به نام خدا

پیش گزارش آزمایش هفتم آزمایشگاه ریزپردازنده

امیرپارسا سلمان خواه

۹۸۳۱۰۳۴

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؟

از eeprom برای نگه داری داده هایی که به طور خیلی سریعی تغییر نمی کنند استفاده می شود. به عنوان مثال پارامتر های مورد نیاز برای کانفیگ کردن یک دستگاه در یک eeprom ذخیره می شوند تا در صورت خاموش شدن و روشن شدن دستگاه، این پارامتر ها را بتوان بازیابی کرد. همچنین در bios کامپیوتر ها و برای بالا اوردن کامپیوتر، اطلاعاتی نظیر آدرس سیستم عامل در eeprom نگه داری می شود تا برنامه ی bootstrap بتواند آن را پیدا و اجرا کند.

از حافظه ی فلش معمولا زمانی استفاده می شود که به حجم وسیع تری از داده ها نیاز است. در واقع حافظه ی فلش خود یک نوع eeprom است اما با این تفاوت که اطلاعات را نه به شکل بایت بایت بلکه به شکل بلاک بلاک منتقل می کند و این باعث می شود که سریع تر باشد. همچنین می توان بار ها روی حافظه های فلش نوشت. این در حالی است که در eeprom ها تعداد محدودی عملیات نوشتن را می توان انجام داد.

حافظه ی ram یک حافظه ی volatile است و با قطع شدن برق، اطلاعات آن از بین می رود. اما eeprom یک حافظه ی non volatile است و با قطع برق اطلاعات آن از بین نمی رود.

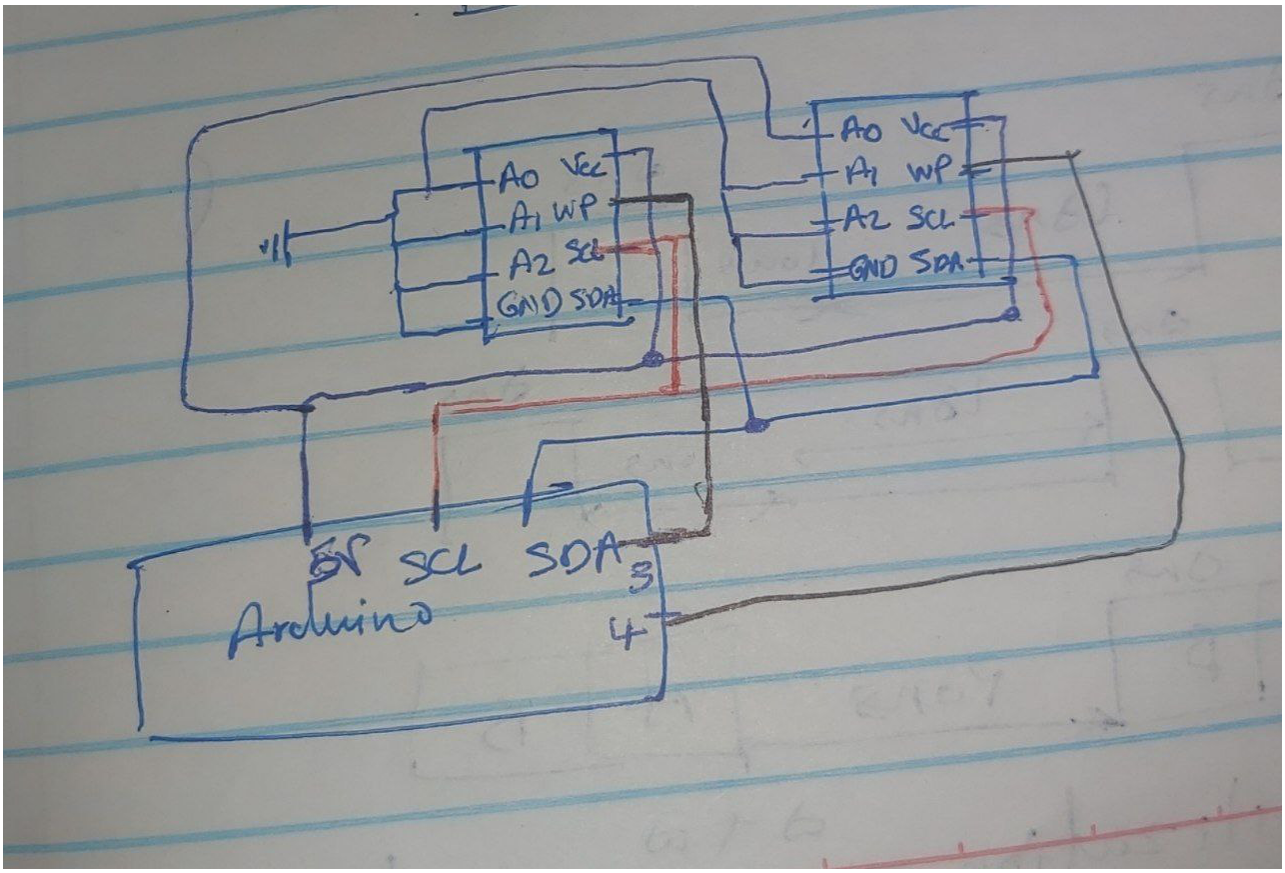
پرسش: اگر بخواهیم برای نگهداری مدهای کاری حافظه Flash را به کار ببریم، فرآیند نوشتن باید چگونه انجام شود که داده‌های دیگری که بر روی همان بلاک هستند از دست نروند؟

باید ابتدا کل بلاک خوانده شود و سپس بخشی از آن که می خواهیم تغییر کند را تغییر دهیم و سپس دوباره آن را بر روی حافظه بنویسیم.

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؟

از آن جا که ۲ پایه ی آدرس داریم، پس می توانیم ۴ تا از این حافظه را روی یک باس داشته باشیم.
بنابراین ۴ تا ۴ کیلوبایت یعنی ۱۶ کیلوبایت حافظه خواهیم 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؟)

دنباله ی فریم ها کاملا با پروتکل I2C مطابق است. به این شکل که ابتدا یک بیت شروع فرستاده می شود و بعد از آن آدرس slave و بیت خواندن یا نوشتن فرستاده می شود. سپس ack از slave دریافت می شود و بعد از آن ارسال دیتا شروع می شود و یک بایت یک بایت دیتا فرستاده می شود و ack دریافت می شود. در اینجا دیتا ها آدرس خانه ی حافظه و مقدار مورد نظر برای نوشتن است. در پایان نیز بیت پایانی فرستاده می شود.

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؟

فرکانس کلاک در master که در اینجا همان آردوینو است تامین می شود.
برای ارسال هر بسته تقریبا به ۴۰ بیت نیاز است و بعد از آن به ۵ میلی ثانیه زمان برای نوشتن دیتا روی eeprom نیاز است. ارسال ۴۰ بیت دیتا نیازمند ۴۰ کلاک است و چون هر کلاک ۰.۱ میلی ثانیه است، ۴ میلی ثانیه برای ارسال بسته طول می کشد. بنابراین با گذشت ۹ میلی ثانیه می توان یک عملیات نوشتن را به شکل کامل انجام داد و سراغ عملیات بعدی رفت.

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()

اینترفیس I2C را راه اندازی می کند.
setClock()

کلاک را تنظیم می کند.
beginTransmission()

با دریافت آدرس، انتقال دیتا را آغاز می کند.

write()

بسته های دیتا را ارسال می کند.
endTransmission()

ارسال اطلاعات را به إتمام می رساند.

requestFrom()

با دریافت آدرس از slave درخواست اطلاعات می کند.
available()

در صورتی که دیتایی دریافت شده باشد، مقدار ۱ منطقی را بر می گرداند.
read()

دیتای دریافت شده را بر می گرداند.

برای ارسال و دریافت اطلاعات می توان از دو تابع زیر استفاده کرد:

```
void writeI2CByte(byte data_addr, byte data){

  digitalWrite(WRITE_PROTECT, LOW);

  Wire.beginTransmission(ADDR);

  Wire.write(data_addr);

  Wire.write(data);
```

```
  Wire.endTransmission();
  digitalWrite(WRITE_PROTECT, HIGH);
  delay(5);
 }

 byte readI2CByte(byte data_addr){
  byte data = NULL;
  Wire.beginTransmission(ADDR);
  Wire.write(data_addr);
  Wire.endTransmission();
  Wire.requestFrom(ADDR, 1);
  delay(5);
  if(Wire.available()){
   data = Wire.read();
  }
  return data;
 }
```

کد ماشین لباس شویی:

```
#include <MsTimer2.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

const byte ROWS = 4;
const byte COLS = 4;
```

```
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define KEYPAD1 6
#define KEYPAD2 13
#define KEYPAD3 12
#define KEYPAD4 11
#define KEYPAD5 10
#define KEYPAD6 9
#define KEYPAD7 8
#define KEYPAD8 7

#define WRITE_PROTECT 2
#define ADDR 0x57

byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4};
//connect to the row pinouts of the keypad
byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8};
//connect to the column pinouts of the keypad

LiquidCrystal_I2C lcd(0x27, 16, 2);
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

byte const INIT_STATE = 0;
byte const PREWASH_STATE = 1;
byte const DETERGENT_STATE =2;
byte const WATER_STATE = 3;
byte const DRYING_STATE = 4;
byte const CONFIG_MENU_STATE = 5;
byte const CONFIG_ENTER_TIME_STATE = 6;
byte const FINISH_STATE = 7;

byte readI2CByte(byte data_addr);

byte prewash_time = 10;
byte detergent_time = 10;
```

```
byte water_time = 10;
byte drying_time = 10;

int input = 0;
byte state = INIT_STATE;

long start_time = 0;
byte remaining_time = 0;
byte editing = 0;
byte paused = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin();
  lcd.begin(); // Init LCD
  lcd.display();
  pinMode(WRITE_PROTECT, OUTPUT);

//  writeI2CByte(0, 10);
//  writeI2CByte(1, 10);
//  writeI2CByte(2, 10);
//  writeI2CByte(3, 10);
//  writeI2CByte(4, 0);
//  writeI2CByte(5, 10);
//  writeI2CByte(6, 0);

  prewash_time = readI2CByte(0);
  detergent_time = readI2CByte(1);
  water_time = readI2CByte(2);
  drying_time = readI2CByte(3);
  state = readI2CByte(4);
  remaining_time = readI2CByte(5);
  paused = readI2CByte(6);

  lcd.clear();
  switch (state){
    case INIT_STATE:
      print_at_center("1) Start", 0);
      print_at_center("2) Config", 1);
```

```
      break;
    case PREWASH_STATE:
      print_at_center("Prewash State", 0);
      print_at_center(String(remaining_time), 1);
      break;
    case DETERGENT_STATE:
      print_at_center("Detergent State", 0);
      print_at_center(String(remaining_time), 1);
      break;
    case WATER_STATE:
      print_at_center("Water State", 0);
      print_at_center(String(remaining_time), 1);
      break;
    case DRYING_STATE:
      print_at_center("Drying State", 0);
      print_at_center(String(remaining_time), 1);
      break;
    case FINISH_STATE:
      print_at_center("Done!", 0);
      break;
  }


  start_time = millis();
}

void loop() {
  // put your main code here, to run repeatedly:
  char key = keypad.getKey();
  switch (state){
    case INIT_STATE:
      if (key == '1'){
        state = PREWASH_STATE;
        writeI2CByte(4, state);
        remaining_time = prewash_time;
        start_time = millis();
      }
      else if (key == '2'){
        state = CONFIG_MENU_STATE;
        lcd.clear();
```

```
      print_at_center("1)PREWASH 2)DET", 0);
      print_at_center("3)WATER 4)DRY", 1);
    }
    break;

  case PREWASH_STATE:
    if (paused == 0 && key == NO_KEY && remaining_time != remaining_time -
(millis() - start_time) / 1000){
      start_time = millis();
      remaining_time -= 1;
      lcd.clear();
      print_at_center("Prewash State", 0);
      print_at_center(String(remaining_time), 1);
      writeI2CByte(5, remaining_time);
      if (remaining_time == 0){
        state = DETERGENT_STATE;
        writeI2CByte(4, state);
        remaining_time = detergent_time;
        start_time = millis();
      }
    }
    else if (key == 'A'){
      paused = 1 - paused;
      writeI2CByte(6, paused);
      if (paused == 0){
        start_time = millis();
      }
    }
    break;

  case DETERGENT_STATE:
    if (paused == 0 && key == NO_KEY && remaining_time != remaining_time -
(millis() - start_time) / 1000){
      start_time = millis();
      remaining_time -= 1;
      writeI2CByte(5, remaining_time);
      lcd.clear();
      print_at_center("Detergent State", 0);
      print_at_center(String(remaining_time), 1);
      if (remaining_time == 0){
```

```
        state = WATER_STATE;
        writeI2CByte(4, state);
        remaining_time = water_time;
        start_time = millis();
      }
    }
    else if (key == 'A'){
      paused = 1 - paused;
      writeI2CByte(6, paused);
      if (paused == 0){
        start_time = millis();
      }
    }
    break;

  case WATER_STATE:
    if (paused == 0 && key == NO_KEY && remaining_time != remaining_time -
(millis() - start_time) / 1000){
      start_time = millis();
      remaining_time -= 1;
      writeI2CByte(5, remaining_time);
      lcd.clear();
      print_at_center("Water State", 0);
      print_at_center(String(remaining_time), 1);
      if (remaining_time == 0){
        state = DRYING_STATE;
        writeI2CByte(4, state);
        remaining_time = drying_time;
        start_time = millis();
      }
    }
    else if (key == 'A'){
      paused = 1 - paused;
      writeI2CByte(6, paused);
      if (paused == 0){
        start_time = millis();
      }
    }
    break;
```

```
    case DRYING_STATE:
      if (paused == 0 && key == NO_KEY && remaining_time != remaining_time -
(millis() - start_time) / 1000){
        start_time = millis();
        remaining_time -= 1;
        writeI2CByte(5, remaining_time);
        lcd.clear();
        print_at_center("Drying State", 0);
        print_at_center(String(remaining_time), 1);
        if (remaining_time == 0){
          state = FINISH_STATE;
          writeI2CByte(4, state);
          lcd.clear();
          print_at_center("Done!", 0);
        }
      }
      else if (key == 'A'){
        paused = 1 - paused;
        writeI2CByte(6, paused);
        if (paused == 0){
          start_time = millis();
        }
      }
      break;

    case FINISH_STATE:
      if (key != NO_KEY){
        state = INIT_STATE;
        writeI2CByte(4, state);
        lcd.clear();
        print_at_center("1) Start", 0);
        print_at_center("2) Config", 1);
        break;
      }

    case CONFIG_MENU_STATE:
      if (key == '1'){
        lcd.clear();
        print_at_center("Set Prewash time:",0);
        print_at_center(String(prewash_time),1);
```

```
    editing = 1;
  }
  else if (key == '2'){
    lcd.clear();
    print_at_center("Set Det time:",0);
    print_at_center(String(detergent_time),1);
    editing = 2;
  }
  else if (key == '3'){
    lcd.clear();
    print_at_center("Set Water time:",0);
    print_at_center(String(water_time),1);
    editing = 3;
  }
  else if (key == '4'){
    lcd.clear();
    print_at_center("Set Drying time:",0);
    print_at_center(String(drying_time),1);
    editing = 4;
  }
  if (key >= '1' && key <= '4'){
    state = CONFIG_ENTER_TIME_STATE;
  }
  break;

case CONFIG_ENTER_TIME_STATE:
  if (key >= '0' && key <= '9'){
    input = input * 10 + key - '0';
    lcd.clear();
    print_at_center(String(input),0);
  }
  else if (key == 'D'){
    switch(editing){
      case 1:
        prewash_time = input;
        writeI2CByte(0, input);
        break;
      case 2:
        detergent_time = input;
        writeI2CByte(1, input);
```

```
          break;
        case 3:
          water_time = input;
          writeI2CByte(2, input);
          break;
        case 4:
          drying_time = input;
          writeI2CByte(3, input);
          break;
      }
      input = 0;
      state = INIT_STATE;
      lcd.clear();
      print_at_center("1) Start", 0);
      print_at_center("2) Config", 1);
    }
    break;

  default:
    break;
 }
}

void print_at_center(String message, byte row){
 int col = (16 - message.length()) / 2;
 lcd.setCursor(col, row);
 lcd.print(message);
}

void writeI2CByte(byte data_addr, byte data){
 digitalWrite(WRITE_PROTECT, LOW);
 Wire.beginTransmission(ADDR);
 Wire.write(data_addr);
 Wire.write(data);
 Wire.endTransmission();
 digitalWrite(WRITE_PROTECT, HIGH);
 delay(5);
}

byte readI2CByte(byte data_addr){
```

```
byte data = NULL;
Wire.beginTransmission(ADDR);
Wire.write(data_addr);
Wire.endTransmission();
Wire.requestFrom(ADDR, 1); //retrieve 1 returned byte
delay(5);
if(Wire.available()){
  data = Wire.read();
}
return data;
                                                                                {
```