"به نام یزدان پاک"

پیش گزارش آزمایش هفتم

اعضای گروه:

آرتا اسدی حقی 9731006

کیانا آقاکثیری 9831006

سارا تاجرنیا 9831016

تاریخ تحویل پیش گزارش :1400/9/20

پرسش: در چه کاربردهایی EEPROM به کار برده می‌شود؟ چرا در اینجا حافظه Flash یا RAM را به کار نمی‌بریم؟ تفاوت حافظه RAM با EEPROM در چیست؟

تفاوت اصلی حافظه EEPROM با Flash در این است که در حافظه EEPROM میتوان بصورت بایت به بایت داده را حذف کرد اما در حافظه Flash حتما باید یک block کامل حذف شود. حافظه RAM هم برخلاف EEPROM توانایی نگهداری داده‌هایش پس از قطع شدن برق را ندارد و حافظه فرار محسوب می‌شود و برای نگهداری داده‌های موقتی CPU استفاده می‌شود.

پس در کاربردهایی که میخواهیم داده‌ای را برای مدت زمان طولانی و با قابلیت غیرفرار بودن نگهداری کنیم و از طرفی دستمان برای حذف کردن بایت به بایت داده‌ها باز باشد، از EEPROM استفاده می‌کنیم.

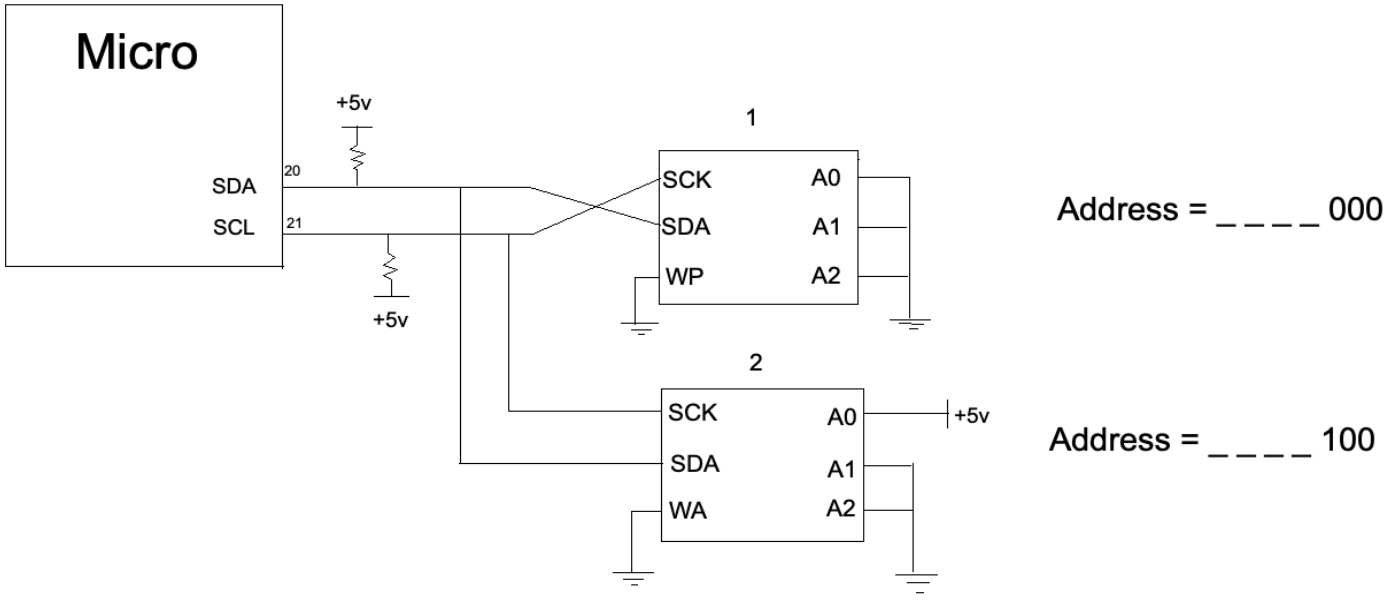
پرسش: اگر بخواهیم برای نگهداری مدهای کاری حافظه Flash را به کار ببریم، فرآیند نوشتن باید چگونه انجام شود که داده‌های دیگری که بر روی همان بلاک هستند از دست نروند؟

پس از هربار دستور پاک کردن در این نوع حافظه، یک بلاک کامل پاک می‌شود؛ یعنی تمام بیت‌های آن 1 می‌شوند. پس از آن می‌توان هرچند تعداد بیت 0 که لازم است در آن بلاک نوشته شود اما اگر قرار باشد یکی از بیت‌های 0 به 1 تبدیل شود، مجدداً کل بلاک باید پاک شود و دوباره فرایند نوشتن در آن صورت گیرد.

پرسش: اگر یک حافظه‌ی EEPROM بیرونی دارای 4KB حافظه و 2 پایه آدرس باشد، در این صورت می‌توان حداکثر چند KB حافظه EEPROM بیرونی بر روی یک باس مشترک داشت؟

با دو پایه آدرس می‌توان حداکثر 4 حافظه EEPROM داشت که هرکدام 4KB حافظه دارند و در مجموع روی یک باس مشترک می‌توان 16KB حافظه داشت.

پرسش: نمودار شماتیک برای اینکه دو AT24C02 را به یک باس مشترک وصل کنیم و حفاظت نوشتن غیر فعال باشد را رسم کنید. (آدرس‌دهی سخت‌افزاری دلخواه - باس را هم به پایه‌های میکروکنترلر متصل کنید)



پرسش: همخوانی این دنباله فریم ها را با پروتکل TWI بررسی کنید. (فریم‌های آدرس و داده را مشخص کنید، دستور خواندن یا نوشتن چگونه مشخص می‌شوند؟)

در عملیات نوشتن بایتی، ابتدا از طرف مستر یک بیت START ارسال می‌شود و پس از آن تمام اسلیوهای روی باس به خط گوش می‌دهند. سپس آدرس اسلیو مدنظر مستر(DEVICE ADDRESS) برای ارتباط داده می‌شود. پس از آن مستر یک بیت 0 می‌دهد که به معنای نوشتن(WRITE) روی دستگاه اسلیو است. در این حالت اسلیوی که آدرسش توسط مستر داده شده است، اگر روی باس باشد، یک بیت صفر به نشان تایید می-دهد(ACK). پس از آن مستر یک بایت داده ارسال می‌کند که آدرس جایی از اسلیو است که می‌خواهد در آن بنویسد(WORD ADDRESS). اسلیو هم یک بیت تایید به نشان دریافت این یک بایت می‌فرستد. سپس مستر یک بایت داده مدنظرش(DATA) را برای اسلیو می‌فرستد تا در آدرس گفته شده ذخیره کند. در آخر اسلیو یک بیت تایید(ACK) دریافت این 1 بایت را می‌فرستد و پس از آن مستر یک بیت 1 می‌دهد که به معنای پایان ارتباط با اسلیو است(STOP).

در عملیات خواندن بایتی، همانند عملیات نوشتن، ابتدا یک بیت استارت برای شروع ارتباط و سپس آدرس اسلیو مدنظر مستر توسط مستر فرستاده می‌شود. سپس مستر اعلام می‌کند که قصد دارد روی اسلیو بنویسد (WRITE) و این نوشتن برای آن است که در شمارنده آدرس اسلیو، آدرس خانه ای از حافظه اسلیو که مدنظر مستر است نوشته شود. پس از این مرحله اسلیو بیت تایید می‌فرستد و مستر پس از فرستادن آدرس خانه مدنظرش و دریافت بیت تایید اسلیو، دوباره ارتباط را آغاز می‌کند(START). پس از این قسمت، مستر مجددا آدرس اسلیو مدنظرش را می‌دهد اما این‌بار اعلام می‌کند که قصد دارد از اسلیو بخواند(READ). پس از تایید اسلیو، اسلیو محتوای خانه ای از حافظه که در قسمت قبل توسط مستر اعلام شده بود، برای مستر می‌فرستد. اگر مستر بیت تایید بدهد اسلیو محتوای خانه بعدی را می‌فرستد اما همانطور که در شکل مشخص است، مستر بیت صفری به نشانه تایید برای اسلیو نمی‌فرستد و اسلیو متوقف می‌شود و پس از آن مستر ارتباط را خاتمه می‌دهد.

پرسش: فرکانس کلاک در کدام دستگاه پیکربندی می‌شود؟ کلاک را کدام دستگاه فراهم می‌کند؟ با توجه به زمان مورد نیاز برای انجام عملیات نوشتن، با فرض این‌که کلاک را 10KHz تنظیم کرده باشیم، در این صورت حداکثر با چه نرخی می‌توان عملیات نوشتن را انجام داد؟

کلاک توسط مستر فراهم می‌شود و در این دستورکار، میکروی ما نقش مستر و درنتیجه تامین کننده کلاک ارتباط را دارد. اگر بخواهیم بصورت byte write داده بنویسیم، داده‌های سربار ارتباط شامل 1 بیت استارت، 7 بیت آدرس اسلو، 1 بیت دستور نوشتن، 1 بیت تایید اسلیو، حداقل 1 بایت آدرس حافظه مدنظر مستر برای نوشتن در آن و 1 بیت تایید مجدد اسلیو است و به ازای هر 1 بایت هم اسلیو 1 بیت تایید می‌فرستد و در انتها هم یک بیت STOP از طرف مستر به اسلیو ارسال می‌شود. در اینصورت، برای ارسال حداقل 1 بایت داده از مستر به اسلیو، 21 بیت داده سربار داریم که باعث می‌شود از هر 29 بیت داده، فقط 8 بیت آن مفید باشد و با کلاک 10KHz می‌توان حدود 2.75Kbit داده بر ثانیه ارسال کرد و نرخ ارسال 2.75 Kbit/sec خواهد بود. البته به لطف قابلیت page write به ازای داده‌های پشت سرهم تا حد زیادی می‌توان این نرخ را افزایش داد و از داده‌های سربار اضافی جلوگیری کرد.

- begin()
- setClock()
- beginTransmission()
- write()
- endTransmission()
- requestFrom()
- available()
- read()

پرسش: هر یک از تابع‌های نوشته‌شده را از راه لینک کتابخانه Wire، در مستندات آردوینو بررسی کنید و کد لازم را برای تولید دنباله‌ی فریم ها برای عملیات نوشتن و خواندن گفته‌شده (با این تابع ها) بنویسید.

begin() :

این تابع باعث شروع کار کتابخانه Wire است و ماژول مدنظر را به عنوان Master یا Slave به باس I2C اضافه می‌کند. به طور معمول فقط یک بار باید این تابع فراخوانی شود.

setClock()

این عملکرد فرکانس ساعت را برای ارتباطات I2C تنظیم می کند. دستگاه های Salve I2C حداقل فرکانس ساعت کاری ندارند، با این وجود ۱۰۰KHz معمولا خط پایه این موارد است.

beginTransmission()

این تابع انتقال داده را به دستگاه Slave I2C شروع می‌کند. همچنین در انتها باید این تابع را پایان دادن فراخوانی کنیم. ;()endTransmission.

write()

با استفاده از این تابع داده ها را از دستگاه Slave در پاسخ به درخواست Master ذخیره می‌کند، یا bytes را برای انتقال از Msater به Slave در صف قرار می دهد، این موارد در میان call های ()Transmission و ()endTransmission، کد پایین یک نمونه از عملکرد این تابع است.

endTransmission()

پس از فراخوانی تابع ;()Wire.beginTransmission برای پایان دادن به روند کار این تابع، از تابع معرفی شده استفاده می‌کنیم.

requestFrom()

توسط Master برای درخواست Bytes از دستگاه Slave استفاده می شود. سپس می توان بایت ها را با توابع ;()available و ;()read دریافت کرد. اگر مقدار false باشد، ()requestFrom پس از درخواست، پیام راه اندازی مجدد را ارسال می کند. این امر از درخواست دیوایس اصلی دیگر بین پیام ها جلوگیری می کند. این امکان را برای یک دستگاه اصلی فراهم می کند تا هنگام کنترل، چندین درخواست ارسال کند.

available()

تعداد بایت های موجود برای بازیابی را با ;()read برمی گرداند. این تابع را باید بعد از فراخوانی یک Master برای درخواست ()From یا در Slave با هندلر ()onReceive فراخوانی کنید.

read()

یک بایت را که از دستگاه Slave به Master منتقل شده است می خواند پس از استفاده از تابع ()requestFrom.