ИСПОЛЬЗОВАНИЕ МИКРОСХЕМЫ DATAFLASH ПАМЯТИ AT45DB642 ФИРМЫ ATMEL В МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМАХ С ИНТЕРФЕЙСОМ SPI

Олег Николайчук
onic@ch.moldpac.md

Статья опубликована: Схемотехника, 2004, №7, с. 18-22

Целью настоящей статьи является ознакомление читателей с проблемами разработки подсистем DataFlash памяти для записи данных большого объема. В рамках настоящей статьи приведено описание программного интерфейса микросхем AT45DB642 и AT45DCB008 фирмы Atmel в микросистемах с интерфейсом SPI.

Создание автономных микроконтроллерных систем сбора и обработки данных, как правило, связано с необходимостью хранения достаточно больших объемов измеренных и обработанных данных. В настоящее время для этих целей обычно используются различные типы Flash памяти. Специально для таких изделий фирмой Atmel разработано семейство микросхем с общим названием DataFlash [1]. Всего фирмой Atmel выпускается 11 типов микросхем с емкостью от 1 Мбита до 128 Мбит (от 125 Кбайт до 16 Мбайт). На страницах нашего журнала уже публиковалась статья, в которой приведены все типы микросхем этого семейства [2]. Наиболее удобной для применения в автономных микроконтроллерных системах сбора и обработки данных из выпускаемых в настоящее время микросхем фирмы Atmel является AT45DB642[2,3], поскольку она единственная оснащена двумя типами интерфейсов: программно-аппаратным параллельным интерфейсом Rapid8 и последовательным интерфейсом SPI. В статье [2] подробно описаны все характеристики и программно-аппаратный интерфейс Rapid8, а также приведена библиотека подпрограмм на языке «С» фирмы Keil[4], поэтому в рамках этой статьи мы не будем приводить подробное описание всех команд микросхемы и всей библиотеки подпрограмм для работы с микросхемой через интерфейс SPI. Мы рассмотрим только основные достоинства интерфейса SPI, типовую схему подключения и основные подпрограммы, необходимые для создания программной библиотеки.

Особо следует отметить, что микросхема AT45DB642 выпускается также в корпусе MMC карточки (Multi Media Card) под названием AT45DCB008. В таком исполнении микросхема имеет только семь контактов SPI интерфейса и полностью идентична микросхеме AT45DB642 по структуре, организации и основным параметрам.

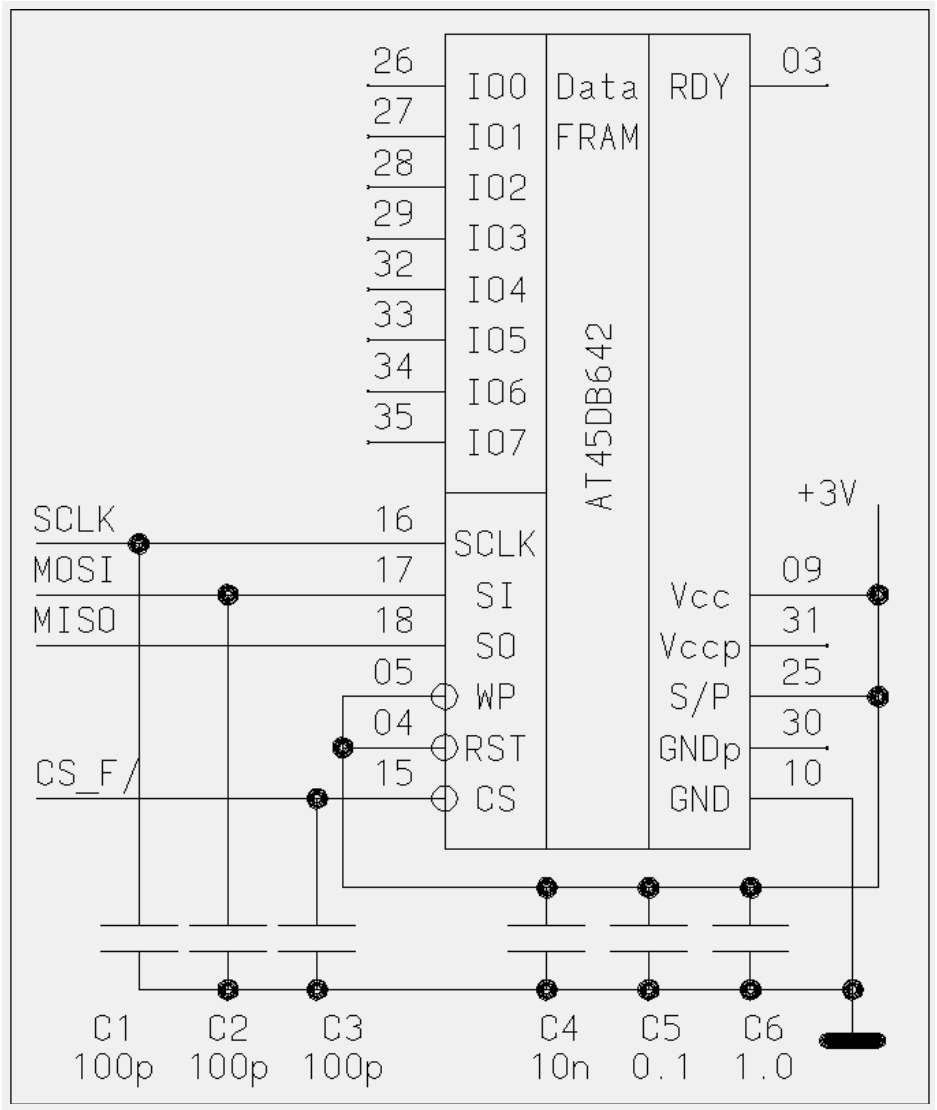
Отметим основные достоинства интерфейса SPI по сравнению с интерфейсом Rapid8.

Во-первых, в микросистемах с интерфейсом SPI используется только четыре вывода микроконтроллера, а не десять выводов (без учета вспомогательных сигналов сброса RST/ и защиты записи WP/), как в интерфейсе Rapid8. Вышеперечисленные вспомогательные сигналы могут использоваться в обоих интерфейсах микросхемы AT45DB642, а в карточке AT45DCB008 эти сигналы вообще отсутствуют. Уменьшение количества используемых выводов микроконтроллера особенно актуально в сложных микроконтроллерных системах, в которых, как правило, ощущается дефицит выводов для связи с различными узлами микросистемы. Еще большую актуальность имеет уменьшение количества используемых выводов в микросистемах, построенных на базе малоформатных микроконтроллеров (с уменьшенным количеством линий ввода/вывода), например, микроконтроллеров C8051F30x/31x/32x/33x/35x фирмы Silicon Laboratories (SiLabs)[5] или AT89C1051/2051/4051 фирмы Atmel[1]. В этих микроконтроллерах обычно каждый вывод у разработчика «на вес золота».

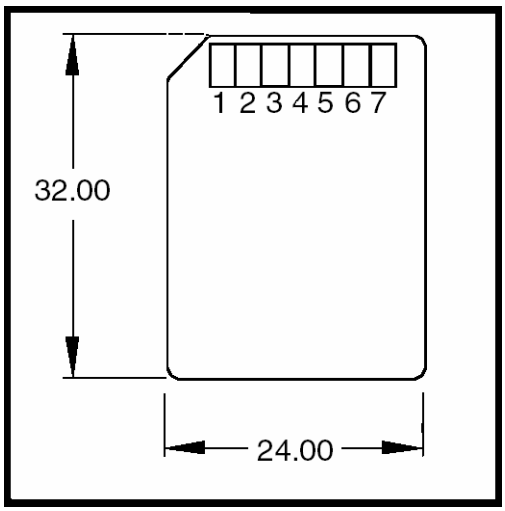
Во-вторых, при использовании SPI интерфейса в микросхеме AT45DB642 не используются буферные (магистральные) узлы, что позволяет значительно сократить энергопотребление.

Основным недостатком SPI интерфейса по сравнению с интерфейсом Rapid8 является более низкая результирующая производительность при операциях записи/воспроизведения, не смотря на то, что предельная частота передачи данных (для микросхемы AT45DB642) по последовательному каналу SPI (в режимах 0 и 3) - до 20 МГц, по параллельному каналу Rapid8 – до 5 МГц.

Типовая схема подключения микросхемы AT45DB642 с интерфейсом SPI приведена на рис.1. Следует отметить, что поскольку микросхема AT45DB642 оснащена двумя интерфейсами, необходимо не забывать подключать вывод переключения режима интерфейсов SER/PAR к напряжению питания для включения интерфейса SPI.



Габаритные размеры карточки AT45DCB008 (толщина – не более 1,5 мм) и расположение контактов показаны на рис.2.



Карточка AT45DCB008 имеет 7 контактов, на которые выведены цепи питания и сигналы интерфейса SPI. Первый контакт – сигнал выборки карточки – CS_F/, второй контакт –

последовательный вход данных MOSI, третий контакт – общий – GND, четвертый контакт – питание (от +2,7 до +3,6 В) – Vp, пятый контакт – сигнал тактирования – SCLK, шестой контакт – у микросхемы AT45DCB008 не используется, седьмой контакт - последовательный выход данных MISO. Следует особенно подчеркнуть, что назначение выводов и размеры карточки AT45DCB008 полностью соответствует назначению выводов стандартных MMC карт, которые также могут работать в режиме SPI. Имеется только два отличия. Первое состоит в том, что выбора карточки (первый контакт) в MMC картах не используется и всегда должен быть соединен с общим проводом GND. Второе отличие состоит в том, что шестой контакт также должен быть соединен с землей. Поэтому, для обеспечения совместимости с сигналами MMC карт, шестой контакт карточки AT45DCB008 обычно также соединяется с землей GND.

На рис.1. кроме собственно микросхемы показаны 6 конденсаторов, установка которых рекомендуется производителем – фирмой Atmel. Первые три служат для исключения «шильев» на фронтах управляющих и информационных сигналов, т.к. рассматриваемые микросхемы очень чувствительны к коротким импульсным помехам. Другие три конденсатора служат для фильтрации помех по питанию. Все указанные на рис.1 конденсаторы обязательно должны устанавливаться не только при использовании микросхемы серии AT45DBxxx, но и при использовании микросхем AT45DCB008 и MMC карт.

Теперь приступим к описанию программного интерфейса. Программный интерфейс также будем рассматривать на примере микроконтроллера C8051F021[6], который наиболее совместим со всеми полноформатными семействами. Напомним читателю, что полное описание всех команд микросхем AT45DB642 и AT45DCB008 приведено в статье[2]. Отметим также, что параллельный интерфейс Rapid8, подробно рассмотренный в вышеупомянутой статье, представляет собой комбинацию стандартного параллельного интерфейса памяти и последовательного интерфейса SPI, но при этом команды и данные в нем передаются последовательно, байт за байтом, в параллельном виде. Иными словами, многие подпрограммы, включенные в состав описанной в [2] библиотеки останутся неизменными и для интерфейса SPI.

Принципиальных программных отличий в библиотеке для интерфейсов Rapid8 и SPI всего четыре:

Первое очевидное отличие заключается в том, что в различных интерфейсах используются различные выводы микроконтроллера. Следовательно, в основном включаемом файле необходимо включить объявления других используемых выводов, например:

```
//******************//
//***    MAIN.H    ***//
//******************//
#ifndef __MAIN__
#define __MAIN__
//Стандартный включаемый файл с описаниями адресов SFR регистров микроконтроллера
#include <Cygnal\C8051F020.h>
// Объявление нестандартного типа byte
#ifndef __BYTE__
#define __BYTE__
   typedef unsigned char byte;
#endif

#define    FALSE       0
#define    ERROR       0
#define    TRUE        1
#define    OK          1
#define    WORK        1
#define            ON          1
#define            OFF         0

//Объявление тактовой частоты микроконтроллера
#define          SYSCLK                22118400

//... Другие объявления ...
```

```
//Объявления линий ввода/вывода  интерфейса SPI
sbit            SCLK        = P0^2;
sbit            MISO        = P0^3;
sbit            MOSI        = P0^4;
sbit            NSS         = P0^5;
sbit            CS          = P3^1;
//... Другие объявления ...
#endif // Main.h
```

Второе отличие заключается в том, что для некоторых команд (операций) интерфейсов AT45DB642 и AT45DCB008 используются различные коды операции – «опкоды» (opcode). Существует достаточно простой способ преодоления этих отличий путем создания общего для обоих интерфейсов включаемого файла “AT45DB642.H”, содержащего «опкоды» всех команд:

```
//******************//
//**  AT45DB642.H  **//
//******************//
#ifndef __AT45DB642__
#define __AT45DB642__

// Определения ОПКОДОВ операций чтения

#define RAPID_CONTINUOUS_ARRAY_READ      0x68
#define SPI_CONTINUOUS_ARRAY_READ        0xE8
#define RAPID_BURST_ARRAY_READ_W_DELAY0x69
#define SPI_BURST_ARRAY_READ_W_DELAY     0xE9
#define RAPID_MAIN_MEMORY_PAGE_READ      0x52
#define SPI_MAIN_MEMORY_PAGE_READ        0xD2
#define RAPID_BUFFER_1_READ              0x54
#define SPI_BUFFER_1_READ                0xD4
#define RAPID_BUFFER_2_READ              0x56
#define SPI_BUFFER_2_READ                0xD6
#define RAPID_STATUS_READ                0x57
#define SPI_STATUS_READ                  0xD7

// Определения ОПКОДОВ других операций
#define BUFFER_1_WRITE                   0x84
#define BUFFER_2_WRITE                   0x87
#define BUFFER_1_ PAGE_WRITE_WITH_ERASE  0x83
#define BUFFER_2_ PAGE_WRITE_WITH_ERASE  0x86

#define BUFFER_1_ PAGE_WRITE_WO_ERASE    0x88
#define BUFFER_2_ PAGE_WRITE_WO_ERASE    0x89
#define PAGE_ERASE                       0x81
#define BLOCK_ERASE                      0x50
#define PAGE_WRITE_THROUGH_BUFFER_1      0x82
#define PAGE_WRITE_THROUGH_BUFFER_2      0x85
#define MAIN_PAGE_TO_BUFFER_1_TRANSFER   0x53
#define MAIN_PAGE_TO_BUFFER_2_TRANSFER   0x55
#define MAIN_PAGE_TO_BUFFER_1_COMPARE    0x60
#define MAIN_PAGE_TO_BUFFER_2_COMPARE    0x61
#define AUTO_ REWRITE_THROUGH_BUFFER_1   0x58
#define AUTO_ REWRITE_THROUGH_BUFFER_2   0x59
#endif // __AT45DB642__
```

При рассмотрении вышеприведенного файла определений легко заметить, что отличие в «опкодах» имеется только для всех операций чтения.

Третье отличие также характерно только для операций чтения, поскольку, как указывалось в [2], операции чтения требуют после передачи трех байтов адреса передавать еще несколько незначащих байтов, необходимых для синхронизации. Так вот, третье отличие состоит собственно в том, что количество этих незначащих байтов различно для рассматриваемых двух интерфейсов. Это наглядно проиллюстрировано в таблице 4 статьи [2]. Операций (или команд) чтения всего 6. Операция чтения статуса вообще не требует передачи незначащих байтов, две команды чтения буфера - предполагают одинаковое количество передаваемых после трех байтов адреса незначащих байтов, равное 1. А вот операции чтения страницы основной памяти, непрерывного чтения и непрерывного чтения с задержкой требуют передачу после трех байтов адреса различного количества незначащих байтов: 60 для интерфейса Rapid8 и 4 для интерфейса SPI. Второе и третье отличие достаточно просто можно нейтрализовать, но об этом мы поговорим несколько позже.

Последнее четвертое отличие связано с особенностями интерфейса SPI. Оно заключается в необходимости включения в состав библиотеки подпрограмм инициализации интерфейса SPI, а также подпрограмм приема и передачи данных. Эти подпрограммы выглядят следующим образом:

```
// Используемые переменные
xdata byte DF_ADDR[4];
xdata byte DF_ERROR;

// Подпрограмма инициализации интерфейса SPI
void SPI0_Init (void)
{
    SPI0CFG = 0x07;    // Данные передаются по переднему
                       // фронту тактовой частоты, 8 битов
    SPI0CN  = 0x03;    // Режим Master; SPI разрешен
    SPI0CKR = SYSCLK/2;//8000000; // Частота SPI <= 4MHz
}

// Подпрограмма получения байта по интерфейсу SPI
byte SPI_GetB (void)
{
  SPIF=0;              // Очистить флаг готовности
  SPI0DAT=0;           // Очистить регистр данных
  while (SPIF == 0);   // Ожидать готовности
  return SPI0DAT;      // Считать данные
}

// Передача байта данных по интерфейсу SPI
void SPI_SetB (byte DB)
{
  SPIF = 0;            // Очистить флаг готовности
  SPI0DAT = DB;        // Записать данные
  while (SPIF == 0);   // Ожидать завершения передачи
}

// Подпрограмма управления выборкой микросхемы
void Set_CS(bit B)
{
   if (B) {Time(1);CS=1;} // Установить бит
   else   {CS=0;Time(1);} // Очистить бит
}

// Подпрограмма формирования адреса
void AddressCalc (int PageAddr, int ByteAddr)
{
   if (PageAddr>8192) DF_ERROR|=0x02;
```

```
        // Если номер страницы больше 8192 – ошибка = 2
  if (ByteAddr>1056) DF_ERROR|=0x01;
        // Если номер байта больше 1056 – ошибка = 1
  // Далее следует вычисление трех байтов передаваемого адреса
  DF_ADDR[0]=(byte)((PageAddr>>5)&0x0FF);
  DF_ADDR[1]=(byte)((ByteAddr>>8)&0x07|(PageAddr&0x1F)<<3);
  DF_ADDR[2]=(byte)(ByteAddr&0x0FF);
  Set_CS(0);
}

// Подпрограмма передачи трех байтов адреса и незначащих байтов
void SPI_AddressWrite (int DCB)
{
int i;

   for(i=0;i<3;i++)  // Передать три байта адреса
                SPI_SetB(SPI_ADDR[i]);
   if (DCB)     // Передать незначащие байты
                for(i=0;i<DCB;i++)      SPI_SetB(0);
}
```

Далее для иллюстрации нейтрализации второго и третьего отличий мы рассмотрит всего две из четырех основные (наиболее часто используемые) команды. Но прежде отметим, что существует два варианта создания и использования библиотеки подпрограмм.

Первый вариант заключается в том, что создается единая библиотека, содержащая функции для работы с одним и другим интерфейсами. В этом случае библиотеку, приведенную в [2], всего лишь следует дополнить приведенными выше функциями инициализации интерфейса SPI, а также продублировать все подпрограммы чтения с отличными названиями (для SPI), «опкодами» и количеством незначащих байтов (4 вместо 60).

Второй вариант заключается в создании единой библиотеки исходных текстов с включением в них директив условного компилирования, а затем компиляции двух различных библиотек для двух различных интерфейсов.

Первый из перечисленных вариантов очевиден и его реализация не представляет особого интереса, поэтому рассмотрим второй вариант на примере основных команд.

В основном включаемом файле библиотеки должно присутствовать объявление выбранного интерфейса, которое можно сделать, например, следующим образом:

```
#define SPI       1 // 1=SPI, 0=RAPID8
```

Отметим, что в нижеследующих примерах все функции, начинающиеся с DF_ приведены и описаны в статье [2].

Первая из основных функций – запись во встроенный буфер оперативной памяти BuffNum начиная с адреса ByteAddr, из буфера Buff оперативной памяти микроконтроллера Len байтов.

```
void BufferWrite (int BuffNum, int ByteAddr, char *Buff, int Len)
{
int i;

    AddressCalc (0,ByteAddr);   // Вычисление адреса - не зависит от интерфейса
// В зависимости от выбранного типа интерфейса используется
// Различные подпрограммы записи байта для передачи ОПКОДА
#ifdef SPI
  SPI_SetB  ((BuffNum) ? BUFFER_2_WRITE : BUFFER_1_WRITE);
  SPI_AddressWrite (0);
#else
  DF_SetB  ((BuffNum) ? BUFFER_2_WRITE : BUFFER_1_WRITE);
  DF_AddressWrite (0);
 #endif
```

```
  for(i=0; i<Len; i++)
  {
#ifdef SPI
     SPI_SetB (Buff[i]);
#else
     DF_SetB (Buff[i]);
#endif
  }
  Set_CS(1);
}
```

Более сложный вид имеет вторая функция – чтения из встроенного буфера оперативной памяти BuffNum начиная с адреса ByteAddr, в буфера Buff оперативной памяти микроконтроллера Len байтов. В ней в зависимости от выбранного типа интерфейса изменяются не только подпрограммы передачи данных, но и ОПКОДЫ.

```
void BufferRead (int BuffNum, int ByteAddr, char *Buff, int Len)
{
int i;
   AddressCalc (0,ByteAddr);   // Вычисление адреса - не зависит от интерфейса
// Различные подпрограммы записи байта для передачи ОПКОДА
#ifdef SPI
  SPI_SetB  ((BuffNum) ?
        SPI_BUFFER_2_READ : SPI_BUFFER_1_READ);
  SPI_AddressWrite (1);
#else
  DF_SetB  ((BuffNum) ?
        RAPID_BUFFER_2_WRITE : RAPID_BUFFER_1_WRITE);
  DF_AddressWrite (1);
 #endif
  for(i=0; i<Len; i++)
  {
#ifdef SPI
     Buff[i]=SPI_GetB ();
#else
     Buff[i]=DF_GetB ();
#endif
  }
  Set_CS(1);
}
```

Две оставшиеся основные команды – запись основной страницы памяти из выбранного буфера и чтение из основной страницы памяти, а также все остальные команды переделываются аналогично из исходных текстов, приведенных в [2].

Приведенные в данной статье сведения позволяют читателю самостоятельно создать необходимую библиотеку, базируясь на полученных в ходе чтения настоящей статьи знаниях.

Литература:

1. http://www.atmel.com/products/
2. О. Николайчук. Использование микросхемы DataFlash памяти AT45DB642 фирмы ATMEL в микроконтроллерных системах с параллельным интерфейсом // Схемотехника, 2004: №2, 37-39; №3, 33-36; №4, 40-41.
3. http://www.atmel.com/dyn/resources/prod_documents/DOC1638.PDF
4. http://www.keil.com
5. http://www.silabs.com
6. https://www.mysilabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Precision_Mixed-Signal/en/C8051F02x.pdf