О MMC и SD картах.

Или о том, как правильно перевести их в SPI режим.

© Romanich
E-mail: dre1983@mail.ru

Владивосток 2008

Часто в сети Интернет можно найти кучу воплей по части неудачной инициализации карт памяти, конкретно Multi Media Card (MMC) и Secure Digital (SD) и криков о помощи разрешить данную проблему. И столько же бесконечное количество примеров, как это сделать. Но ни один пример практически не будет гарантированно работать со всеми MMC и SD, так как есть очень тонкие особенности, не позволяющие коду инициализации работать безотказно на всех картах.

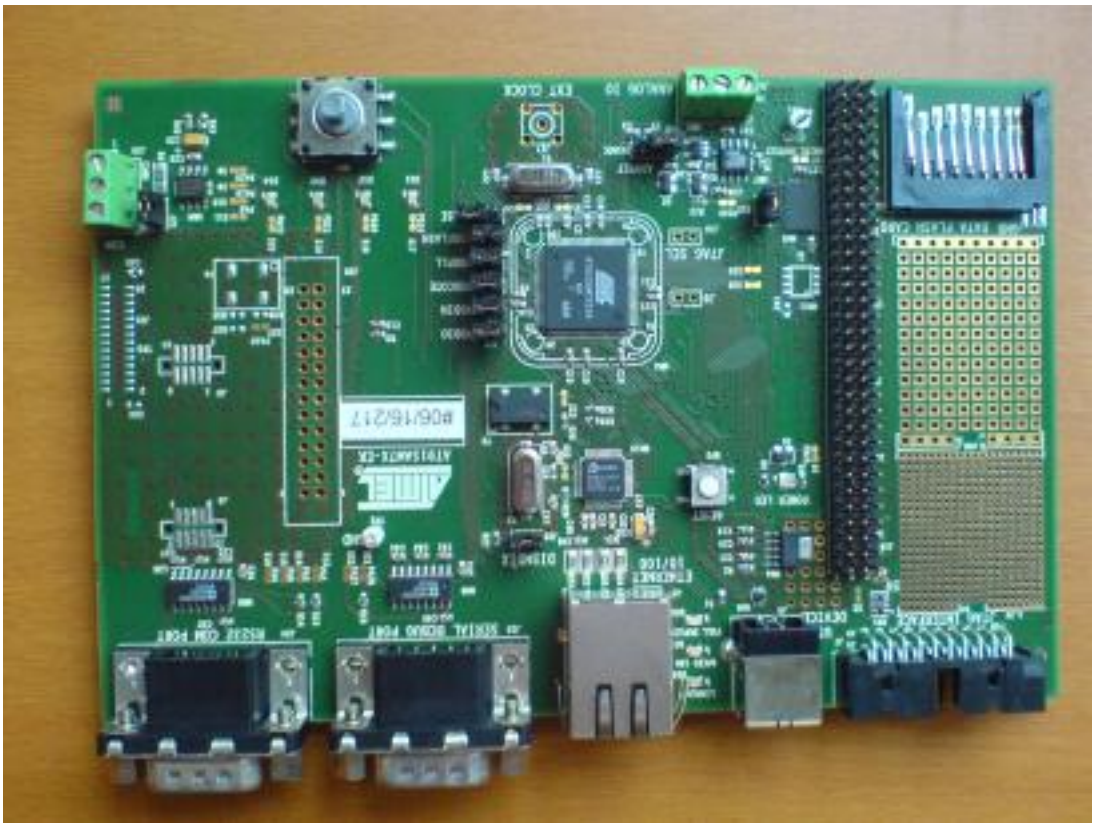
Цель данной статьи – показать тонкости перевода карт памяти в режим SPI, который очень распространен и позволяет работать со всеми картами одинаково. Статья базируется на личных практических опытах, проведённые мной в настоящем времени.

Начну с того, в статье рассмотрены карты MMC и SD. Остальные карты выходят за рамки обсуждения данной статьи. Это относится к mini SD, SD micro и прочим Duo Stick'ам, Smart Media'м, TransMedia'м.

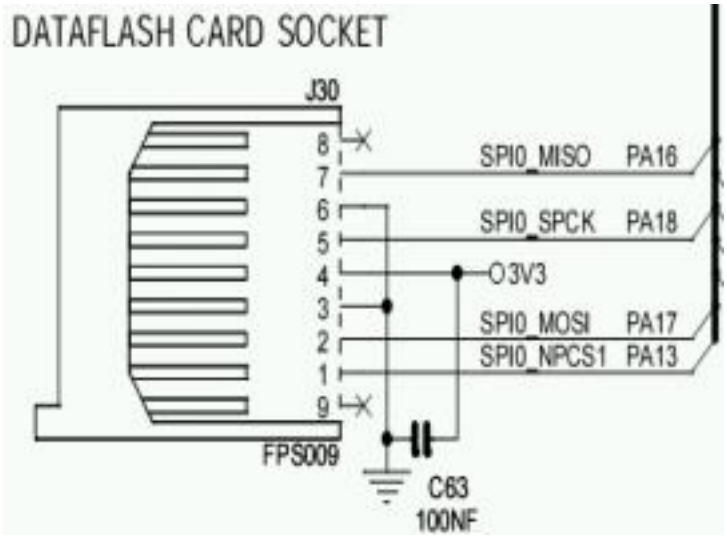
В качестве опытных образцов выступили следующие карты:

- SD 16MB производства Panasonic (JAPAN)
- MMC 16MB производства Canon (JAPAN)
- MMC+ 128MB производства Kingston (TAIWAN)

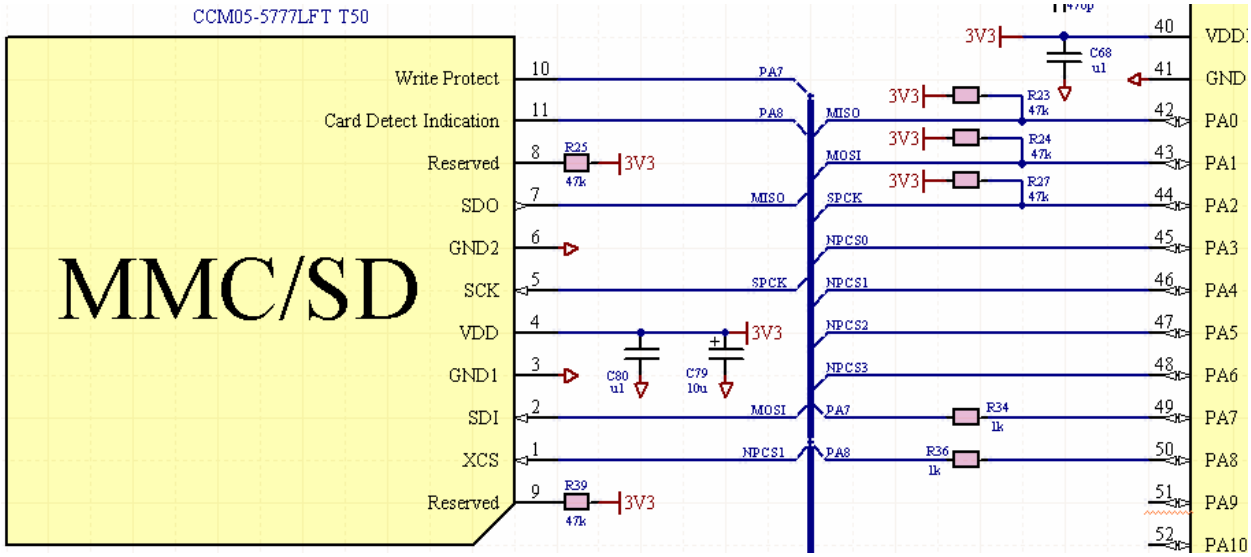
В качестве хоста использовалась Evaluation Board AT91SAM7X-EK на базе ARM7 контроллера AT91SAM7X256 (тактовая частота установлена около 48 MHz).



На отладочной плате выведен разъём для подключения рассматриваемых карт памяти. Карты подключены к SPI контроллера по схеме ниже:



Выводы 8 и 9 висят в воздухе (не подтянуты), тем не менее, MMC карты работали без сбоев. Но всё же рекомендую в своих конструкциях эти выводы подключать через резистор (номинал несколько десятков kΩ) к плюсу питания. Например, в Advanced MicroMachine применено такое подключение:



Когда с подключением более-менее разобрались, переходим к программированию. Тактовая частота контроллера 48 МГц. На этапе инициализации используется SPI с частотой синхроимпульсов 400 кГц. После завершения инициализации частота синхроимпульсов 24 МГц (все операции чтения/записи делались на этой частоте).

Все проверенные мной три карточки прекрасно работали с SPI в режимах 0 и 3. Для AT91SAM7X256 полярность и фаза определена так:

Mode 0: CPOL=0, NCPHA=1 (в программе константа битового поля SPI_MODE=2)
Mode 3: CPOL=1, NCPHA=0 (в программе константа битового поля SPI_MODE=1)

Обратите внимание, что в битовом поле задаётся **инверсное** значение фазы SPI. Это обстоятельство следует учесть, при переносе программы с платформы ARM7 на платформу AVR, например.

В программе используется “ручное” управление выводом NPCS1, так как его требуется опускать на несколько байт передачи (хотя SAM7 контроллеры могут аппаратно это делать, просто конечная цель была получить рабочий код для карточек, а не сидеть долго над даташитом контроллера).

Теперь перейдём непосредственно к фрагментам кода.

```
CardCS(1);
 for(b=0;b<10;b++) SPI(0xFF); //Посылаем 10 байтов 0xFF при неактивном чипселекте
```

Это следует учитывать, если на SPI висят ещё другие устройства, так как хост может посылать байты 0xFF другому выбранному устройству, что дополнительно приведёт к сбросу карты (которая не выбрана).

```
CardCS(0);
 SPI(0x40); //Даём команду CMD0 (GO_IDLE_STATE)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0x95); //Correct CRC
```

При вызове команды GO_IDLE_STATE должен быть указан правильный CRC, иначе возможны ошибки. В следующих командах CRC можно указывать любым.

```
/*
 Опрашиваем пока не встретится 0xFF
 Это ОЧЕНЬ ВАЖНО!!!
 Иначе при вызове следующей команды в ответе R1 будут вечно устанавливаться бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(b!=0xFF) b=SPI(0xFF);
```

Очень существенный фрагмент!!! Без которого карты SD будут вечно возвращать ответ R1 со значением 5 вместо 1 или 0, тоесть ошибку. Ни о каком переводе в режим SPI с такой ошибкой быть не может. При возникновении такой ошибки следующие команды игнорируются до тех пор, пока текущая команда не будет возвращать ответ с отсутствующим битом 2.

Далее происходит нечто интересное. Во всех статьях по SD картам написано, что вместо команды CMD1 (которая применяется в MMC картах) нужно использовать CMD55 и CMD41, тоесть делать ACMD41. Всё это верно, **НО**: моя SD карточка также корректно выполняет CMD1 !!! И даже успешно переводится в SPI режим, читается и пишется **J** Выдвигаю предположение, что данная SD карта **совместима** с MMC !

Пробовал применить ACMD41 вместо CMD1 для карт MMC. Ничего хорошего это не принесло – обе мои MMC карточки (MMC и MMC+) “повисли”.

```
/*
 Ждём примерно 0.25-1 секунды
 Если не ждать, а снова посылать CMD55 и ACMD41, то пробуждение карты в некоторых случаях может быть очень долгим (до 25 секунд)
 */
 Simple Delay (1000000);
```

Тоже важный момент, достойный внимания. Особенно когда питают схему от слаботочного источника питания. Не знаю почему, но на компьютере, который у меня на работе (от его разъёма USB питалась отладочная плата с SD карточкой) эта задержка необходима! Иначе слишком долго ждать придется, пока карта перейдёт в SPI режим. На домашнем компьютере эта задержка оказалась ненужной (тоже от разъёма USB питаю, но там ток отдачи больше). Предполагаю, что это связано с током отдачи источника питания. Нужность задержки проверялась только на SD карте, на картах MMC она не мешает, поэтому можно оставить.

Далее. Разработчики карт намудрили с битами в байте ответа R1 и в Token-байте. Сразу же после выполнения команды чтения идёт ответный байт R1, который расписывается так:

```
/*
 Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 Для всех карт:
 0x00 - Ошибок нет
 Для SD:
 0x20 - Address Error (Адрес не кратен 512)
 Для MMC:
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон)
 Для MMC+:
 0x20 - Address Error (Адрес не кратен 512)
 0x40 - Parameter Error (Адрес вышел за диапазон)
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон и не кратен 512)
 */
```

Тоесть в этом месте:

1) для SD карты ошибка выхода адреса за диапазон прозрачна

2) для MMC карты взводятся почему-то оба бита Parameter и Address Error при ошибке выхода адреса за диапазон

3) лишь только MMC+ честно выставляет биты ошибки в трёх разных случаях

Вывод: нужно R1=0, иначе какая-то ошибка.

```
/*
 Опрашиваем пока не встретим что-нибудь отличное от 0xFF
 Для всех карт:
 0xFE - Data Token (Первый служебный байт данных)
 Для SD:
 0x08 - Error Token Out Of Range (Адрес вышел за диапазон)
 */
```

Тоесть здесь:

1) Для SD именно в этом месте фиксируется ошибка выхода адреса за диапазон

2) Остальным картам всё равно

Вывод: нужно Token=0xFE, иначе какая-то ошибка.

Это касаемо чтения секторов. При записи следующая картина:

```
/*
 Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 Для всех карт:
 0x00 - Ошибок нет
 Для SD:
 0x20 - Address Error (Адрес не кратен 512)
 Для MMC:
 0x20 - Address Error (Адрес не кратен 512)
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон)
 Для MMC+:
 0x20 - Address Error (Адрес не кратен 512)
 0x40 - Parameter Error (Адрес вышел за диапазон)
 0x60 - Parameter Error Address Error (Адрес вышел за диапазон и не кратен 512)
 */
```

Что-то MMC решила при записи сектора выставить бит Address Error в случае если адрес не кратен 512 и по-прежнему выставляет оба бита в случае ошибки выхода адреса за диапазон.

Ниже дана тестовая программа целиком. Она инициализирует необходимое железо, далее записывает на сектор карты информацию, затем её читает с проверкой на правильность. В случае, если записано правильно, по RS232 отсылается серия из бесконечных 1. В противном случае – серия из 0.

```
#include "TYPE.H"         //Описание типов
#include "AT91SAM7X256.H" //Описание регистров AT91SAM7X256

u32 RandomSeed; //Для инициализации и хранения случайного числа

u32 Random(u32 mod) //Случайное число 0..0xFFFFFFFF
{
 RandomSeed=(RandomSeed*25173)+13849;
 if(!mod) return RandomSeed;
 return (RandomSeed%mod);
}

void SimpleDelay(u32 d) //Простенькая задержка
{
 while(--d);
}

void PrepareClock(void) //Enable SYS, PIOA, SPI0 Clock
{
 PMC_PCER=(1<<1)|(1<<2)|(1<<4);
 PMC_PCDR=~((1<<1)|(1<<2)|(1<<4));
}

void PreparePIOA(void) //PIOA Initialization
{
 PIOA_PER=~((1<<27)|(1<<28)|(1<<16)|(1<<17)|(1<<18));
 PIOA_PDR=(1<<27)|(1<<28)|(1<<16)|(1<<17)|(1<<18);
 PIOA_OER=(1<<12)|(1<<13)|(1<<14)|(1<<15);
 PIOA_ODR=~((1<<12)|(1<<13)|(1<<14)|(1<<15));
 PIOA_SODR=(1<<12)|(1<<13)|(1<<14)|(1<<15);
 PIOA_CODR=~((1<<12)|(1<<13)|(1<<14)|(1<<15));
 PIOA_IFER=0x00000000;
 PIOA_IFDR=0xFFFFFFFF;
 PIOA_IER=0x00000000;
 PIOA_IDR=0xFFFFFFFF;
 PIOA_MDER=0x00000000;
 PIOA_MDDR=0xFFFFFFFF;
 PIOA_PPUER=0x00000000;
 PIOA_PPUDR=0xFFFFFFFF;
 PIOA_ASR=(1<<27)|(1<<28)|(1<<16)|(1<<17)|(1<<18);
 PIOA_BSR=0x00000000;
 PIOA_OWER=0xFFFFFFFF;
 PIOA_OWDR=0x00000000;
 PIOA_ODSR=0xFFFFFFFF;
}

void CardCS(u32 cs) //Card ChipSelect (Software NPCS1)
{
 PIOA_OWER=(1<<13);
 PIOA_OWDR=~(1<<13);
 PIOA_ODSR=(cs<<13);
}

void PrepareUART(void) //UART Initialization
{
 DBGU_CR=(1<<2)|(1<<3)|(1<<5)|(1<<7);
 DBGU_MR=0x00000800;               //Normal Mode, No Parity
 DBGU_BRGR=(47923200/16/57600);    //Скорость 57600 kbps
 DBGU_CR=(1<<4)|(1<<6);
}

void OutUART(u32 b) //Передача байта по UART
{
 while(!(DBGU_SR&2));
 DBGU_THR=b;
}

void PrepareSPI0(void) //SPI0 Initialization
{
 SPI0_CR=1;      //Enable SPI
 SPI0_MR=1;      //Master Mode
 SPI0_IER=~0x3FF; //Прерывания запрещены
 SPI0_IDR=0x3FF;
}

#define POLPHA 2 //SD и MMC работают режимах SPI 0(POLPHA=2) и 3(POLPHA=1)

void SpeedSlow(void) //Маленькая скорость SPI
{
 SPI0_CSR=(120<<8)|POLPHA; //48MHz/120=400kHz
}
```

```
void SpeedFast(void) //Большая скорость SPI
{
 SPI0_CSR=(2<<8)|POLPHA; //48MHz/2=24MHz
}

u8 SPI(u8 b) //Передача и приём байта по SPI
{
 while(!(SPI0_SR&2)); //ожидание пока буфер передачи не пустой
 SPI0_TDR=b;          //запись в регистр передачи
 while(!(SPI0_SR&1)); //ожидание пока буфер приема пустой
 return SPI0_RDR;     //чтение из регистра приема
}

void PrepareSD(void) //Перевод в режим SPI карты SD (для MMC карт не работает)
{
 u8 b;
 SpeedSlow(); //Медленный обмен
 CardCS(1);
 for(b=0;b<10;b++) SPI(0xFF); //Посылаем 10 байтов 0xFF при неактивном чипселекте
 CardCS(0);
 SPI(0x40); //Даём команду CMD0 (GO_IDLE_STATE)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0x95); //Correct CRC
 while(b==0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 /*
 Опрашиваем пока не встретится 0xFF
 Это ОЧЕНЬ ВАЖНО!!!
 Иначе при вызове следующей команды в ответе R1 будут вечно устанавливаться бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(b!=0xFF) b=SPI(0xFF);
 //А теперь инициализируем карту до тех пор, пока она не выйдет из Idle State
 Again:
 SPI(0x77); //Даём команду CMD55 (APP_CMD)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0xFF);
 while(b==0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 /*
 Опрашиваем пока не встретится 0xFF
 Это ОЧЕНЬ ВАЖНО!!!
 Иначе в ответе R1 будут вечно устанавливаться бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(SPI(0xFF)!=0xFF);
 SPI(0x69); //Даём команду ACMD41 (APP_SEND_OP_COND)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0xFF);
 while(b==0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 /*
 Опрашиваем пока не встретится 0xFF
 Это ОЧЕНЬ ВАЖНО !!!
 Иначе в ответе R1 будут вечно устанавливаться бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(SPI(0xFF)!=0xFF);
 /*
 Ждём примерно 0.25-1 секнунды
 Если не ждать, а снова посылать CMD55 и ACMD41, то пробуждение карты в некоторых случаях может быть очень долгим (до 25 секунд)
 */
 SimpleDelay(1000000);
 if(b!=0) goto Again; //Если карта не проснулась (в R1 всё ещё установлен бит 0) то заново посылаем ОБЕ(!!!) команды CMD55 и ACMD41
 CardCS(1);
 SpeedFast(); //Быстрый обмен
}

void PrepareMMC(void) //Перевод в режим SPI карты MMC (??? моя SD карта на 16MB от Panasonic также успешно переводится этой функцией !!!)
{
 u8 b;
 SpeedSlow(); //Медленный обмен
 CardCS(1);
 for(b=0;b<10;b++) SPI(0xFF); //10 байтов 0xFF при неактивном чипселекте
 CardCS(0);
 SPI(0x40); //Даём команду CMD0 (GO_IDLE_STATE)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0x95); //Correct CRC
 while(b==0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 /*
```

```
 Опрашиваем пока не встретится 0xFF
 Здесь это необязательно, но следующая команда CMD1 в первый раз даст
 ответ R1, в котором устанавится бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(b!=0xFF) b=SPI(0xFF);
 //А теперь инициализируем карту до тех пор, пока она не выйдет из Idle State
 Again:
 SPI(0x41); //Даём команду CMD1 (SEND_OP_COND)
 SPI(0);
 SPI(0);
 SPI(0);
 SPI(0);
 b=SPI(0xFF);
 while(b==0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 /*
 Опрашиваем пока не встретится 0xFF
 Это ОЧЕНЬ ВАЖНО!!!
 Иначе в ответе R1 будут вечно устанавливаться бит 2 (Illigal Command) и бит 0 (In Idle State)
 */
 while(SPI(0xFF)!=0xFF);
 /*
 Ждём примерно 0.25-1 секнунды
 Если не ждать, а снова посылать CMD55 и ACMD41, то пробуждение карты в некоторых случаях может быть очень долгим (до 50
секунд)
 */
 //SimpleDelay(1000000);
 if(b!=0) goto Again; //Если карта не проснулась (в R1 всё ещё установлен бит 0) то заново посылаем команду CMD1
 CardCS(1);
 SpeedFast(); //Быстрый обмен
}

u8 Buffer[512]; //Буфер сектора MMC/SD

void InSector(u32 Address) //Чтение сектора 512 байт (адрес должен быть кратен 512)
{
 u8 b;
 u16 i;
 CardCS(0);
 SPI(0x51); //Даём команду CMD17 (READ_SINGLE_BLOCK)
 SPI(Address>>24);
 SPI(Address>>16);
 SPI(Address>> 8);
 SPI(Address    );
 b=SPI(0xFF);
 /*
 Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 Для всех карт:
 0x00 - Ошибок нет
 Для SD:
 0x20 - Address Error (Адрес не кратен 512)
 Для MMC:
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон)
 Для MMC+:
 0x20 - Address Error (Адрес не кратен 512)
 0x40 - Parameter Error (Адрес вышел за диапазон)
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон и не кратен 512)
 */
 while(b==0xFF) b=SPI(0xFF);
 while(b!=0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится 0xFF
 /*
 Опрашиваем пока не встретим что-нибудь отличное от 0xFF
 Для всех карт:
 0xFE - Data Token (Первый служебный байт данных)
 Для SD:
 0x08 - Error Token Out Of Range (Адрес вышел за диапазон)
 */
 while(b==0xFF) b=SPI(0xFF);
 for(i=0;i<512;i++) Buffer[i]=SPI(0xFF); //Читаем сектор 512 байт
 SPI(0xFF); //Читаем два байта CRC
 SPI(0xFF);
 while(SPI(0xFF)!=0xFF); //Опрашиваем пока не встретится 0xFF
 CardCS(1);
}

void OutSector(u32 Address) //Запись сектора 512 байт (адрес должен быть кратен 512)
{
 u8 b;
 u16 i;
 CardCS(0);
 SPI(0x58); //Даём команду CMD24 (WRITE_BLOCK)
 SPI(Address>>24);
 SPI(Address>>16);
 SPI(Address>> 8);
 SPI(Address    );
 b=SPI(0xFF);
 /*
 Опрашиваем пока не встретится что-нибудь отличное от 0xFF (ответ R1)
 Для всех карт:
 0x00 - Ошибок нет
 Для SD:
```

```
 0x20 - Address Error (Адрес не кратен 512)
 Для MMC:
 0x20 - Address Error (Адрес не кратен 512)
 0x60 - Parameter Error, Address Error (Адрес вышел за диапазон)
 Для MMC+:
 0x20 - Address Error (Адрес не кратен 512)
 0x40 - Parameter Error (Адрес вышел за диапазон)
 0x60 - Parameter Error Address Error (Адрес вышел за диапазон и не кратен 512)
 */
 while(b==0xFF) b=SPI(0xFF);
 while(b!=0xFF) b=SPI(0xFF); //Опрашиваем пока не встретится 0xFF
 SPI(0xFE); //Посылаем Data Token
 for(i=0;i<512;i++) SPI(Buffer[i]); //Пишем сектор 512 байт
 SPI(0xFF); //Пишем два байта CRC
 SPI(0xFF);
 while(SPI(0xFF)!=0xFF); //Опрашиваем пока не встретится 0xFF
 CardCS(1);
}

#define SECTOR 12345

main(void)
{
 u16 i;
 PrepareClock();
 PreparePIOA();
 PrepareUART();
 PrepareSPI0();
 PrepareSD();
 //Заполняем буфер псевдослучайными числами
 RandomSeed=5;
 for(i=0;i<512;i++) Buffer[i]=Random(256);
 //Пишем буфер в сектор № 12345
 OutSector(SECTOR<<9);
 //Очищаем буфер
 for(i=0;i<512;i++) Buffer[i]=0;
 //Читаем сектор № 12345 в буфер
 InSector(SECTOR<<9);
 //Проверяем считанный (ранее записанный) сектор
 RandomSeed=5;
 for(i=0;i<512;i++) if(Buffer[i]!=Random(256)) while(1) OutUART(0); //Если хоть один байт не совпал, то вечно слать 0 по COM-порту
 while(1) OutUART(1); //Если все байты в секторе правильны, то вечно слать 1 по COM-порту
}
```

Ну вот и всё, собственно, что я хотел рассказать про MMC и SD карты. Успехов во всех творческих начинаниях ! **J**

Отдельное спасибо Игорю Афонькину (James DiGreze), Вадиму Акимову (LVD) и Игорю Внукову (HardWareMan) за проявленный интерес и посильную помощь.

Apparatchik Romanich
18.07.08