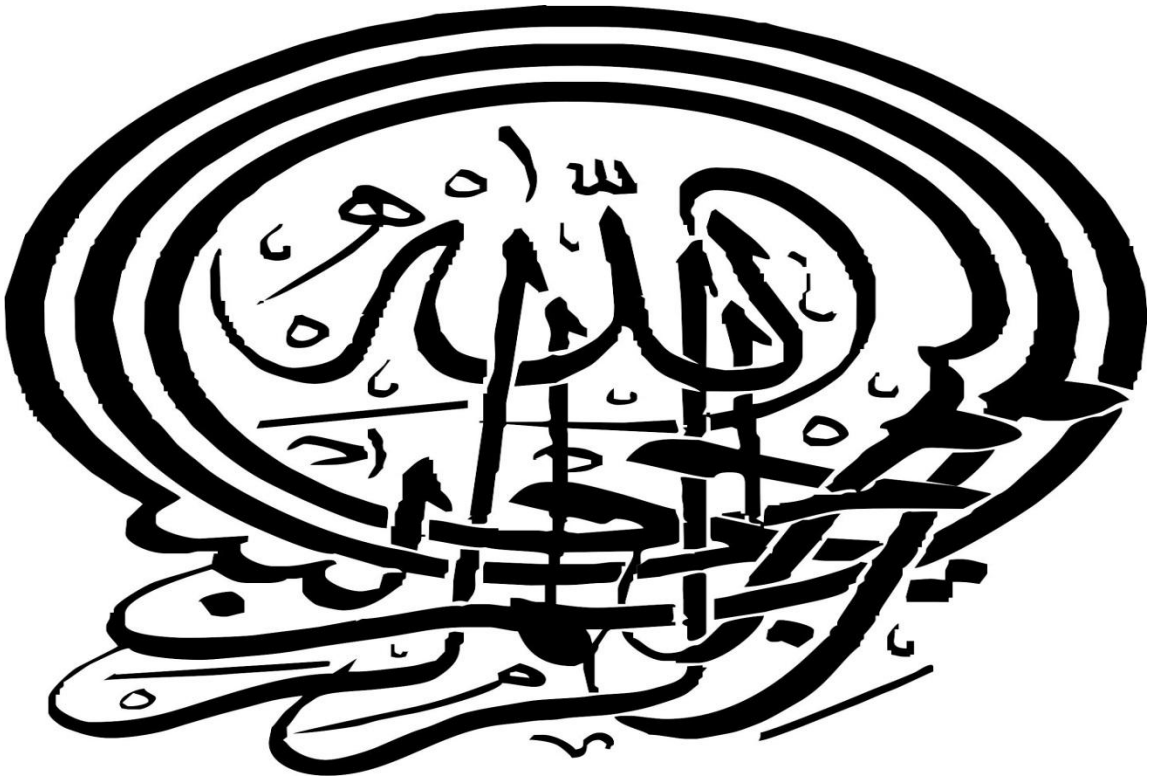
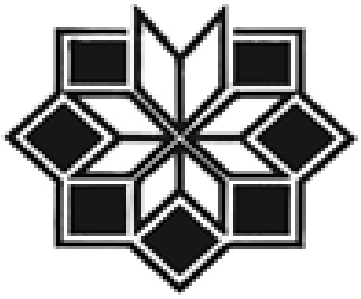


دانشگاه فنی و حرفه ای

دانشکده انقلاب اسلامی تهران



پایان نامه ی پروژه ی دانشجویی

عنوان پروژه : music player

نام دانشجویان :

رشته /مقطع : الکترونیک/ کاردانی پیوسته

استاد راهنما : مهندس صموتی

مدیر پروژه : مهندس زارعی

91_92

مقدمه :

در این پروژه از میکروکنترلر ATMEGA32 که از جمله میکروکنترلر های خانواده ی AVR است استفاده شده است . در زیر مطالبی جهت آشنایی با خانواده ی AVR و به خصوص ATMEGA32 آمده است .

ساده ترین معماري ميكرو كنترلر ، متشكل از يك ريز پردازنده ، حافظه و درگاه ورودي / خروجي است .

ریز پردازنده نیز متشکل از واحد پردازش مرکز (CPU) و واحد کنترل (CU) است .

CPU درواقع مغز يك ريز پردازنده است و محلي است كه در آنجا تمام عمليات رياضي و منطقي ، انجام مي شود . واحد كنترل ، عمليات داخلي ريز پردازنده را كنترل مي كند و سيگنال هاي كنترلي را به ساير بخشهاي ريز پردازنده ارسال مي كند تا دستورالعمل ها ي مورد نظر انجام شوند .

حافظه بخش خيلي مهم از يك سيستم ميكرو كامپيوتري است. ما مي توانيم بر اساس به كارگيري حافظه، آن را به دو گروه دسته بندي كنيم: حافظه برنامه و حافظه داده . حافظه برنامه ، تمام كد برنامه را ذخيره مي كند . اين حافظه معمولاً از نوع حافظه فقط خواندني (ROM) مي باشد . انواع ديگري از حافظه ها نظير EPROM و حافظه هاي فلش EEPROM براي كاربردهايي كه حجم توليد پاييني دارند و همچنين هنگام پياده سازي برنامه به كار مي روند . حافظه داده از نوع حافظه خواندن / نوشتن (RAM) مي باشد . در كاربردهاي پيچيده كه به حجم بالايي از حافظه RAM نياز داريم ، امكان اضافه كردن تراشه هاي حافظه بيروني به اغلب ميكرو كنترلر ها وجود دارد .

درگاهها ورودي / خروجي (I/O) به سيگنال هاي ديجيتال بيروني امكان مي دهند كه با ميكرو كنترلر ارتباط پيدا كند . درگاههاي I/O معمولاً به صورت گروههاي 8 بيتي دسته بندي مي شوند و به هر گروه نيز نام خاصي اطلاق مي شود . به عنوان مثال ، ميكروكنترلر 8051 داراي 4 درگاه ورودي / خروجي 8 بيت مي باشد كه P3, P2, P1, P0 ناميده مي شوند . در تعدادي از ميكرو كنترلر ها ، جهت خطوط درگاه I/O قابل برنامه ريزي مي باشد . لذا بيت هاي مختلف يك درگاه را مي توان به صورت ورودي يا خروجي برنامه ريزي نمود . در برخي ديگر از ميكروكنترلرها (از جمله ميكروكنترلرهاي 8051) درگاههاي I/O به صورت دو طرفه مي باشند . هر خط از درگاه I/O اين گونه ميكرو كنترلرها را مي توان به صورت ورودي و يا خروجي مورد استفاده قرار داد . معمولاً ، اين گونه خطوط خروجي ، به همراه مقاومتهاي بالا كش بيروني به كار برده مي شوند .

ميكروهاي AVR داراي انعطاف پذيري غير قابل مقايسه و بي همتايي هستند.آنها قادر به تركيب هر نوع كدي با يك معماري كارامد از طريق زبانهاي C و Assembly هستند و قادرند از طريق اين برنامه ها تمام پارامترهاي ممكن در يك سيكل يا چرخه ماشين را با دقت بسيار بالا هماهنگ كنند . ميكرو AVR داراي معماريي است كه ميتواند در تمام جهات مورد استفاده شما ، عمل كند ميكرو AVR معماريي دارد كه براي شما كارايي 16 بيتي ارائه مي دهد كه البته قيمتش به اندازه يك 8 بيتي تمام مي شود .

ميكرو كنترلر AVR به منظور اجراي دستورالعملهاي قدرتمند در يک سيكل كلاک(ساعت) به اندازه كافي سريع است و مي تواند براي شما آزادي عملي را كه احتياج داريد به منظور بهينه سازي توان مصرفي فراهم كند.

بهره هاي كليدي AVR :

1) داراي بهترين MCU براي حافظه فلش در جهان ! (MCU: **M**aster **C**ontrol **U**nit)

2) داراي سيستمي با بهترين هماهنگي

3) داراي بالاترين كارايي و اجرا در CPU (يك دستورالعمل در هر سيكل كلاك)

4) داراي كدهايي با كوچكترين سايز

5) داراي حافظه خود برنامه ريز

6) داراي واسطه JTAG كه با IEEE 1149.1 سازگار است

(IEEE: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers.)

1) داراي سخت افزار ضرب كننده روي خود

2) داراي بهترين ابزارها براي پيشرفت و ترقي

3) داراي حالات زيادي براي ترفيع دادن يا Upgrade .

واژگان كليدي AVR :

ميكروكنترلر AVR بر مبناي معماري RISC (كاهش مجموعه ي دستورالعملهاي كامپيوتر) پايه گذاري شده و مجموعه اي از دستورالعملها را كه با 32 ثبات كار ميكنند تركيب مي كند .

به كارگرفتن حافظه از نوع Flash كه AVR ها به طور يكسان از آن بهره مي برند از جمله مزاياي آنها است .

يك ميكرو AVR مي تواند با استفاده از يك منبع تغذيه 2.7 تا 5.5 ولتي از طريق شش پين ساده در عرض چند ثانيه برنامه ريزي شود يا Program شود .

ميكروهاي AVR در هرجا كه باشند با 1.8 ولت تا 5.5 ولت تغذيه مي شوند البته با انواع توان پايين (LowPower) كه موجودند .

خانواده هاي محصولات AVR :

Tiny AVR :

ميكروكنترلري با اهداف كلي و با بيش از 4 كيلو بايت حافظه فلش و 128 بايت حافظه استاتيك و قابل برنامه ريزي است . (منظور از حافظه استاتيك SRAM و حافظه قابل برنامه ريزي EEPROM است) .

نكات كليدي و سودمند مدل Tiny :

1) آنها به منظور انجام يك عمليات ساده بهينه سازي شده و در ساخت وسايلي كه به ميكرو هاي كوچك احتياج است كاربرد فراوان دارند .
2) كارايي عظيم آنها براي ارزش و بهاي وسايل موثر است .

Mega AVR :

اين نوع ميكروها قابليت خود برنامه ريزي دارند و مي توان آنها را بدون استفاده از مدارات اضافي برنامه ريزي كرد

همچنين بيش از 256K بايت حافظه فلش و 4K بايت حافظه استاتيك و قابل برنامه ريزي دارند .

نكات كليدي و سودمند مدل Mega :

1) حافظه سريع از نوع فلش با عملكرد خود برنامه ريز و بلوكه ي بوت (Boot Block)
2) دقت بسيار بالاي 8-كانال در تبديل آنالوگ به ديجيتال 10 بيتي
3) USART و SPI و TWI بر طبق واسطه هاي سريال
4) واسطه ي JTAG بر طبق IEEE 1149.1

Interface is a byte oriented interface **W**ire **T**wo : TWI

USART: **U**niversal **S**erial **A**synchronous **R**eceiver/**T**ransmitter

SPI: **S**erial **P**eripheral **I**nterface

JTAG available only on devices with 16KB Flash and up

واسط JTAG فقط در ميكروهاي با بيش از 16 كيلوبايت حافظه فلش موجود است .

: LCD AVR

اين نوع ميكرو داراي درايور براي نمايشگر LCD با قابليت كنترل اتوماتيك تباين و مقايسه تصوير مي باشد . باعث تمديد عمر باتري مي شود و در حالت فعال داراي توان مصرفي پايييني است .

اين خانواده از AVR ها با بالاترين يكپارچگي و انعطاف پذيري ممكن طراحي شده اند و با داشتن درايور LCD و كنترلر اتوماتيك

وضوح تصوير ،بهترين واسطه را با انسان دارند و داراي توان مصرفي پايين و كارايي بالايي هستند.اولين عضو اين خانواده 100 سگمنت داشت و داراي يك UART و SPI به منظور ارتباط به صورت سريال بود .

نكات كليدي وسودمند مدل LCD :

1) كارايي فوق العاده با سرعت يك ميليون دستورالعمل در ثانيه به ازاي يك مگاهرتز
2) واسطه ها براي ارتباط با انسان: وقفه هاي صفحه كليد و درايور نمايشگر LCD
3) آنها اين اجازه را به طراح سيستم مي دهند كه توان مصرفي را در برابر سرعت پردازش تا جايي كه امكان دارد بهينه كند .

مفهوم توان مصرفي پايين در LCD AVR :

توان مصرفي پايين آنها براي استفاده بهينه از باتري و همچنين كاربرد ميكرو در وسايل سيار و سفري طراحي شده كه ميكروهاي جديد AVR با توان مصرفي كم از شش روش اضافي در مقدار توان مصرفي ، براي انجام عمليات بهره مي برند .

اين ميكروها تا مقدار 1.8 ولت قابل تغذيه هستند كه اين امر باعث طولاني تر شدن عمر باتري مي شود

در ميكروهاي با توان پايين ، عمليات شبيه حالت Standby است يعني ميكرو مي تواند تمام اعمال

داخلي و جنبي را متوقف كند و كريستال خارجي را به همان وضعيت شش كلاك در هر چرخه رها كند

نكات كليدي و سودمند حافظه ي فلش خود برنامه ريز :

1) قابليت دوباره برنامه ريزي كردن بدون احتياج به اجزاي خارجي
2) 128 بايت كوچك كه به صورت فلش سكتور بندي شده اند
3) داشتن مقدار متغير در سايز بلوكه ي بوت (Boot Block)
4) خواندن به هنگام نوشتن
5) بسيار آسان براي استفاده
6) كاهش يافتن زمان برنامه ريزي
7) كنترل كردن برنامه ريزي به صورت سخت افزاري

ISP در ميكروكنترلرها :

1) واسطه سه سيمي محلي براي بروزرساني سريع
2) آسان و موثر در استفاده

واسطه JTAG در ميكروكنترلرها :

1) واسطه اي كه تسليم قانون IEEE 1149.1 است و مي تواند به صورت NVM برنامه ريزي كند يعني هنگام قطع جريان برق داده ها از بين نروند.استفاده از فيوزبيتها و بيتهاي قفل .

2) بیشتر براي دیباگ کردن آنچیپ و به منظور تست استفاده مي شود .

فیوزبیت ها در میکروکنترلر :

این ها یک سری بیت کلی هستنند که منطق 0 به معنای برنامه ریزی شدن و منطق 1به معنای برنامه ریزی نشدن بیت است .

اساسا کار این بیت ها تغییر مشخصات سخت افزاری میکروکنترلر است و هر مدل از میکروکنترلر به نسبت طراحی دارای فیوز بیت های مختلفی است .
به فرض مثال اگر فیوز بیت به نام EESAVE برنامه ریزی شود محتویات حافظه EEPROM در زمان پاک کردن میکروکنترلر محفوظ میماند ولی اگر این بیت برنامه ریزی نشود حافظه EEPROM در زمان پاک شدن میکرو پاک میشود .

میکروکنترلری که در این پروژه به کار رفته است ATMEGA32 می باشد بنابر این از این به بعد سعی می شود در مورد این میکرو به طور خاص صحبت شود .

نمای ظاهری این میکروکنترلر در شکل زیر آمده است :

مشخصات سخت افزاری

ATMEGA32 :

شکل ظاهری و پایه ها :

ATMEGA32 در سه نوع بسته بندی PDIP با 40 پایه و TQFP با 44پایه و MLF با 44 پایه ساخته میشود که در بازار ایران بیشتر نوع PDIP موجود میباشد .

ATMRGA32 دارای چهار پورت 8بیتی (1 بایتی) دارد که علاوه بر اینکه بعنوان یک پورت معمولی میتوانند باشند کارهای دیگری نیز انجام میدهند . بطور مثال PORTA میتواند بعنوان ورودی ADC (تبدیل ولتاژ آنالوگ به کد دیجیتال) استفاده شود که این خاصیت های مختلف پورت در برنامه ای که نوشته می شود تعیین خواهد شد .
ولتاژ مصرفی این آی سی از4.5V تا 5.5V میتواند باشد .

فرکانس کار هم تا 16MHz میتواند انتخاب شود که تا 8MHz نیازی به کریستال خارجی نیست و در داخل خود آی سی میتواند تامین شود . فرکانس کار از جمله مواردی است که باید در برنامه تعیین شود . لازم به ذکر است که این فرکانس بدون هیچ تقسیمی به CPU داده میشود . بنابراین این خانواده از میکروکنترلرها سرعت بیشتری نسبت خانواده های دیگر دارند .

برنامه ای که برای میکروکنترلر در کامپیوتر نوشته میشود وقتی که برای استفاده در آی سی ریخته میشود (توسط پروگرامر مخصوص آن خانواده) در مکانی از آن آی سی ذخیره خواهد شد بنام ROM . حال در ATMEGA32 مقدار این حافظه به (32 کیلوبایت) می رسد .

در این آی سی مکانی برای ذخیره موقت اطلاعات یا همان RAM هم وجود دارد که مقدارش 2KB است.در RAM اطلاعات فقط تا زمانی که انرژی الکتریکی موجود باشد خواهد ماند و با قطع باتری اطلاعات از دست خواهند رفت . به همین منظور در ATMEGA32 مکانی برای ذخیره اطلاعات وجود دارد که با قطع انرژی از دست نخواهند رفت . به این نوع حافظه ها EEPROM گفته میشود که در این آی سی مقدارش 1KB است و تا 100,000 بار میتواند پر و خالی شود .

مهمترین مشخضات این میکروکنترلر 40 پایه عبارت است از :

1) کارایی بالا و توان مصرفی کم
2) 32رجیستر (ثبات) 8 بیتی
3) سرعت با سقف 16 میلیون دستور در ثانیه درفرکانس16 Mhz
4) 32کیلو بایت حافظه FLASH داخلی قابل برنامه ریزی با قابلیت ده هزار بار نوشتن و پاککردن
5) 2کیلو بایت حافظه داخلیSRAM
6) قابلیت ارتباط JTAG
7) دو تایمر/ شمارنده هشت بیتی
8) یک تایمر/شمارنده شانزده بیتی
9) چهار کانالPWM
10) هشت کانال مبدل A/D ده بیتی
11) یک مقایسه کننده آنالوگ داخلی
12) WATCHDOG قابل برنامه ریزی با اسیلاتور داخلی
13) ارتباط سریال برای برنامه ریزیISP :

USARTسریال قابل برنامه ریزی

1) دارای شش حالت SLEEP
2) منابع وقفه داخلی و خارجی
3) اسیلاتور داخلی RC
4) کار با ولتاژ 4.5 تا 5.5
5) فرکانس کاری 0 تا 16 مگاهرتز
6) 32خط داده ورودی و خروجی قابل برنامه ریزی
7) عملکرد کاملا ثابت
8) 1024بایت حافظه EEPROM داخلی قابل برنامه ریزی با قابلیت صد هزار بار نوشتن و خواندن

فیوز بیت های ATMEGA32 :

ATMEGA32 دارای دو بایت فیوز بیت می باشد .

OCDEN : در صورتی که بیت های قفل برنامه ریزی نشده باشند ، برنامه ریزی این بیت به همراه بیت JTAGEN باعث می شود که سیستم ON CHIP DEBUG فعال شود . برنامه ریزی شدن این بیت به قسمت هایی از میکرو اجازه می دهد که در مدهای SLEEP کار کنند که این خود باعث افزایش مصرف سیستم می شود . این بیت به صورت پیش فرض برنامه ریزی نشده (1) است .

JTAGEN : بیتی برای فعال سازی برنامه ریزی میکرو از طریق استاندارد ارتباطی IEEE (JTAG) که در حالت پیش فرض فعال است و میکرو می تواند از این ارتباط برای برنامه ریزی خود استفاده نماید . پایه های PC2 ..... 5 در این ارتباط استفاده می شوند .

SPIEN : در حالت پیش فرض برنامه ریزی شده و میکرو از طریق سریال SPI برنامه ریزی می شود .

CKOPT : انتخاب کلاک که به صورت پیش فرض برنامه ریزی نشده است را بر عهده دارد . عملکرد این بیت بستگی به بیت های CKSEL دارد .

EESAVE : در حالت پیش فرض برنامه ریزی نشده و در زمان پاک شدن (ERASE) میکرو حافظه EEPROM پاک می شود ولی در صورتی که برنامه ریزی شود ، محتویات EEPROM در زمان پاک شدن میکروکنترلر محفوظ می ماند .

BOOTSZ0,1 : برای انتخاب مقدار حافظه ی BOOT برنامه ریزی می شود و در زمان برنامه ریزی شدن فیوز بیت BOOTRST اجرای برنامه از این آدرس حافظه ی BOOT آغاز خواهد شد .

BOOT RST : انتخاب بردار ری ست BOOT که در حالت پیش فرض برنامه ریزی نشده و آدرس بردار ری ست $0000 است و در صورت برنامه ریزی این بیت ، آدرس بردار ری ست به آدرسی که فیوز بیت های BOOTSZ0 و BOOTSZ1 مشخص کرده اند تغییر می یابد .

BODLEVEL : زمانی که این بیت برنامه ریزی نشده (پیش فرض) است ، اگر ولتاژ VCC از 2.7 V کمتر شود ، ری ست داخلی میکرو فعال شده و سیستم را ری ست می کند . زمانی که این بیت برنامه ریزی شده باشد ، اگر ولتاژ پایه ی VCC از 4 V پایین تر شود ، ری ست داخلی میکرو فعال شده و میکرو را ری ست می کند .

BODEN : برای فعال کردن عملکرد مدار BROWN – OUT این بیت بایستی برنامه ریزی شده باشد . این بیت به صورت پیش فرض برنامه ریزی نشده است .

SUT1 , SUT0 : برای انتخاب زمان START – UP به کار برده می شوند .

در جدول زیر بایت پر ارزش فیوز بیتهای ATMEGA32 را مشاهده می کنید :

FUSE HIGH BYTE	BIT NO	DESCRIPTION	DEFAULT VALUE
OCDEN	7	ENABLE OCD (ON CHIP DEBUG ENABLE)	1 (UNPROGRAMMED , OCD ENABLE)
JTAGEN	6	ENABLE JTAG	0 (PROGRAMMED , JTSG ENABLE)
SPIEN	5	ENABLE SERIAL PROGRAM AND DATA DOWNLOADING	0 (PROGRAMMED , SPI PROG . ENABL)
CKOPT	4	OSCILLATOR OPTION	1 (UNPROGRAMMED)
EESAVE	3	EEPROM MEMORY IS PRESERVED THROUGH THE CHIP ERASE	1 (UNPROGRAMMED , EEPROM NOT PRESERVED)
BOOTSZ1	2	SELECT BOOT SIZE	0 (PROGRAMMED)
BOOTSZ0	1	SELECT BOOT SIZE	0 (PROGRAMMED)
BOOTRST	0	SELECT RESET VECTOR	1 (UNPROGRAMMED)

در جدول زیر بایت کم ارزش فیوز بیتهای ATMEGA32 را مشاهده می کنید :

FUSE LOW BYTE	BIT NO	DESCRIPTION	DEFAULT VALUE
BOD LEVEL	7	BROWN OUT DETECTOR TRRIGER LEVEL	1 (UNPROGRAMMED)
BODEN	6	BROWN OUT DETECTOR ENABLE	1 (UNPROGRAMMED , BOD DISABLE)
SUT1	5	SELECT START UP TIME	1 (UNPROGRAMMED)
SUT0	4	SELECT START UP TIME	0 (PROGRAMMED)
CKSEL3	3	SELECT CLOCK SOURCE	0 (PROGRAMMED)
CKSEL2	2	SELECT CLOCK SOURCE	0 (PROGRAMMED)
CKSEL1	1	SELECT CLOCK SOURCE	0 (PROGRAMMED)
CKSEL0	0	SELECT CLOCK SOURCE	1 (UNPROGRAMMED)

درجدول زیر طرز انتخاب مقدار حافظه BOOT توسط فیوز بیت های BOOTSZ0 , 1 مشاهده می شود :

BOOTSZ1	BOOTSZ0	BOOT SIZE	PAGES	Application Flash Addresses	Boot Flash Addresses	Boot Reset Addresses
1	1	256 words	4	$0000 - $3EFF	$3F00 - $3FFF	$3F00
1	0	512 words	8	$0000 –$ 3DFF	$3E00 - $3FFF	$3E00

0	**1**	**1024 words**	**16**	**$0000 – $3BFF**	**$3C00 - $3FFF**	**$3C00**
0	**0**	**2048 words**	**32**	**$0000 – $37FF**	**$3800 - $3FFF**	**$3800**

در جدول زیر نحوه ی انتخاب آدرس بردار ری ست توسط فیوز بیت BOOTRST نشان داده شده است :

BOOTRST	RESET ADDRESS
1 (UNPROGRAMED)	**RESET VECTOR = APPLICATION RESET (ADDRESS = $0000)**
0 (PROGRAMED)	**RESET VECTOR = BOOT LOADER RESET**

در جدول زیر سطوح مختلف ولتاژ برای مدار BROWN OUT نشان داده شده است :

BODEN , BODLEVEL	BROWN OUT DETECTOR
11	**DISABLE**
10	**DISABLE**
01	**AT VCC = 2.7 V**
00	**AT VCC = 4 V**

پیکره بندی پورت ها در ATMEGA32 :

برای تعیین جهت پایه ی پورت ها از این پیکره بندی استفاده می کنیم . جهت یک پایه می تواند ورودی و یا خروجی باشد .

CONFIG PORT x = state

CONFIG PIN X . Y = state

Y و X بسته به میکروکنترلر می توانند به ترتیب پایه های 0 تا 7 باشند پورت های A,B,C,D,E,F,G برای میکرو های مختلف باشند . state نیز می تواند یکی از گزینه های زیر باشد :

1) INPUT یا 0 : در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده صفر می شود و پایه یا پورت به عنوان ورودی استفاده می شود .

2) OUT PUT یا 1 : در این حالت رجیستر جهت داده پایه یا پورت انتخاب شده یک می شود و یا پایه یا پورت به عنوان خروجی استفاده می شود .

بررسی پورت های میکروکنترلر ATMEGA 32 :

پورت A :

پورت A یک I/O دو طرفه ی هشت بیتی است . سه آدرس از مکان حافظه ی I/O به پورت A اختصاص دارد . یک آدرس برای رجیستر داده ی پورت A ، دومین رجیستر جهت داده ی DDRA و سومی پایه ی ورودی پورت A ، یعنی PIN A است . آدرس پایه های ورودی PORT A فقط قابل خواندن است . در صورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های PORT A دارای مقاومت PULL UP مجزا هستند . بافر خروجی PORT A جریانی تا 20 میلی آمپر را sink می کند و در

نتیجه قابلیت راه اندازی LED را به صورت مستقیم دارد . هنگامی که پایه های PA0 – PA7 با مقاومت های PULL DOWN خارجی ، به عنوان خروجی استفاده می شوند ، آنها به عنوان SOURCE جریان می شوند و این حالت زمانی اتفاق می افتد که مقاومت های PULL UP داخلی فعال باشند .

استفاده از PORT A به عنوان یک I/O عمومی دیجیتال :

جدول فوق تاثیر تغییرات DDRA n را بر روی پایه های PORT A نشان می دهد :

تمام هشت پایه ی موجود زمانیکه به عنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند .

Pan ، پایه ی I/O عمومی : بیت DDAn در رجیستر DDRA مشخص کننده ی جهت پایه است . با توجه به جدول بالا اگر DDAn یک باشد ، Pan به عنوان یک پایه ی خروجی مورد استفاده قرار می گیرد و اگر صفر باشد ، P An به عنوان یک پایه ی ورودی در نظر گرفته می شود . اگر PORT An یک باشد هنگامیکه پایه به عنوان ورودی تعریف می شود ، مقاومت PULL UP فعال می شود که برای غیر فعال کردن مقاومت باید PORT An برابر صفر شود یا این که پایه به عنوان خروجی تعریف شود . پایه های پورت A

DDAn	PORT An	I/O	PULL UP	Comment
0	0	INPUT	NO	Tri – state
0	1	INPUT	YES	PA n Will source current if ext . pulled low
1	0	OUT PUT	NO	Push – pull Zero out put
1	1	OUT PUT	NO	Push – pull one out put

زمانی که ری ست اتفاق می افتد به حالت Tri – state می روند .

کاربرد های PORT A :

پورت A به عنوان ADC هم استفاده می شود . این نکته بسیار مهم است که اگر تعدادی از پایه های PORT A به عنوان خروجی تعریف می شوند ، در زمان نمونه برداری از سیگنال آنالوگ توسط ADC سوئیچ نشوند . این کار ممکن است عملیات تبدیل ADC را نا معتبر کند .

پورت B :

پورت B یک I/O دو طرفه هشت بیتی است . سه آدرس از مکان حافظه ی I/O به پورت B اختصاص دارد . یک آدرس برای رجیستر داده پورت B ، دومین رجیستر جهت داده DDRB و سومین ، پایه ی ورودی PORT B ، PIN B است . آدرس پایه های ورودی پورت B فقط قابل خواندن است ، در صورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . پایه های پورت دارای مقاومت PULL UP مجزا هستند . بافر خروجی پورت B جریانی تا 20 میلی آمپر را sink می کنند و لذا می توانند یک LED را مستقیما راه اندازی کنند . هنگامی که PB0 تا PB7 با مقاومت های PULL DOWN به عنوان خروجی استفاده شوند ، آنها SOURCE جریان می شوند و این حالت زمانی اتفاق می افتد که مقاومت های PULL UP داخلی فعال باشند .

Bit	7	6	5	4	3	2	1	0
	PORTB.7	PORTB.6	PORTB.5	PORTB4	PORTB.3	PORTB.2	PORTB.1	PORTB.0

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

رجیستر های پورت B عبارت اند از :

رجیستر جهت داده ی پورت B (DDRB) مطابق جدول زیر است :

Bit	7	6	5	4	3	2	1	0
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
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

بایت آدرس پایه های ورودی پورت B (PINB) مطابق جدول زیر است :

Bit	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Read/Write	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
Initial Value	N / A	N / A	N / A	N / A	N / A	N / A	N / A	N / A

کاربر دهای پورت B :

- PORT B.7 – SCK

SCK : کلاک خروجی MASTER و کلاک ورودی SLAVE برای ارتباط SPI است . زمانی که SPI به عنوان SLAVE شکل دهی می شود این پایه با توجه به تنظیم DDB7 ورودی و در حالت MASTER خروجی تعریف می شود .

❖ PORTB.6 – MISO

MISO : وردی داده MASTER و خروجی داده SLAVE که برای ارتباط SPI استفاده می شود . زمانی که SPI به عنوان MASTER شکل دهی می شود ، این

پایه با توجه به تنظیمات DDB.6 ورودی و در حالت SLAVE به عنوان خروجی استفاده می شود .

❖ PORTB.5 – MOSI

MOSI : ورودی داده SLVE و خروجی داده MASTER که برای ارتباط SPI استفاده می شود . زمانی که SPI به عنوان MASTER شکل دهی می شود ، این پایه با توجه به تنظیمات DDB5 خروجی و در حالت SLAVE به عنوان ورودی استفاده می شود .

❖ PORTB.4 – SS :

SS : زمانی که SPI به عنوان SLAVE شکل دهی می شود PB.4 با توجه به DDB4 ورودی تعریف می شود و در SLAVE یا LOW شدن این پایه ی SPI فعال می شود . این پایه در MASTER می تواند خروجی یا ورودی تعریف شود .

❖ PORTB.3 – OC0 , AIN1 :

AIN1 : ورودی منفی مقایسه کننده آنالوگ است .

OC0 : دیگر کاربرد این پایه به عنوان خروجی مد مقایسه ای Timer/Counter0 است . پایه ی PB3 با یک کردن DDB7 می

تواند برای خروجی مد مقایسه ای Timer/Counter0 شکل دهی شود .

PORTB.2 – INT2 , AIN0 :

AINO : ورودی مثبت مقایسه کننده آنالوگ است .

INT2 : دیگر کاربرد این پایه به عنوان منبع خارجی وقفه دو است . پایه ی PB2 می تواند به عنوان منبع وقفه خارجی برای میکرو استفاده شود .

❖ PORTB.1 – T1 :

T1 : ورودی کلاک برای Timer/Counter1 است .

❖ PORTB.0 – XCK , T0 :

T0 : ورودی کلاک برای Timer/Counter0 است .

XCK : این پایه نیز می تواند به عنوان کلاک خارجی USART استفاده شود . این پایه فقط زمانی که USART در مد آسنکرون کار می کند فعال می شود .

پورت C :

پورت C یک I/O دو طرفه هشت بیتی است . سه آدرس از مکان حافظه I/O به PORTC اختصاص دارد . یک آدرس برای رجیستر داده PORTC ، دومین رجیستر جهت داده DDRC و سومین ، پایه ی ورودی پورت C ، PINC است . آدرس پایه های ورودی پورت C فقط قابل خواندن است ، در صورتیکه رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های

پورت دارای مقاومت PULL UP مجزا هستند . بافر خروجی پورت C می تواند تا 20 میلی آمپر را sink کند و در نتیجه LED را می تواند مستقیما راه اندازی کند . هنگامی که PC0 تا PC7

DDCn	PORT Cn	I/O	PULL UP	Comment

با مقاومت های PULL DOWN خروجی استفاده می شوند ، آنها SOURCE جریان می شوند و این در حالتی است که مقاومت های PULL UP داخلی فعال باشند .

استفاده از پورت C به عنوان یک I/O عمومی دیجیتال :

تمام هشت پایه ی موجود زمانی که به عنوان پایه های I/O دیجیتال استفاده می شوند دارای عملکرد مساوی هستند . PCn ، پایه ی I/O عمومی : بیت DDCn در رجیستر DDRC مشخص کننده ی جهت پایه است .

طبق جدول زیر اگر DDCn یک باشد ، PCn به عنوان یک پایه ی خروجی مورد استفاده قرار می گیرد و اگر DDCn صفر باشد ، PCn به عنوان یک پایه ی ورودی در نظر گرفته می شود . اگر PORTCn یک باشد ، هنگامی که پایه به عنوان ورودی در نظر گرفته می شود ، مقاومت PULL UP فعال می شود و برای خاموش کردن این مقاومت باید PORTCn صفر شود یا این که پایه به عنوان خروجی تعریف شود . پایه های پورت در زمانی که ری ست اتفاق می افتد ، به حالت Tri – state می روند .

0	**0**	**INPUT**	**NO**	**Tri – state**
0	**1**	**INPUT**	**YES**	**PA n Will source current if ext . pulled low**
1	**0**	**OUT PUT**	**NO**	**Push – pull Zero out put**
1	**1**	**OUT PUT**	**NO**	**Push – pull one out put**

دیگر کاربردهای پورت C :

در جدول زیر کاربردهای دیگر پورت C به طور خلاصه آمده است :

PIN	Alternate Function
PC7	**TOSC2 (Timer Osillator Pin 2)**
PC6	**TOSC1 (Timer Osillator Pin 1)**
PC5	**TDI (JTAG Test Data In)**
PC4	**TDO (JTAG Test Data out)**
PC3	**TMS (JTAG Test Mode Select)**
PC2	**TCK (JTAG Test Clock)**
PC1	**SDA (TOW Wire Serial Bus Data Input/Out put Line)**
PC0	**SCL (TOW Wire Serial Bus Clock Line)**

❖ PORTC.7 – TOSC2 :

TOSC2 : زمانی که تایمر – کانتر 2 در مد آسنکرون کار می کند به این پایه و پایه ی TOSC1 کریستال ساعت متصل می شود . در این حالت دیگر نمی توان این پایه را به عنوان I/O استفاده نمود .

❖ PORTC.6 – TOSC1 :

TOSC1 : زمانی که تایمر – کانتر 2 در مد آسنکرون کار می کند به این پایه و پایه ی TOSC2 کریستال ساعت متصل می شود . در این حالت دیگر نمی توان این پایه را به عنوان I/O استفاده نمود .

❖ PORTC.5 – TDI :

TDI : در زمان ارتباط JTAG ، به عنوان ورودی داده سریال عمل می کند و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود .

❖ PORTC.4 – TDO :

TDO : در زمان ارتباط JTAG ، به عنوان خروجی داده ی سریال عمل می کند و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود .

❖ PORTC.3 – TMS :

TMS : در زمان ارتباط JTAG استفاده می شود و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود .

❖ PORTC.2 – TCK :

TCK : در زمان ارتباط JTAG استفاده می شود و دیگر نمی توان از این پایه به عنوان I/O استفاده نمود .

❖ PORTC.1 – SDA :

SDA : در زمان ارتباط 2 – Wire به عنوان خط داده استفاده می شود .

❖ PORTC.0 – SCL :

SCL : در زمان ارتباط 2 – Wire به عنوان خط کلاک استفاده می شود .

پورت D :

پورت D یک I/O دو طرفه ی هشت بیتی است . سه آدرس لز مکان حافظه ی I/O به PORTD اختصاص دارد . یک آدرس برای رجیستر داده PORTD ، دومین رجیستر جهت داده DDRD و سومین ، پایه ی ورودی پورت D ، PIND است . آدرس پایه های ورودی PORTD فقط قابل خواندن است در صورتی که رجیستر داده و رجیستر جهت داده هم خواندنی و هم نوشتنی هستند . تمام پایه های پورت دارای مقاومت PULL UP مجزا هستند . بافر خروجی PORTD می تواند تا 20 میلی آمپر را sink کند و در نتیجه می تواند یک LED را مستقیما راه اندازی کند . هنگامی که PD0 تا PD7 با مقاومت های PULL DOWN به عنوان خروجی استفاده می شوند ، آنها SOURCE جریان می شوند و این حالت زمانی اتفاق می افتد که مقاومت های PULL UP داخلی فعال باشند .

رجیستر جهت داده پورت D (DDRD) مطابق جدول زیر است :

Bit	7	6	5	4	3	2	1	0
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
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

بایت آدرس پایه های ورودی پورت D (PIND) مطابق جدول زیر است :

Bit	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Read/Write	R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W
Initial Value	N / A	N / A	N / A	N / A	N / A	N / A	N / A	N / A

کاربردهای پورت D :

در مطالب زیر کاربرد هر یک از پایه های PORTD آمده است :

❖ PORTD.7 – OC2 :

OC2 : خروجی مد مقایسه ای تایمر – کانتر 2 می باشد . PD.7 با یک شدن DDD7 می تواند به عنوان پایه ی خروجی مد مقایسه ای Timer/counter2 شکل دهی شود . این پایه همچنین برای خروجی PWM تایمر نیز استفاده می شود .

❖ PORTD.6 – ICP :

ICP : PD6 می تواند به عنوان پایه ی ورودی CAPTURE ، Timer/counter1 عمل کند .

❖ PORTD.5 – OC1A :

OC1A : خروجی مد مقایسه ای Timer/counter1 . این پایه با یک شدن DDD5 می تواند برای خروجی مد مقایسه ای Timer/counter1 شکل دهی شود . این پایه همچنین برای خروجی PWM تایمر 1 نیز استفاده می شود .

❖ PORTD.4 – OC1B :

OC1B : خروجی مد مقایسه ای Timer/counter1

این پایه با یک شدن DDD4 می تواند برای خروجی مد مقایسه ای Timer/counter1 شکل دهی شود . این پایه همچنین برای خروجی PWM تایمر استفاده می شود .

❖ PORTD.3 – INT1 :

INT1 : منبع وقفه خارجی یک .

پایه ی PD3 می تواند به عنوان منبع وقفه خارجی برای میکرو استفاده شود .

❖ PORTD.2 – INT0 :

INT0 : منبع وقفه خارجی صفر

پایه ی PD2 می تواند به عنوان منبع وقفه خارجی برای میکرو استفاده شود .

❖ PORTD.1 – TXD :

TXD : ارسال داده (پایه ی خروجی داده برای USART)

زمانی که ارسال USART فعال می شود ، پایه با توجه به DDD1 به عنوان خروجی شکل دهی می شود.

❖ PORTD.0 – RXD :

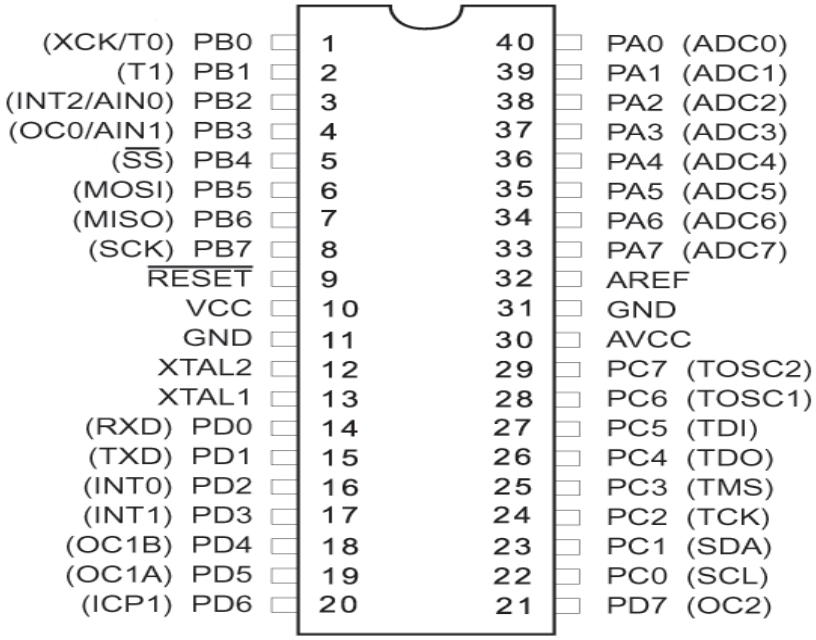
RXD : دریافت داده (پایه ی ورودی داده برای USART)

زمانی که دریافت USART فعال می شود ، پایه با توجه به DDD0 به عنوان ورودی شکل دهی می شود .

PIN	ALTERNATION FUNCTION
PD7	OC2 (T / C2 OUT PUT COMPARE MATCH OUT PUT)
PD6	ICP (T / C1 INPUT CAPTURE PIN)
PD5	OC1A (T / C1 OUT PUT COMPARE A MATCH OUT PUT)
PD4	OC1B (T / C1 OUT PUT COMPARE B MATCH OUT PUT)
PD3	INT1 (EXTERNAL INTERUPT 1 OUT PUT)
PD2	INT0 (EXTERNAL INTERUPT 0 OUT PUT)
PD1	TXD (USART OUT PUT LINE)
PD0	RXD (USART IN PUT LINE)
PIN	ALTERNATION FUNCTION
PD7	OC2 (T / C2 OUT PUT COMPARE MATCH OUT PUT)
PD6	ICP (T / C1 INPUT CAPTURE PIN)
PD5	OC1A (T / C1 OUT PUT COMPARE A MATCH OUT PUT)
PD4	OC1B (T / C1 OUT PUT COMPARE B MATCH OUT PUT)
PD3	INT1 (EXTERNAL INTERUPT 1 OUT PUT)
PD2	INT0 (EXTERNAL INTERUPT 0 OUT PUT)
PD1	TXD (USART OUT PUT LINE)

PD0	RXD (USART IN PUT LINE)

در شکل زیر طرز قرار گرفتن پایه ها ی ATMEGA 32 را مشاهده می کنید :



نمایشگر چیست؟

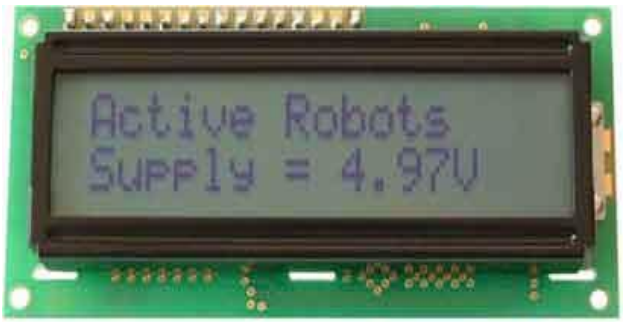
نمایشگر قطعه‌اي الکترونیکی است که با اتصال آن به میکروکنترلر می‌توان هرگونه تصویری را به‌نمایش درآورد. نمایشگرها در مدل‌های بسیار متنوع برای کاربردهای مختلف در

بازار وجود دارند. ازLCD های رنگی‌ای که در موبایل‌ها استفاده می‌شوند گرفته تا مدل‌های بسیار ابتدایی مانند 7 segmentقبلاً با آن آشنا شده‌ایم. در این جلسه ما با نوعی نمایشگر LCD آشنا خواهیم شد که به‌وسیله ی آن می‌توان تمام نمادهایی که در سیستم کدگذاری ASCII وجود دارند را به نمایش در آورد. همان‌طور که قبلاً اشاره شد، این نمادها شامل تمام حروف الفبای بزرگ و کوچک، اعداد لاتین و ....هستند. این نوع LCD را در اصطلاح تجاریLCD های کاراکتری (Alphanumeric LCD)می‌گویند.

معرفی LCD کارکتری :

LCD های کارکتری خود به چند نوع دیگر از لحاظ اندازه تقسیم بندی میشوند . که از LCD هایی با 1 سطر و 1 ستون آغاز میشوند تا اندازهایی مثل 4 سطر و 40 ستون که البته تمام آنها از 16پایه تشکیل شده اند.

برای راه اندازی LCD توسط AVR نیازی به دانستن جزئیات طرز کار LCD نیست . برای کار با LCD علاوه بر پایه های تغذیه و CONTRAST (تنظیم روشنایی) که باید مانند شکل مداری پایین بایاس شوند نیاز به 6 پایه ی دیگر است که عبارتند از پایه های : RS , E , DB4 , DB5 , DB6 , DB7 .



نمایشگرها در ساخت رباتها و دستگاه‌های هوشمند الکترونیکی کاربرد بسیار زیادی دارند. با ذکر چند مثال شما را با کاربرد این نمایشگرها بیشتر آشنامی‌کنیم.

در ربات مین‌یاب برای اعلام مختصات مین‌ها به داور، باید روبات مجهز به نمایشگری باشد که بتوان این اطلاعات را بر روی آن به نمایش درآورد.

در ربات فوتبالیست، نمایشگر در زمان مسابقه کاربرد مستقیمی ندارد، اما در مراحل عیب‌یابی و تنظیمات اولیه سنسورها کاربرد زیادی دارد. مثلاً برای تنظیم حساسیت هر سنسور، اطلاعات آن بر روی صفحه نمایش به کاربر نشان داده می‌شود و کاربر می‌تواند آن را سریع تر تنظیم کند. به‌عنوان مثال برای تنظیم سنسورهای نوری می‌توان ولتاژ خروجی آن را توسط ADC اندازه‌گیری کرد و بر روی LCD نمایش داد.

از دیگر موارد کاربرد این نوعLCD ها می‌توان به دستگاه‌های تلفن خانگی اشاره کرد که به‌کمک آن، داده‌هایی مثل شماره‌ی تلفن فرد تماس‌گیرنده، دفترچه تلفن و . را نمایش می‌دهد.

LCDهای کارکتری در سایزهای مختلفی وجود دارند. سایز این نوع LCD را بر اساس تعداد کاراکترهایی که در هر سطر و ستون نمایش داده می‌شوند، تعیین می‌کنند. پرکاربردترین سایزLCD های کاراکتری 2*16 است، یعنی این LCD می‌تواند 2 ردیف 16 تایی کاراکتر را هم‌زمان روی صفحه نمایش دهد.

چگونه از lcd استفاده کنیم؟

در ساختمان داخلی این LCDها مدارات متعددی وجود دارد که اطلاعاتی که برای نمایش دادن به LCD فرستاده می‌شود را پردازش کرده و اطلاعات مورد نظر ما را روی صفحه به‌نمایش در می‌آورند.

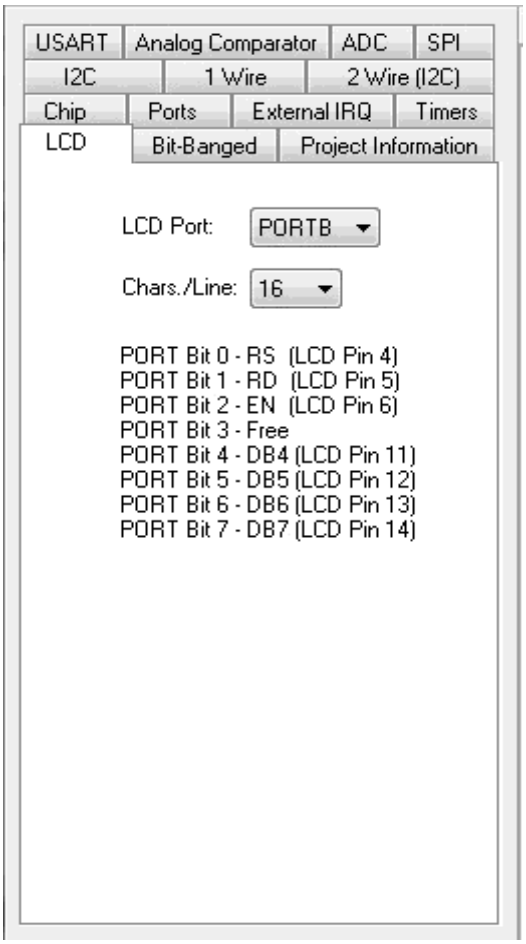
این اطلاعات باید از طریق پایه‌های LCD به آن منتقل شوند.

برقراری ارتباط و نمایش اطلاعات بر روی LCD کار چندان ساده ای نیست، اما CodeVision در اینجا هم به کمک ما آمده است و کار را

بسیار ساده کرده است.توضیح در مورد نحوه‌ی استفاده از LCD را از تنظیمات نرم افزاری آن در محیط codevision شروع می‌کنیم.

تنظیمات اولیه در code vision برای راه اندازی lcd :

Codevisionرا باز کرده پروژه‌ی جدیدی بسازید. سپس در Code Wizardتنظیمات مربوط به لبه‌ی Chip را انجام دهید.
حالا سراغ لبه‌ی LCD می‌رویم.
برای راه اندازیLCD های کارکتری، باید تمام پایه‌های یکی از پورت‌های میکروکنترلر را به پایه‌های مربوطه در LCD متصل کنیم.
ابتدا باید تعیین کنیم می‌خواهیم کدام پورت را به LCD اختصاص دهیم.
سپس باید با تعیین تعداد کاراکترهای قابل نمایش در هر سطر از LCDنوع آن را مشخص کنیم .مثلاً اگر LCD ما 2*16 است، باید عدد 16 را انتخاب کنیم.
سپس نحوه‌ی اتصال پایه‌های میکروکنترلر به LCD را با توجه به نوع LCD به شما نشان می‌دهد.



برای مثال ترتیب اتصال پایه‌ها برای LCD 16*2 بر روی پورت "B" در زیر نشان داده شده است.
پایه PB.0 به پایه‌ی چهارم LCD متصل شود.
پایه PB.1 به پایه‌ی پنجم LCD متصل شود.
پایه PB.2 به پایه‌ی ششم LCD متصل شود.
پایه PB.3 به جایی متصل نمی‌شود.
پایه PB.4 به پایه‌ی یازدهم LCD متصل شود.
پایه PB.5 به پایه‌ی دوازدهم LCD متصل شود.
پایه PB.6 به پایه‌ی سیزدهم LCD متصل شود.
پایه PB.7 به پایه‌ی چهاردهم LCD متصل شود.

بعد از اینکه طبق ترتیب ذکر شده پایه‌ها را متصل کردیم، و تنظیمات اولیه را در CodeWizard انجام دادیم، سراغ برنامه‌نویسی آن می رویم CodeVision .توابعی را آماده کرده است که به کمک آن‌ها می‌توانیم به‌سادگی اطلاعات موردنظر خودمان روی LCD نمایش دهیم.

4 دستور اصلی برای نمایش اطلاعات روی lcd :

1- lcd_putchar(‘ ‘);

این دستور برای نمایش یک کاراکتر بر روی LCD استفاده می‌شود. مثلاً دستور زیر حرف F را بر روی LCD نمایش می‌دهد :

lcd_putchar(‘F’);

2- lcd_putsf(“ “);

این دستور برای نمایش یک رشته از حروف بر روی LCD استفاده می‌شود. مثلاً دستور زیر جمله‌ی it is a test را بر روی LCD نمایش می‌دهد:

lcd_putsf(“it is a test”);

3- lcd_clear();

این دستور برای پاک کردن LCD مورد استفاده قرار می‌گیرد. این دستور هر کاراکتری را که روی LCD در حال نمایش باشد پاک می‌کند.

4- lcd_gotoxy(,);

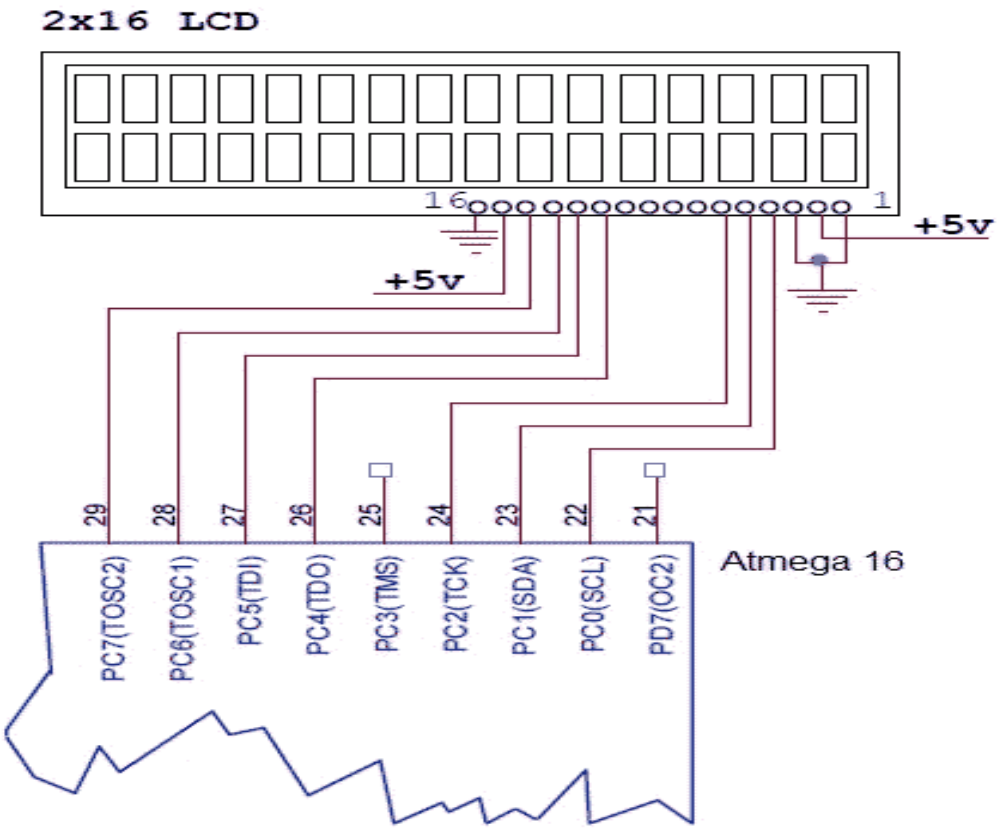
به‌کمک این دستور می‌توان تعیین کرد کاراکتر یا جمله‌ی مورد نظر ما در کدام سطر و ستون در LCD نوشته شود .مثلاً دستورهاي زیر lcd را پاک کرده و واژه‌ی Hello را از وسط سطر دوم می‌نویسد.
شماره‌گذاری سطرها و ستون‌ها از 0 شروع می‌شود. پس سطر شماره‌ی 1 ، سطر دوم است.

```
lcd_clear();
lcd_gotoxy(1,7);
lcd_putsf("Hello");
```

سایر پایه های lcd :

LCD نیز مانند هر قطعه‌ی الکترونیکی دیگر نیاز به 2 پایه برای تغذیه + و – دارد. در LCD های 16*2 اختلاف پتانسیل مورد نیاز برای تغذیه باید 5 ولت باشد. پایه شماره‌ی 1 باید به GND و پایه‌ی شماره‌ی 2 باید به 5 ولت متصل شود. پایه‌ی شماره‌ی 3 نیز برای تنظیم نور زمینه در LCD تعبیه شده است. در حالت معمولی باید این پایه مستقیماً به GND متصل شود.

پایه‌های 15 و 16 نیز برای تغذیه‌ی نور پشت زمینه هستند. پایه‌ی 15 به Vcc (5 ولت) و پایه‌ی 16 به GND متصل می‌شود
.



معرفی خطهای lcd :

شماره و نام خط	عملکرد
Vss -1	زمین
Vcc -2	ولتاژ 5 ولت برای کنترلر
Vee -3	ولتاژ تنظیم درخشندگی (contrast)
RS -4	انتخابگر ثبات دستور / داده
RW -5	انتخابگر خواندن / نوشتن
Enable -6	فعال کننده
Bus 14-7	8 خط گذرگاه داد یا

دستور	
ولتاژ 5 ولت برای لامپ پشت صفحه	15-
زمین برای لامپ پشت صفحه	16-

Vee : برای تنظیم درخشندگی کاراکترها بکار می رود که باید ولتاژی بین صفر و 5 ولت به این پایه اعمال نمود. برای بیشترین درخشندگی این پایه را به زمین متصل کنید.

انتخابگر ثبات داده / دستور مشخص می کند که چه چیزی به LCD فرستاده می شود. اگر این خط صفر باشد کنترلر LCD بایت موجود روی خطوط 7 تا 14 را بعنوان یک دستور تلقی کرده و اگر این پایه یک باشد اطلاعات را بعنوان یک کد اسکی که باید کاراکتر معادل آنرا نمایش دهد در نظر می گیرد.

انتخابگر خواندن / نوشتن جهت اطلاعات را نشان می دهد. اگر این پایه صفر باشد اطلاعات به LCD ارسال می شود و اگر یک باشد عمل خواندن از LCD صورت می گیرد.

RS	R/W	MODE
0	0	Command write
0	1	Command read
1	0	Data write
1	1	Data read

فعال کننده : برای هر دستور یا داده ای که به LCD میفرستیم یا میخواهیم از آن بخوانیم باید یک پالس پائین رونده (یعنی تغییر از سطح یک به صفر) را به این پایه اعمال کنیم تا دستور یا داده بوسیله کنترلر LCD پردازش شود.

در خطوط 7 تا 14 خط 7 کم ارزشترین بیت(LSB) و خط 14 پر ارزش ترین بیت (MSB) می باشد.

در صورت تمایل به روشن کردن لامپ پشت صفحه ولتاژ 5 ولت را به پایه 15 اعمال و پایه 16 را به زمین متصل می کنیم.

برای آزمایش می توان LCD را به پورت چاپگر متصل و اطلاعاتی را به آن ارسال نمود. در این حالت بطور معمول خطوط داده پورت به خطوط 7 تا 14 و سه خط کنترلی به پایه های 4 تا 6 اتصال داده می شود توجه داشته باشید که ولتاژ تغذیه و لامپ پشت صفحه LCD توسط منبع خارجی تامین می شود.

روش فرستادن یک کاراکتر:

خط خواندن نوشتن را صفر کنید تا نوشتن انتخاب شود.
خط داده / دستور را یک کنید تا داده انتخاب شود.
کد اسکی کاراکتر مورد نظر را روی خطوط 0D تا 7D قرار دهید.
خط انتخاب را ابتدا یک و سپس صفر کنید. حداقل 450 نانو ثانیه باید این خط را صفر نگه دارید تا داده پردازش شود. بعد از آن حالت خط تاثیری نخواهد داشت

برای آزمایش می توان LCD را به پورت چاپگر متصل و اطلاعاتی را به آن ارسال نمود. در این حالت بطور معمول خطوط داده پورت به خطوط 7 تا 14 و سه خط کنترلی به پایه های 4 تا 6 اتصال داده می شود توجه داشته باشید که ولتاژ تغذیه و لامپ پشت صفحه LCD توسط منبع خارجی تامین می شود.
خط انتخاب را ابتدا یک و سپس صفر کنید. حداقل 450 نانو ثانیه باید این خط را صفر نگه دارید تا داده پردازش شود. بعد از آن حالت خط تاثیری نخواهد داشت.

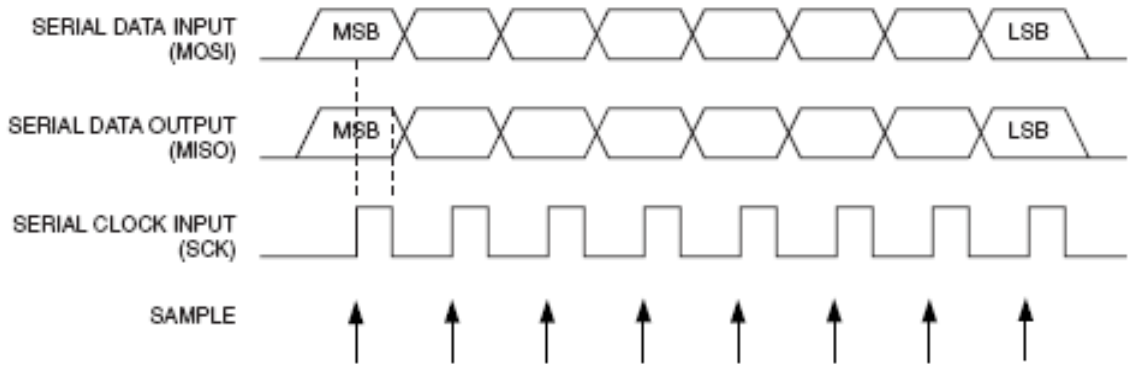
بخش حافظه ی پروژه (MMC) :

حال به بررسی مموري کارتهاي SD/MMC میپردازیم .این نوع حافظه ها که از نوع FLASH میباشند و به صورت مرسوم در دستگاههاي قابل حملی مانند MP3 PLAYERها و COOL DISK ها و موبایلها استفاده میشوند . از خصوصیات بارز آن میتوان به سرعت بالا و حجم بسیار بالاي آن اشاره کرد .این حجم بالا که از چندین مگابایت تا چندین گیگابایت میباشد به همان قدر که براي تولید کنندگان این دستگاهها وسوسه انگیز است که براي مهندسان الکترونیک.شاید دلیل عمده این توجه از طرف مهندسان الکترونیک فضاي زیادي باشد که در هیچ مدار میکروکنترلري به آن دسترسی ندارند .البته قطعات مختلفی براي رفع این نیاز مانند خانواده AT45DBXXX از شرکت اتمل تولید شد که نوعی FLASH بوده ولی از لحاظ ظرفیت هرگز به گرد MMCیا SD نخواهند رسید .ویژگی بارز دیگر این حافظه قابلیت پشتیبانی از استانداردهاي FAT است که با این ویژگی امکان دسترسی به دادهها در کامپیوتر بوجود میآید.

بررسی مموري کارت از دید سخت افزاری:

MMC وSD تنها از نظر سرعت وحجم بافر با هم تفاوت دارند ولی از نظر نوع فضا و آدرس دهی ها و ...با هم فرقی ندارند .پروتکل ارتباطی مموري به دو شکل IDE MODE یا ارتباط موازي چهاربیتی و نوع SPI MODE یا ارتباط سریال سنکرون میباشد که متاسفانه مطالب زیادي درباره نوع اول موجود نیست اما نوع دوم ارتباط مرسومی در میکروکنترلرهاست . این پروتکل ارتباطی به طور معمول داراي سه خط اصلی ارتباطی میباشد که یکی از این سه خط وظیفه ارسال داده ، دیگري دریافت داده و خط سوم

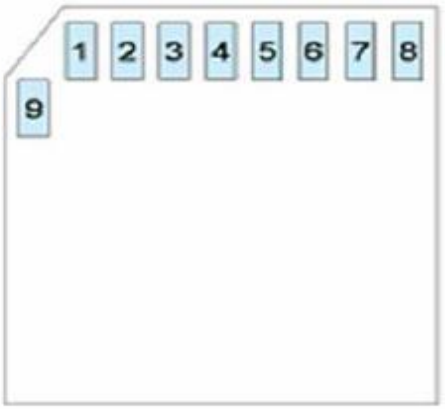
حامل کلاك ميباشد .در اين ارتباط طرفين در دو حالت قرار ميگيرند يکی MASTER که توليد کننده پالس و حالت دوم SLAVE که هيچگونه اعمالی روي کلاك انجام نميدهد و فقط دريافت کننده آن است.در اين ارتباط يک خط نيز حامل سيگنال SLAVE SELECT يا SS ميباشد که با فعال شدن آن سيستم دريافت کننده اين سيگنال به مد SLAVE ميرود اما اين خط به صورت OPTIONAL يا اختياري است و در اکثر موارد نيازي به آن نيست. در ارتباط با موري کارتها اين خط به نوعی بعنوان CS يا CHIP SELECT به کار ميرود .در شکل زير يک نمودار سيگنالی اين ارتباط را مشاهده ميکنيد:



نکته مهم ديگر اينکه در اين ارتباط ميتوان انتخاب کرد که ابتدا MSB ارسال شود يا LSB يا در بحث موري کارت MSB ابتدا ارسال ميشود.

MEMORY CARD PIN CONFIGURATION

شکل زير وضعيت پايه هاي موري کارت و ابعاد فيزيکی موري را نمايش می دهد:



Pin 1	Chip Select
Pin 2	SDI
Pin 3	V ground
Pin 4	V supply (+3.3V)
Pin 5	SCLK
Pin 6	V ground
Pin 7	SDO
Pin 8	Reserved (only available in SD Cards)
Pin 9	Reserved (only available in SD Cards)

تغذيه موري بين 2.7 ولت تا 3.6 ولت ميباشد که براي اطمينان از صحت کار آن توصيه شده با ولتاژ 3.6 ولت تغذيه شود و حافظه بسيار به ولتاژ کاري خود حساس است و امکان آسيب ديدن آن بدليل اضافه ولتاژ و نويز

بسیار زیاد است .فرکانس کاري حافظه در مد SPI تا 50 مگا هرتز بوده و در این پروژه که با میکروکنترلر AVR انجام شده فرکانس کاري حد اکثر8 مگا هرتز میباشد .

نکته : براي ارسال هر داده یا دستور به موري باید ابتدا CS فعال شود بجز مواردي که ذکر خواهد شد .

نکته : این نوع حافظه شامل چندین سکتور است که همه سکتورها در همه انواع موري کارت 512 بایت است.

نحوه ارتباط:

1.RESET: براي راه اندازي حافظه ابتدا باید از نظر سخت افزاري ریست شود .براي این کار مخالف همه دستورها باید CS غیر فعال شود سپس ده بار عدد 0XFF از طریق SPI به موري ارسال شود .بلافاصله عدد 0X40 که دستور RESET می باشد ارسال و چهار بار عدد 0X00 ارسال شود .در ادامه عدد 0X95 ارسال شود و تا زمانیکه عدد 0X01 از طرف موري دریافت نشد به موري عدد 0XFF ارسال شود .

2. INITIALIZATION MEMORY : براي INIT کردن موري ابتدا CS را فعال کرده و عدد 0X41 را ارسال میکنیم و بلافاصله چهار بار صفر را ارسال کرده و تا زمانیکه ازطرف موری عدد 0X00دریافت نشد به موری عدد 0XFF را ارسال میکنیم .

3.خواندن یک سکتور از موري : بعد از فعال شدن CS وعدد 0X51 را به آن ارسال می کنیم و به صورت BIG INDIAN دو برابر آدرس مورد نظر را به موري ارسال کرده و یک بارعدد 0X00 را ارسال کرده و تا زمانیکه عدد 0XFE را ارسال کرده و تا زمانیکه عدد 0XFF را ارسال میکنیم .در این لحظه موري آماده ارسال داده هاي موجود در سکتور مورد نظر بوده و با هر بار ارسال 0XFF یکی از 512 بایت را ارسال کند .

4.نوشتن در یک سکتور : براي نوشتن در یک سکتور ابتدا عدد **0X52** ارسال شده سپس دو برابر آدرس سکتور مورد نظر ارسال و سه بار 0XFF ویک بار 0XFE ارسال شده در این لحظه میتوان 512 بایت داده را ارسال کرد بعد از این داده ها دو بار 0XFF ارسال شود ومی توان CS را غیر فعال کرد .

MICROSOFT FAT

یکی از خارق العاده ترین تکنولوژیهایی که شرکت مایکروسافت ارائه داده **FAT می باشد. این** استاندارد که براي مکان حافظه هاي بسیار حجیم بوجود آمد در ابتدا با استاندارد **FAT12** ارائه شد که قابلیت آدرس دهی 12 بیتی کلاسترها را داشت و تا 16 مگابایت مموري یا هارددیسک را پشتیبانی میکرد ولی امروزه بدلیل حجم بالاي ظرفیت این نوع استاندارد منسوخ شده است.در ادامه نمونه قویتري با نام FAT16 را ارائه کرد که قابلیت آدرس دهی 16 بیتی کلاسترها را داشت و تا ظرفیت 4 گیگا بایت را پشتیبانی میکرد این نمونه نیز با روي کار آمدن فضاهاي بسیار بالا در هارد دیسکها کنار رفته و فقط در سیستمهاي قابل حملی مانند COOL DISKو MP3 PLAYER ها استفاده میشود. این نمونه بدلیل پائین بودن تعداد سکتور به ازاي هر کلاستر در سیستم هاي قدیمی سرعت کمتري را داراست زیرا در حجم برابر به محاسبات بیشتري نسبت به نمونه جدیدتر خود نیاز دارد تا ادامه فایل ذخیره شده را در حافظه پیدا کند.به این دلیل و محدودیت ظرفیت نمونه دیگري ارائه شد که با نام FAT32 قابلیت آدرس دهی 32 بیتی کلاسترها را دارا بود.این نمونه که تا چندي پیش در کامپیوترهاي شخصی حرف اول را در مورد مدیریت هارددیسک ها می زد تا ظرفیت 32گیگابایت حافظه را پشتیبانی میکرد و از سرعت بالایی نسبت به همه نمونه هاي قبلی برخوردار بود.اما با نمونه آخر با نام NTFS فعلا تمام حرفها و حدیثها تمام شد، این استاندارد که با نمونه هاي قبل فرق داشته و ظرفیت 2 ترابایت را در حافظه پشتیبانی میکند و بسیار جمع و جورتر از همه طراحی شد و از ریخت و پاش فضاي حافظه بسیار جلوگیري شده و از لحاظ سرعت و میزان محاسبات مورد نیاز بسیار بهینه سازی شد.

در ادامه به بحث در مورد بعضی از مفاهیم موجود در این استاندارد و کلمات کلیدي آن میپردازیم و با توضیح لازم تفاوتهاي موجود بین نمونه هاي مختلف را شرح می دهیم.البته چون مبحث اصلی مموري کارتهاي SDوMMC میباشد و بغیر از ویندوز VISTA بقیه ویندوزها قابلیت پشتیبانی NTFS را روي این نوع حافظه ندارند اینجانب به شخصه از کار روي این استاندارد صرف نظر کرده و بعلاوه در مورد FAT12 دلیل اینکه در بازار مموري با ظرفیت 16 مگابایت موجود نیست تحقیقی به عمل نیامد.

مفاهیم کلیدي:

1) SECTOR: فضاي مموري کارتها یا هارد دیسکها به تعدادي سکتور تقسیم میشوند که براي مموري کارتهاي SDوMMC سکتور شامل 512 بایت میباشد و مموري کارت در هنگام نوشتن و خواندن بطور مستقیم

با بایتها سر و کار ندارد بلکه با سکتورها کار میکند و به این ترتیب براي اینکه بتوان به شکل معمول با موري کار کرد بهتر است که میکروکنترلري استفاده کرد که از 512 بایت بیشتر SRAM داشته باشد.

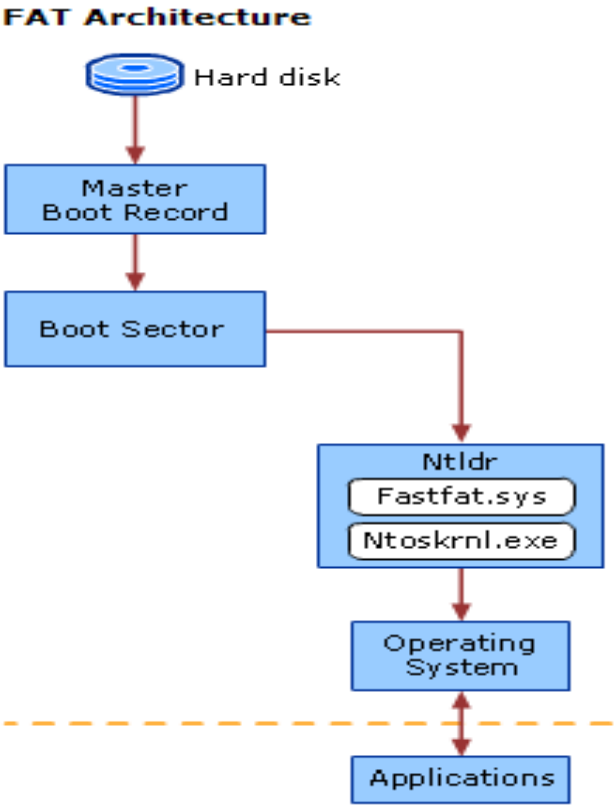
2)CLUSTER: از دید سخت افزاري کلاستر وجود خارجی ندارد و فقط در استاندارد FAT فایل سیستم بجاي اینکه با سکتورها کار کند که تعداد زیادي میباشند با کلاستر کار میکند، و کلاستر به مجموعهاي از سکتورها اطلاق میشود که براي هر نوع FAT اعم از NTFS و FAT32، FAT16، FAT12 متفاوت و وابسته به ظرفیت حافظه میباشد. در جدول زیر این مطلب به روشنی مشخص شده:

Volume size	FAT16 cluster size	FAT32 cluster size	NTFS cluster size
7 MB–16 MB	2 KB	Not supported	512 bytes
17 MB–32 MB	512 bytes	Not supported	512 bytes
33 MB–64 MB	1 KB	512 bytes	512 bytes
65 MB–128 MB	2 KB	1 KB	512 bytes
129 MB–256 MB	4 KB	2 KB	512 bytes
257 MB–512 MB	8 KB	4 KB	512 bytes
513 MB–1,024 MB	16 KB	4 KB	1 KB
1,025 MB–2 GB	32 KB	4 KB	2 KB
2 GB–4 GB	64 KB	4 KB	4 KB
4 GB–8 GB	Not supported	4 KB	4 KB
8 GB–16 GB	Not supported	8 KB	4 KB
16 GB–32 GB	Not supported	16 KB	4 KB
32 GB–2 TB	Not supported	Not supported	4 KB

از آنجایی که هر سکتور در این نوع موري 512 بایت است پس به طور مثال براي ظرفیت 129مگابایت تا 256 مگابایت در FAT32 که کلاستر سایز 2 کیلوبایت ذکر شده یعنی4 سکتور.

3) MBR :MBR (MASTER BOOT RECORD) یک کد اجرایی کوچکی است که در بخشی از حافظه قرار میگیرد تا سیستم از آن به اصطلاح BOOT شود.

.در شکل روبه رو نحوه بوت شدن یک سیستم عامل را ملاحظه میکنید.



4)اجزای FAT: در جدول زیر این اجزا مشخص است:

	ores information about the layout of the volume and the file e that loads Windows Server 2003.
	art of the first FAT, including the boot sector.
	of the partition.
	within the file system.

معمولا سکتور صفر در موري کارتها شامل BOOT SECTOR بوده و بعد از آن یک فضایی رزرو شده و فضاي FAT اول قرار داردو FATدوم برای ریکاوری وBACKUP دقیقا شبیه به FATاول قرار میگیرد.

5)ROOT DIRECTORY: به شاخه اصلی موري کارت ROOT DIRECTORY میگویند که در آن آدرس فایلها و فولدرهاي موجود و نام و مشخصاتشان نوشته شده است.

.*در ادامه براي روشن شدن موضوع هرFAT را به صورت جداگانه بررسی میکنیم.

FAT32: همانطور که گفته شد در موري کارتها سکتور بوت معمولا در سکتور صفر قرار دارد. این سکتور شامل اطلاعات اصلی فضاي حافظه است.

1) ده بایت اول این سکتور اطلاعاتی از شرکت پیاده کننده وجود دارد .

2) Byte per sector:بایت دوازدهم و سیزدهم تعداد هر بایت در هر سکتور را مشخص میکند که در اینجا با توجه به توضیحات داده شده همیشه عدد 512 قرار دارد.

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x0B	2 bytes	00 02	**Bytes Per Sector.** The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096.

3) Sector per cluster: این فضاي یک بایتی تعداد سکتور به ازاي هر کلاستر را معرفی میکند.

0x0D	1 byte	10	**Sectors Per Cluster.**The number of sectors in a cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128. The Windows Server 2003 implementation of FAT32 allows for creation of volumes up to a maximum of 32 GB. However, larger volumes created by other operating systems (Windows 95 OSR2 and later) are accessible in Windows Server 2003.

4) Reserved sector: این داده دوبایتی نشان دهنده آدرس شروع FAT می باشد .در واقع این آدرس محل شروع جدول تخصیص حافظه یا memory allocation table میباشد که مهمترین بخش فضاي حافظه است.

با توجه به اینکه بیشتر مواقع آدرس سکتور اصلی صفر است بطور معمول آدرس شروع FATنیز همیشه 36 میباشد .البته این آدرس در مواردي که شرکت راه انداز نیاز به قرار دادن اطلاعاتی دور از دسترس کاربران خود داشته باشد فرق خواهد کرد اما اگر موري براي کاربا ویندوز طراحی شده باشد این آدرس همیشه 36 است .

0x0E	2 bytes	24 00	**Reserved Sectors.** The number of sectors that precede the start of the first FAT, including the boot sector.

5) Number of FATs: این داده یک بایتی بوده و تعداد FAT های موجود در موري را نشان میدهد و به طور معمول حاوي عدد 2 بوده که نشانگر دو FAT در موري است که همانطور که قبلا گفته شد FAT دوم براي بازیابی اطلاعات است که به دلایل نا معلومی از آن استفاده نمی شود. روش کار به صورتی است که فایلها و فولدرها در FAT اول ودوم تخصیص و آدرس دهی میشوند اما در هنگام پاك کردن یا فرمت سریع یا quick formatفقط FAT اول پاك شده و FATدوم تا زمانی که فایل یا فولدر جدیدي ایجاد نشود اطلاعات مربوط به داده هاي قبلی را دارد پس میتوان با رجوع به FAT دوم اطلاعات از دست رفته را بازگرداند اما همانطور که گفته شد این روش استفاده نمیشود و در هنگام پاك کردن یا فرمت سریع هر دو FAT پاک میشوند.

0x10	1 byte	02	**Number of FATs.** The number of copies of the FAT on the volume. The value of this field is always 2.

6) small sectors و Root entries : این دو داده که هرکدام دوبایت میباشند در FAT32همیشه عدد صفر را در خود ذخیره کرده اند.

0x11	2 bytes	00 00	**Root Entries (FAT12/FAT16 only).** For FAT32 volumes, this field must be set to zero.
0x13	2 bytes	00 00	**Small Sectors (FAT12/FAT16 only).** For FAT32 volumes, this field must be set to zero.

7) Largs sectors: یکی از مهمترین داده هاي موجود در فایل سیستم که چهار بایتی و تعداد سکتورهاي موجود در حافظه را نمایش میدهد. آدرس آن بایت 33 به بعد بوده و با توجه به جدول نمونه حافظه 128 مگابایتی این اعداد ملاحظه میشوند 00 197 03 00 : که با محاسبه آن خواهیم داشت 0X3C500 یا 247040 که ضرب در 512 فضاي کل حافظه 126484480 بایت خواهد شد که بر 1024 خواهیم داشت 123520 کیلوبایت و بر 1024 خواهیم داشت 120.625مگابایت.

8) Sectors per FAT: این داده چهار بایتی تعداد سکتور اشغالی هر FAT یا همان memory allocation table را نشان میدهد و به زبان دیگر تنها و تنها راه پیدا کردن شماره سکتور root directory استفاده از این داده میباشد .

روش محاسبه به صورت زیر است:

Rootdirsectornumber=reservedsector + (sectorperfat * numberoffats)

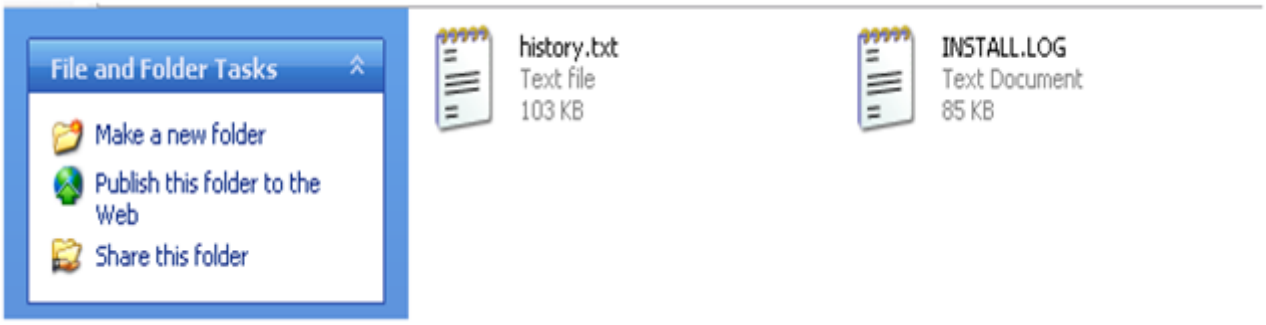
که در حافظه 128 مگابایتی شاخه اصلی برابر است با:

Rootdirsectornumber = 36 + (2 * 958) = 1952

این آدرس سکتور شاخه اصلی است .حال با سیستم طراحی شده domino این آدرس را می خوانیم و با محتوي موري كارت مقايسه میکنیم:

INSTALL LOG]úc/9/9ju_9_—P_HISTORY TXT _rúc/9/9Ybø4X_š_

که در محیط ویندوز داریم:



.*در این سکتور داده‌هایی در مورد محتوي شاخه وجود دارد که در ادامه توضیح خواهیم داد .امابه طور کلی اسم فایل و فولدرهاي موجود قابل رویت است.

0x24	4 bytes	2A 22 00 00	**Sectors Per FAT (FAT32 only).** The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and reserved sectors (described in this table) to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory.

.*دو داده دوبایتی بدون کاربرد بعد از sector per FAT قرار دارد.

0x28	2 bytes	00 00	Not used by Windows Server 2003.
0x2A	2 bytes	00 00	**File System Version (FAT32 only).** The high byte is the major revision number; the low byte is the minor revision number. This field supports the ability to extend the FAT32 media type in the future with concern for old FAT32 drivers mounting the volume. Both bytes are zero in Windows Server 2003, Windows 2000, and Windows Me and earlier.

Root cluster number: از آنجایی که شاخه اصلی ممکن است فایل و فولدرهاي زيادي را درخود جاي دهد به همین دلیل این شاخه شامل چند کلاستر است که در این داده این تعداد مشخص میشود .طول این داده چهار بایت است و به قول مایکروسافت بطور معمول شامل عدد 2 است یعنی شاخه اصلی دو کلاستر است .

0x2C	4 bytes	02 00 00 00	**Root Cluster Number (FAT32 only).** The cluster number of the first cluster of the root directory. This value is typically, but not always, 2.

*.بعد از این داده چند بایت قرار دارد که کاربردي در پروژه ندارد ..

0x30	2 bytes	01 00	**File System Information Sector Number (FAT32 only).** The sector number of the File System Information (FSINFO) structure in the reserved area of the FAT32 volume. The value is typically 1. A copy of the FSINFO structure is kept in the Backup Boot Sector, but it is not kept up-to-date.
0x32	2 bytes	06 00	**Backup Boot Sector (FAT32 only).** A value other than zero specifies the sector number in the reserved area of the volume where a copy of the boot sector is stored. The value of this field is typically 6. No other value is recommended.
0x34	12 bytes	00 00 00 00 00 00 00 00 00 00 00 00	**Reserved (FAT32 only).** Reserved space for future expansion. The value of this field must always be zero.

BPB:بعد از داده‌هاي بالا چند بایت براي انحصاري کردن موري تخصیص داده شده که مربوط به نام و شماره مشخصه موري و ... است.

Extended BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x40	1 byte	80	**Physical Drive Number**. Related to the BIOS physical drive number. Floppy drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. This value is only relevant if the device is a boot device.
0x41	1 byte	00	**Reserved**. FAT32 volumes are always set to zero.
0x42	1 byte	29	**Extended Boot Signature**. A field that must have the value 0x28 or 0x29 to be recognized by Windows Server 2003.
0x43	4 bytes	F1 9E 5E 5E	**Volume Serial Number**. A random serial number that is created when a volume is formatted and that helps to distinguish between disks.
0x47	11 bytes	NO NAME	**Volume Label**. A field that was once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x52	8 bytes	FAT32	**System ID**. A text field with a value of FAT32.

بقیه داده ها مربوط به داده های قابل اجرا است که در این پروژه بکار نیامده و فقط در همه موری ها ثابت بوده و در هنگام فرمت کردن باید در محل مناسب نوشته شوند.

ساختار **Root directory:**

این داده ها مربوط به یکی از فایلها است که خوانده شده:

```
point1:(I);73
point2:(N);78
point3:(S);83
point4:(T);84
point5:(A);65
point6:(L);76
point7:(L);76
point8:( );32
point9:(L);76
point10:(O);79
point11:(G);71
point12:( );32
point13:();0
point14:(]);93
point15:(ú);250
point16:(c);99
point17:(/);47
point18:(9);57
point19:(/);47
point20:(9);57
point21:();0
point22:();0
point23:(j);106
point24:(u);117
point25:(_);4
point26:(9);57
```

```
point27:(_);3
point28:();0
point29:(—);151
point30:(P);80
point31:(_);1
point32:();0
```

ساختار داده‌ها و اطلاعات موجود در این سکتورها همانند بقیه دایرکتوریها بوده و تفاوتی بین FAT32 &FAT16 در این مورد نیست. در این محیط بهازای هر فایل یا فولدر 32 بایت فضا اشغال شده وبه ترتیب یازده بایت اولیه آن مربوط به اسم و پسوند فایل یا فولدر میباشد .فرمت این اسم به صورت 8.3 میباشد یعنی هشت کارکتر اسم و سه کارکتر پسوند .در صورتیکه طول اسم کمتر باشد فضاي موجود با کد اسکی SPACE یا 0X20 پر میشود و نکته قابل توجه این است که کارکتر '.' در این فضا وجود ندارد و سیستمی که در حال کار با این فضا است باید بجاي '.'به تعداد فضاي خالی کد 0X20 را قرار دهد و در هنگام خواندن نیز سه بایت آخر پسوند در نظر گرفته شده و فضاهاي خالی حذف بقیه اسم در نظر گرفته میشوند. بایت بعدي معروف به بایت هویت یا Attribute بوده که در آن اطلاعاتی از قبیل فایل /فولدر بودن یا hidden/file system/VID/read only/Archive مشخص میشود. جدول زیر توضیح این داده هاست:

Attrib Bit	Function	LFN	Comment
0 (LSB)	Read Only	1	Should not allow writing
1	Hidden	1	Should not show in dir listing
2	System	1	File is operating system
3	Volume ID	1	Filename is Volume ID
4	Directory	x	Is a subdirectory (32-byte records)
5	Archive	x	Has been changed since last backup
6	Ununsed	0	Should be zero
7 (MSB)	Ununsed	0	Should be zero

به طور مثال اگر فضاي 32 بایتی موجود مربوط به فولدر باشد بیت چهارم این داده یک میباشد .یا در داده‌هاي مربوط به موري point(12):32 یعنی بیت پنجم یک است پس فایل Archiveمی باشد:

چند داده بعد مربوط به آخرین زمان دسترسی و زمان تولید فایل میباشد:

Creation time and date	5 bytes	Time and date file was created.
Last access date	2 bytes	Date file was last accessed.
Last modification time and date	4 bytes	Time and date file was last modified.

بایت 27 و 28 مربوط به آدرس کلاستر شروع فایل یا فولدر میباشد . این آدرس مربوط به فضاي عمومی نبوده بلکه آدرس شروع در جدول تخصیص حافظه را نشان میدهد که توضیح خواهیم داد. از بایت 29 تا 32 نیز ظرفیت فایل را نمایش میدهد که در مثال فوق داریم 00 01 80 151 که ظرفیت فایل 86167 بایت خواهد بود.

نکته :براي پاك کردن فایل یا فولدر کافی است بجاي کاراکتر اول اسم آن عدد **0XE5 را قرار داد.** در این صورت سیستمی که فایل یا فولدري با شروع0XE5 را مشاهده کرد آن را نادیده میگیرد. درصورتیکه بایت اول هر 32 بایت صفر باشد سیستم باید آنرا آخر فولدر در نظر بگیرد و براي ایجادفایل یا فولدر جدید از اینجا شروع کند.

مد عملیاتی SPI

مد عملیاتی SPI به طور مشترک در کارت های MMC و SDC پشتیبانی می شود. کارت حافظه به سادگی از طریق واسط SPI با میکروکنترلرهایی که دارای این واسط می باشند ارتباط برقرار می کند. مد اطلاعاتی در SPI باید در حالت صفر (CPHA = 0, CPOL = 0) انتخاب شود. برای اطلاعات بیشتر در مورد واسط SPI به فصل دوازدهم مراجعه نمایید.

دستور و پاسخ در مد SPI

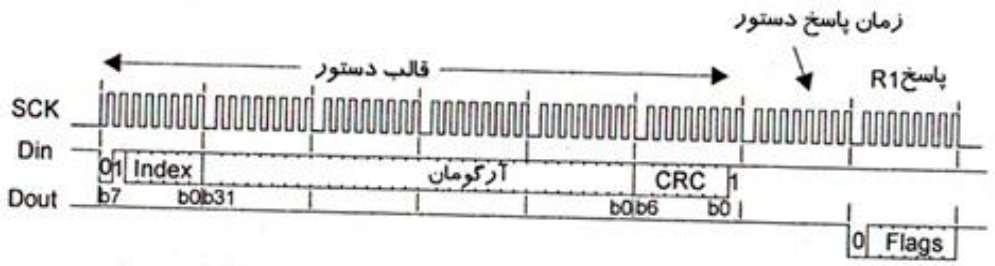
در مد SPI میزبان به عنوان Master و کارت حافظه به عنوان Slave شناخته می شود. بنابراین تولید کلاک روی پایه CLCK توسط

میزبان انجام می پذیرد. حداکثر فرکانس پالس ساعت قابل اعمال روی پایه CLCK حدود 20مگاهرتز است.

اصول ارتباط بین میزبان و کارت بر اساس ارسال دستور از طرف میزبان و پاسخ از طرف کارت انجام می شود. هنگامی که یک قالب دستور به کارت ارسال شود یک پاسخ به دستور (R1 , R2 یا R3) به میزبان باز گردانده خواهد شد.

قالب دستورات به صورت شش بایتی بوده و پس از ارسال، میزبان باید تا زمانی که یک پاسخ مؤثر را دریافت کند به خواندن بایت ها ادامه دهد. زمان پاسخ به دستور (NCR) صفر تا هشت بایت برای SDC و یک تا هشت بایت برای MMC است. سیگنال CS نیز باید در طول ارسال پایین (Low) نگه داشته شود. همچنین رشته CRC در مد انتخابی است اما از آن در هنگام ارسال قالب دستورات استفاده می شود.

شکل زیرمطالب فوق را به خوبی نشان می دهد.

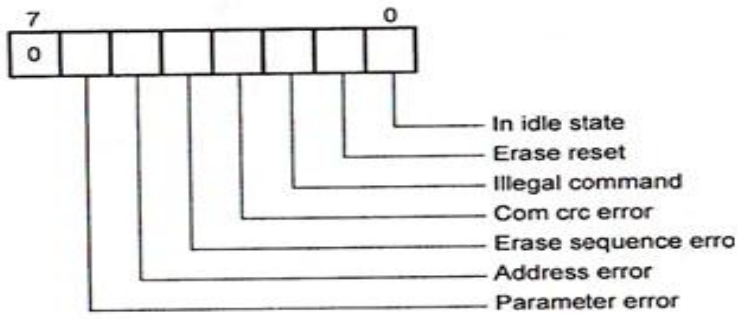


پاسخ در مد SPI

سه قالب پاسخ دستوری به نام های R1، R2 و R3 وجود دارد که با توجه به نوع دستور یکی از این پاسخ ها از طرف کارت ارسال می شوند.

فرمت پاسخ R1

این پاسخ برای بیشتر دستورات باز می گردد. فرمت پاسخ R1 در شکل زیرنشان داده شده است.



❖ In Idle State

یک بودن این بیت نشان دهنده قرار گرفتن کارت در وضعیت بیکار (Idle) است.

❖ Erase Reset

یک شدن این بیت نشان دهنده این است که دستور Erase خارج از مرحله دریافت شده است.

❖ Illegal Command

یک بودن این بیت بیانگر اشتباه بودن دستور ارسالی است.

❖ Communication CRC Error

نشان دهنده خطا در فیلد CRC آخرین دستور می باشد.

❖ Erase Seuence Error

نشان دهنده خطا در اجرای دستور پاک کردن متوالی (Erase Sequence) می باشد.

❖ Address Error

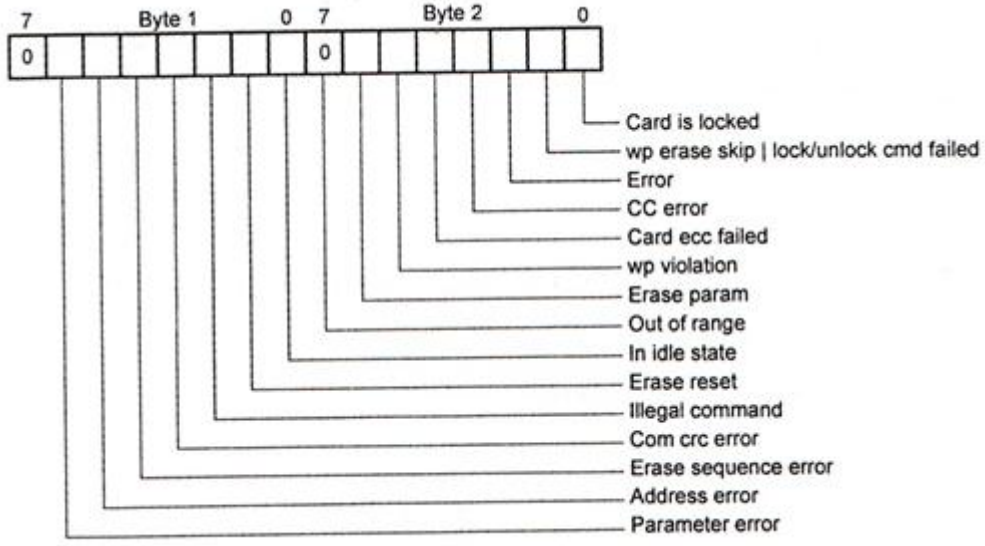
نشان دهنده خطا در آدرس می باشد.

❖ Parameter Error

نشان دهنده خطا در فیلد آرگومان می باشد، به طوری که آدرس یا طول بلوک ها خارج از محدوده کارت حافظه است.

فرمت پاسخ R2

پاسخ R2 دارای طول دو بایت (CMD13) SEND_ STATUS توسط کارت فرستاده می شود.



همان طور که در شکل فوق مشخص است، اولین بایت مشابه پاسخ R1 و محتوای بایت دوم به شرح زیر می باشد.

❖ Erase Param

این بیت زمانی که یک سکتور نامعتبر جهت پاک کردن توسط میزبان انتخاب شود، یک خواهد شد.

❖ Write Protect Violation

خطای محافظ نوشتن

❖ Card ECC Failed

یک شدن این بیت بیانگر این است که کد تصحیح خطا (ECC) در درون کارت موجب تصحیح اطلاعات نشده است.

❖ CC Error

یک شدن این بیت بیانگر خطا در کنترلر داخلی کارت می باشد.

❖ Error

هنگامی که یک خطای کلی یا ناشناخته در حین عملیات رخ دهد این بیت یک می شود

❖ Write Protect Erase Chip

یک شدن این بیت نشان دهنده این است که فقط بخشی از فضای آدرس پاک شده است.

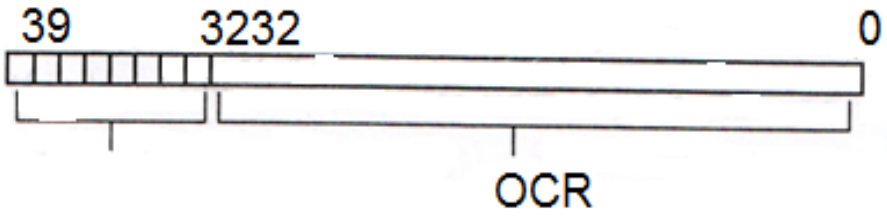
❖ **Card is Locked**

در هنگام قفل بودن کارت این بیت یک خواهد شد. قابلیت قفل در برخی از کارت ها پشتیبانی می شود.

مقدار 00H در پاسخ R1 به معنای موفقیت آمیز بودن است و هنگامی که خطایی رخ دهد، بیت مربوطه یک خواهد شد.

فرمت پاسخ R3

این پاسخ فقط یک دستور (CMD58) Read_OCR برگشت داده می شود.



همان طور که در شکل فوق مشخص است طول پاسخ R3 پنج بایت بوده که بایت اول مشابه پاسخ R1 و چهار بایت دیگر شامل محتویات رجیستر OCR می باشند.

*انتخاب مد SPI در کارت حافظه

بعد از باز نشانی یا روشن شدن MMC / SD (Power_ On) وارد مد عملکرد داخلی خود (مد SD Bus برای SD و مد MMC Bus برای MMC) می شود. جهت انتخاب مد SPI باید مراحل زیر را انجام داد.

1. **Power On:** پس از رسیدن ولتاژ تغذیه به 2/2 ولت، یک میلی ثانیه منتظر مانده و سپس پایه های CS و SDI را یک نموده و بیش از هفتاد و چهار پالس را به پایه SCLK اعمال می کنیم.
2. **ریست نرم افزاری:** جهت ریست کارت، دستور CND0 را به CS پایین ارسال می کنیم. در این هنگام کارت تا زمانی که CMD0 آشکار شود از سیگنال CS نمونه برداری می کند. اگر CS

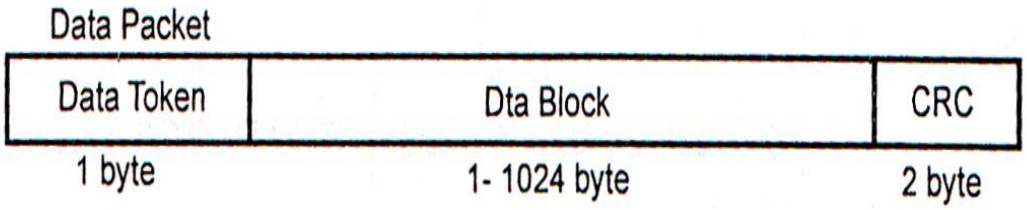
پایین بود، کارت وارد مد SPI می شود. باید توجه داشت رشته CRC در دستور CMD0 باید دارای یک مقدار معتبر (95H) باشد و پس از وارد شدن کارت به مد SPI تست CRC غیرفعال شده و مقدار CRC قابل چشم پوشی است. پس از پذیرفته شدن دستور CMD0 کارت وارد وضعیت Idle شده و بیت In Idle State در پاسخ R1 یک خواهد شد. (R1=01H)

3.مقداردهی اولیه: در وضعیت Idle کارت فقط دستورات CMD0، CMD1 و CMD58 را پذیرا است و دستورات دیگر مردود می باشند. هنگامی که کارت یک CMD1 را آشکار نمود، شروع به مقداردهی اولیه می کند. به منظور پایان دادن به مقداردهی، کنترلر میزبان بایستی مجدداً دستور CMD1 را ارسال نماید و پاسخ را بررسی کند. هنگامی که کارت با موفقیت مقداردهی اولیه شد، وضعیت Idle در پاسخ R1 پاک می شود (R1=00H)

فرآیند مقداردهی اولیه ممکن است چند میلی ثانیه به طول بیانجامد این مقدار بستگی به ظرفیت کارت دارد. پس از مقداردهی اولیه می توان عمل Read و Write را انجام داد.

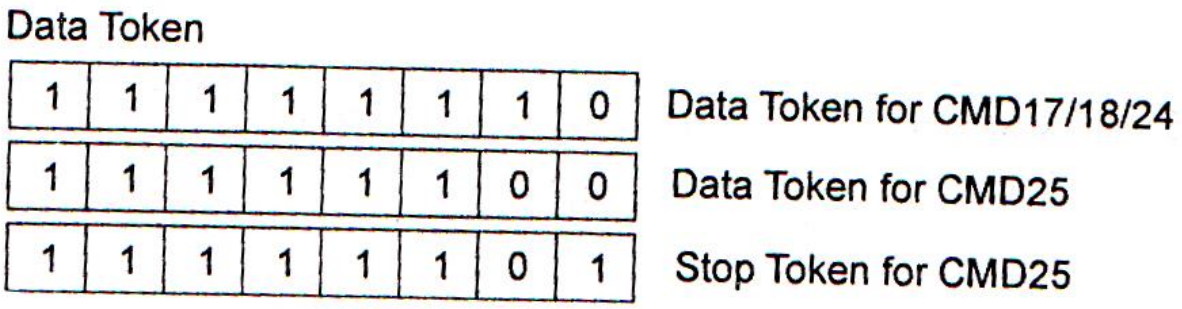
*** انتقال داده**

در یک انتقال داده یک یا بیشتر بلوک های داده بعد از پاسخ ارسال یا دریافت می شوند. بلوک داده به عنوان یک بسته اطلاعات همراه با بایت نشانه داده (data Token) و دو بایت CRC ارسال می شود. فرمت پاکت داده مطابق شکل زیر است.



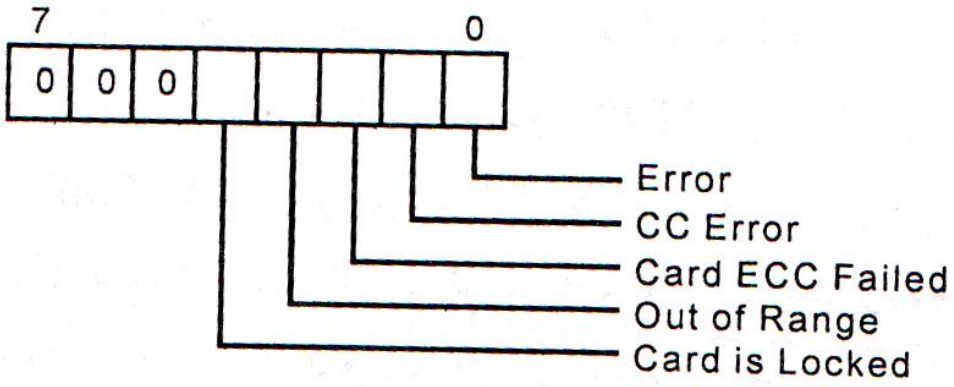
***نشانه های داده (Data Tokens)**

قالب بایت نشانه داده برای دستورات مختلف در شکل 6 نشان داده شده است.



*نشانه خطا (Error Token)

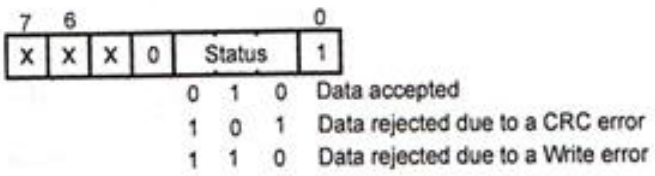
اگر عملیات خواندن ناموفق باشد و کارت نتواند اطلاعات لازم را ارسال کند، با ارسال یک بایت به عنوان نشانه خطا این موضوع را به اطلاع میزبان می رساند.



*پاسخ داده (Data Response)

هر مرحله نوشتن داده در بلوک با ارسال پاسخ داده توسط کارت تصدیق خواهد شد.

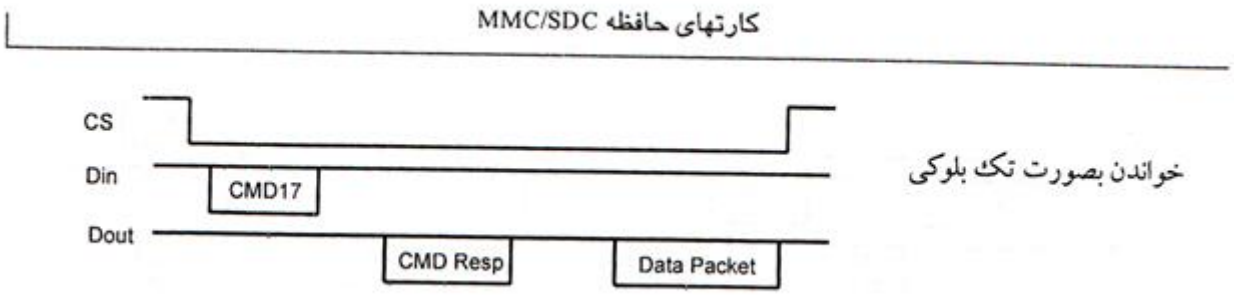
پاسخ داده دارای طول یکبایت مطابق شکل زیر می باشد.



*خواندن تک بلوکی (سکتور)

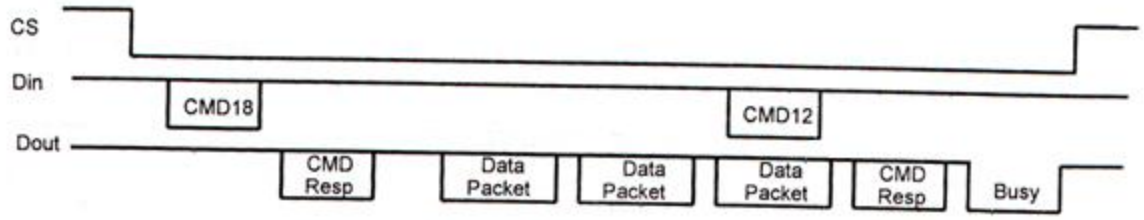
خواندن به صورت تک بلوکی توسط دستور CMD17 انجام می شود، برای این منظور در قسمت آرگومان از دستور CMD17، آدرس سکتور مورد نظر را قرار داده و با ارسال آن به کارت، یک وضعیت خواندن آغاز شده و بلوک داده به میزبان ارسال می شود.

اندازه بلوک (سکتور) به صورت پیش فرض 512 بایت می باشد که به کمک دستور CMD16 قابل تغییر است. چنانچه در طول عملیات خواندن هر خطایی رخ دهد، یک نشانه خطا به جای بسته داده برگشت داده می شود.



*خواندن بلوک های متوالی

دستور CMD18 چندین بلوک را به طور متوالی از آدرس تخصیص یافته می خواند. عملیات خواندن تا هنگامی که دستور CMD12 آشکار نشود ادامه خواهد یافت.

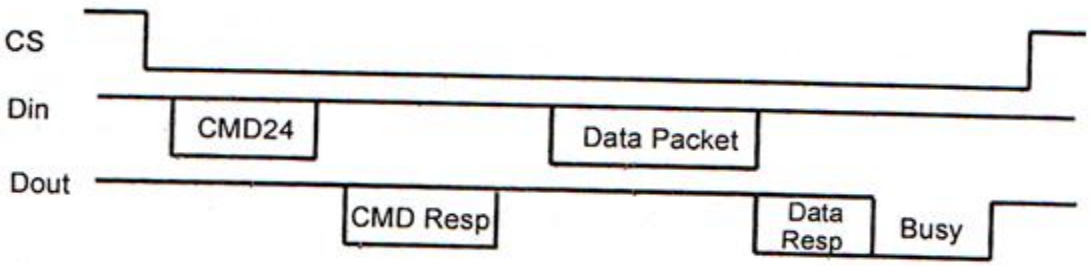


خواندن بصورت متوالی

*نوشتن تک بلوکی

هنگامی که یک دستور نوشتن پذیرفته می شود، میزبان یک بسته داده را به کارت ارسال می کند. قالب بسته مشابه دستور خواندن بلوک است. پس از این که یک بسته داده ارسال شد، کارت یک پاسخ داده (Data Response) را به دنبال بسته داده ارسال می کند.

معمولاً اندازه بلوک (سکتور) در اکثر کارت ها 512 بایت می شود که در برخی از کارت ها قابلیت تغییر آن وجود دارد.



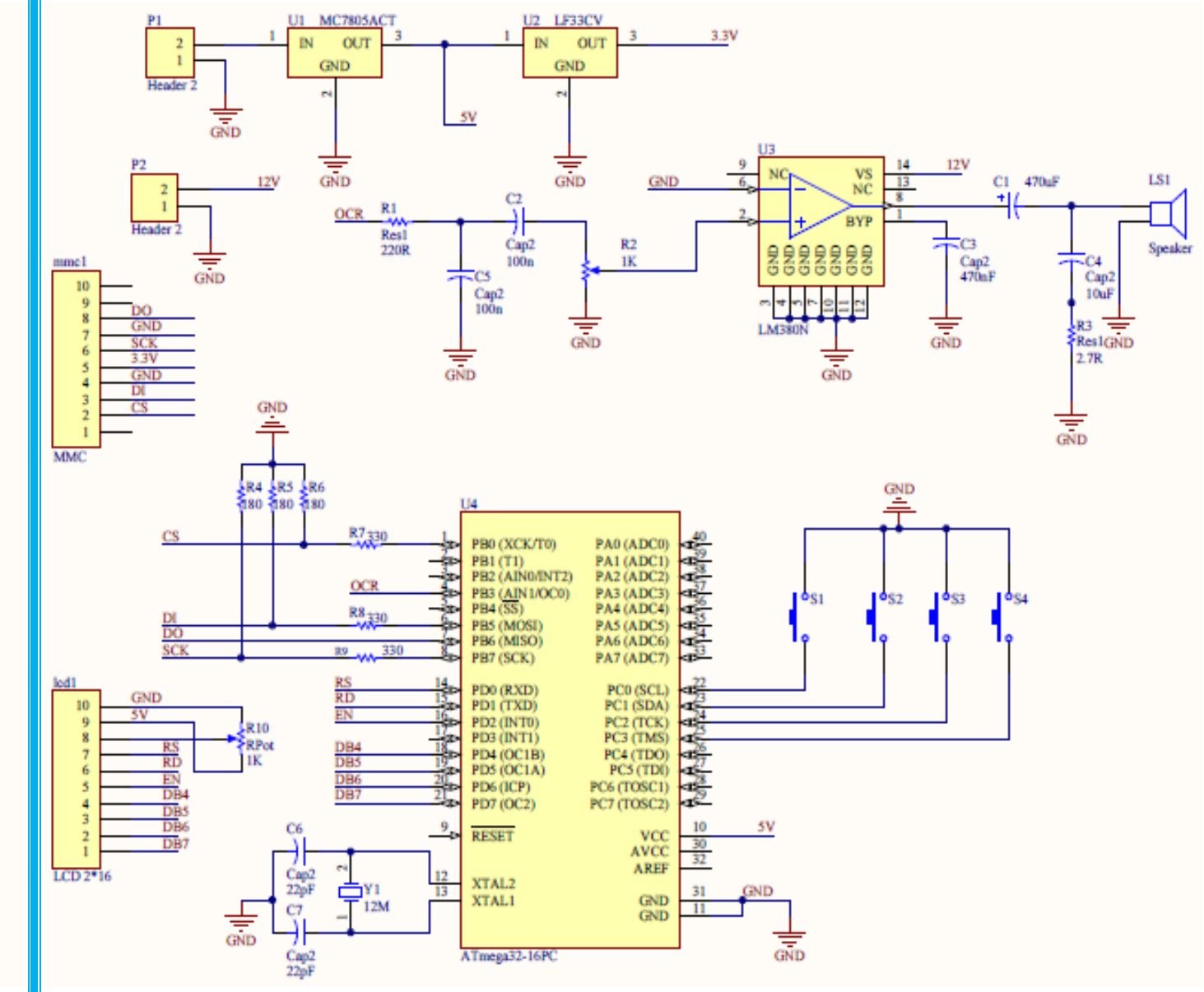
نوشتن تک بلوکی

*نوشتن بلوک های متوالی

دستور CMD25 چندین بلوک را به طور متوالی در آدرس تخصیص یافته می نویسد. عملیات نوشتن تا زمانی که نشانه STOP قطع نشود ادامه دارد.

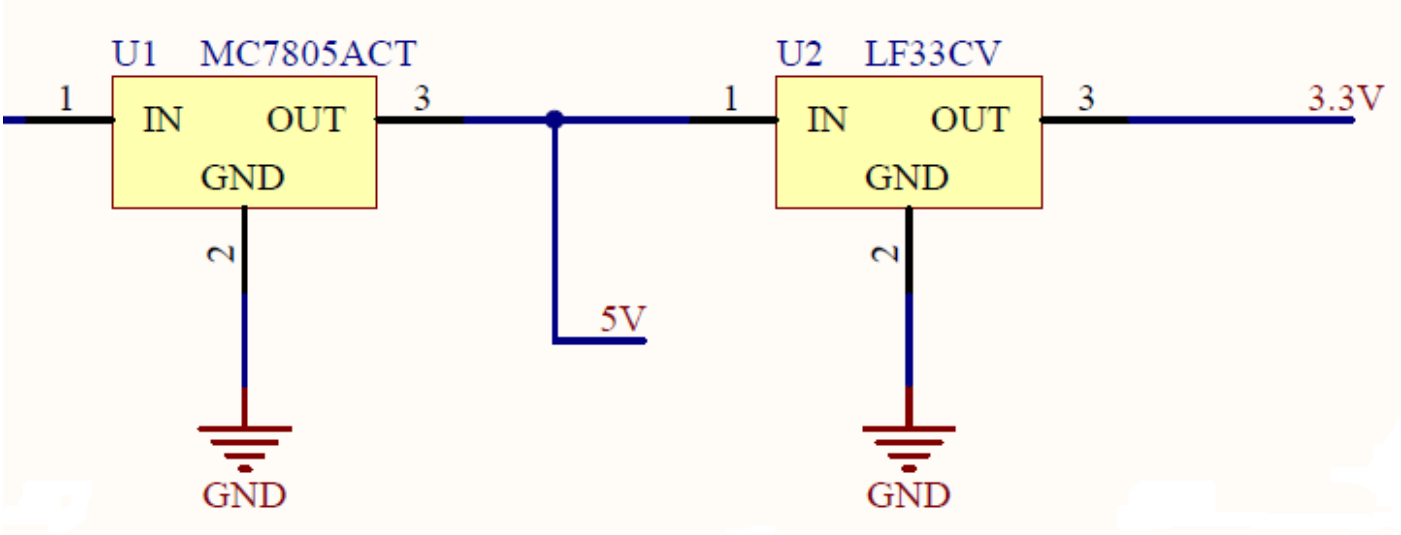
مدار طراحی شده برای پروژه:

درشکل زیر مدار طراحی شده برای پروژه موزیک پلیر با حافظه ی MMC آمده است:



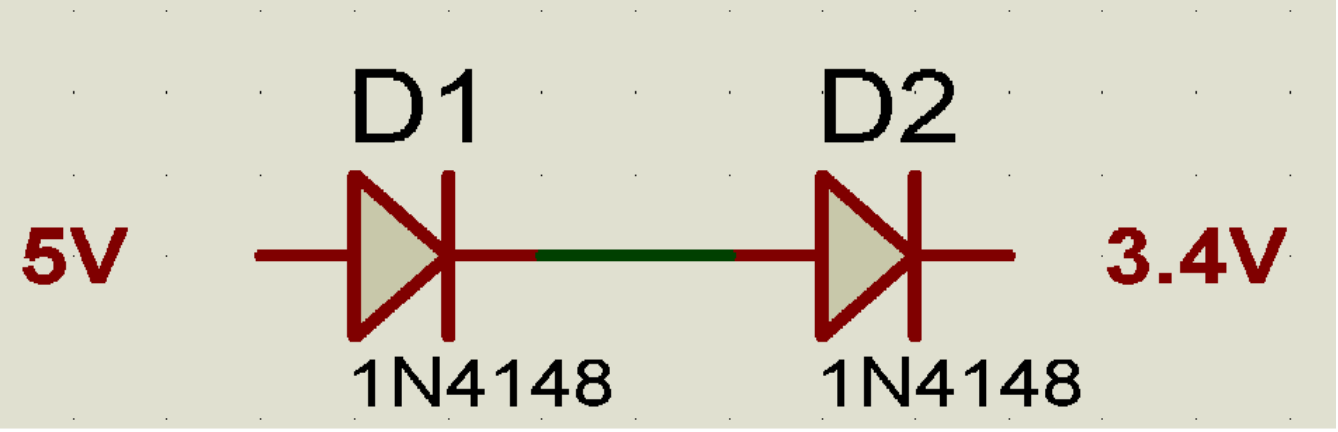
حال به بررسی انواع بخش های پروژه می پردازیم:

1)مدار رگولاتور برای تامین ولتاژ کاری مدار.

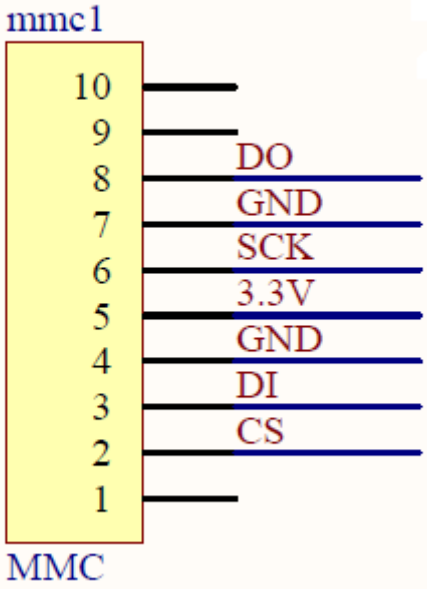


مدار فوق بخش تامین ولتاژ تغذیه ی میکرو و RAM MMC است.

با استفاده از یک رگولاتور 7805 میتوان یک ولتاژ 5ولتی را برای تغذیه ی میکرو استفاده کرد ولی برای کارکرد RAM یک ولتاژ 3.3ولتی نیاز است که با استفاده از IC رگولاتور LF33 این ولتاژ تامین شده است .دلیل استفاده نکردن از تقسیم ولتاژ مقاومتی این است که RAM MMC در لحظه ی روشن شدن جریان زیادی تا 1A می کشد که مقاومت نمی تواند این جریان را تحمل کند لذا هم مقاومت خواهد سوخت وهم مدار تثبیت ولتاژ خوبی ندارد . راه کم هزینه تری هم وجود دارد و استفاده از دو دیود به صورت سری است به 5 ولت متصل میشود.و خروجی آن 3.4 ولت میدهد اما پیشنهاد نمیشود زیرا درمواقعی MMC جریان لحظه ای حدود 1 آمپر میکشد و باعث میشود که موری در موقع راه اندازی دچار مشکل شود و راه اندازی نشود.بهترین و مطمین ترین راه استفاده از رگولاتور است.



2)بخش حافظه ی پروژه .



درشکل روبه رو پایه های 1,9,10به صورت آزاد هستند چون این پایه ها در MODE MMC استفاده می شوند ودر MODE SPI به صورت آزاد می باشند.

پایه ی DO : (Data Out)

این پایه به (Master Input Slave Output)MISOمیکروکنترلر متصل میشود .

پایه ی DI : (Data In)

به پایه ی MOSI(Master Out Slave In) متصل میشود.

پایه ی SCLK:

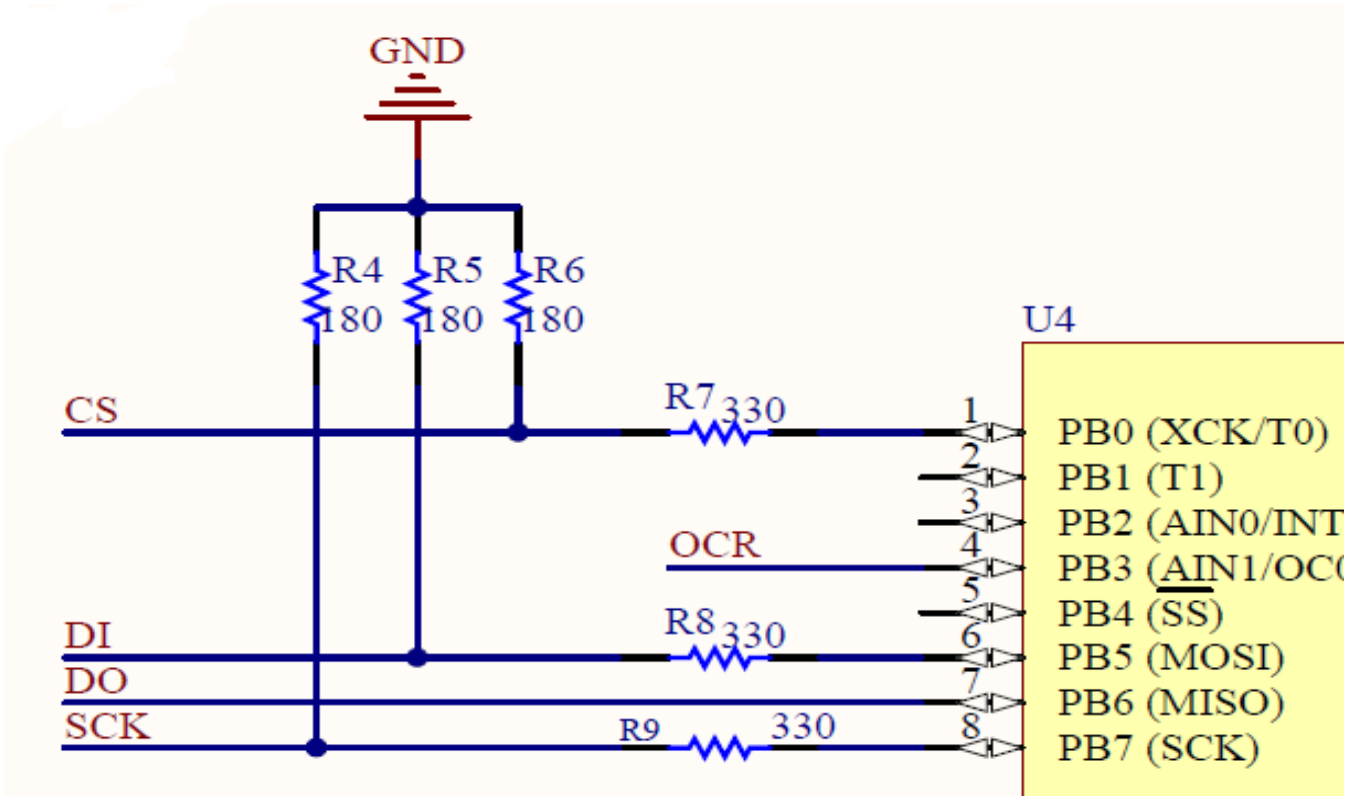
کلاک موری میباشد و به پایه ی SCK متصل میشود.کلاک موری تا 25MHZمیتواند باشد

پایه ی CS : (CHIP SELECT)

این پایه برای انتخاب حافظه ی مورد نظر از بین سایر حافظه ها یا مدارات متصل به میکرو از طریق رابط SPI است.

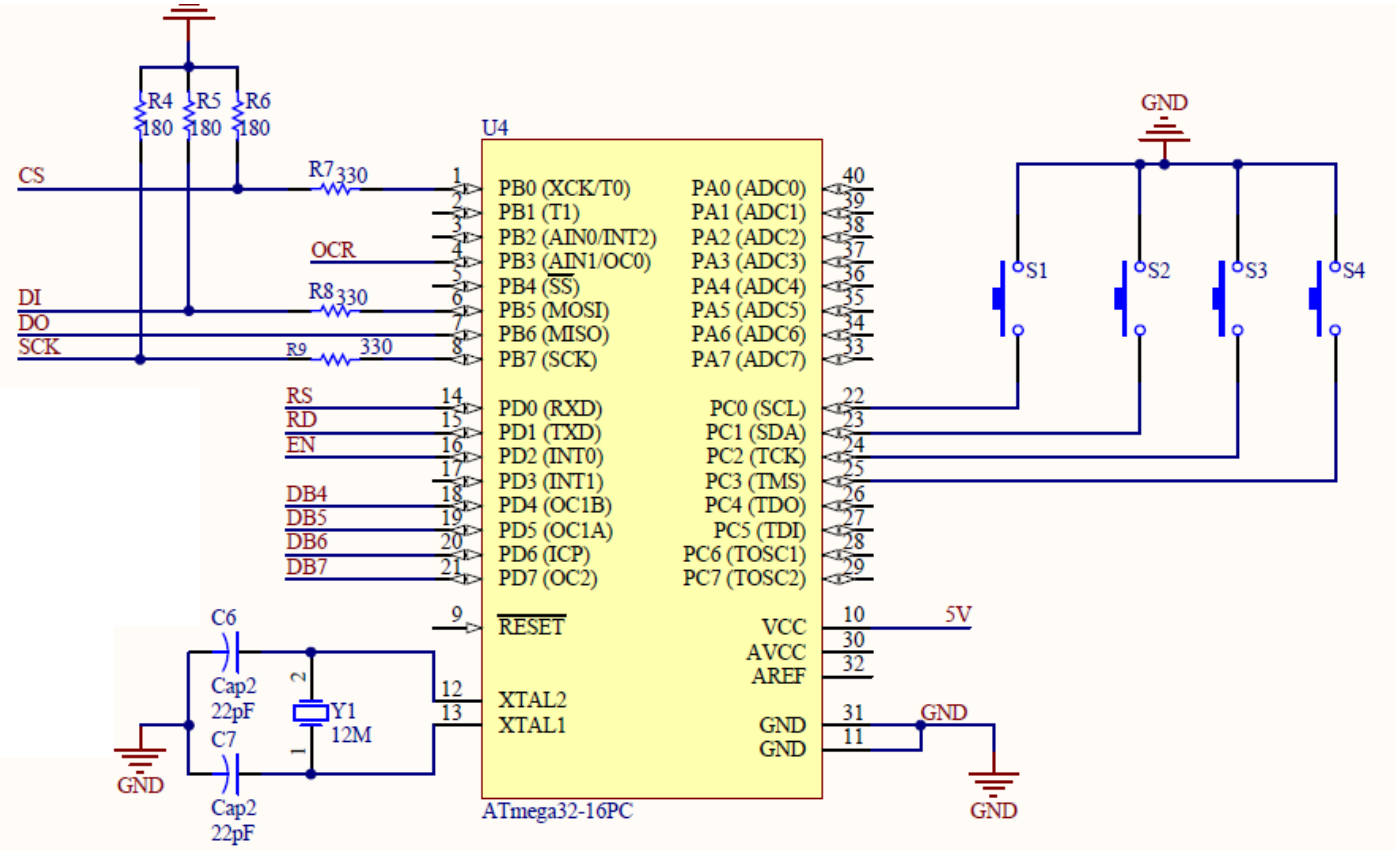
به این صورت که وقتی میکرو میخواهد یک حافظه را انتخاب کند با صفر کردن این پایه حافظه مورد نظر را انتخاب میکند تا برای تبادل اطلاعات با میکرو آماده شود.

همانطور که گفته شد تغذیه ی موری 3.3V است بنابراین اگر ولتاژ بیشتر شود باعث صدمه به موری میشود. منطق میکرو کنترلر 0V و 5V است.بنابراین اگر پایه هایی از میکرو که باید دیتا را به موری بدهد چون ولتاژش5 ولت است باعث صدمه به موری میشود بنابراین باید این ولتاژ را توسط تقسیم مقاومتی به 3.3 برسانیم توجه شود فقط پایه هایی که باید دیتا را از میکرو به موری بدهد تقسیم مقاومتی شود و پایه هایی از موری مثل که باید دیتا را از موری به میکرو بدهد نیازی به تقسیم مقاومتی ندارد زیرا در منطق میکرو ولتاژ 3.3 یک منطقی تعریف شده است.



پایه هایی که از موری به میکرو فرمان می دهد مثل DO می تواند مستقیم متصل شود زیرا 3.3 در منطقه تعریف شده وجود دارد.

3)بخش کنترل کننده ی مدار و قسمت تولید صوت.



کنترل مدار فوق برعهده ی میکرو ی AT MEGA 32 می باشد.

اساس ساختار صوت به صورت آنالوگ است که برای ذخیره شدن روی حافظه باید به کدهای دیجیتالی تبدیل شود که این کار توسط کامپیوتر وبا نمونه برداری از سیگنال آنالوگ وتبدیل آن به دیجیتال انجام می پذیرد. میکروی فوق وظیفه ی خواندن اطلاعات از حافظه را برعهده دارد.میکرو این اطلاعات دیجیتال را که در حافظه اند راازطریق رابط SPI گرفته وبه سیگنال آنالوگ تبدیل کرده و به خروجی تحویل میدهد.این دریافت وتبدیل داده ها ازحافظه در چند مرحله انجام می پذیرد که در زیر آورده شده است:

ابتدا میکرو توسط پایه CS, RAM را فعال سازی می کند سپس اطلاعات را توسط پایه DI به RAM ارسال می کند تا RAMخودش را جهت تبادل اطلاعات ونحوه ی ارتباط بامیکرو و انتخاب سکتور های مورد نظری که داده ها روی آن ذخیره شده اند آماده کند به عبارت دیگر میکرو توسط این پایه به RAMفرمان می دهد.توسط پایه ی SCKوپالس های آن RAM خود را بامیکرو سنکرون میکند یعنی به طور همزمان بامیکرو کار می کند.پایه یDO نیز برای دریافت داده ها از RAM می باشد که در ادامه به صورت نرم افزاری توضیح داده خواهد شد.

برنامه ی نوشته ی شده برای میکرو:

```
#include <mega32.h>
#include <alcd.h>
#include <delay.h>
#include <spi.h>
#include <stdio.h>
#include "mmclib/mmc.h"
void m_play(void);
void m_next(void);
void m_pause(void);
void m_back(void);

char buffread[512];
char a;
bit s=1;
int i;
unsigned long int m_begin=81891,m_end=94000;
void main(void){

DDRB=(1<<DDB0)|(1<<DDB4)|(1<<DDB5)|(1<<DDB7)|(1<<DDB3);
DDRC=0X00;
PORTC=0XFF;
SPCR=0x50;

SPCR=0x50;
SPSR=0x01;
TCCR0=0x00;// TCCR0=0x00//TCCR0=0x69
TCNT0=0x00;
OCR0=0x00;
TIMSK=0x00;
lcd_init(16);
```

```
  while(s==1){
    s=mmc_init();
    lcd_gotoxy(0,0);
    lcd_putsf("searching mmc");
    }
  lcd_clear();
  lcd_gotoxy(0,0);
  lcd_putsf("card detect ok");
  delay_ms(200);
  lcd_clear();
  lcd_gotoxy(0,0);
  lcd_putsf("Searching");
  delay_us(20);
  lcd_gotoxy(0,1);

  lcd_putsf("music player");
   delay_ms(100);
 while(1){
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf("press play key");
   delay_us(20);
   lcd_gotoxy(0,1);
   lcd_putsf(" music player");

  if(PINC.0==0){
  while(PINC.0==0);
  m_play();
  }

   }

}


////////////////////////////////////////
 void m_play(void){
 TCCR0=0x69;
 lcd_clear();
 lcd_gotoxy(0,0);
 lcd_putsf("playing");
 delay_us(20);
 lcd_gotoxy(0,1);
 lcd_putsf("music player");
```

```
 while(1){
mmc_read(m_begin,buffread);
    for(i=0;i<512;i++){

    while(!(TIFR&0X02));
    TIFR |=0X02;
    OCR0=buffread[i];
     if(PINC.0==0){
     while(PINC.0==0);
     delay_ms(35);
     m_pause();
     }
     if(PINC.1==0){
     while(PINC.1==0);
     delay_ms(30);
     m_next();
     }
     if(PINC.2==0){
     while(PINC.2==0);
     delay_ms(30);
     m_back();
     }

      }
   m_begin++;
   if(m_begin==m_end) m_next();

   }
  }
////////////////////////////////
  void m_pause(void){
  TCCR0=0x00;
  lcd_clear();
  lcd_gotoxy(0,0);
  lcd_putsf("pause");
  delay_us(20);
  lcd_gotoxy(0,1);
  lcd_putsf("  music player");
     while(1){
     if(PINC.0==0){
     while(PINC.0==0);
     delay_ms(30);
     m_play();
     }
  }
 }
```

```
//////////////////////////////////////
     void m_next(void){
     a++;
     if(a==8) a=1;

      switch(a){
case  1:m_begin=124507;m_end=134979;lcd_clear();lcd_putsf("1");break;
case  2:m_begin=69419 ;m_end=81827 ;lcd_clear();lcd_putsf("2");break;
case  3:m_begin=55475 ;m_end=69395 ;lcd_clear();lcd_putsf("3");break;
case  4:m_begin=50475 ;m_end=55451 ;lcd_clear();lcd_putsf("4");break;
case  5:m_begin=22147 ;m_end=38440 ;lcd_clear();lcd_putsf("5");break;
case  6:m_begin=8475  ;m_end=22075 ;lcd_clear();lcd_putsf("6");break;
case  7:m_begin=81883 ;m_end=94115 ;lcd_clear();lcd_putsf("7 ");break;
default:m_begin=124507;m_end=134979;lcd_clear();lcd_putsf("8")");
        }

     }
    }


   void m_back(void){
       a--;
       if(a==0) a=7;
      switch(a){
case 1:  m_begin=124507;m_end=134979;lcd_clear();lcd_putsf("1");break;
case 2:  m_begin=69419;m_end=81827;lcd_clear();lcd_putsf("2");break;
case 3:  m_begin=55475;m_end=69395;lcd_clear();lcd_putsf("3");break;
case 4:  m_begin=50475;m_end=55451;lcd_clear();lcd_putsf("4");break;
case 5:  m_begin=22147;m_end=38440;lcd_clear();lcd_putsf("5");break;
case 6:  m_begin=8475;m_end=22075;lcd_clear();lcd_putsf("6");break;
case 7:  m_begin=81883;m_end=94115;lcd_clear();lcd_putsf("7 ");break;
default: m_begin=124507;m_end=134979;lcd_clear();lcd_putsf("8");
       }



   }
```

در برنامه ی فوق خطوط اول برای تعریف میکرو وLCD ومد SPI وکتابخانه ی MMC می باشد.

درخطوط بعدی 4عدد تابع تعریف شده که برای خواندن و مکث و جلو رفتن به آهنگ بعدی به کار می رود.سپس چند متغیر تعریف شده که در متن برنامه از آن ها استفاده شده است.در تابع اصلی ابتدا رجیستر ها ی SPI وتایمر و وقفه را مقدار دهی می کنیم .در متن برنامه از کتابخانه ی MMC نیز استفاده شده است که در ادامه توضیح داده خواهد شد.

وارد حلقه ی اصلی برنامه می شویم.در متن اصلی با یک متغیری سرو کار داریم که مقدار آن متغیر از

تابع MMC خوانده می شودوسپس تصمیمات لازم برای مقدار آن توسط متن برنامه انجام می گیرد.

برای مدار 3عددکلید تعریف شده است که این 3کلید کار پخش وتوقف و آهنگ قبلی وبعدی را بر عهده دارند مثل با فشردن PINC.0 ترانه شروع به خواندن می کند ویا متوقف می شود.

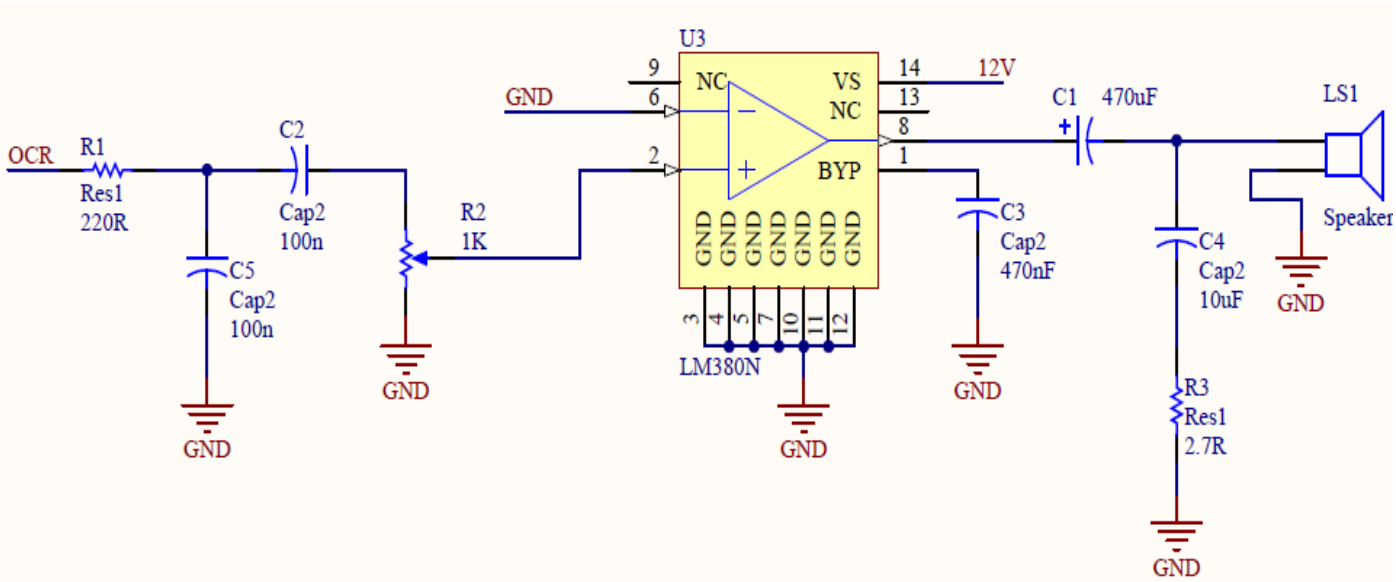
وقتی MMCرا به میکرو متصل کردیم میکرو پس از شناسایی MMC عملیات مربوط به خواندن را شروع می کند.وقتی وارد تابع m_play شد از سکتور شروع آهنگ (m_begin) 512بایت می خواند و با یک حلقه ی forاین بایت های خوانده شده را در آرایه ی buffreadمی ریزد ودر هرلحظه OCR0 را با مقدار آرایه برابر می کند. در ادامه عملکرد کلید ها تعریف شده اند که چه کاری با فشردن آنها انجام گیرد.اگر کلید PINC.0 فشرده شودوارد این تابع (m_pause)میشود.این تابع جهت خواندن ومکث استفاده می شود.وقتی وارد تابع شد تایمر را غیر فعال می کند وسپس عباراتی روی LCDنمایش می دهد و وقتی کلید PINC.0 دوباره فشرده شد به تابع m_playبرمیگردد وخواندن را ادامه می دهد.در تابع m_next وقتی هربار که وارد تابع می شود یک عدد به مقدار متغیر a می افزاید وبسته به مقدار این متغیر وارد یکی از case ها می شود. در این case ها نام ومشخصات داده های موجود در MMC نوشته شده اند که میکرو بر حسب این ها از MMC می خواند.

تابع m_back نیز همانند تابع m_next است منتها در این تابع از مقدار aکاسته می شود.

فرکانس کار میکرو:

میکرو با فرکانس یک اسیلاتور کریستالی 12مگاهریز بیرونی کار می کند.

بخش طبقه صوت مدار:



بخش طبقه ی صوت برای آشکار سازی صوت به کار رفته است .میکرو پس از خواندنداده های موجود در MMC آنها را در پایه ی OCR0 قرار می دهد.چون داده ها در پایه ی OCR0به صورت پالس می باشند لذا باید به صورت آنالوگ تبدیل شوند که ای کار توسط فیلتر پایین گذر (R1وC5) انجام میگیرد .برای تقویت این سیگنال از OP AMP LM380 استفاده شده است.صوت پس از تقویت به ورودی بلند گو ارسال می گردد. speaker موجود 8اهمی انتخاب شده است زیرا خروجی OPAMPنیز دارای مقاومت 8اهم می باشد.