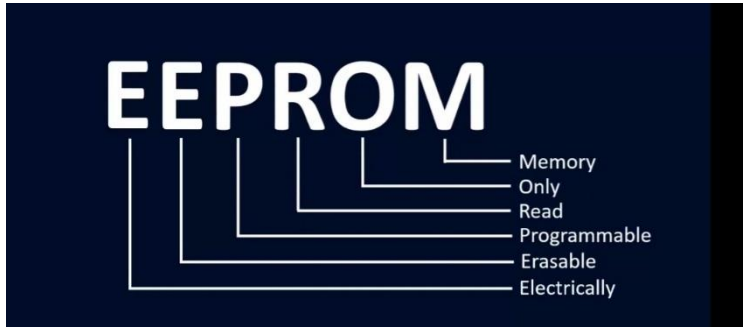
پیش‌گزارش آزمایشگاه _ آز7: آشنایی با پروتکل TWI و کار با داده بیرونی و حافظه EEROM و خواندن و نوشتن روی آن

- در چه کاربردهایی EEPROM بکاربرده می شود؟ چرا در اینجا حافظه RAM یا Flash را بکار نمی بریم؟ تفاوت حافظه EEPROM با RAM در چه است؟

حافظه EEPROM حافظه ای قابل برنامه ریزی است که می توان آن را پاک و بار ها پروگرم کرد. کاربرد آن در رد هایی است که نیاز به تنظیمات و پیکربندی اولیه برای عملکرد خود دارند به طوری که اگر ماشین خاموش و روشن شود تنظیمات از بین نروند و همچنین هر موقع که صلاح دانست بتوان تنظیمات را تغییر داد و مطمئن بود که در صورت تغییر تنظیمات، پیکر بندی جدید روی آن نصب شده و دائمی شده است.

به کار بردن حافظه RAM به دلیل موقت بودن آن، در هر بار روشن کردن ماشین باید از ابتدا برنامه ریزی شود.
حافظه ROM نیز حافظه فقط خواندنی است که نمی توان آن را دوباره برنامه ریزی کرد، مقادیر داخل آن را پاک کرد و یا به صورت الکتریکی و نرم افزاری آن را تغییر داد. فلش ROM از جهتی عملکردی یک گام به عقب بود زیرا پاک کردن فقط در قطعات بزرگ انجام می شد. با این وجود ، محدود کردن پاک کردن به قطعات بزرگ امکان ذخیره اطلاعات را بسیار فشرده تر از آنچه با EEPROM امکان پذیر شده بود ، فراهم کرد. برای نوشتن در فلش باید داده های قبلی را پاک و دوباره آپدیت کرد و هزینه بر است.



حافظه RAM حافظه موقتی است که از ترانزیستور، خازن یا .. ساخته شده است و در صورت قطع شده جریان برق مقادیر و داده های درون سلول های خود را از دست می دهد.
در مقابل حافظه EEPROM یک حافظه دائمی قابل برنامه ریزی مجدد و پاک شدن است که در صورت قطع شدن جریان برق مقادیر داده شده را در خود نگه می دارد و از بین نمی رود.

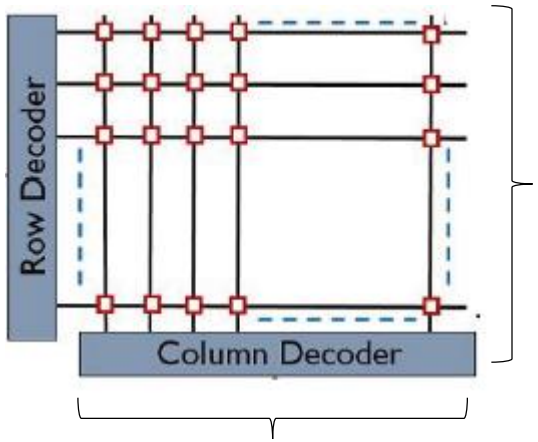
- اگر بخواهیم برای نگهداری مدهای کاری حافظه **Flash** را به کار ببریم، فرآیند نوشتن باید چگونه انجام شود که داده های دیگری که بر روی همان بلاک هستند از دست نروند؟

بسیاری از دستگاههای فلش دارای چرخه نوشتن سریعتر هستند اما سرعت پاک کردن آنها آهسته تر از حالت معمول دستگاههای EEPROM است .برای نوشتن یک بایت برای بسیاری از دستگاههای EEPROM 1-10میلی ثانیه و برای پاک کردن آنها به 5-50 میلی ثانیه نیاز است ؛ دستگاه های فلش به طور کلی کمتر از 100 usنیاز دارند تا بنویسید ، اما بعضی از آنها برای پاک کردن به صدها میلی ثانیه نیاز دارند.

می تواند بایت های با cursor دقیق، واحد را بخواند ، بنویسد و پاک کند. منطق کنترل آن به گونه ای تنظیم شده است که همه بایت ها به صورت جداگانه قابل دسترسی باشند. برای بازنویسی آن نیازی به یادآوری (RAM) محتوای صفحه است. به این معنی که داده بلاک خود را ذخیره و داده جدید را پس از نوشتن، با یاد آوری و باز خوانی داده های بلاکی که ذخیره کرده یود و افزودن به آن به ادامه می توان داده را نوشت بدون آنکه داده های دیگر از بین بروند.

- اگر یک حافظه ی **EEPROM** بیرونی دارای **4 KB**حافظه و **2** پایه آدرس باشد، در این صورت می توان حداکثر چند **KB** حافظه **EEPROM** بیرونی بر روی یک باس مشترک داشت؟

4KB



512row

8bit

2 line address

Decoder 2^n = 512

= 2^9

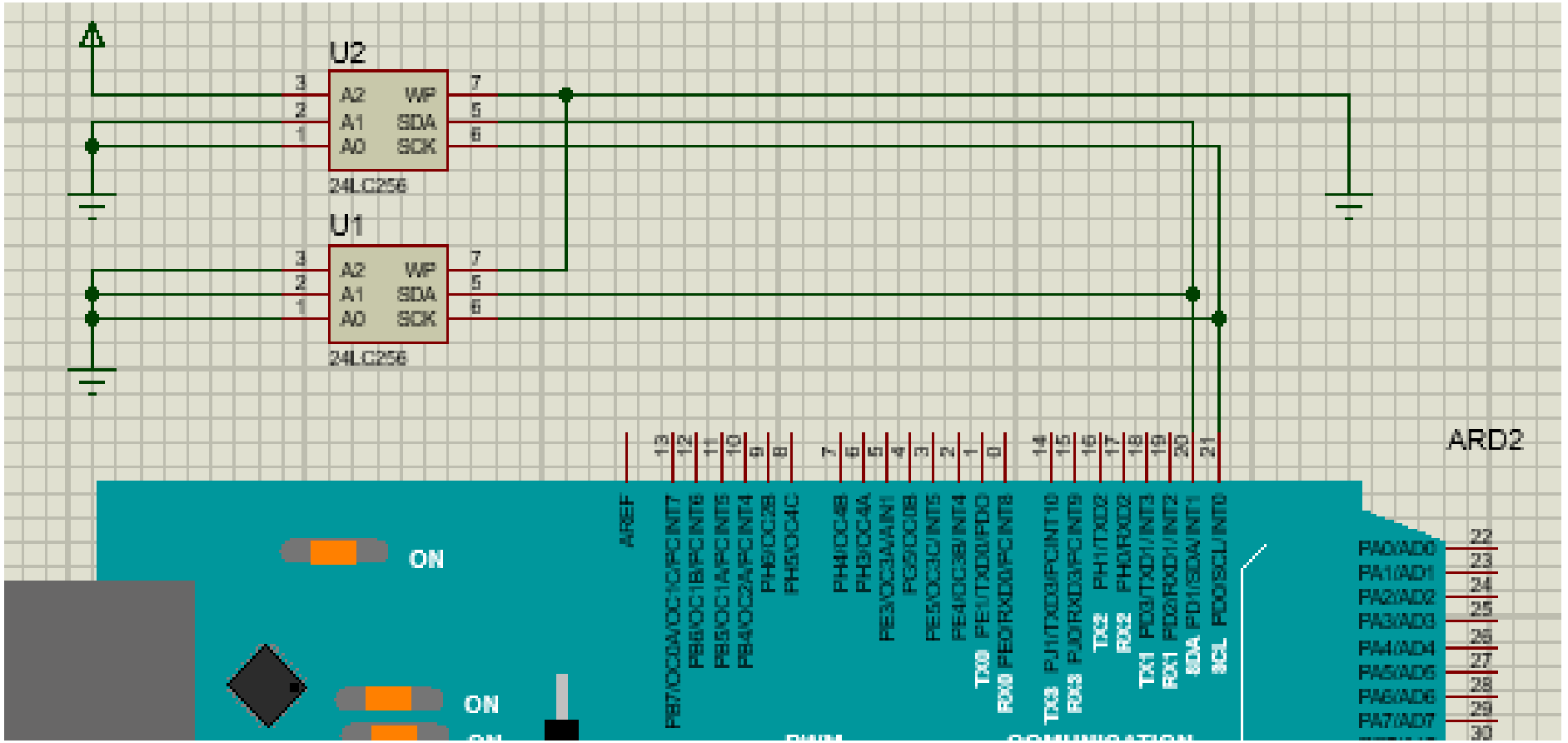
----> n = 9

حداقل 9 بیت می توان داشت

9<=2^k ---> k = 4

حداکثر 2^4 یعنی 16 بیت می توان روی باث مشترک داشت.

• نمودار شماتیک برای این که دو **02AT24C** را به یک باس مشترک وصل کرد و حفاظت نوشتن غیر فعال باشد را رسم کنید. (آدرس دهی سخت افزاری دل خواه - هم چنین باس را به پایه های میکروکنترلر نیز متصل کنید)



• هم خوانی این دنباله فریم ها را با پروتکل بررسی کنید. (فریم های آدرس و داده را مشخص کنید، دستور خواندن یا نوشتن چگونه مشخص می شوند؟)

FIGURE 6-1: BYTE WRITE

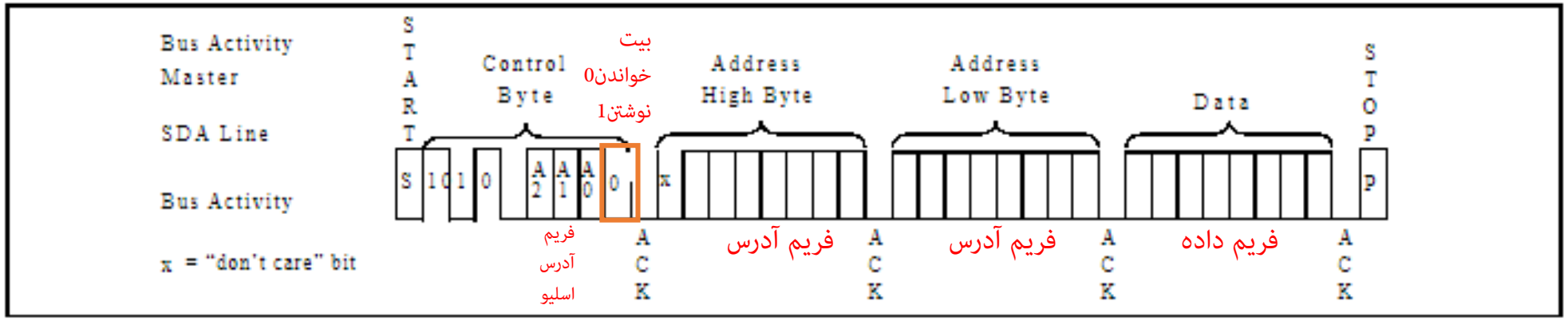
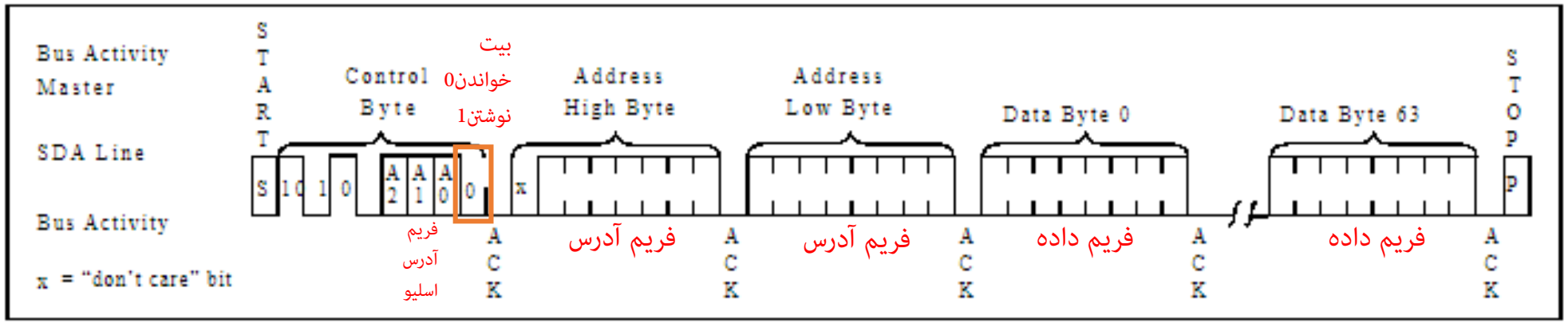


FIGURE 6-2: PAGE WRITE



- فرکانس کلاک در کدام دستگاه پیکربندی می شود؟ کلاک را کدام دستگاه فراهم می کند؟ با توجه به زمان مورد نیاز برای انجام عملیات نوشتن، با فرض این که کلاک را **KHz10** تنظیم کرده باشیم، در این صورت حداکثر با چه نرخی می توان عملیات نوشتن را انجام داد؟

فرکانس کلاک درون برد Arduino که همان مستر است فراهم و پیکر بندی می شود و از میکرو به اسلیو ها داده و روی باث با آنها هماهنگ می شود.

f = 10000 Hz

T= 1/10000 Hz

یک کلاک را در زمان T ارسال می شود.
در هر کلاک 1 بیت ارسال و دریافت می شود.
برای ارسال یک پک کامل داده به 20 کلاک نیاز داریم

$$\text{نرخ هر نوشتن} = \frac{20}{10000} = \frac{1}{500}$$

- مستندات آردوینو بررسی کنید و کد لازم را برای تولید دنباله ی ، **Wire**پرسش: هر یک از تابع های نوشته شده را از راه لینک کتابخانه فریم ها برای عملیات نوشتن و خواندن گفته شده (با استفاده از این تابع ها) بنویسید.

توابع کتابخانه wire :

begin()

راه اندازی کتابخانه wire برای وصل کردنbus پروتکل I2C مستر به اسلیو. تنها یک بار صدا زده می شود

Syntax

```
Wire.begin()
```

---

setClock()

فرکانس کلاک را برای این پروتکل تغییر می دهد که Hz100 بیس آن است.

Syntax

```
Wire.setClock(clockFrequency)
```

---

beginTransmission(address)

شروع انتقال داده به اسلیو با آدرس آن و ارسال بیت استارت

Syntax

```
Wire.beginTransmission(addr)
```

---

write()

ارسال پیام به اسلیو انتخاب شده با آدرس و گرفتن ack از آن

Syntax

```
Wire.write(value)
Wire.write(string)
Wire.write(data, length)
```

---

endTransmission()

اتمام انتقال به اسلیو و ارسال بیت استاپ

Syntax

```
Wire.endTransmission()
Wire.endTransmission(stop)
```

---

requestForm()

مستر با این تابع به طور مستقل در خواست خواندن یا نوشتن با استفاده از آدرس اسلیوه مربوطه انجام می دهد

Syntax

```
Wire.requestFrom(address, quantity)
Wire.requestFrom(address, quantity, stop)
```

---

available()

تعداد بیت های فعال را با استفاده از این تابع بازیابی می کند. باید پس از requestForm در مود مستر فرخوانی شود

Syntax

```
Wire.available()
```

---

read()

یک بایت را می خواند که پس از فراخوانی درخواست (یا) از دستگاه slave به یک master منتقل شده یا از master به slave ارسال شده است

Syntax

```
Wire.read()
```