>
#include<stdio.h>
#include<delay.h>
#define  Slave1  10
#define  Slave2  20
#define  Slave3  30
```

در اینجا پورتی که lcd به آن متصل است را به کامپایلر معرفی می کنیم.که در این برنامه Lcd به پورت d متصل است.

```
#asm              //LCD port D
  .equ __lcd_port=0x12
#endasm

#include <lcd.h>
```

متغییر زیر مربوط به توابع lcd است به این فرم که کاراکتری که قرار است در Lcd نوشته شود در این متغییر ذخیره شده و سپس در Lcd نوشته می شود.

```
char data_lcd[15];
```

برنامه صفحه کلید

```
////////////////////////---  key function  ---////////////////////////
```

اعدادی که در صفحه کلید قرار است نشان داده شود در این تابع ذخیره شده است.

```
int code[4][4]={{5,1,4,7},{0,2,5,8},{14,3,6,9},{3,11,1,2}};
```

صفحه کلید به پورت B متصل می باشد. در این برنامه ابتدا چهار بیت کم ارزش این پورت که به صورت ورودی در نظر گرفته شده، را صفر و چهار بیت با ارزش را یک میکنیم.که چهار بیت با ارزش به عنوان خروجی در نظر گرفته شده است.

```
int scan_key(void){
   int  i,data,num_key=0xFF,zz=0x10;
   PORTB=0xF0;
```

در این خط برنامه چهار بیت کم ارزش یعنی ورودی از صفحه کلید به میکرو، را دریافت کرده و منتظر می ماند که کاربر کلیدی را فشار دهد. چنانچه کلیدی فشرده شود برنامه به خط بعدی رفته و در غیر این صورت منتظر فشردن کلید می ماند.

```
m1:  data=((PINB) & (0x0F));      // wait for press key
   if(data==0x00) goto m1;
```

تاخیر ۲۰ میلی ثانیه ای برای گرفتن لرزه های ناشی از صفحه کلید یا نویز استفاده می شود..به این فرم که دوباره کلید فشرده شده را از صفحه کلید می خواند و چنانچه همان کلید قبلی باشد به خط بعدی رفته و در غیر این صورت نویز می باشد.و برنامه دوباره منتظر فشردن صفحه کلید می ماند.

```
   delay_ms(20);
   PORTB=0xF0;
   data=((PINB) & (0x0F));        // for debaunce
   if(data==0x00) goto m1;
```

در این حلقه هر بار یکی از بیت های خروجی(سطر) را یک کرده و در خط بعدی ورودی صفحه کلید(ستون) را می خواند و در متغیر Data ذخیره می کند.. چنانچه کلیدی در آن ستون فشرده نشده باشد، ستون بعدی را یک کرده و این سیکل را تکرار کرده تا data با یکی از اعداد زیر برابر شود. در این صورت برنامه به تابع code رفته و بر اساس عدد که به این تابع ارجاع داده می شود، کلید فشرده شده را پیدا می کند و در متغیر num_key می ریزد.

```
   for(i=0;i<4;i++)  {
            PORTB=zz;
               data=(PINB ) & (0x0F));
               if(data==0x01)  num_key=code[0][i];
               if(data==0x02)  num_key=code[1][i];
               if(data==0x04)  num_key=code[2][i];
               if(data==0x08)  num_key=code[3][i];
            delay_ms(20);
            zz=zz<<1;    }
```

در این خطوط برنامه به منظور جلو گیری از چند بار خوانده شدن صفحه کلید، منتظر برداشتن دست از روی صفحه کلید می شود. و در پایان کلید فشرده شده را بر می گرداند.

```
m2:  PORTB=0xF0;
   data=((PINB) & (0x0F));
   if(data != 0x00) goto m2;
  return num_key;   }
```

برنامه نوشتن در Slave

```
//////////////////-- write i2c  --//////////////////
```

این تابع دو مقدار را به عنوان ورودی دریافت کرده که آدرس و دیتایی که قرار است نوشته شود، می باشد.سپس برای تعریف کردن سرعت انتقال رجیستر TWBR را مقدار دهی می کنیم.

```
void  Master_Send(unsigned int addres,unsigned int dat){
  TWBR=0x02;
```

برای فرستادن شرط شروع مقدار زیر در رجیستر کنترلی نوشته شده و برنامه منتظر پاسخ Slave به آن می ماند.

```
  TWCR=0xA4;
  while (!(TWCR & 0x80));
```

آدرس Slave را در رجیستر TWDR ریخته و آن را بر روی باس می نویسیم.و در خط بعدی منتظر پاسخ از طرف Slave می مانیم.

```
    TWDR=addres;
    TWCR=0x84;
  while (!(TWCR & 0x80));
```

سپس دیتا را در رجیستر TWDR ریخته و آن را بر روی باس می نویسیم. و منتظر پاسخ دریافت این دیتا از طرف slave می مانیم.و در نهایت شرط تووقف را ارسال می کنیم.

```
    TWDR=dat;
    TWCR=0x84;
  while (!(TWCR & 0x80));
    TWCR=0x94;
}
```

این تابع مر بوط به نوشتن دیتا در lcd می باشد. به این نحو که دو مقدار C1 و C2 را برای قرار دادن مکان نما در lcd و مقدار بعدی را به عنوان دیتا دریافت می کند.

```
//////////////////-- LCD DISPLAY DATA--///////////
 void lcd_data(int c1,int c2,int deta){
```

در این خط مکان نما را به نقطه مورد نظر انتقال داده می شود.و در خط بعدی دیتا را به فرم دسی مال در متغیر data_lcd ذخیره کرده و در نهایت آنرا بر روی Lcd می نویسد.

```
      lcd_gotoxy(c1,c2);
        print(data_lcd,"%d",deta);
      lcd_puts(data_lcd);   }
```

این زیر برنامه مربوط به تبدیل آنالوگ به دیجیتال می باشد.

```
//////////////////-- ADC --////////////////////////
 #define ADC_VREF_TYPE 0x60
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
```

```
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
```

برنامه main در این قسمت نوشته شده است که بعد از مشخص کردن پایه های میکرو به عنوان ورودی یا خروجی، دستور العمل های لازم را بر روی lcd می نویسد. سپس منتظر می ماند تا کاربر کد slave را وارد کرده و در صورت درست بودن کد، منتظر دریافت دیتا از کاربر می شود. ولی اگر چنانچه کد slave اشتباه وارد شود پس از اطلاع دادن به کاربر دوباره منتظر دریافت کد slave می شود.کد Slave اولی ۱۰ ، کد Slave دومی ۲۰ و کد Slave سومی ۳۰ می باشد.همچنین از کد ۰۰ هم برای فراخوانی عمومی استفاده شده است، یعنی اینکه تمام Slave ها این دیتا را دریافت خواهند کرد.

```
////////////////--- main program ---//////////////////
void main(void)
{
```

متغیر address مربوط به آدرس Slave ایی می باشد که کاربر آنرا وارد کرده و متغیر Data مربوط به دیتایی می باشد که در آن slave نوشته می شود.متغیر num1 و num2 مربوط به رقم یکان و دهگان آدرس دریافتی از صفحه کلید می باشد.

```
int address,data,num1,num2;
float var;
```

پورت ها را به عنوان ورودی و خروجی تعریف شده است.

```
PORTB=0xFF;
DDRB=0xF0;
PORTC=0x00;
DDRC=0xFF;
```

مر بوط به تابع آنالوگ به دیجیتال می باشد.

```
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x83;
SFIOR&=0xEF;
SFIOR|=0x10;
```

توابع Lcd به کامپایلر معرفی می شود. که این تابع باید قبل از هر تابع دیگر Lcd تعریف شود.

```
lcd_init(16);
```

در این خطوط ابتدا صفحه lcd را پاک کرده و در سطر صفر و ستون صفر جمله زیر نوشته می شود. " Enter Micro Code " و در سطر یک ستون صفر بعد از یک ثانیه جمله زیر نوشته می شود " Then Enter Data " نوشته می شود و بعد از تاخیر ۵ ثابیه ایی جهت خواندن راحتتر کاربر دوباره Lcd را پاک می کنیم.

```
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Enter Micro Code");
lcd_gotoxy(0,1);
delay_ms(1000);
lcd_putsf("Then Enter Data");
 delay_ms(5000);
 lcd_clear();
```

در ابن حلقه که بی نهایت می باشد، ابتدا مکان نما به سطر ستون صفر منتقل شده و عبارت " Micro Code is= " نوشته شده و همزمان با آن led متصل به پورت c یک می شود.سپس صفحه کلید فراخوانی شده و عددی که کاربر وارد می کند در به عنوان رقم دهگان در متغیر num2 ذخیره می شود. و در سطر صفرم ، ستون ۱۴ نمایش داده می شود.

```
while (1)
   {
try_again: lcd_gotoxy(0,0);
        lcd_putsf("Micro Code is=  ");
        PORTC.2=1;
        num2=scan_key();
        lcd_data(14,0,num2);
```

در این خطوط نیز دوباره صفحه کلید فرا خوانده شده و رقم یکان را برای آدرس slave از کاربر می خواهد. و در سطر صفر ستون ۱۵ نمایش می دهد. و برای تکمیل آدرس آنرا تبدیل به عدد دو رقمی می کند.. سپس در خط بعدی با فرستادن صفر بر روی پورت led را خاموش می کند.

```
        num1=scan_key();
        lcd_data(15,0,num1);
        address=num2*10+num1;
        PORTC.2=0;
        delay_ms(100);
```

برای اطمینان از صحت کد ادرسی که برای slave فرستاده شده، چنانچه کدی که وارد شد، آدرس هر کدام از این slave ها نباشد یا آدرس صفر که برای فراخوانی عمومی در نظر گرفته شده است، نباشد، برنامه پیغام زیر را بر روی lcd نمایش می دهد " This Address isWrong. Try again " که این پیغام به مدت ۲ ثانیه نمایش داده می شود.سپس صفحه نمایش پاک شده و منتظر وارد کردن آدرس از طرف کاربرمی ماند.

```
if(!(address==00 ||address ==slave1 || address==slave2 || address==slave3))
  { lcd_gotoxy(0,0);
lcd_putsf("This Address is ");
lcd_gotoxy(0,1);
lcd_putsf("Wrong. try again");
delay_ms(2000);
lcd_clear();
goto try_again;   }
```

سپس بعد از درست وارد کردن آدرس درست، برنامه منتظر وارد کردن دیتا می ماند و led را روشن کرده و برنامه صفحه کلید را فراخوانی می کند.بعد از فشردن صفحه کلید، led خاموش شده و بعد از تاخیر ۱۰۰ میلی ثانیه ای، عدد وارد شده را بر روی سطر یک ستون چهارده نشان می دهد.

```
lcd_gotoxy(0,1);
lcd_putsf("Data Micro is=  ");
 PORTC.2=1;
data=scan_key();
PORTC.2=0;
delay_ms(100);
 lcd_data(14,1,data);
```

چنانچه آدرسی که وارد شده، آدرس صفر باشد، آنگاه دیتای دریافتی را برای تمام slave ها چاپ می کند.

```
 if(address!=0)
      Master_Send(address,data);
    else   {
      Master_Send(0x0A,data);
      delay_ms(50);
      Master_Send(0x14,data);
      delay_ms(50);
      Master_Send(0x1E,data);
      }
 delay_ms(1000);
};
}
```

##

برنامه زیر مربوط به اولین slave می باشد.آدرس این slave ، ۱۰ یا همان 0x0A می باشد. در این برنامه slave از طریق وقفه منتظر آدرس دهی master می ماند. و بعد از دریافت داده آن را بر روی پورت D میکرو نمایش می دهد.

```
#include <mega8.h>
```

آدرس slave اول برابر ۱۰ می باشد، سپس متغیر Receive_Data که به صورت سرتاسری تعریف شده است.

```
#define  Slave1  10
unsigned int Receive_Data;
```

در برنامه وقفه زیر ابتدا منتظر می شود که پرچم TWINT یک شود که نشان دهنده اتمام ارسال است. در خط بعدی پالس تصدیق را روی گذرگاه ایجاد می کند. و منتظر یک شدنپرچم TWINT می ماند.

```
interrupt [TWI] void twi_isr(void)
{
  while (!(TWCR & 0x80));
    TWCR=0xc4;
  while (! (TWCR & 0x80));
```

سپس دیتای دریافتی از master را که در رجیستر TWDR قرار دارد را در متغیر Receive_Data قرارمی دهد. و بر روی پورت D نمایش می دهد.در خط بعدی پالس تصدیق دریافت دیتا را بر روی باس قرار می دهد.و بعد از مقدار دهی دوباره رجیستر های مربوطه از برنامه وقفه خارج می شود.

```
    Receive_Data=TWDR ;
     PORTD=Receive_Data;
    TWCR=0xc4;
  while (! (TWCR & 0x80));
    TWCR=0x94;
    TWAR=slave1;
    TWCR=0x45;
}
```

در برنامه اصلی ابتدا پورت D را به عنوان خروجی در نظر گرفته و در خط بعدی وقفه سراسری را فعال کرده سپس رجیستر TWBR را برای سرعت انتقال فعال کرده و رجیستر TWAR را برای آدرس slave تایین کرده و برنامه وارد حلقه بی نهایت می شود.

```
void  main()
{
  DDRB=0xFF;

  #asm ("sei");
  DDRD=0xFF;
  TWBR=0x02;
  TWAR=slave1;
```

```
  TWCR=0x45;

  while (1);
}
```

##

برنامه زیر مربوط به دومین slave می باشد.آدرس این slave ، ۲۰ یا همان 0x14 می باشد. در این برنامه slave از طریق وقفه منتظر آدرس دهی master می ماند. و بعد از دریافت داده آن را بر روی پورت D میکرو نمایش می دهد.

```
#include <mega8.h>
```

آدرس slave دوم برابر ۲۰ می باشد، سپس متغیر Receive_Data که به صورت سرتاسری تعریف شده است.

```
#define  Slave2  20
unsigned int Receive_Data;
```

در برنامه وقفه زیر ابتدا منتظر می شود که پرچم TWINT یک شود که نشان دهنده اتمام ارسال است. در خط بعدی پالس تصدیق را روی گذرگاه ایجاد می کند. و منتظر یک شدنپرچم TWINT می ماند.

```
interrupt [TWI] void twi_isr(void)
{
  while (!(TWCR & 0x80));
    TWCR=0xc4;
  while (! (TWCR & 0x80));
```

سپس دیتای دریافتی از master را که در رجیستر TWDR قرار دارد را در متغیر Receive_Data قرارمی دهد. و بر روی پورت D نمایش می دهد.در خط بعدی پالس تصدیق دریافت دیتا را بر روی باس قرار می دهد.و بعد از مقدار دهی دوباره رجیستر های مربوطه از برنامه وقفه خارج می شود.

```
    Receive_Data=TWDR ;
     PORTD=Receive_Data;
    TWCR=0xc4;
  while (! (TWCR & 0x80));
    TWCR=0x94;
    TWAR=slave2;
    TWCR=0x45;
}
```

در برنامه اصلی ابتدا پورت D را به عنوان خروجی در نظر گرفته و در خط بعدی وقفه سراسری را فعال کرده سپس رجیستر TWBR را برای سرعت انتقال فعال کرده و رجیستر TWAR را برای آدرس slave تایین کرده و برنامه وارد حلقه بی نهایت می شود.

```
void  main()
{
  DDRB=0xFF;

  #asm ("sei");
  DDRD=0xFF;
  TWBR=0x02;
  TWAR=slave2;
  TWCR=0x45;

  while (1);
}
```

##

برنامه زیر مربوط به سومین slave می باشد.آدرس این slave ، ۳۰ یا همان 0x1E می باشد. در این برنامه slave از طریق وقفه منتظر آدرس دهی master می ماند. و بعد از دریافت داده آن را بر روی پورت D میکرو نمایش می دهد.

```
#include <mega8.h>
```

آدرس slave سوم برابر 30 می باشد، سپس متغیر Receive_Data که به صورت سرتاسری تعریف شده است.

```
#define  Slave3 30
unsigned int Receive_Data;
```

در برنامه وقفه زیر ابتدا منتظر می شود که پرچم TWINT یک شود که نشان دهنده اتمام ارسال است. در خط بعدی پالس تصدیق را روی گذرگاه ایجاد می کند. و منتظر یک شدنپرچم TWINT می ماند.

```
interrupt [TWI] void twi_isr(void)
{
  while (!(TWCR & 0x80));
    TWCR=0xc4;
  while (! (TWCR & 0x80));
```

سپس دیتای دریافتی از master را که در رجیستر TWDR قرار دارد را در متغیر Receive_Data قرارمی دهد. و بر روی پورت D نمایش می دهد.در خط بعدی پالس تصدیق دریافت دیتا را بر روی باس قرار می دهد.و بعد از مقدار دهی دوباره رجیستر های مربوطه از برنامه وقفه خارج می شود.

```
    Receive_Data=TWDR ;
     PORTD=Receive_Data;
    TWCR=0xc4;
  while (! (TWCR & 0x80));
    TWCR=0x94;
    TWAR=slave3;
    TWCR=0x45;
}
```

در برنامه اصلی ابتدا پورت D را به عنوان خروجی در نظر گرفته و در خط بعدی وقفه سراسری را فعال کرده سپس رجیستر TWBR را برای سرعت انتقال فعال کرده و رجیستر TWAR را برای آدرس slave تایین کرده و برنامه وارد حلقه بی نهایت می شود.

```
void  main()
{
  DDRB=0xFF;

  #asm ("sei");
  DDRD=0xFF;
  TWBR=0x02;
  TWAR=slave3;
  TWCR=0x45;

  while (1);
}
```

بخش دوم

برنامه های زیر را هم می توان در میکروها قرار داد. ولی این برنامه ها زمانی درست کار می کنند که تعداد master و Salve ها یکی باشند. چنانچه تعداد slave ها بیش از یکی شود، آنگاه کدی که از master به Slave فرستاده می شود، همواره درست نمی باشد. علت این امر استفاده از توابع i2c می باشد که به صورت نرم افزار با این پروتکل ارتباط برقرار می کند. که عیب آن این می باشد که با میکروکنترلر به عنوان master ، و با بقیه ابزارهای جانبی به عنوان Slave رفتار می کند. ولی در برنامه هایی که در چند صفحه قبل نوشته شد، به صورت سخت افزرای با پروتکل i2c برخورد کرده و این مشکل را ندارد.

تفاوت این دو برنامه در قسمت مربوط به توابع i2c می باشد.و دیگر توابع استفاده شده در این دو برنامه یکسان می باشند.

برنامه زیر مربوط به master بوده آدرس های مربوط به slave و اطلاعاتی که باید در آن slave قرار گیرد را از کاربر دریافت می کند .

```
#include <mega8.h>
#include<stdio.h>
#include<delay.h>
#asm                                // i2c conect port C
  .equ __i2c_port=0x15
  .equ __sda_bit=0
  .equ __scl_bit=1
#endasm
#include <i2c.h>
#asm            //LCD port D
  .equ __lcd_port=0x12
#endasm
#include <lcd.h>
char data_lcd[15];                          // key conect port B
 ////////////////////////---  key function  ---/////////////////////////
int code[4][4]={{1,2,3,10},{4,5,6,11},{7,8,9,12},{13,0,14,15}};
int scan_key(void){
    int i,data,num_key=0xFF,zz=0x10;
    PORTB=0xF0;
m1:  data=((PINB) & (0x0F));     // wait for press key
    if(data==0x00) goto m1;
    delay_ms(20);
    PORTB=0xF0;
    data=((PINB) & (0x0F));        // for debaunce
    if(data==0x00) goto m1;
```

```
    for(i=0;i<4;i++)  {
                PORTB=zz;
                    data=((PINB ) & (0x0F));
                    if(data==0x01)  num_key=code[0][i];
                    if(data==0x02)  num_key=code[1][i];
                    if(data==0x04)  num_key=code[2][i];
                    if(data==0x08)  num_key=code[3][i];
                delay_ms(20);
                zz=zz<<1;     }
m2:  PORTB=0xF0;
    data=((PINB) & (0x0F));
    if(data != 0x00) goto m2;
  return num_key;   }
 ////////////////////-- write i2c  --////////////////////
 void write(unsigned int address,unsigned int data)
{     int ak;
      i2c_start();
      do{ak= i2c_write(address); }while(ak != 1);
       i2c_write(data);
       i2c_stop();
       delay_ms(10);    }
 ////////////////////-- LCD DISPLAY DATA--////////////
  void lcd_data(int c1,int c2,int deta){
          lcd_gotoxy(c1,c2);
          sprintf(data_lcd,"%d",deta);
          lcd_puts(data_lcd); }
 ////////////////////-- ADC --////////////////////////
 #define ADC_VREF_TYPE 0x60
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
 ////////////////--- main program ---////////////////////
void main(void)
{
```

```
int address,data,num1,num2;
float var;
PORTB=0xFF;
DDRB=0xF0;
PORTC=0x00;
DDRC=0xFF;
// 2 Wire Bus initialization
// Generate Acknowledge Pulse: On
// 2 Wire Bus Slave Address: 11h
// General Call Recognition: On
// Bit Rate: 50.000 kHz
TWBR=0x0B;
TWAR=0x23;
TWCR=0x44;


ADMUX=ADC_VREF_TYPE;
ADCSRA=0x83;
SFIOR&=0xEF;
SFIOR|=0x10;


i2c_init();
lcd_init(16);
 write(22,03);
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf("Enter Micro Code");
   lcd_gotoxy( 0,1);
   delay_ms(1000);
   lcd_putsf("Then Enter Data");
    delay_ms(5000);
    lcd_clear();
while (1)
     {   //  var=read_adc(4);
        //  var=var*5;
          lcd_gotoxy(0,0);
          lcd_putsf("Micro Code is=  ");
          PORTC.2=1;
          num2=scan_key();
          lcd_data(14,0,num2);
          num1=scan_key();
```

```
        lcd_data(15,0,num1);
        address=num2*10+num1;
        PORTC.2=0;
        delay_ms(100);


        lcd_gotoxy(0,1);
        lcd_putsf("Data Micro is=  ");
         PORTC.2=1;
        data=scan_key();
        PORTC.2=0;
        delay_ms(100);
         lcd_data(14,1,data);
         write(address,data);
        write(address,data);
         delay_ms(1000);
    };
}
```

برنامه زیر مربوط به اولین slave می باشد. آدرس این Slave برابر ۱۰ می باشد.

```
#include <mega8.h>
#include<delay.h>


#asm
  .equ __i2c_port=0x15
  .equ __sda_bit=0
  .equ __scl_bit=1
#endasm
#include <i2c.h>


void main(void)
{
int data,address;
PORTD=0x00;
DDRD=0xFF;
PORTB=0x00;
DDRB=0xFF;
i2c_init();
```

```
 while (1)  {
  do{address=i2c_read(1);}while(address!=۱۰);
    data=i2c_read(0);
    PORTD=data;
       };
}
```

برنامه زیر مربوط به دومین slave می باشد.آدرس این Slave ، ۲۰ می باشد

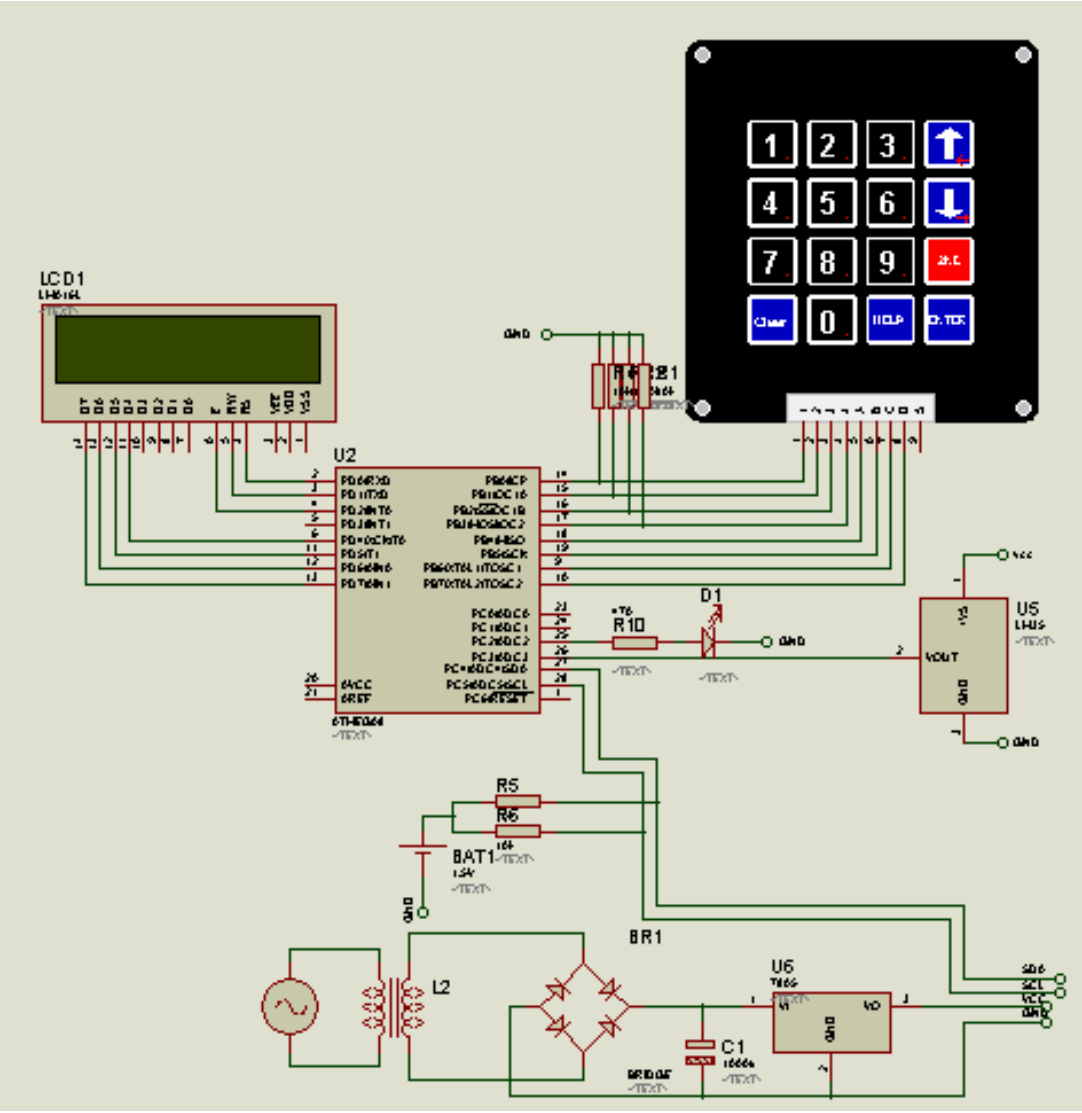
```
#include <mega8.h>
#include<delay.h>

#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=0
   .equ __scl_bit=1
#endasm
#include <i2c.h>

void main(void)
{
int data,address;
PORTD=0x00;
DDRD=0xFF;
PORTB=0x00;
DDRB=0xFF;
i2c_init();
 while (1)  {
  do{address=i2c_read(1);}while(address!=۲۰);
    data=i2c_read(0);
    PORTD=data;
       };
}
```

برنامه زیر مربوط به سومین slave می باشد.آدرس این slave ، ۳۰ می باشد.

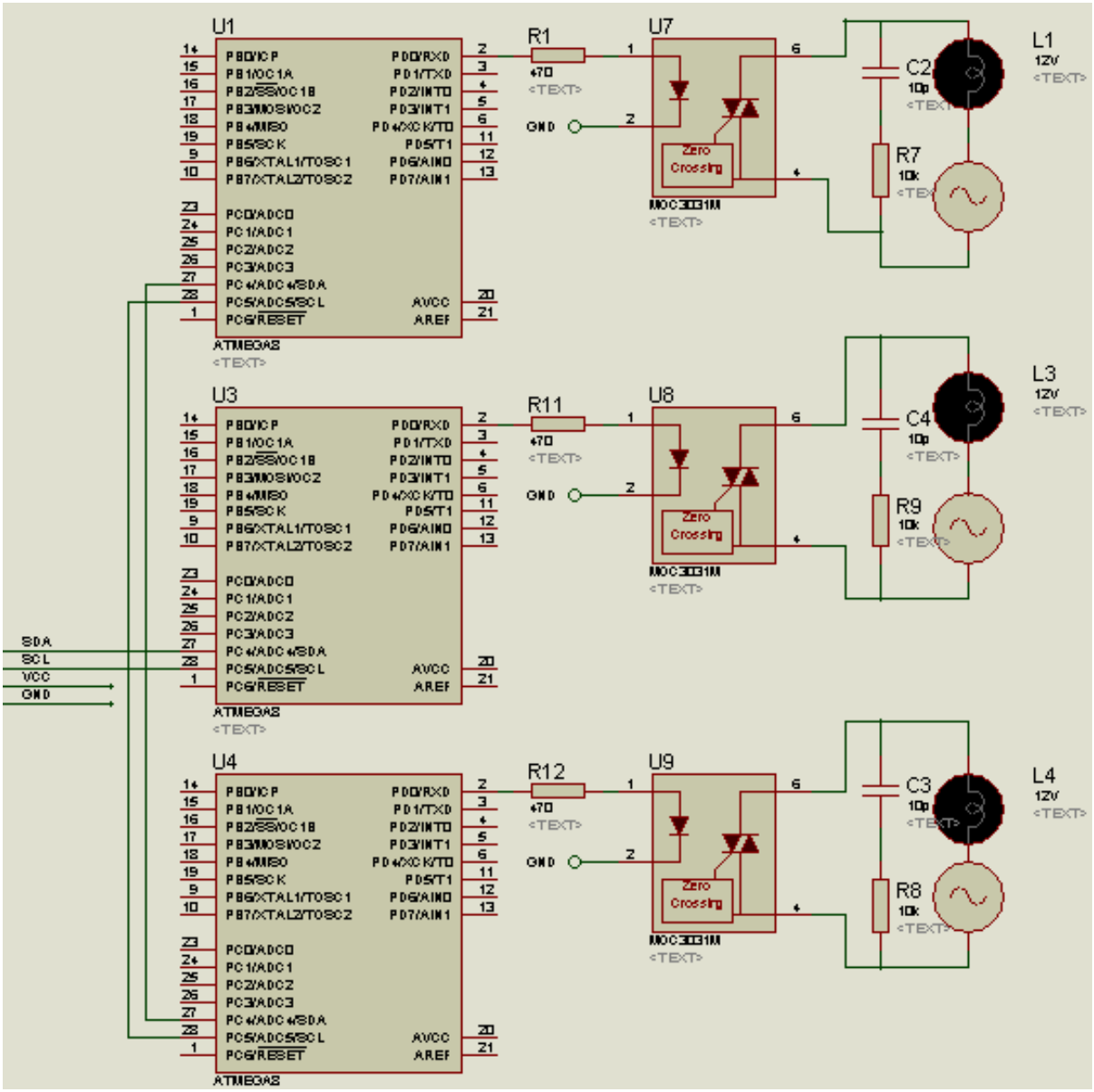
```
#include <mega8.h>
```

```
#include<delay.h>


#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=0
   .equ __scl_bit=1
#endasm
#include <i2c.h>


void main(void)
{
int data,address;
PORTD=0x00;
DDRD=0xFF;
PORTB=0x00;
DDRB=0xFF;
i2c_init();
 while (1)  {
  do{address=i2c_read(1);}while(address!=٣٠);
    data=i2c_read(0);
    PORTD=data;
       };
}
```

شماتیک مدار



شکل ٦-١- مدار MASTER



شکل ٦-٢- مدار تمام Slave ها

پیشنهاد

یکی از کاربردهایی که می توان با این پروتکل انجام داد این می باشد که سیستمی متشکل از چند SLAVE ویک master را ایجاد نماییم که هدف آن مانیتور کردن یک پروسه می باشد.. به عنوان مثال برای اینکه دما، رطوبت و فشار چند اتاق را همزمان کنترل کنیم، میتوانیم در هر اتاقی یک slave قرار داده که کارشان ثبت دما ، فشار و رطوبت می باشد. و master در زمان های مشخص از هر کدام از Slave ها گزارش کارشان را دریافت نماید و بعد از پردازش آن، عکس الملل های لازم(دستور هر کدام از slave ها) را انجام دهد.

کار بعدی که می توان با این شبکه انجام داد استفاده از این پروتکل در مونیتورینگ قلب می باشد. به این نحو که هر کدام از slave ها با مدار جانبی در کنار یک بیمار قلبی قرار می گیرد به این فرم که سیگنال قلب توسط میکروفنی که به قلب بیمار متصل است، به یک تقویت کننده متصل شده و توسط ADC میکرو کنترلر از آنالوگ به دیجیتال تبدیل شده و توسط Slave در زمان بندی مشخصی که توسط master معین می شود، از slave به master منتقل می شود. و master هم آنرا می تواندتوسط پورت موازی، سری یا USB به کامپیوتر انتقال داده و آنرا به عنوان یک پیکسل دریافت کرده و این پیکسل ها وقتی در کنار هم قرار گیرند شکل سیگنال قلبی هر بیمار بر روی صفحه مانیتور نشان داده شده و پردازش های لازمراروی ان انجام دهد. کار بعدی دیگری که می توان روی آن انجام داد، استفادهاز این پروتکل به عنوان PLC می باشد که یک سری ورودی ها و خروجی هایی دارد که master همان نقش PLC را ایفا کرده و هر کدام از این Slaveها می توانند به عنوان ورودی ها و خروجی هایی که در هر لحظه اطلاعات را به پروسه مرکزی(master) انتقال می دهند

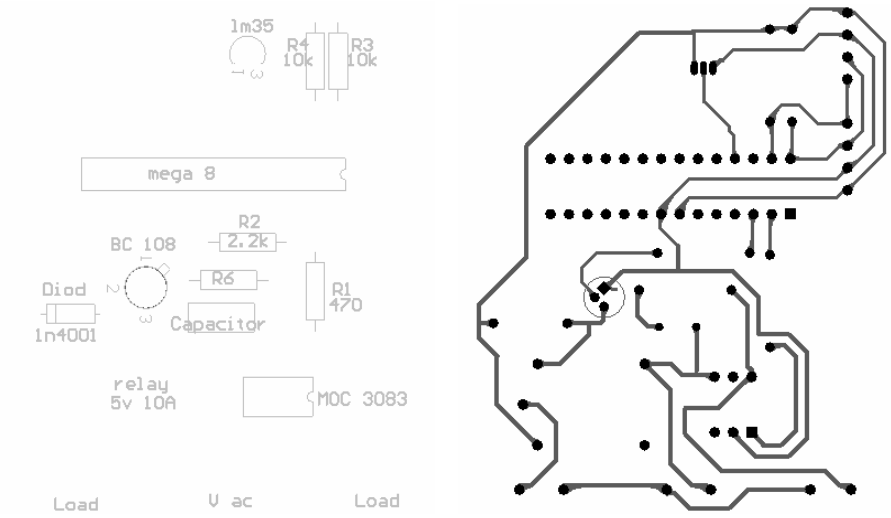
نتیجه گیری

این پروژه به فرم عملی برای کنترل دستگاه ها مناسب بوده و دستگاه های جانبی بدون اینکه نیاز به قطع کردن کل دستگاه برای وصل کردن وسیله جدیدی باشد، می تواند مورد استفاده قرار گیرد.. کار با این پروتکل به دلیل وجود تابع های از قبل طراحی شده توسط طراح، ساده بوده و به دلیل دو سیمه بودن این پروتکل مدار های PCB آن نیز کوچک می باشد. به دلیل کوچک شدن PCB و کم بودن مدارات جانبی، هزینه این

مراجع

[1]. Philips Semiconductors
http://www.semiconductors.philips.com Accessed : March 2006
[2]. Philips Semiconductors
http://www.semiconductors.philips.com/acrobat_download/literature/9398/3934001
1.pdf
[3]. http://www.i2c-bus.org/highspeed/Accessed : March 2006
[4]. THE I2C-BUS SPECIFICATION - VERSION 2.1 - JANUARY 2000
[5]. I2C BUS – Quarndon Electronics Ltd. – www.quarndon.co.uk
[6]. Westermo Handbook – Industrial data communication – Edition 3.0 – westermo teleindustri AB, Sweden
[7]. And the other Internet based references, papers, eBooks , catalogs and articles.

شکل Pcb مربوط به Slave ها



شکل Pcb مر بوط به master