**پیش­گزارش دستورکار هفتم آزمایشگاه ریزپردازنده و زبان اسمبلی**

نگار موقتیان، مریم موسوی – گروه 3

1. **کاربرهای EEPROM و تفاوت آن با حافظه­های RAM و Flash**

حافظۀ EEPROM برخلاف حافظۀ RAM یک حافظۀ دائمی می­باشد، به این معنا که با قطع شدن تغذیۀ آن و خاموش شدن مدار اطلاعات داخل آن باقی می­ماند. حافظۀ Flash نیز یک حافظۀ دائمی می­باشد، اما مزیت EPPROM نسبت به آن این است که نوشتن داده­ها می­تواند بایت به بایت انجام شود، در حالی که در حافظه­های Flash تنها می­توانیم داده­ها را به صورت بلوکی نوشته و بخوانیم (و به همین دلیل از آن­ها برای نگهداری برنامۀ میکروکنترلر استفاده می­شود که آن را یکباره نوشته و یکباره پاک می­کنیم). به علاوه تعداد دفعاتی که می­توانیم بر روی این حافظه­ها داده بنویسیم از حافظۀ Flash بیش­تر است.

در نتیجه زمانی که می­خواهیم داده­هایی مانند تنظیمات کاربر را زمان اجرا ذخیره کنیم، اما بتوانیم پس از قطع شدن تغذیۀ مدار نیز دوباره به آن­ها دسترسی داشته باشیم از EEPROM استفاده می­کنیم.

1. **فرآیند نوشتن در حافظه­های Flash**

همانطور که در قسمت قبل نیز اشاره شد نوشتن بر روی حافظه­های Flash باید بلوک به بلوک انجام شود (در این نوع از حافظه­ها به طور معمول از بلوک­های 4KB ای استفاده می­شود). برای نوشتن یک بایت بر روی این حافظه اگر مقدار روی حافظه یک باشد می­توانیم آن را به صفر تغییر دهیم اما بالعکس این کار امکان پذیر نیست و باید کل بلوک حاوی داده را پاک کرده و آن را دوباره بنویسیم.

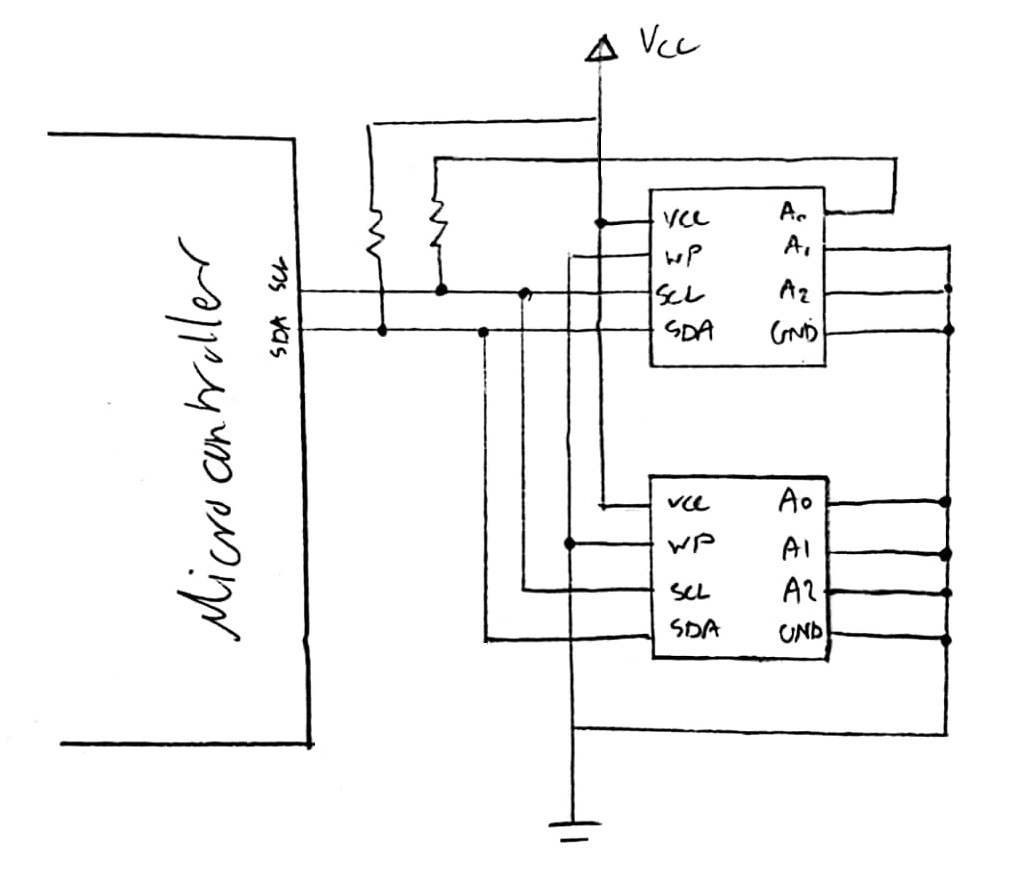
برای اینکه دیگر داده­های قبلی نوشته شده بر روی حافظه طی این فرآیند از بین نرود می­توانیم کل بلوک حافظه را خوانده و در یک بافر ذخیره کنیم، سپس دادۀ دلخواه را بر روی این بافر تغییر داده و دوباره کل محتوای بافر را بر روی بلوک فوق بنویسیم. بدیهیست این فرآیند زمانبر بوده و کارا نیست، بنابراین برای ذخیره­سازی چنین داده­هایی بهتر است از حافظه­های EEPROM استفاده کنیم.

1. **اگر یک حافظۀ EEPROM بیرونی دارای 4KB حافظه و 2 پایۀ آدرس باشد، در این صورت می­توان حداکثر چند KB حافظۀ EEPROM بیرونی بر روی یک باس مشترک داشت؟**

برای این که این حافظه­های بیرونی بتوانند بر روی یک باس مشترک کار کنند باید آدرسی یکتا داشته باشند تا میکروکنترلر بتواند در هر لحظه مشخص کند بر روی کدام یک می­خواهد داده بنویسد و یا از کدام یک داده بخواند. از آن­جایی که حافظۀ فوق 2 پایۀ آدرس دارد، می­توان 4 آدرس یکتا به آن تخصیص داد، بنابراین می­توانیم 4 تا از این حافظه­ها را به میکروکنترلر متصل کنیم. در نتیجه در نهایت 16KB حافظۀ EEPROM بیرونی خواهیم داشت.

1. **اتصال دو AT24C02 به میکروکنترلر توسط یک باس مشترک**

به یکی از حافظه­های AT24C02 آدرس 000 و به دیگری آدرس 001 می­دهیم، از این طریق میکروکنترلر می­تواند مشخص کند که بر روی کدام یک از حافظه­ها می­خواهد داده بنویسد و یا از کدام یک می­خواهد داده بخواند.



1. **همخوانی دنبالۀ فریم­های خواندن از حافظه و پروتکل TWI**

در پروتکل TWI برای شروع ارتباط یک سیگنال START توسط master فرستاده شده و برای پایان آن نیز یک سیگنال STOP فرستاده می­شود. سپس به صورت سریال آدرس slave ای که master می­خواهد با آن ارتباط برقرار کند بر روی باس داده قرار می­گیرد. پس از آن یک تک بیت فرستاده می­شود که مشخص می­کند master می­خواهد داده­ای را بخواند یا بنویسد (مقدار صفر برای نوشتن و یک برای خواندن). حال slave یک بیت ACK می­فرستد. سپس داده­ها انتقال می­یابند. زمان نوشتن master به ازای هر بایت یک ACK دریافت می­کند تا زمانی که بخواهد به ارتباط خاتمه دهد، زمان خواندن نیز slave داده­ها را به master ارسال کرده و master به ازای هر بایت داده یک ACK می­فرستد، جز برای بایت آخر که نفرستادن ACK آن به معنای این است که master دیگر نمی­خواهد داده­ای بخواند.

برای نوشتن و خواندن داده از روی حافظۀ EEPROM فوق نیز روندی مشابه طی می­شود.

زمان نوشتن ابتدا master ارتباط را شروع کرده، آدرس حافظۀ مورد نظر را مشخص کرده، و با فرستادن صفر مشخص می­کند می­خواهد بر روی slave داده بنویسد. سپس یک ACK از slave دریافت کرده و آدرس دادۀ مورد نظر را بر روی باس قرار می­دهد. پس از دریافت ACK آدرس داده را بر روی باس گذاشته و با دریافت ACK بعدی ارتباط را خاتمه می­دهد.

زمان خواندن نیز ابتدا master ارتباط را شروع کرده، آدرس حافظۀ مورد نظر را مشخص کرده، و با فرستادن صفر مشخص می­کند می­خواهد بر روی slave داده بنویسد. سپس یک ACK از slave دریافت کرده و آدرس دادۀ مورد نظر را بر روی باس قرار می­دهد. پس از دریافت ACK ارتباط دیگری را شروع کرده و اینبار پس از مشخص کردن آدرس حافظۀ مورد نظر مشخص می­کند که می­خواهد از روی آن داده­ای را بخواند. سپس یک ACK ارسال کرده و داده را از slave دریافت می­کند. این بار اما ACK ارسال نمی­کند و از این طریق اعلام می­کند که داده­ای دیگری را نمی­خواهد بخواند. سپس با فرستادن سیگنال STOP ارتباط را خاتمه می­دهد.

1. **اطلاعات مربوط به فرکانس کلاک**

فرکانس کلاک همواره توسط master (در این­جا میکروکنترلری که از آن استفاده می­کنیم) مشخص شده و توسط master تولید می­شود.

طبق پروتکل توضیح داده شده نوشتن هر بایت بر حافظه 29 کلاک زمان می­برد. بنابراین در هر ثانیه می­توان تقریباٌ 344 بایت بر روی این حافظه نوشت.

البته این مقدار در واقعیت کمتر است، زیرا نوشتن داده­ها بر روی حافظه نیز زمان می­برد. با توجه به آن­چه در دستورکار اشاره شده­است نوشتن بر حافظه در حداکثر 5 میلی­ثانیه انجام می­شود (و این زمان نسبت به 29 کلاک صرف شده برای برقراری ارتباط، ارسال داده و خاتمۀ آن بیش­تر است)، لذا نوشتن هر بایت از داده بر روی حافظه 50 کلاک زمان می­برد.

1. **توابع کتابخانۀ Wire**

* begin(): ارتباط master و slave را از طریق پروتکل I2C آغاز می‌کند. اگر پارامتری نداشته باشد دستگاه فوق به عنوان master عمل می‌کند و در غیر این صورت پارامتر وارد شده آدرس slave را تعیین می‌کند.
* setClock(): برای تعیین فرکانس ارتباط استفاده می‌شود.
* beginTransmission(): ارتباط را برای شروع ارسال داده به آدرس داده شده آغاز می‌کند.
* write(): داده را بر روی باس می‌نویسد (با فرض اینکه ارتباط پیش از این برقرار شده­است).
* endTransmission(): ارتباط را طبق پروتکل I2C پایان می‌دهد.
* requestFrom(): برای درخواست خواندن داده از slave توسط master صدا زده می‌شود.
* available(): تعداد بایت‌هایی که آماده دریافت توسط read() هستند را می‌دهد.
* read(): بایت­های ارسال شده بر روی باس را می‌خواند.

با توجه به این دستورات می­توان عملیات خواندن و نوشتن از حافظه را مانند زیر انجام داد.

**Write:**

Wire.beginTransmission(DEVICE\_ADDRESS);

Wire.write(WORD\_ADDRESS);

Wire.write(DATA);

Wire.endTransmission();

**Read:**

Wire.beginTransmission(DEVICE\_ADDRESS);

Wire.write(WORD\_ADDRESS);

Wire.endTransmission();

Wire.requestFrom(DEVICE\_ADDRESS, SIZE);

Wire.read();