Dr. Bob Davidov

# Application of flash memory in control devices

# Применение флеш-памяти в устройствах управления

***Цель работы:*** Знакомство с SPI интерфейсом устройств памяти.

***Задача работы:*** Подключение и использование флеш-памяти для накопления данных.

***Приборы и принадлежности:*** Персональный компьютер, контроллер Arduino UNO и WAVGAT, микросхема памяти W25Q64FVSIG.

## ВВЕДЕНИЕ

Флеш-память широко используется для хранения программ и данных в портативных устройствах с низким потреблением. Для успешного применения этого вида электронной памяти в системах управления необходимо знать организацию циклов чтения и записи данных, время выполнения которых влияет на суммарную задержку системы, и, в конечном счете, на устойчивость системы. В этой работе рассматривается флеш-память с SPI интерфейсом, подключаемая к одноплатному контроллеру WAVGAT (клон контроллера Arduino UNO).

## ОБЩИЕ СВЕДЕНИЯ

Флеш-память (*Flash-Memory*) относится к твердотельной полупроводниковой энергонезависимой перезаписываемой памяти (EEPROM). Количество циклов записи ограничено, но оно больше чем у жестких дисков. К другим недостаткам можно отнести относительно невысокое быстродействие: скорость чтения памяти лежит в диапазоне 10..400 Мб/с, скорость записи в память уступает чтению в два и более раз.

Флеш-память можно организовать на базе карт памяти и микросхем (***Рисунок 1***). Для чтения карт необходимо специальное устройство (***Рисунок 1***).

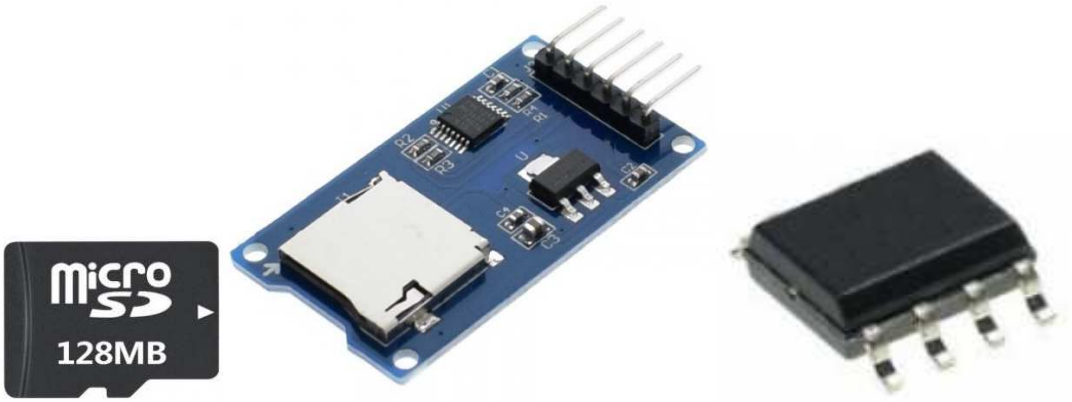
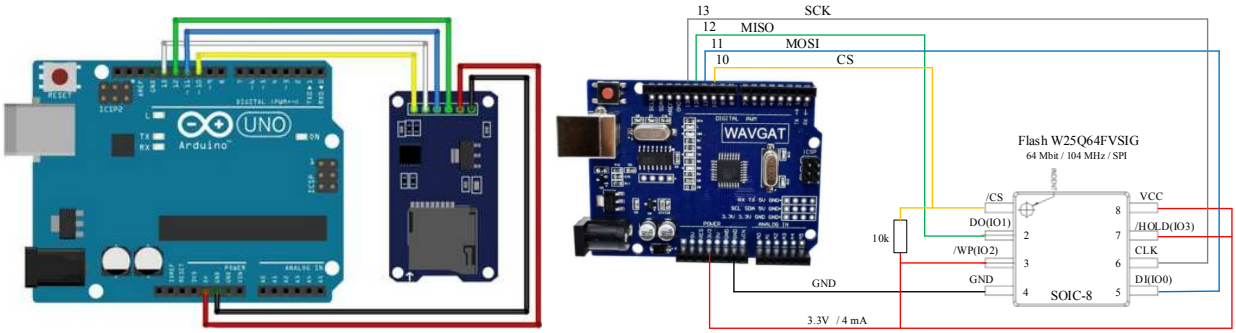



**Рисунок 1.** Карта Флеш-памяти (слева), модуль MicroSD SPI для Ардуино (в середине) и микросхема памяти W25Q64FVSIG.

Модуль MicroSD SPI и микросхема памяти W25Q64FVSIG подключаются к SPI интерфейсу контроллера Arduino UNO (WAVGAT), как показано на **Рисунок 2**.




**ВНИМАНИЕ**. Напряжение логического уровня единицы контроллера Arduino UNO **5 В**, контроллера WAVGAT **3,3 В**. Для подключения к контроллерам устройств с другими уровнями, например, памяти W25Q64FVSIG к Arduino UNO, необходимо использовать преобразователи уровней 3,3В/5В.

**Рисунок 2.** Подключение модуля MicroSD SPI и SPI микросхемы W25Q64FVSIG к плате Arduino UNO (слева) и клону WAVGAT UNO (справа). Микросхема памяти может работать работает и без подтягивающего резистора 10 Ком. Выводы защиты от записи /WP и приостановки устройства /HOLD микросхемы W25Q64FVSIG подключены к питанию 3.3В.

SPI интерфейс контроллеров UNO включает следующие цифровые линии.

- 12: сигнал MISO (Master In Slave Out) - вход контроллера выход устройства;
- 11: сигнал MOSI (Master Out Slave In) - выход контроллера вход устройства;
- 11: SCK (Serial Clock) - Тактовые импульсы, выход контроллера;
- 10: SS (Slave Select) – включение/отключения устройства, выход контроллера. (может использоваться и другой порт).

**Таблица 1.** Сравнительные характеристики модуля MicroSD SPI и микросхемы W25Q64FVSIG

| **Параметр** | **Модуль** *MicroSD SPI* | **Микросхема** *W25Q64FVSIG* |
|---|---|---|
| Емкость памяти | Поддерживаемые карты*: micro SD карты (< = 2 Гб), micro SDHC карты (< = 32 Гб) | 64 Мб (8МБ) |
| Интерфейс | SPI | SPI |
| Напряжение питания | 4.5 .. 5.5 В или 3.3 В | 2.7 .. 3.3 В |
| Потребляемый ток | от 0.2 мА до 200 мА | от 1 мкА до 4 мА |
| Максимальная тактовая частота | | 104 МГц |
| Количество перезаписей | | > 100 000 |
| Рабочая температура | | от -40 до +85 С |
| Габариты | 42 х 24 х 12 мм | Корпус SOIC-8: 3,8х4,9х1,5мм |
| Относительная стоимость | Модуль: 59 руб<br>Карта 128Мб: 150 руб | 60 руб |

*Перед использованием micro SD карты ее необходимо отформатировать в FAT16 или FAT32.

КОМАНДЫ СТАНДАРТНОГО SPI ИНТЕРФЕЙСА

Перед передачей команды устройству контроллер должен выставить на низкий (LOW) уровень сигнал SS (порт 10 или другой, связанный со входом CP (Chip Select) устройства) SPI интерфейса (см. **Рисунок 2**). Связь с устройством заканчивается переводом сигнал SS на высокий (HIGH уровень).

**Таблица 2.** Команды стандартного SPI интерфейса [1]

| INSTRUCTION NAME | BYTE 1 | BYTE 2 | BYTE 3 | BYTE 4 | BYTE 5 | BYTE 6 |
|---|---|---|---|---|---|---|
| *CLOCK NUMBER* | *(0 – 7)* | *(8 – 15)* | *(16 – 23)* | *(24 – 31)* | *(32 – 39)* | *(40 – 47)* |
| Write Enable | 06h | | | | | |
| Volatile SR Write Enable | 50h | | | | | |
| Write Disable | 04h | | | | | |
| Read Status Register-1 | 05h | (S7-S0)[2] | | | | |
| Read Status Register-2 | 35h | (S15-S8)[2] | | | | |
| Write Status Register | 01h | (S7-S0) | (S15-S8) | | | |
| Page Program | 02h | A23-A16 | A15-A8 | A7-A0 | D7-D0 | D7-D0[3] |
| Sector Erase (4KB) | 20h | A23-A16 | A15-A8 | A7-A0 | | |
| Block Erase (32KB) | 52h | A23-A16 | A15-A8 | A7-A0 | | |
| Block Erase (64KB) | D8h | A23-A16 | A15-A8 | A7-A0 | | |
| Chip Erase | C7h/60h | | | | | |
| Erase / Program Suspend | 75h | | | | | |
| Erase / Program Resume | 7Ah | | | | | |
| Power-down | B9h | | | | | |
| Read Data | 03h | A23-A16 | A15-A8 | A7-A0 | (D7-D0) | |
| Fast Read | 0Bh | A23-A16 | A15-A8 | A7-A0 | dummy | (D7-D0) |
| Release Powerdown / ID[4] | ABh | dummy | dummy | dummy | (ID7-ID0)[2] | |
| Manufacturer/Device ID[5] | 90h | dummy | dummy | 00h | (MF7-MF0) | (ID7-ID0) |
| JEDEC ID[6] | 9Fh | (MF7-MF0) Manufacturer | (ID15-ID8) Memory Type | (ID7-ID0) Capacity | | |
| Read Unique ID | 4Bh | dummy | dummy | dummy | dummy | (UID63-UID0) |
| Read SFDP Register | 5Ah | 00h | 00h | A7–A0 | dummy | (D7-0) |
| Erase Security Registers[3] | 44h | A23-A16 | A15-A8 | A7-A0 | | |
| Program Security Registers[3] | 42h | A23-A16 | A15-A8 | A7-A0 | D7-D0 | D7-D0[3] |
| Read Security Registers[3] | 48h | A23-A16 | A15-A8 | A7-A0 | dummy | (D7-D0) |
| Enable QPI | 38h | | | | | |
| Enable Reset | 66h | | | | | |
| Reset | 99h | | | | | |

## SPI БИБЛИОТЕКИ

Для работы с SPI устройствами пакет программ контроллера WAVGAT, располагаемый в разделе пользователя ..\Users\xxx\Documents\Arduino\libraries\ имеет библиотеку SPI. Для сравнения, раздел ..\libraries\ контроллера WAVGAT необходимо дополнить библиотекой SPIflash [2]. Обе SPI библиотеки содержат команды чтения, записи и стирания флеш-памяти.

Сравнительные примеры программ для работы с флеш-памятью приведены в таблицах ниже.

## ЧТЕНИЕ ОСНОВНЫХ ПАРАМЕТРОВ ФЛЕШ ПАМЯТИ

**Таблица 3**. Примеры чтения параметров микросхемы памяти с использованием библиотек **SPI** и **SPIflash.**

| SPI | SPIflash |
|---|---|
| #include <SPI.h><br><br>const int SSPin = 10;<br>// JEDEC ID > Manufacturer ID> Memory Type > Device ID<br>const byte JDCID = 0x9F;<br><br>void setup() {<br>  Serial.begin(9600);<br>  pinMode(SSPin, OUTPUT);<br>  SPI.begin();<br><br>  // скорость передачи, порядок бит, режим SPI<br>  SPISettings mySet(100000, MSBFIRST, SPI_MODE0);<br><br>  SPI.beginTransaction(mySet);<br>  digitalWrite(SSPin, LOW);<br>  SPI.transfer(JDCID);<br>  for (int i=0; i<3; i++) {<br>  byte data = SPI.transfer(0);<br>  Serial.print(data,HEX);<br>  }<br>  digitalWrite(SSPin, HIGH);<br>  SPI.endTransaction();<br>}<br><br>void loop() {<br>  // цикл пустой<br>} | #include <SPIflash.h><br>// Signal order: MOSI/DI, MISO/DO, SCK/CLK, CE/CS<br>SPIflash myFlash(11, 12, 13, 10);<br>long maxpage;<br><br>void setup()<br>{<br>  Serial.begin(9600);<br>  myFlash.begin();<br>  pinMode(13, OUTPUT);<br>  digitalWrite(13, LOW);<br>}<br><br>void loop()<br>{<br>  if (myFlash.ID_device!=0)<br>  {<br>  Serial.print("The connected device is a ");<br>  Serial.print(myFlash.Text_device);<br>  Serial.print(", which is a ");<br>  Serial.print(myFlash.Capacity, DEC);<br>  Serial.print("Mbit ");<br>  Serial.print(myFlash.Text_type);<br>  Serial.print(" from ");<br>  Serial.print(myFlash.Text_manufacturer);<br>  Serial.println(".");<br>  Serial.println();<br>  Serial.println("*****************************");<br>  Serial.print("Manufacturer ID : 0x");<br>  Serial.println(myFlash.ID_manufacturer, HEX);<br>  Serial.print("Memory type  : 0x");<br>  Serial.println(myFlash.ID_type, HEX);<br>  Serial.print("Device ID  : 0x");<br>  Serial.println(myFlash.ID_device, HEX);<br>  Serial.println("*****************************");<br>  Serial.print("Number of pages : 0x");<br>  Serial.println(myFlash.Pages, HEX);<br>  Serial.println("*****************************");<br>  Serial.println();<br>  maxpage = myFlash.Pages;<br>  Serial.print("Waiting for Flash chip to be ready... ");<br>  myFlash.waitForReady(); |

| | |
|---|---|
| | Serial.println("Chip is ready.");<br>digitalWrite(13, HIGH);<br>}<br>while(1) {};<br>} |
| *ОТВЕТ на мониторе IDE:*<br><br>EF4017 | *ОТВЕТ на мониторе IDE:*<br><br>The connected device is a W25Q64FV, which is a 64Mbit SPI Serial Flash from Winbond.<br><br>**********<br>Manufacturer ID : 0xEF<br>Memory type : 0x40<br>Device ID : 0x17<br>**********<br>Number of pages : **0x8000**<br>**********<br><br>Waiting for Flash chip to be ready... Chip is ready. |
| | Примечание. Объем страницы: 256 байт (**0x100**). Всего страниц: **0x8000** |

ЧТЕНИЕ SPI ФЛЕШ-ПАМЯТИ

Чтение может выполняться многократно.

Последовательность чтения данных [3]:

- послать команду на чтение (команда 03h/0Bh, 1 байт) и начальный адрес (3 байта),
- считать требуемое количество байт. Поскольку SPI – синхронный интерфейс, то для чтения необходимо передавать в ПЗУ любые байты, например, нули, в ответ вернутся хранимые в ПЗУ байты (byte data = SPI.transfer(0);)

**Таблица 4**. Примеры чтения данных с использованием библиотек **SPI** и **SPIflash**

| **SPI** | **SPIflash** |
|---|---|
| #include <SPI.h><br><br>const int SSPin = 10;<br>const byte READ = 0x03;<br>const byte ADDR1 = 0x00; // (A23-A16),<br>const byte ADDR2 = 0x00; // (A15-A8)<br>const byte ADDR3 = 0x00; // (A7-A0)<br><br><br>void setup() {<br>Serial.begin(9600);<br>pinMode(SSPin, OUTPUT);<br>SPI.begin();<br>}<br><br>void loop() {<br>//rate, dataOrder: MSBFIRST or LSBFIRST, dataMode: SPI_MODE0, SPI_MODE1, SPI_MODE2, or SPI_MODE3<br>SPISettings mySet(100000, MSBFIRST, SPI_MODE0);<br>// digitalWrite(SSPin, HIGH); | #include <SPIflash.h><br><br>#define SERIAL_BAUD 9600 //115200<br><br>// Signal order: MOSI/DI, MISO/DO, SCK/CLK, CE/CS<br>SPIflash myFlash(11, 12, 13, 10);<br><br>void setup(){<br>Serial.begin(SERIAL_BAUD);<br><br>Serial.print("Start...");<br>myFlash.begin();<br>pinMode(13, OUTPUT);<br>digitalWrite(13, LOW);<br>}<br><br>void loop(){<br>uint32_t page=0;<br>Serial.println("Flash content:");<br>digitalWrite(13, HIGH); |

```
  SPI.beginTransaction(mySet);
  digitalWrite(SSPin, LOW);
  SPI.transfer(READ);
  SPI.transfer(ADDR1);
  SPI.transfer(ADDR2);
  SPI.transfer(ADDR3);
  for (int i=0; i<16; i++) {
    byte data = SPI.transfer(0);
//    Serial.print((char)data);
    Serial.print(data,HEX);
  }
  Serial.println();
  digitalWrite(SSPin, HIGH);
  SPI.endTransaction();

  while(1) {};
}
```

```
  myFlash.readPage(page); // page is 0; 1; 2; .. of 256 byte
  digitalWrite(13, LOW);
  for (int i=0; i<256; i++) {
    int v = myFlash.buffer[i];
    Serial.print(v,HEX);
    Serial.print(" ");
  }
  Serial.println();
  Serial.println("end of page reading");
  while(1) {} // Inf delay
}
```

## ЗАПИСЬ ДАННЫХ В SPI ФЛЕШ-ПАМЯТЬ

ВНИМАНИЕ! Запись может выполняться только в очищенную память. Память очищается секторами по 4КБ или блоками по 32 КБ или 64КБ (см. **Таблица 2**)

Запись может выполняться отдельными байтами с указанием адреса каждого байта или более быстро - непрерывным потоком до 256 байт, когда указывается адрес только первого байта.

ВНИМАНИЕ! Непрерывная побайтовая запись должна укладываться целиком в станицы по 256 байт (0..255, 256..511, 512..767, 768..1023, и т.д.). В противном случае, когда адрес первого байта лежит в пределах одной страницы, например в ее середине, и размер записи больше половины страницы, байты переходящие на следующую страницу не будут записаны. Чтобы записать данные на новую страницу, необходимо указать адрес первого “переходящего” байта. Проблема с ”переходящими” байтами отсутствует, если длина записей с одним адресом кратна 2 и не превышает 256 байт.

Рассматриваемая флеш-память 8МБ содержит 32768 или 0x8000 страниц (по 256 байт). Тестовое время записи байта при тактировании 100 КГц равно 76 мкс (как 9сек/256/256).

Последовательность записи данных [3]:

- выставить разрешение на запись (команда 06h, 1 байт),
- послать команду на запись (команда 02h, 1 байт), начальный адрес (3 байта) и данные побайтно от 1 до 256 байт,
- выставить запрет записи (команда 04h, 1 байт)

Примечание.

1. Каждая постраничная запись должна начинаться разрешением, а заканчиваться запретом записи.
2. Между записями страниц необходимо ввести задержку (1 мс, достаточно).

**Таблица 5**. Примеры записи данных с использованием библиотек **SPI** и **SPIflash**

**SPI**

```
#include <SPI.h>

const int SSPin = 10;
const byte WREN = 0x06;
const byte WRDI = 0x04;
const byte READ = 0x03;
const byte PP = 0x02;
const byte ADDR1 = 0x0; // (A23-A16),
const byte ADDR2 = 0x0; // 4K + 1
const byte ADDR3 = 0x0; // (A7-A0)

int data[256];

void setup() {
  pinMode(SSPin, OUTPUT);
  SPI.begin();

  SPISettings mySet(100000, MSBFIRST, SPI_MODE0);

  // Выставление разрешения записи:
  SPI.beginTransaction(mySet);
  digitalWrite(SSPin, LOW);
  SPI.transfer(WREN);
  digitalWrite(SSPin, HIGH);
  SPI.endTransaction();

  // Запись массива данных в ПЗУ:
  SPI.beginTransaction(mySet);
  digitalWrite(SSPin, LOW);
  SPI.transfer(PP);
  SPI.transfer(ADDR1);  // (A23-A16)
  SPI.transfer(ADDR2);  // (A15-A8)
  SPI.transfer(ADDR3);  // (A7-A0)

  // fill data array
  for (int i=0; i<256; i++) {
    data[i] = random(255);
  }
  // write data array
  for (int i=0; i<sizeof(data); i++) {
    SPI.transfer(data[i]);
  }
  digitalWrite(SSPin, HIGH);
  SPI.endTransaction();

  // Выставление запрета записи:
  SPI.beginTransaction(mySet);
  digitalWrite(SSPin, LOW);
  SPI.transfer(WRDI);
  digitalWrite(SSPin, HIGH);
  SPI.endTransaction();
}

void loop() {
  //
}
```

**SPIflash**

```
#include <SPIflash.h>

int strbuf[256];

// Signal order: MOSI/DI, MISO/DO, SCK/CLK, CE/CS
SPIflash myFlash(11, 12, 13, 10);

int buf[256];

void setup(){
  Serial.begin(9600); // Baud rate

  // fill strbuf array
  for (int i=0; i<256; i++) {
    buf[i] = random(255);
  }

  Serial.print("Start...");

  myFlash.begin();
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
}

void loop(){
  uint32_t page=0; // page is 0; 1; 2; .. of 256 byte
  Serial.println("Write content");
  for (int i=0; i<256; i++) {
    myFlash.buffer[i] = buf[i];
    Serial.print(buf[i],HEX);
    Serial.print(" ");
  }
  Serial.println();
  digitalWrite(13, LOW);
  myFlash.writePage(page);
  digitalWrite(13, HIGH);
  Serial.println("end of writing");

  while(1) {} // Inf delay
}
```

ОЧИСТКА SPI ПАМЯТИ

Очистка SPI памяти выполняется полностью (команда 60h), секторами по 4КБ (команда 20h) или блоками по 32КБ (команда 52h) или 64КБ (команда D8h), см. **Таблица 2**.

Последовательность стирания блоками [3]:

- выставить разрешение на запись (команда 06h, 1 байт),
- послать команду на стирание (команда 20h, 1 байт) и адрес (3 байта),
- выставить запрет записи (команда 04h, 1 байт)

Все биты очищенных ячеек содержат единицы: 0xFF.

Примечание.

1. Каждая постраничная запись должна начинаться разрешением, а заканчиваться запретом записи.
2. Между записями страниц необходимо ввести задержку (тестовая задержка – не менее 60 мс).

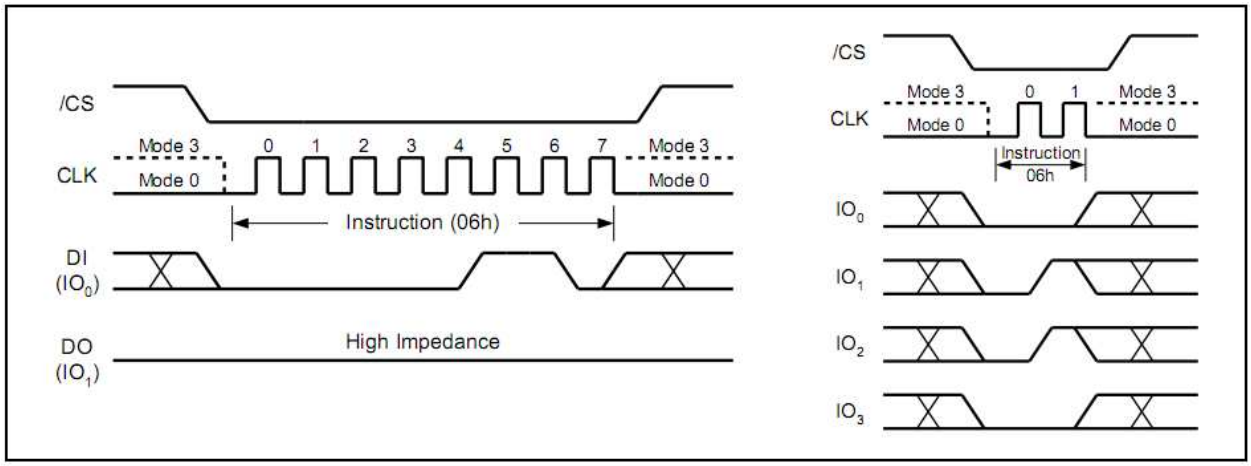



Figure 5. Write Enable Instruction for SPI Mode (left) or QPI Mode (right)

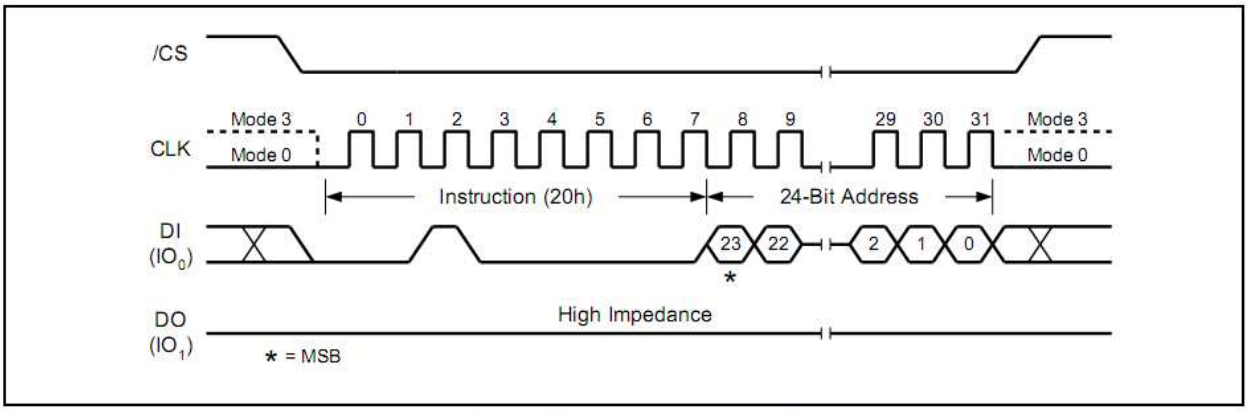



Figure 21a. Sector Erase Instruction (SPI Mode)

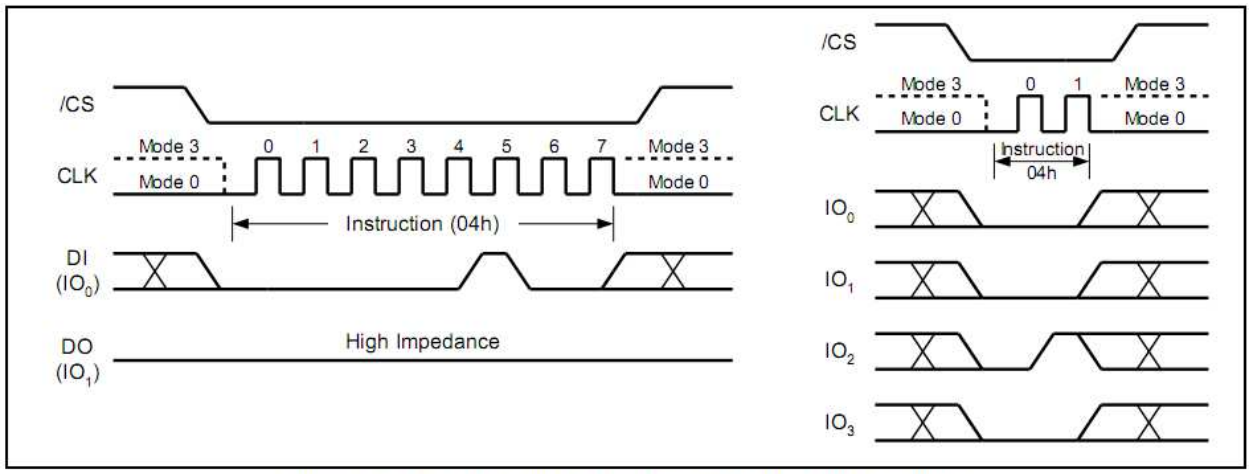



Figure 7. Write Disable Instruction for SPI Mode (left) or QPI Mode (right)

**Рисунок 3.** Диаграмма очистки одного сектора флеш-памяти [1].

**Таблица 6**. Примеры очистки блока памяти с использованием библиотеки **SPI** и всей памяти с использованием библиотеки **SPIflash.**

| SPI | SPIflash |
|---|---|
| **// Очистка блока 4К с указанного адреса**<br><br>#include <SPI.h><br><br>const int SSPin = 10;<br>const byte WREN = 0x06; // SPI Write Enable<br>const byte WRDI = 0x04; // SPI Write Disable<br>const byte SER = 0x20; // SPI Sector Erase (4KB)<br>const byte ADDR1 = 0x0; // Ini address (A23-A16), Flash capacity 8MB<br>const byte ADDR2 = 0x00; // Ini address (A15-A8), Flash capacity 8MB<br>const byte ADDR3 = 0x0; // Ini address (A7-A0), Flash capacity 8MB<br><br>void setup() {<br>pinMode(SSPin, OUTPUT);<br>SPI.begin();<br><br>SPISettings mySet(100000, MSBFIRST, SPI_MODE0);<br><br>// Выставление разрешения записи:<br>SPI.beginTransaction(mySet);<br>digitalWrite(SSPin, LOW);<br>SPI.transfer(WREN);<br>digitalWrite(SSPin, HIGH);<br>SPI.endTransaction();<br><br>// Очистка сектора:<br>SPI.beginTransaction(mySet);<br>digitalWrite(SSPin, LOW);<br>SPI.transfer(SER);<br>SPI.transfer(ADDR1);<br>SPI.transfer(ADDR2);<br>SPI.transfer(ADDR3);<br>digitalWrite(SSPin, HIGH);<br>SPI.endTransaction(); | **// Очистка всей памяти**<br><br>#include <SPIflash.h><br><br>char input = 0;<br>int databuf[256];<br><br>// Signal order: MOSI/DI, MISO/DO, SCK/CLK, CE/CS<br>SPIflash myFlash(11, 12, 13, 10);<br><br>void setup(){<br>Serial.begin(9600);<br><br>// fill databuf array<br>for (int i=0; i<256; i++) {<br>databuf[i] = random(255);<br>}<br>myFlash.begin();<br>pinMode(13, OUTPUT);<br>}<br><br>void loop(){<br>Serial.println("Erasing chip");<br><br>digitalWrite(13, HIGH);<br>myFlash.eraseChip();<br>digitalWrite(13, LOW);<br><br>Serial.println("Done...");<br><br>while(1) {};<br>} |

| | |
|---|---|
| // Выставление запрета записи:<br>SPI.beginTransaction(mySet);<br>digitalWrite(SSPin, LOW);<br>SPI.transfer(WRDI);<br>digitalWrite(SSPin, HIGH);<br>SPI.endTransaction();<br>}<br><br>void loop() {<br>//<br>} | |
| | **Примечание**. Очистка всей памяти 8МБ длится 19 секунд. |

## ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

**Задание 1.** Программа чтения и записи данных флеш-памяти (микросхема W25Q64FVSIG).

1. Используя примеры программ ***Таблица 3*** .. ***Таблица 6*** и видоизмененный пример SPIflash библиотеки [2] (данный ниже) разработайте программу чтения, записи и стирания микросхемы W25Q64FVSIG флеш-памяти. Каждый процесс запускается передачей контроллеру символа ‘r’, ‘w’ и ‘c’ соответственно. Чтение выполняется потоком 256 байт. Записываемые данные (256 байт) формируются генератором случайных чисел.

```
// This sketch is an example of using the SPIflash library with W25Q64FVSIG
// SPI Flash chip. This sketch listens to a few serial commands
// Hence type the following commands to interact with the SPI flash memory array:
// - 'w' writes the first 256bytes of the flash chip
// - 'c' erases the entire memory chip
// - 'r' read flash memory
// Get the SPIFlash library from here: http://www.rinkydinkelectronics.com/library.php?id=65

#include <SPIflash.h>
//#include <SPI.h>

#define SERIAL_BAUD    9600  //115200
char input = 0;
int strbuf[256];

////////////////////////////////////////////
// flash(SPI_CS, MANUFACTURER_ID)
```

```
// SPI_CS          - CS pin attached to SPI flash chip (8 in case of Moteino)
// MANUFACTURER_ID - OPTIONAL, 0x1F44 for adesto(ex atmel) 4mbit flash
//                           0xEF30 for windbond 4mbit flash
//////////////////////////////////////////
SPIflash myFlash(11, 12, 13, 10);

void setup(){
  Serial.begin(SERIAL_BAUD);

  // fill strbuf array
  for (int i=0; i<256; i++) {
    strbuf[i] = random(255);
  }

  Serial.print("Start...");

  myFlash.begin();
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
}

void loop(){
  // Handle serial input (to allow basic DEBUGGING of FLASH chip)
  // ie: display first 256 bytes in FLASH, erase chip, write bytes at first 10 positions, etc
  if (Serial.available() > 0) {
    uint32_t page=0;
    input = Serial.read();
    if (input == 'r') // reading flash area
    {
      Serial.println("Flash content:");
      digitalWrite(13, HIGH);
      myFlash.readPage(page);
      digitalWrite(13, LOW);
      for (int i=0; i<256; i++) {
        int v = myFlash.buffer[i];
        Serial.print(v,HEX);
        Serial.print(" ");
      }
      Serial.println();
      Serial.println("end of reading");
    }
    else if (input == 'w')
```

```
    {
      Serial.println("Write content");
      for (int i=0; i<256; i++) {
        myFlash.buffer[i] = strbuf[i];
        //Serial.print(strbuf[i],HEX);
        //Serial.print(" ");
      }
      Serial.println();
      digitalWrite(13, LOW);
      myFlash.writePage(page);
      digitalWrite(13, HIGH);
      Serial.println("end of writing");
    }
    else if (input == 'c')
    {
      Serial.println("Clearing content");

      digitalWrite(13, HIGH);
      myFlash.eraseChip();
      digitalWrite(13, LOW);

      Serial.println("end of clearing");
    }
  }
}
```

2. Используя монитор Arduino IDE проверьте работоспособность программы.
3. Измерьте время записи байта, чтения байта, стирания блока 4КБ памяти и всей памяти. Для измерения времени стирания блока используйте пример ***Таблица 6*** с библиотекой SPI.
4. По диаграмме **Рисунок 3** рассчитайте время стирания 4КБ блока флеш-памяти. Сравните с измеренным временем.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова максимальная скорость записи данных отдельными байтами (не потоком) во флеш- память W25Q64FVSIG при частоте синхронизации 100 КГц?
2. Какова максимальная скорость чтения данных отдельными байтами (не потоком) из флеш- памяти W25Q64FVSIG при частоте синхронизации 100 КГц?
3. Какое время стирания страницы минимального объема флеш-памяти W25Q64FVSIG?

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**


1. WINBOND W25Q64FV SPIFLASH, Revision L https://www.winbond.com/resource-files/w25q64fv_revl1_100713.pdf
2. Rinky-Dink Electronics. http://www.rinkydinkelectronics.com/library.php
3. Чтение и запись флеш-памяти с помощью Arduino на примере микросхемы 25L8005 https://soltau.ru/index.php/arduino/item/508-chtenie-i-zapis-flesh-pamyati-s-pomoshchyu-arduino-na-primere-mikroskhemy-25l8005
4. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov