64-мегабитная MMC-карта и PIC-микроконтроллер

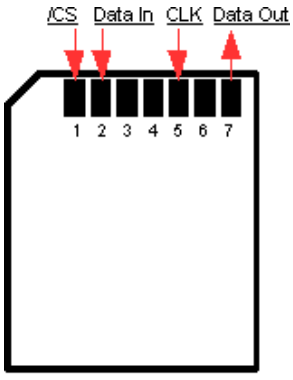
Здесь рассматривается запись-чтение карты памяти MMC с помощью PIC16F877 по SPI-интерфейсу (64-мегабайтная MMC и PIC-микроконтроллер). Программная реализация SPI-интерфейса даёт возможность подключить одну карту MMC к любому PIC-микроконтроллеру.

В статье приводятся электрическая схема, теория по интерфейсу SPI, теория по MMC, сигнальные диаграммы, программа на Си (компилятор CC5X).

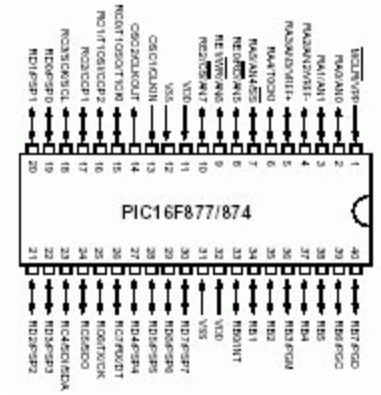
+

Характеристики MMC-карты на 64 Мб:
- огромная FLASH-память
- очень быстрая
- 300 000 циклов записи
- чтение без ограничений
- связь с PIC по 4 линиям

Характеристики PIC16F877:
- SPI-интерфейс
- RS232-интерфейс
- 8 входов для АЦП (10 бит)



MMC-карта



Вывод	Обозначение	Функция	Вывод	Обозначение	Функция
1	/CS	Выбор микросхемы	**18**	RC2	Цифровой выход
2	Data IN	Вход данных	**24**	SDO	Выход данных SPI
3	GND	Общий минус	**-**	-	
4	Vdd	Питающее напряжение	**-**	-	
5	CLK	Синхронизация	**18**	SCL	Синхронизация SPI
6	GND	Общий минус	**-**	-	
7	Data OUT	Выход данных	**23**	SDI	Вход данных SPI

Схема включения и электрическая схема

Для MMC необходимо питающее напряжение в диапазоне 2.7 – 3.6 В. В любом случае мы должны использовать стабилизатор напряжения для 3.3 В, например, CS5203. PIC-микроконтроллер может работать в диапазоне напряжений от 2 до 5.5 В. Таким образом, можно сойтись на снабжении обеих микросхем питанием 3.3 В. В этом случае можно подключить MMC непосредственно к микроконтроллеру.

Исходя из соображений, что АЦП и последовательная связь при 5 В функционируют более стабильно, PIC16F877 всё-таки питается в этой схеме от 5 В. Выходы микроконтроллера SDO, SCL, /CS подключены через делители напряжения, так что на входы MMC подаётся напряжение не более 3.2 В. Входы микроконтроллера не нуждаются в каком-либо усилении с 3.3 до 5 В, так как PIC всё равно понимает 3.3 В как логическую «1».

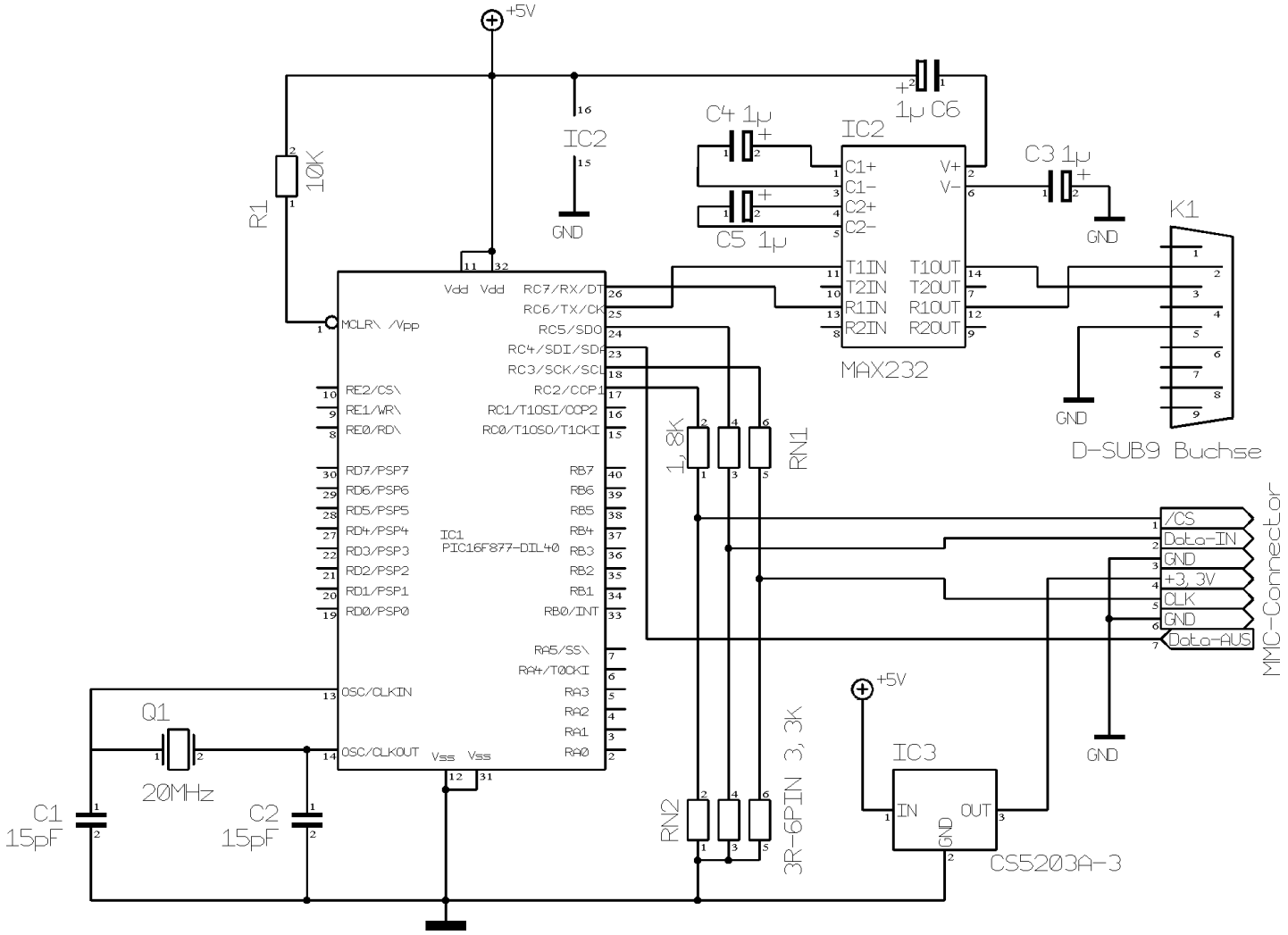


Рис. 1. Схема электрическая

MAX232 используется для обеспечения последовательной связи с компьютером (на скорости 115200 бод).

Для подключения к MMC в качестве переходника прекрасно подходит старый разъём от флоппи-дисковода (см. **Рис. 2** ниже), либо можно использовать предлагаемый http://www.segor.de так называемый «MMC-разъём» ("MMC-Connector") за 4.90 евро.

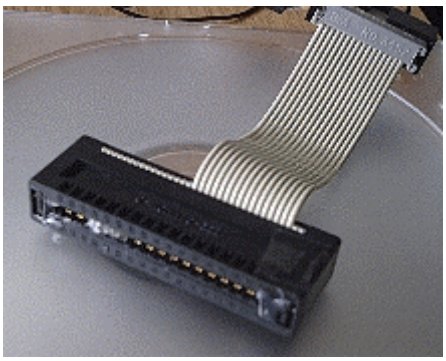


Рис. 2. Разъём для MMC из разъёма для флоппи-дисковода

Как, собственно, функционирует SPI?

SPI - это последовательный синхронный интерфейс. Синхронным он называется потому что каждый бит принимается /передаётся во время тактового импульса.

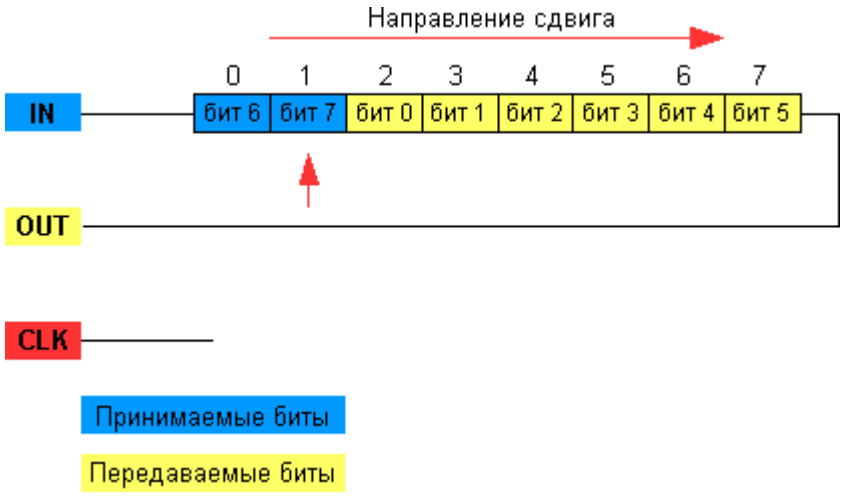


Рис.3. Приём и передача байта по интерфейсу SPI

На **Рис. 3** мы видим регистр, через который передаётся и принимается каждый байт. Здесь изображён момент, когда уже произошли два тактовых импульса, и биты 7 и 6 выданы на линию OUT. Одновременно принимаются 2 бита (также 7 и 6) нового байта. После 8 тактовых импульсов посылаемый байт окажется полностью «снаружи», а принимаемый байт будет полностью загружен в регистр. Если PIC работает как мастер (ведущий), он сам генерирует тактовые импульсы, в которых нуждается ведомое устройство, чтобы своевременно сдвигать регистр и таким образом быть в состоянии принимать/передавать правильные биты. В PIC-микроконтроллере таким регистром является регистр SSPBUF, в который и помещается один байт, чтобы он начал его передавать, а новый принимать. Однако, нельзя слишком рано считывать SSPBUF (пока данные сдвигаются). Поэтому мы проверяем флаг BF, который установлен во время процесса передачи.

MMC-карты могут работать на тактовой частоте до 20 МГц (CLK). Интерфейс SPI PIC-микроконтроллера может настраиваться максимально на 5 МГц. Однако, если PIC соединяется с MMC длинными проводниками, то тактовую частоту следует снизить, иначе форма сигнала исказится, и это приведёт к ошибкам. (У меня провода длиной 15 см.)

Временная диаграмма на **Рис. 4** показывает возможности настройки интерфейса SPI на PIC (красным цветом подчёркнуты значения битов CKP, CKE и SMP, которые нужно установить в PIC-микроконтроллере для работы по SPI с MMC, и соответствующие им временные диаграммы).

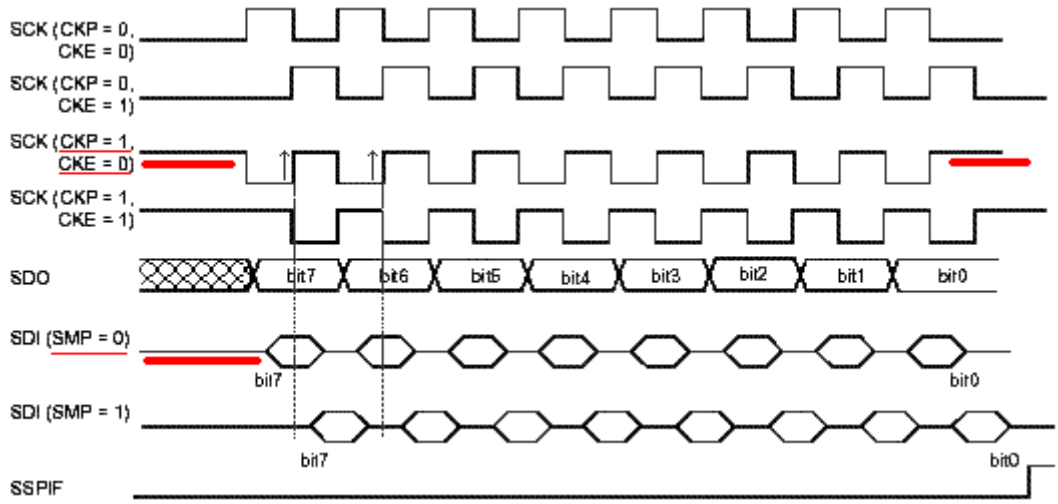
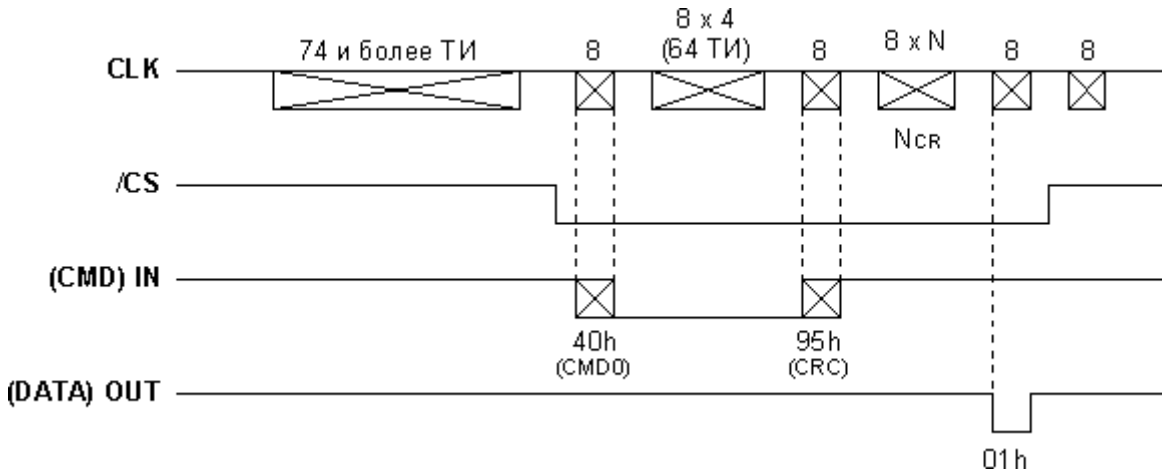


Рис. 4. Временные диаграммы различных режимов SPI

Инициализация MMC в режиме SPI

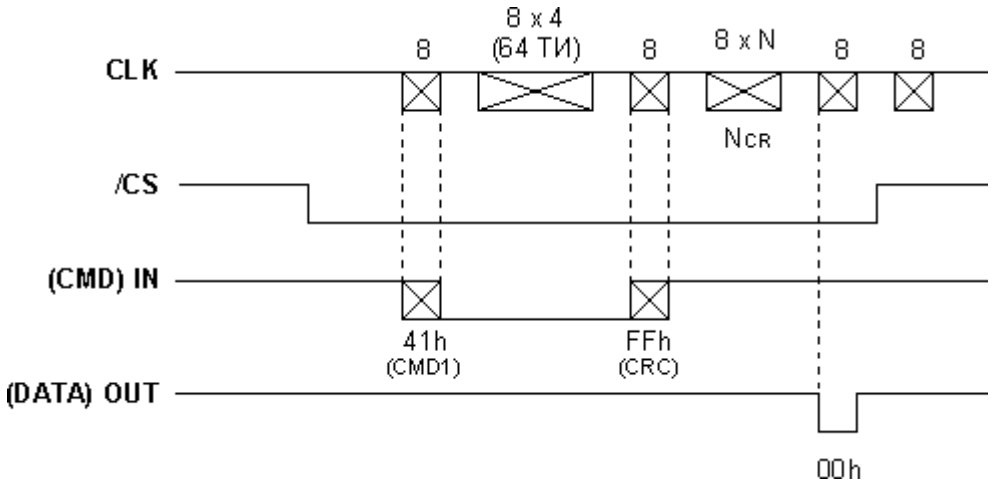
Примечание: на схемах, приведённых ниже, ТИ – тактовый импульс, N – неопределённое количество.

Сброс (CMD0)



0. Подача напряжения
1. Посылка 80 тактовых импульсов
2. Активация сигнала /CS, т.е. /CS=0 (так как активным уровнем является 0)
3. Посылка команды CMD0
4. Байт ответа = 0x01? Если нет – ОШИБКА, если да – шаг 5
5. Посылка 8 тактовых импульсов

Инициализация (CMD1)



6. Посылка команды CMD1
7. Байт ответа = 0x00? Если нет – вернуться на шаг 6, если да – перейти к шагу 8.
8. Посылка 8 тактовых импульсов.

Инициализация MMC в режиме SPI завершена.

Рабочая программа, функциональное описание

`SerString("Hallo Welt")`	Эта функция служит для выдачи сообщений.

`while(!TXIF); //Проверка, пуст ли регистр TXREG`
`TXREG ='W'; //Последовательная передача буквы "W"`
	Эти команды используются для последовательной передачи символов.

`a=SPI(b);`	Функция передаёт символ **b** и принимает символ в переменную **a**.
`Command(0x49,0,512,0xFF );`	Передаёт в MMC команду и адрес.
`MMC_Init()`	Здесь настраивается SPI, выполняется сброс MMC и инициализация MMC в режиме SPI.

```
#include <C:\cc5\16F877.H>
// Speicherschutz 12-13-4-5=выкл., Debug 11=выкл., ProgrammFlash 9=вкл., EEpromRead 8=вкл., NiederVoltProgr
7=выкл.
// NiederVoltReset 6=вкл., EinschaltTimer 3=вкл., WachDogTimer 2=выкл., Oszilator 01=XC
#pragma config |= 0b.11.111101.11.00.10

#pragma bit CS @ PORTC.2        //Выход для выбора микросхемы (сигнала CS)

#pragma origin 100              //С адреса 100 в программной памяти
//*********************************************************************

void InitUSART()
{
BRGH=1;                 // Высокая скорость передачи данных
//SPBRG=129;            // (9600 бод при входной тактовой частоте 20 МГц)
//SPBRG=64;             // (19200 бод при входной тактовой частоте 20 МГц)
//SPBRG=32;             // (38400 бод при входной тактовой частоте 20 МГц)
SPBRG=10;               // (115200 бод при входной тактовой частоте 20 МГц)
SPEN = 1;               // Set_Serial_Pins;
SYNC = 0;               // Set_Async_Mode;
TX9 = 0;                // Set_8bit_Tx;
RX9 = 0;                // Set_8bit_Rx;
CREN = 1;               // Enable_Rx;
TXEN = 1;               // Enable_Tx;
RCIE=0;                 // Выключить прерывание от Rx
}
//*********************************************************************

void SerString(const char *str) // Последовательная передача строки
{
char ps;
ps = *str;              // Указатель на начало строки в ps
```

```
while(ps>0)             // Проверка, не достигнут ли конец строки
        {
        str++;                  // Увеличение указателя для доступа к следующему символу
        if (ps== 0) break;      // Проверка, не достигнут ли конец строки
        while(!TXIF);           // Проверка, свободен ли регистр TXREG
        TXREG =ps ;             // Передача байта данных
        ps = *str;              // Запоминаем содержание указателя в ps
        }
}
//*********************************************************************

char SPI(char d)                // Передача по SPI-интерфейсу
{                               //
SSPBUF=d;
while (!BF);                    // Ожидание окончания передачи
return SSPBUF;                  // и одновременно принимаем
}
//*********************************************************************

char Command(char befF,uns16 AdrH,uns16 AdrL,char befH )
{                               // Передача команды в MMC
char a;
SPI(0xFF);
SPI(befF);
SPI(AdrH.high8);
SPI(AdrH.low8);
SPI(AdrL.high8);
SPI(AdrL.low8);
SPI(befH);
SPI(0xFF);
return SPI(0xFF);               // Возвращаем ответ
}
//*************************************************************************

bit MMC_Init()
{
// Инициализация SPI
SMP=0;          // Вход действителен в середине синхроимпульса
CKE=0;          // Приём данных по переднему фронту
CKP=1;          // Высокий уровень - это пассивное состояние
SSPM1=1;        // Скорость f/64(312kHz), Мастер
//SSPM0=1;      // Скорость f/16(1,25MHz), Мастер
SSPEN=1;        // Включить SPI

CS=1;           // MMC отключена

char i;         // Переменные

//MMC стартует в режиме SPI, сброс
for(i=0; i < 10; i++)SPI(0xFF);         // 10*8=80 тактовых импульсов
CS=0;                                   // Включить MMC

// CMD0
if (Command(0x40,0,0,0x95) !=1)goto Fehler ;    // Сброс

st:                                             // Если MMC не существует, остаётся
// CMD1                                         // Здесь находится программа
if (Command(0x41,0,0,0xFF) !=0) goto st ;       // Повторять CMD1 до тех пор, пока
/*
// Считваем CID
if (Command(0x4A,0,0,0xFF) !=0) goto Fehler;    // Идентификационный номер карты
for(i=0; i < 20; i++)
{
while(!TXIF);
TXREG =SPI(0xFF);
}

// Считываем CSD                                 // Специфические данные карты (свойства карты)
if (Command(0x49,0,0,0xFF) !=0) goto Fehler;
for(i=0; i < 20; i++)
{
while(!TXIF);
TXREG =SPI(0xFF);
}
*/
// Установить длину (по умолчанию 512 байтов)
//if (Command(0x50,0,16,0xFF) !=0) goto Fehler; // 16-байтовый режим для чтения

return 1;
Fehler:
```

```
return 0;
}
//*****************************************************************************

void main(void)
{
INTCON=0;                               // Отключить прерывания
ADCON1=0x6;                             // PortA как цифровые выходы
TRISC=0b.1101.0011;                     // sck rc3-0, sdo rc5-0, CS rc2-0.
TRISB=0b.0000.0010;                     // RB2>TX, RB1>RX
uns16 i;                                // Переменная 0...65535
InitUSART();                            // Инициализация последовательной передачи данных
SerString("Ich bin ON ");               // Стартовое сообщение
if (MMC_Init()) SerString("MMC ON ");   // Инициализация MMC и сообщение, если всё нормально!
//*****************************************************************************

// 512-байтовый режим чтения
if (Command(0x51,0,512,0xFF) !=0) SerString("Lese_resp_Fehler ");
while(SPI(0xFF) != 0xFE);                       // Ожидание 0xFE - начала каждой передачи данных
for(i=0; i < 512; i++)
        {
        while(!TXIF);                           // Проверка, пуст ли регистр TXREG
        TXREG =SPI(0xFF);                       // Передача байта данных
        }
SPI(0xFF);                              // В конце два незначимых байта
SPI(0xFF);
//*****************************************************************


// 512-байтовый режим записи
if (Command(0x58,0,512,0xFF) !=0) SerString("Schreib_resp_Fehler ");
SPI(0xFF);
SPI(0xFF);
SPI(0xFE);
for(i=0; i < 512; i++)
        {
        SPI('M');
        }
SPI(255);                               // в конце передаём два незначимых байта
SPI(255);
i=SPI(0xFF);
i &=0b.0001.1111;
if (i != 0b.0000.0101) SerString("Schreib_Fehler ");
while(SPI(0xFF) !=0xFF);                // ожидаем окончание состояния занятости.

//*****************************************************************
while(1);                               //здесь программа заканчивается
```

Выполнение программы

Сначала настраиваются порты.

Затем инициализируется последовательный интерфейс для работы на скорости 115200 бод, без бита паритета, с 1 стоп-битом, и выдаётся сообщение "Ich bin ON".

Затем выполняется инициализация MMC; если она успешна, выдаётся сообщение "MMC ON".

Далее 512 байтов считываются и передаются по последовательной связи в ПК.

Затем записываются 512 байтов с буквой "M".

Запись и чтение выполняются с адреса 512. С адреса 512 начинается зарезервированная область, отформатированная под FAT16, так что сохраняются даже данные и файловая система на вашей карте MMC.

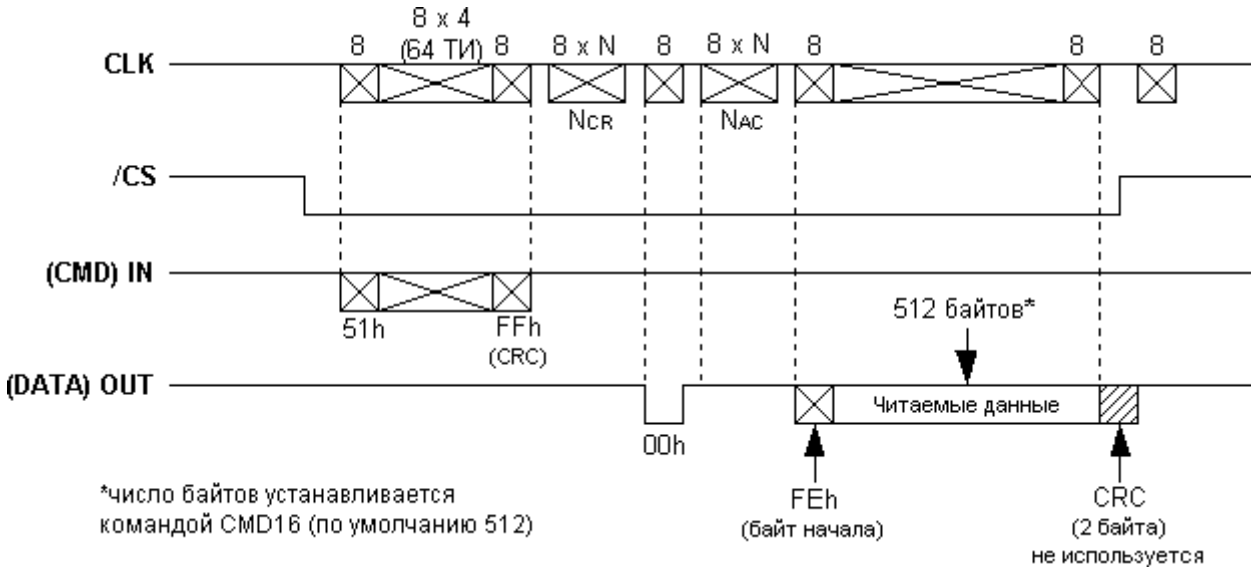
При ошибках могут выдаваться следующие сообщения:

- Lese_resp_Fehler - нет ответа на команду чтения.
- Schreib_resp_Fehler – нет ответа на команду записи.
- Schreib_Fehler – нет подтверждения после записи.

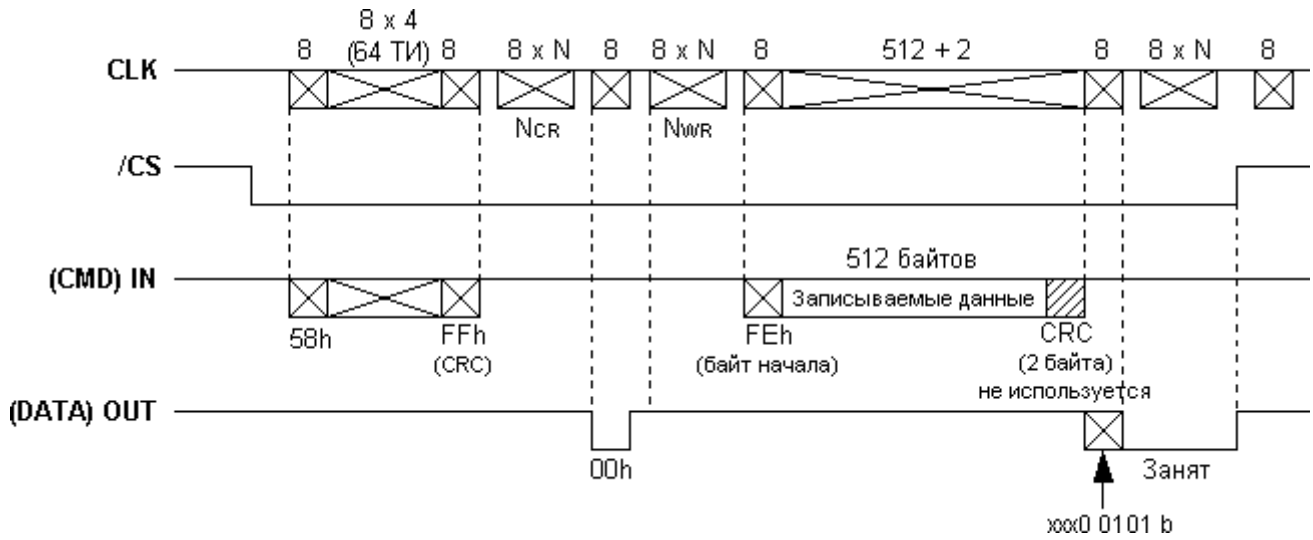
С этой программой можно читать и записывать данные по любому адресу. Можно устанавливать любую величину блока чтения.

Ниже приведены временные диаграммы процесса чтения и процесса записи.

Чтение



Запись



Скачать: Электрическую схему + Исходный код + HEX-файл**.**

Программный интерфейс SPI

Не все типы PIC имеют интерфейс SPI, поэтому я написал подпрограмму, которая делает возможной связь по SPI с любым PIC-микроконтроллером. В программе можно свободно выбирать выводы для подключения MMC.

Этот выбор происходит в начале программы:

```
#pragma bit CS @ PORTC.2                // Выход для выбора микросхемы (CS)
#pragma bit SCK @ PORTC.3               // Выход синхронизации
#pragma bit SDO @ PORTC.5               // Выход выходящих данных
#pragma bit SDI @ PORTC.4               // Вход входящих данных
```

Не забудьте настроить входы/выходы соответствующим образом!

```
TRISC=0b.1101.0011;             // SCK rc3-0, SDO rc5-0, CS rc2-0
```

Конечно, можно взять любой порт и выводы!
Функция `char SPI(char d)` заменяется на:

```
char SPI(char out)              // Частота программного SPI примерно 620 КГц
{
char in,i;
for (i=0;i<8;i++)
        {
        nop2();
        SCK=0;                  // Задний фрон синхроимпульса
        nop();
        SDO=out.7;              // Подготавливаются данные
        nop();
        out=out<<1;
        nop2();                 // Задержка, чтобы дать ведомому возможность подготовить данные
        in=in<<1;
        SCK=1;                  // Передний фрон синхроимпульса
        nop();
        in.0=SDI;               // Считывание данных
        }
return in;
}
```

При более низких тактовых частотах программа тоже работает, только медленнее.
MMC можно подключить к: PIC16F870, PIC16F871, PIC16F872, PIC16F873, PIC16F874, PIC16F876, PIC16F877, PIC16F84, PIC12F675, PIC12F629.
Здесь находится программа для тестирования: программный SPI-интерфейс в MMC. Она приспособлена для PIC16F877, так как я не хотел менять аппаратную часть.

Открытые вопросы

− MMC может записывать только блоки по 512 байтов, где я должен буферизовать их?
− Как могу я использовать FAT-таблицу, чтобы считать собранный файл из MMC-кард-ридера?

Я ищу прямой ответ на эти вопросы. Если что-то придёт мне на ум, я помещу это здесь, в рамках моей Web-страницы.

Нужна ли промежуточная буферизация?

Это не нужно, так как каждый байт можно передавать в MMC с любой скорость, даже 1 байт в час. Только они будут записаны в память лишь после 512 байтов.

FAT

Описание системы FAT, чтения MMC с системой FAT16, программа-пример.