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

**Олег Николайчук**
onic@ch.moldpac.md

**Статья опубликована:** **Схемотехника, 2004, №2, с. 37-39**
**Схемотехника, 2004, №3, с. 33-36**
**Схемотехника, 2004, №4, с. 40-41**

*Целью настоящей статьи является ознакомление читателей с проблемами разработки подсистем DataFlash памяти для записи данных большого объема. В рамках настоящей статьи приведено описание микросхемы AT45DB642 фирмы Atmel, имеющей параллельный (Rapid8) и последовательный (SPI) интерфейсы, рассмотрена принципиальная схема подключения микросхемы AT45DB642 через параллельный интерфейс к микроконтроллеру C8051F021 фирмы Cygnal, а также приведен и описан комплект подпрограмм для осуществления операций с DataFlash памятью.*

## Введение

При разработке современных автономных микроконтроллерных систем сбора данных довольно часто возникает необходимость в хранении достаточно больших объемов информации, достигающих нескольких единиц или десятков мегабайт. 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). Типы и основные параметры выпускаемых микросхем приведены в таблице 1.

Таблица 1

| ТИП | Емкость, бит | Напряжение питания, В | Количество выводов | Тип интерфейса | Размер страниц, байт | Корпус |
|---|---|---|---|---|---|---|
| AT45DB011B | 1M | 2.7 | 8, 9, 14 | Последовательный SPI | 264 | CBGA, SOIC, TSSOP |
| AT45DB021B | 2M | 2.7 | 8, 9, 28 | Последовательный SPI | 264 | CBGA, SOIC, TSOP |
| AT45DB041B | 4M | 2.7 | 8,14, 28 | Последовательный SPI | 264 | CBGA, SOIC, TSOP |
| AT45DB081B | 8M | 2.7 | 8, 14, 28 | Последовательный SPI | 264 | CASON, CBGA, SOIC, TSOP |
| AT45DB1282 | 128M | 2.7 | 40, 44 | Последовательный SPI | 1056 | CBGA, TSOP |
| AT45DB161B | 16M | 2.7 | 8, 24, 28 | Последовательный SPI | 528 | CASON, CBGA, SOIC, TSOP |
| AT45DB321B | 32M | 2.7 | 28, 32, 44 | Последовательный SPI | 528 | CBGA, SOIC, TSOP |
| AT45DB642 | 64M | 2.7 | 40 | Последовательный SPI, Параллельный Rapid8 | 1056 | TSOP |
| AT45DCB002 | 2M | 2.7 | 7 | Последовательный SPI | 528 | BoF |
| AT45DCB004 | 4M | 2.7 | 7 | Последовательный SPI | 528 | BoF |
| AT45DCB008 | 8M | 2.7 | 7 | Последовательный SPI | 1056 | BoF |

Все выпускаемые микросхемы имеют пониженное напряжение питания, от 2,7 до 3,6 В, что позволяет их использовать в микроконтроллерных системах с автономным питанием. Микросхемы DataFlash выпускаются с двумя типами интерфейса: последовательным – SPI и параллельным Rapid8.

Естественно, что микросхемы с последовательным интерфейсом занимают меньшее количество выводов микроконтроллера, однако предполагают наличие в микроконтроллере аппаратно реализованного интерфейса SPI. Однако, недостатком этого типа интерфейса является более низкая результирующая производительность при операциях записи/воспроизведения за счет необходимости передачи дополнительных управляющих данных. К сожалению, только одна из выпускаемых микросхем - AT45DB642[2] оснащена как последовательным SPI интерфейсом, так и параллельным интерфейсам Rapid8. Именно эту микросхему и ее подключение к микроконтроллеру через параллельный интерфейс Rapid8 мы и будем рассматривать в настоящей статье.

**Особенности микросхемы AT45DB642**

Как уже упоминалось выше, микросхема AT45DB642[2] оснащена двумя интерфейсами – последовательным SPI и параллельным Rapid8. Выбор интерфейса осуществляется с помощью специального вывода, что позволяет использовать в микроконтроллерной системе либо один из них при фиксированной подаче на этот вывод соответствующего напряжения, либо оба интерфейса одновременно при управлении этим выводом от микроконтроллера. Весь объем памяти микросхемы AT45DB642 составляет 64 мегабита. Память имеет иерархическую трехуровневую организацию. Нижний уровень содержит 8192 страницы по 1056 (1К+32 байта) байт каждая. Каждые восемь страниц памяти составляют один блок, т.е. всего имеется 1024 блока по 8448 (8К+256) байта в каждом блоке. В свою очередь блоки объединяются в сектора. Нулевой сектор содержит только один нулевой блок, первый сектор содержит 1-31 блоки, а остальные 2-32 сектора содержат по 32 блока. Столь сложная структура необходима для осуществления оптимальной организации адресации, записи и стирания, а также аппаратной защиты записи нулевого и первого секторов. В микросхеме поддерживаются операции блочного и страничного стирания. Для ускорения записи в микросхеме имеется два независимых буфера оперативной памяти объемом 1056 байт каждый (т.е. объем одной страницы). Во время записи информации в один из буферов оперативной памяти, второй, предварительно заполненный буфер, может автоматически перезаписываться в основную DataFlash память. Поскольку время записи в буфер оперативной памяти на предельной скорости работы интерфейса больше времени перезаписи буфера в основную DataFlash память, можно обеспечить режим непрерывной записи поступающих от микроконтроллера данных. Для считывания данных из основной DataFlash памяти также имеются различные режимы. Возможно непосредственное считывание из основной памяти, считывание через буфер, а также непрерывное считывание всего объема основной DataFlash памяти с выбранного адреса. Это очень удобно, например, для передачи записанных данных через последовательный порт микроконтроллера в персональный компьютер или, например, для воспроизведения записанных звуковых сигналов. Микросхема имеет низкое энергопотребление – около 4 мА в режиме чтения, не более 35 мА в режиме записи и менее 2 мкА в пассивном режиме. Предельная частота передачи данных по последовательному каналу SPI (в режимах 0 и 3) до 20 МГц, по параллельному каналу Rapid8 – до 5 МГц. Микросхема выпускается в 40 выводном корпусе TSOP40. Назначение выводов микросхемы приведено в таблице 2.

Вход переключения режима интерфейсов SER/PAR предназначен для активации последовательного или параллельного интерфейсов. Если в микроконтроллерной системе используется какой либо один из интерфейсов, вывод SER/PAR может быть подключен к напряжению питания (последовательный режим) или общему проводу (параллельный режим). Если в системе используются оба интерфейса, вывод SER/PAR должен быть подключен к управляющей линии вывода микроконтроллера. Если в системе используются оба интерфейса, то при активации последовательного режима все входы/выходы параллельного интерфейса I/O0- I/O7 находятся в высокоимпедансном режиме и присутствующие на этих выводах потенциалы игнорируются. Соответственно, при активации параллельного режима, выход последовательных данных SO находится в высокоимпедансном режиме, а уровни на входе последовательных данных SI – игнорируются. Смена типа интерфейса может производиться в любое время при соблюдении двух условий: 1. Сигнал выбора кристалла CS/ должен быть пассивным, т.е. равным высокому логическому уровню («единице»). 2. Интервал между переключением сигнала выбора кристалла CS/ и сигнала выбора интерфейса SER/PAR должен быть более 100 нс.

Наличие двух интерфейсов позволяет использовать микросхему для работы с двумя микроконтроллерами, однако при этом не следует забывать анализировать сигнал готовности кристалла перед выполнением операций записи и стирания.

Вывод SER/PAR имеет внутреннюю «подтяжку уровня» к логической единице и, следовательно, может находиться «в воздухе», т.е. подразумевается, что кристалл включен в режим последовательного интерфейса. Однако, производитель рекомендует с целью повышения помехоустойчивости соединять вывод SER/PAR с линией напряжения питания, если это возможно.

Таблица 2

| Номер вывода | Наименование | Назначение |
|---|---|---|
| 1-2 | NC | Не подсоединять |
| 3 | RDY/BUSY | Готов (1) / занят (0) |
| 4 | RST/ | Сброс кристалла |
| 5 | WP/ | Аппаратная защита записи |
| 6-8 | NC | Не подсоединять |
| 9, | VCC | Основное вход напряжения питания |
| 10 | GND | Основной общий вывод |
| 11-14 | NC | Не подсоединять |
| 15 | CS/ | Выбор кристалла |
| 16 | SCK/CLK | Вход тактирования последовательного/параллельного интерфейса |
| 17 | SI | Вход последовательного интерфейса |
| 18 | SO | Выход последовательного интерфейса |
| 19-24 | NC | Не подсоединять |
| 25 | SER/PAR | Выбор последовательного/параллельного интерфейса |
| 26 | I/O0* | Бит 0 шины параллельного интерфейса |
| 27 | I/O1* | Бит 1 шины параллельного интерфейса |
| 28 | I/O2* | Бит 2 шины параллельного интерфейса |
| 29 | I/O3* | Бит 3 шины параллельного интерфейса |
| 30 | GNDP* | Дополнительный общий вывод параллельного интерфейса |
| 31 | VCCP* | Дополнительный вход напряжения питания параллельного интерфейса |
| 32 | I/O4* | Бит 4 шины параллельного интерфейса |
| 33 | I/O5* | Бит 5 шины параллельного интерфейса |
| 34 | I/O6* | Бит 6 шины параллельного интерфейса |
| 35 | I/O7* | Бит 7 шины параллельного интерфейса |
| 36-40 | NC | Не подсоединять |

Вход последовательных данных SI используется только интерфейсом SPI. В случае, если используется только параллельный режим, т.е. когда SER/PAR=0, не рекомендуется подсоединять какие либо сигналы ко входу SI.

Выход последовательных данных SO используется также только в последовательном режиме. В случае, если используется только параллельный режим, т.е. когда SER/PAR=0, также не рекомендуется подсоединять какие либо сигналы ко выходу SO.

Входы-выходы данных I/O0- I/O7 используются в параллельном режиме для передачи в кристалл кодов операции, адреса и данных, а также для чтения данных.

Вход тактирования SCK/CLK используется как счетный вход в режимах последовательного и параллельного интерфейсов. Данные всегда стробируются на запись в микросхему DataFlash по переднему фронту (перепаду с низкого уровня в высокий) импульсов на этом входе, и всегда стробируются на чтение из микросхемы по заднему фронту импульсов.

Вход выбора кристалла CS/ служит для активации обмена с микросхемой. Очевидно, что этот вход может служить для наращивания объема памяти стандартным способом. Если этот сигнал пассивен, данные на линиях входов SI и I/O0- I/O7 не воздействуют на кристалл, а линии выходов SO и I/O0- I/O7 находятся в высокоимпедансном состоянии. Перепад с высокого уровня на низкий активирует начало операции микросхемы DataFlash.

Вход аппаратной защиты записи WP/ при низком уровне запрещает запись в первые 256 страниц основной памяти (0 и 1 сектора). Вход WP/ имеет внутреннюю «подтяжку уровня» к логической единице и, следовательно, может находиться «в воздухе», т.е. подразумевается, что запись разрешена во все страницы. Однако, производитель рекомендует с целью повышения

помехоустойчивости соединять вход WP/ с линией напряжения питания, если это возможно и конечно же если вход не используется схемой.

Вход сброса кристалла RST/ предназначен для прерывания текущей операции и установки внутреннего цифрового автомата в исходное состояние. Микросхема может находиться в режиме сброса сколь угодно долгое время. Следует отметить, что микросхема имеет встроенный узел сброса по включению питания, однако этот узел не воздействует на вход сброса. Вход RST/ имеет внутреннюю «подтяжку уровня». Однако, производитель рекомендует с целью повышения помехоустойчивости соединять вход RST/ с линией напряжения питания, если это возможно и конечно же если вход не используется схемой.

Выход готовности READY/BUSY предназначен для информирования внешней схемы о выполнении или завершении работы внутреннего цифрового автомата в режимах стирания или записи. Выход выполнен с «открытым истоком».

Входы дополнительного питания параллельного интерфейса VCCP и GNDP предназначены для питания приемопередатчиков параллельного интерфейса и должны использоваться только когда, когда используется параллельный режим интерфейса.

Для повышения надежности работы схемы производитель рекомендует все используемые управляющие сигналы: CS/, WP/, SCK/CLK, RST/ шунтировать на землю конденсаторами 100 пФ. На входы питания, непосредственно у выводов микросхемы рекомендуется использование трех соединенных параллельно конденсаторов с номиналами 1 мкФ, 100 нФ и 10 нФ.

**Операции микросхемы AT45DB642**

Микросхема имеет две группы операций: A – группа операций, использующих основную DataFlash память и B - группа операций, не использующих основную DataFlash память.

К группе A относятся восемь основных и две дополнительные операции :

1. Чтение страницы основной памяти в буфер 1;
2. Чтение страницы основной памяти в буфер 2;
3. Запись буфера 1 (2) в основную память со стиранием;
4. Запись буфера 1 (2) в основную память без стирания;
5. Стирание страницы основной памяти;
6. Стирание блока основной памяти;
7. Программирование страницы основной памяти через буфер;
8. Автоперезапись страницы основной памяти;
9. Передача страницы основной памяти в буфер 1 (2);
10. Сравнение страницы основной памяти с буфером 1 (2).

К группе B относятся три операции:

11. Чтение буфера 1 (2);
12. Запись в буфер 1 (2);
13. Чтение регистра статуса.

Как уже упоминалось выше, операциями чтения, записи и стирания основной памяти управляет встроенный цифровой автомат, который работает по своей внутренней программе, т.е. его время работы не зависит от внешних условий. Такой режим называется внутренним «авто-таймированием». Так вот, все операции группы A выполняются с внутренним «авто-таймированием», а операции группы B зависят от внешних сигналов. В связи с этим, в микросхему подана одна из команд группы A, то до ее завершения ни одна из команд группы A в микросхему подана быть не может. В то же время, если микросхема выполняет команду группы A, то она в тоже время может выполнять одну или несколько (последовательно) команд группы B.

Все команды микросхемы AT45DB642 подразделяются на 3 группы: команды чтения, команды записи и стирания, а также дополнительные команды. Каждой команде соответствует определенный код - код операции – «опкода» (opcode). Название всех команд и их коды операций приведены в таблице 3. После кода операции в большинстве команд передается трехбайтный код адреса, состоящий из одиннадцати битного адреса байтов в странице BA10-BA0 (Byte Address) и тринадцати битного адреса страницы PA12-PA0 (Page Address). Адрес байтов расположен в младших битах 24 битного (трехбайтного) кода адреса, а затем начиная с одиннадцатого бита располагается адрес страницы. Далее могут следовать несколько незначащих байтов, необходимых для синхронизации. Если в команде не используется адрес байтов или страниц, то

его биты обычно заполняются нулями. Более подробно с особенностями вычисления и передачи байтов адреса мы поговорим при рассмотрении программного примера.

Следует напомнить, что основной упор в рассмотрении команд мы будем делать на параллельном режиме работы микросхемы.

Таблица 3

| № | Название команды | Оригинальное название | Код операции Rapid8 / SPI |
|---|---|---|---|
| | **Команды чтения** | | |
| 1 | Непрерывное чтение основной памяти | Continuous Array Read | 0x68 / 0xE8 |
| 2 | Непрерывное чтение основной памяти с синхронной задержкой | Burst Array Read with Synchronous Delay | 0x69 / 0xE9 |
| 3 | Чтение страницы основной памяти | Main Memory Page Read | 0x52 / 0xD2 |
| 4 | Чтение буфера оперативной памяти 1 | Buffer 1 Read | 0x54 / 0xD4 |
| 5 | Чтение буфера оперативной памяти 2 | Buffer 2 Read | 0x56 / 0xD6 |
| 6 | Чтение регистра статуса | Status Register Read | 0x57 / 0xD7 |
| | **Команды записи и стирания** | | |
| 7 | Запись в буфер 1 | Buffer 1 Write | 0x84 |
| 8 | Запись в буфер 2 | Buffer 2 Write | 0x87 |
| 9 | Программирование страницы основной памяти из буфера 1 со встроенным стиранием | Buffer 1 to Main Memory Page Program with Built-in Erase | 0x83 |
| 10 | Программирование страницы основной памяти из буфера 2 со встроенным стиранием | Buffer 2 to Main Memory Page Program with Built-in Erase | 0x86 |
| 11 | Программирование страницы основной памяти из буфера 1 без встроенного стирания | Buffer 1 to Main Memory Page Program without Built-in Erase | 0x88 |
| 12 | Программирование страницы основной памяти из буфера 2 без встроенного стирания | Buffer 2 to Main Memory Page Program without Built-in Erase | 0x89 |
| 13 | Стирание страницы основной памяти | Page Erase | 0x81 |
| 14 | Стирание блока основной памяти | Block Erase | 0x50 |
| 15 | Программирование страницы основной памяти через буфер 1 | Main Memory Page Program Through Buffer 1 | 0x82 |
| 16 | Программирование страницы основной памяти через буфер 2 | Main Memory Page Program Through Buffer 2 | 0x85 |
| | **Дополнительные команды** | | |
| 17 | Передача страницы основной памяти в буфер 1 | Main Memory Page to Buffer 1 Transfer | 0x53 |
| 18 | Передача страницы основной памяти в буфер 2 | Main Memory Page to Buffer 2 Transfer | 0x55 |
| 19 | Сравнение страницы основной памяти с буфером 1 | Main Memory Page to Buffer 1 Compare | 0x60 |
| 20 | Сравнение страницы основной памяти с буфером 2 | Main Memory Page to Buffer 2 Compare | 0x61 |
| 21 | Авто-перезапись страницы основной памяти через буфер 1 | Auto Page Rewrite Through Buffer 1 | 0x58 |
| 22 | Авто-перезапись страницы основной памяти через буфер 2 | Auto Page Rewrite Through Buffer 2 | 0x59 |

**Команды чтения**

Команда «Непрерывное чтение основной памяти» (Continuous Array Read) – предназначена для организации непрерывного чтения всего массива основной DataFlash памяти, начиная с заданного адреса. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x68 (0xE8), далее передаются три байта начального адреса, а за ними следует 4 незначащих байта для последовательного интерфейса или 60 незначащих байтов для параллельного интерфейса. Каждый из байтов сопровождается стробирующим импульсом на входе SLK/CLK. После этого следует подавать стробирующие импульсы, и под каждый из стробирующих импульсов на шине данных будет появляться байт

данных, после чего встроенный цифровой автомат будет автоматически наращивать адрес байта в странице. При достижении последнего адреса страницы цифровой автомат автоматически будет продолжать чтение с первого байта следующей страницы. При достижении последнего адреса последней страницы чтение будет продолжаться с первого адреса первой страницы. Чтение завершится только после дезактивации сигнала выборки кристалла (CS/=1). Во время выполнения этой команды содержимое обоих страниц буферной памяти не нарушается.

Команда «Непрерывное чтение основной памяти с синхронной задержкой» (Burst Array Read with Synchronous Delay) – также предназначена для организации непрерывного чтения всего массива основной DataFlash памяти, начиная с заданного адреса. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x69 (0xE9), далее передаются три байта начального адреса, а за ними следует 4 незначащих байта для последовательного интерфейса или 60 незначащих байтов для параллельного интерфейса. Отличие этой команды от предыдущей заключается лишь в том, что после завершения чтения каждой страницы микроконтроллерная система должна подавать 32 незначащих тактовых цикла перед началом чтения следующей страницы. Чтение завершится только после дезактивации сигнала выборки кристалла (CS/=1). Во время выполнения этой команды содержимое обоих страниц буферной памяти не нарушается.

Команда «Чтение страницы основной памяти» (Main Memory Page Read) – предназначена для непосредственного чтения данных из любой из 8192 страниц основной памяти не нарушая содержимого буферов оперативной памяти. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x52 (0xD2), далее передаются три байта начального адреса, а за ними следует 4 незначащих байта для последовательного интерфейса или 60 незначащих байтов для параллельного интерфейса. После этого под каждый из 1056 тактовых импульсов микросхема будет выдавать содержимое заданной страницы памяти. По истечении 1056 циклов следует дезактивировать сигнала выборки кристалла (CS/=1). Во время выполнения этой команды содержимое обоих страниц буферной памяти не нарушается.

Команды «Чтение буфера оперативной памяти 1(2)» (Buffer 1(2) Read) – предназначена для чтения одной из двух страниц буферной оперативной памяти. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x56 для первого буфера или 0xD6 для второго буфера, далее передаются три байта начального адреса, а за ними следует 1 незначащий байта. В передаваемом трехбайтном коде адреса значащими являются только младшие 11 бит адреса байта в странице, которые определяют адрес первого читаемого байта. Остальные биты кода адреса игнорируются. При достижении чтения последнего байта буфера чтение будет продолжаться с первого байта этого же буфера. Чтение завершится только после дезактивации сигнала выборки кристалла (CS/=1).

Команда «Чтение регистра статуса» (Status Register Read) – предназначена для определения состояния выполнения операций типа «A», т.е. операций с основной DataFlash памятью (смотри выше).

Статусный регистр имеет только два значащих бита. Бит 7 – RDY/BUSY – дублирует состояние одноименного вывода. Высокий логический уровень этого бита соответствует состоянию готовности – не выполняется операций типа «A». Низкий логический уровень соответствует состоянию авто-таймирования. В этом случае возможно использование только операций типа «B». Бит 6 – COMP (Compare) – флаг сравнения – используется в командах сравнения и будет описан немного ниже. Биты 5-2 всегда для микросхемы AT45DB642 читаются как «1». Их можно использовать для идентификации типа микросхемы, т.к. у других DataFlash микросхем фирмы Atmel они имеют другое значение. Биты 1-0 не детерминированы и читаются произвольным образом.

Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x57 (0xD7), далее под каждый тактовый импульс на шине данных появляется обновленное состояние статусного регистра до тех пор, пока не будет дезактивации сигнала выборки кристалла (CS/=1).

**Команды записи и стирания**

Команды «Запись в буфер 1(2)» (Buffer 1(2) Write) – предназначена для записи данных в один из буферов оперативной памяти. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача в микросхему опкода 0x84 для первого буфера или 0x87 для второго буфера, далее передаются три байта начального адреса. В

передаваемом трехбайтном коде адреса значащими являются только младшие 11 бит адреса байта в странице, которые определяют адрес первого записываемого байта. Остальные биты кода адреса игнорируются.

Далее микроконтроллерная система последовательно выставляет на шину адреса записываемые байты, стробируемые тактовыми импульсами. При достижении записи последнего байта буфера запись будет продолжаться с первого байта этого же буфера. Запись завершится только после дезактивации сигнала выборки кристалла (CS/=1).

Команды «Программирование страницы основной памяти из буфера 1(2) со встроенным стиранием» (Buffer 1(2) to Main Memory Page Program with Built-in Erase) – предназначена для записи предварительно сформированного содержимого одного из буферов в страницу основной памяти. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x83 для буфера 1 или 0x86 для буфера 2, далее передаются три байта кода адреса. В коде адреса 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, в которую будет произведена запись. Младшие 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция стирания выбранной страницы памяти и перезапись в нее содержимого адресуемой страницы. Стирание и запись данных происходят в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

Команды «Программирование страницы основной памяти из буфера 1(2) без встроенного стирания» (Buffer 1(2) to Main Memory Page Program without Built-in Erase) – аналогична предыдущей, однако стирания выбранной страницы не производится. Это несколько сокращает время записи, однако накладывает на микроконтроллерную систему заботу о предварительном стирании необходимого массива основной памяти. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x88 для буфера 1 или 0x89 для буфера 2, далее передаются три байта кода адреса. Остальные процессы происходят аналогично.

Команда «Стирание страницы основной памяти» (Page Erase) – предназначена для стирания выбранной страницы. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x81, далее передаются три байта кода адреса. В коде адреса 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, в которую будет произведена запись. Младшие 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция стирания выбранной страницы памяти. Стирание данных происходят в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

Команда «Стирание блока основной памяти» (Block Erase) – предназначена для стирания выбранного блока основной памяти, состоящего из 8 страниц. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x50, далее передаются три байта кода адреса. В коде адреса только биты PA12-PA3 анализируются, а три младших бита адреса страницы (PA2-PA0) и 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция стирания выбранного блока основной памяти. Стирание данных происходят в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

Команды «Программирование страницы основной памяти через буфер 1(2)» (Main Memory Page Program Through Buffer 1(2)) – эта команда является комбинацией команд «Запись в буфер 1(2)» (Buffer 1(2) Write) и «Программирование страницы основной памяти из буфера 1(2) со встроенным стиранием» (Buffer 1(2) to Main Memory Page Program with Built-in Erase). Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача в микросхему опкода 0x82 для первого буфера или 0x85 для второго буфера, далее передаются три байта начального адреса. В передаваемом трехбайтном коде адреса младшие 11 бит адреса байта в странице определяют адрес первого записываемого в страницу байта, а 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, в которую будет произведена запись. Далее микроконтроллерная система последовательно выставляет на шину адреса записываемые байты, стробируемые тактовыми импульсами. При достижении записи последнего байта буфера запись будет продолжаться с первого байта этого же буфера. Запись

байтов в буфер завершится только после дезактивации сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция стирания выбранной страницы памяти и перезапись в нее содержимого адресуемого буфера. Стирание и запись данных происходят в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

**Дополнительные команды**

Команды «Передача страницы основной памяти в буфер 1(2)» (Main Memory Page to Buffer 1(2) Transfer). Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x53 для буфера 1 или 0x55 для буфера 2, далее передаются три байта кода адреса. В коде адреса 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, из которой будет произведено чтение. Младшие 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция чтения данных из выбранной страницы основной памяти в выбранный буфер. Команда выполняется в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

Команды «Сравнение страницы основной памяти с буфером 1(2)» (Main Memory Page to Buffer 1(2) Compare) – выполняют сравнение выбранной страницы основной памяти с выбранным буфером оперативной памяти. Команда может быть использована для текущего контроля. Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x60 для буфера 1 или 0x61 для буфера 2, далее передаются три байта кода адреса. В коде адреса 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, из которой будет произведено чтение. Младшие 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция сравнения данных из выбранной страницы основной памяти с выбранным буфером оперативной памяти. Команда выполняется в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень. Результат операции сравнения индицируется в бите 6 статусного регистра COMP (Compare). Бит устанавливается в «1» при равенстве сравниваемых данных и в «0» при неравенстве.

Команды «Авто-перезапись страницы основной памяти через буфер 1(2)» (Auto Page Rewrite Through Buffer 1(2)) – команда носит ограниченное применение и может использоваться если несколько байт в странице или буфере модифицируются случайным образом. Команда является комбинацией двух команд: «Передача страницы основной памяти в буфер 1(2)» (Main Memory Page to Buffer 1(2) Transfer) и «Программирование страницы основной памяти из буфера 1(2) со встроенным стиранием» (Buffer 1(2) to Main Memory Page Program with Built-in Erase). Выполнение команды начинается с активации сигнала выборки кристалла CS/=0. После этого следует передача опкода 0x58 для буфера 1 или 0x59 для буфера 2, далее передаются три байта кода адреса. В коде адреса 13 бит адреса страницы определяют номер одной из 8192 страниц основной памяти, в которую будет произведена запись. Младшие 11 бит адреса байта игнорируются. Далее следует дезактивировать сигнала выборки кристалла (CS/=1). После этого, автоматически запускается операция чтения страницы основной памяти в буфер, стирания страницы основной памяти и перезапись в нее содержимого буфера. Стирание и запись данных происходят в режиме авто-таймирования и в течение всего этого времени на выходе RDY/BUSY и одноименном бите статусного регистра присутствует низкий логический уровень.

Полное представление о формате команд дает таблица 4.

В таблице использованы следующие сокращения:

В колонке «Интерфейс»: символ «S» – последовательный интерфейс; символ «P» – параллельный интерфейс; символ «A» – оба интерфейса. В колонках «Старший, средний и младший байт адреса»: символ «P» – означает значащие биты адреса страницы; символ «B» – означает значащие биты адреса байта в странице; символ «x» - означает незначащие биты.

В следующей части статьи будет приведена принципиальная схема подключения микросхемы DataFlash памяти AT45DB642 к микроконтроллеру C8051F021 через параллельный интерфейс Rapid8, а также приведена и описана библиотека функций для работы с DataFlash, написанная на языке «C» (Keil).

Таблица 4

| КОМАНДА | Интерфейс | Опкод | Старший байт кода адреса | | | | | | | | Средний байт кода адреса | | | | | | | | Младший байт кода адреса | | | | | | | | Кол. незначащих байтов, P/S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | PA12 | PA11 | PA10 | PA09 | PA08 | PA07 | PA06 | PA05 | PA04 | PA03 | PA02 | PA01 | PA00 | PB10 | PB09 | PB08 | PB07 | PB06 | PB05 | PB04 | PB03 | PB02 | PB01 | PB00 | |
| Непрерывное чтение | P<br>S | 0x68<br>0xE8 | P | P | P | P | P | P | P | P | P | P | P | P | P | B | B | B | B | B | B | B | B | B | B | B | 60<br>4 |
| Непрерывное чтение с задержкой | P<br>S | 0x69<br>0xE9 | P | P | P | P | P | P | P | P | P | P | P | P | P | B | B | B | B | B | B | B | B | B | B | B | 60<br>4 |
| Чтение страницы | P<br>S | 0x52<br>0xD2 | P | P | P | P | P | P | P | P | P | P | P | P | P | B | B | B | B | B | B | B | B | B | B | B | 60<br>4 |
| Чтение буфера 1 | P<br>S | 0x54<br>0xD4 | x | x | x | x | x | x | x | X | x | x | x | x | x | B | B | B | B | B | B | B | B | B | B | B | 1<br>1 |
| Чтение буфера 2 | P<br>S | 0x56<br>0xD6 | x | x | x | x | x | x | x | X | x | x | x | x | x | B | B | B | B | B | B | B | B | B | B | B | 1<br>1 |
| Запись в буфер 1/2 | A | 0x84<br>0x87 | x | x | x | x | x | x | x | X | x | x | x | x | x | B | B | B | B | B | B | B | B | B | B | B | Нет |
| Прогр. страницы из буфера 1/2 со стиранием | A | 0x83<br>0x86 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Прогр. страницы из буфера 1/2 без стирания | A | 0x88<br>0x89 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Стирание блока | A | 0x50 | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Стирание страницы | A | 0x81 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Прогр. страницы через буфер 1/2 | A | 0x82<br>0x85 | P | P | P | P | P | P | P | P | P | P | P | P | P | B | B | B | B | B | B | B | B | B | B | B | Нет |
| Передача страницы в буфер 1/2 | A | 0x53<br>0x55 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Сравнение страницы с буфером 1/2 | A | 0x60<br>0x61 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Авто-перезапись страницы 1/2 | A | 0x58<br>0x59 | P | P | P | P | P | P | P | P | P | P | P | P | P | x | x | x | x | x | x | x | x | x | x | x | Нет |
| Чтение статуса | P<br>S | 0x57<br>0xD7 | Не передается | | | | | | | | | | | | | | | | | | | | | | | | Нет |

**Принципиальная схема подсистемы памяти на микросхеме AT45DB642**

В первой части статьи мы рассмотрели общую структуру и особенности микросхемы AT45DB642[2], привели назначение выводов, описали архитектуру памяти и основные операции. Напомним, что микросхема оснащена двумя интерфейсами: последовательным (SPI) и специализированным параллельным (Rapid8). Микросхема AT45DB642 ориентирована на использование в специализированных микроконтроллерных системах с высоким быстродействием, низким энергопотреблением и напряжением питания. Как раз такими качествами обладают современные сверхбыстродействующие микроконтроллеры фирмы SiLabs[3] (новое название микроконтроллеров фирмы Cygnal). В качестве примера рассмотрим подключение микросхемы DataFlash памяти AT45DB642 к микроконтроллеру C8051F021[4], который наиболее совместим со всеми полноформатными семействами. Типовая принципиальная схема подключения микросхемы AT45DB642 через интерфейс Rapid8 приведена на рис.1.

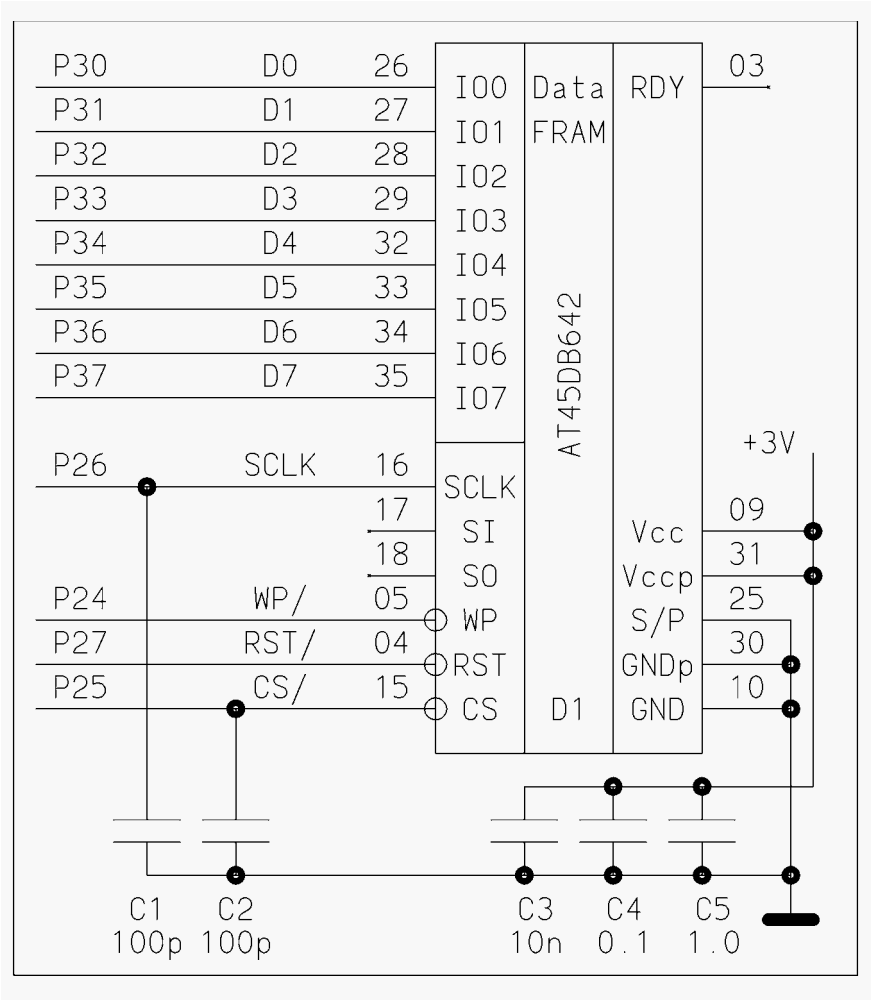


Рис.1. Типовая принципиальная схема подключения AT45DB642 через интерфейс Rapid8

Линии данных D0-D7, которые на изображении микросхемы обозначены как линии ввода/вывода IO0-IO7 (Input/Output), соединены с одним из портов ввода/вывода микроконтроллера, в нашем случае используется третий порт (P30-P37). Изготовитель микросхемы AT45DB642 рекомендует с целью снижения вероятности ошибок записи при переходных процессах во время перехода этих линий в высокоимпедансное состояние использовать на них резисторы «подтяжки» к напряжению питания. На принципиальной схеме эти резисторы не показаны, т.к. используемый микроконтроллер имеет встроенные программируемые элементы «подтяжки». Вывод 25 выбора интерфейса S/P микросхемы AT45DB642 соединен с общим проводом, т.е. микросхема в таком включении работает только через параллельный интерфейс. Основной GND и вспомогательный GNDp выводы микросхемы соединены с общим проводом питания. Основной Vcc и вспомогательный Vccp выводы питания микросхемы соединены с источником напряжения питания 2,7 – 3,3 В. Напомним, что вспомогательные

выводы питания Vccp и GNDp используются только в режиме параллельного интерфейса для питания выходных буферных схем ввода/вывода. Непосредственно у выводов питания должны быть расположены три, соединенных параллельно конденсатора развязки (C3-C5) с номиналами 10 нФ, 100 нФ и 1 мкФ соответственно. Включение этих конденсаторов рекомендуется фирмой Atmel для снижения вероятности ошибок записи, поскольку в время операций записи и стирания потребляемый ток микросхемы AT45DB642 может импульсно увеличиваться с 4 до 35 мА. Основные линии сигналов управления: линия тактирования SCLK и линия выборки микросхемы CS/ соединены с линиями вывода другого свободного порта микроконтроллера, в нашем случае с линиями P26 и P25 соответственно. Поскольку эти линии работают в динамическом режиме, производитель рекомендует их шунтировать конденсаторами C1 и C2 с емкостью 100 пФ. Вспомогательные линии управления: линия защиты записи WP/ и линия сброса цифрового автомата записи RST/ также соединены со свободными линиями вывода микроконтроллера, в нашем случае с линиями P24 и P27 соответственно. Эти линии допускается не шунтировать конденсаторами. Во многих приложениях они могут вообще не использоваться, и в этом случае их необходимо соединять с напряжением питания. Вывод готовности RDY микросхемы в нашем случае не используется, поскольку он дублируется соответствующим битом готовности программно доступного статусного регистра. Этот вывод обычно используется в тех случаях, когда микросхема AT45DB642 подключается не к микроконтроллеру, а к устройству управления, выполненному на «жесткой» логике. Неиспользуемые линии последовательного интерфейса SI и SO рекомендуется оставлять свободными, т.к. они имеют встроенные подтягивающие резисторы, и в режиме параллельного интерфейса их состояние игнорируется встроенным цифровым автоматом микросхемы. Следует также отметить, что производитель рекомендует при разводке печатной платы располагать микросхему AT45DB642 непосредственно возле управляющего устройства (микроконтроллера) и обеспечить длину линий связи не более 40 мм.

### Программный интерфейс микросхемы AT45DB642

Прежде всего обратим внимание читателей на то, что описываемый ниже программный интерфейс написан на языке C для компилятора фирмы Keil[5]. Особенностью этого компилятора является требование к объявлению SFR переменных в каждом их программных модулей. Обычно это делается в общем включаемом файле main.h, который включается во все программные модули проекта (программы). Ниже приведен только фрагмент этого файла, касающийся используемых в нашем случае переменных:

```
//*****************************//
// 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 ...

//Объявление порта ввода/вывода данных
#define          DATABUS       P3
#define          DATABUS_CF  P3MDOUT

//Объявления линий ввода/вывода  UART0
sbit             TX0           = P0^0;
sbit             RX0           = P0^1;
//Объявления линий ввода/вывода  DataFlash
sbit             FL_CLK        = P2^6;
sbit             FL_CS         = P2^5;
sbit             FL_RST        = P2^7;
sbit             FL_WP         = P2^4;

//... Другие объявления ...

#endif // Main.h
```

Теперь перейдем к рассмотрению собственно функций программного интерфейса. Все функции программного интерфейса в рассматриваемом примере включены в один файл DataFlash.c. Объявления функций вынесены в отдельный включаемый файл DataFlash.h, который будет приведен ниже. Поскольку обычно в основной программе используются не все возможные функции, которые поддерживает микросхема AT45DB642, в модуле используется механизм условного компилирования. Т.е., все основные используемые функции компилируются безусловно, а функции используемые редко компилируются в зависимости от значения флага условного компилирования FL_EXTEND. В этом случае на программиста возлагается принятие решения о том, какие функции должны компилироваться всегда, т.е. безусловно, а какие – условно, в зависимости от состояния флага FL_EXTEND. Если программист правильно перераспределил функции, компилятор никаких сообщений не выдаст. При ошибочных действиях программиста компилятор может либо выдать сообщение об ошибке - вызове отсутствующей функции, либо выдать предупреждение о наличии в проекте неиспользуемых функций, которые автоматически игнорируются.

Естественно, что существует более гибкий механизм, заключающийся в создании библиотеки функций. В этом случае приведенный модуль должен быть разбит на ряд модулей, каждый из которых содержит отдельную функцию или группу функций, все модули должны компилироваться отдельно, а затем полученные объектные модули должны быть включены в библиотеку. Создание библиотеки производится по известной методике, приведенной, например, в[5]. Достоинством этого механизма является то, что грамотном создании библиотеки программист освобождается от необходимости определения основных и условно-компилируемых функций, а компилятор автоматически будет извлекать и включать в основную программу только те библиотечные функции, которые необходимы.

Отметим также, что в приводимом программном модуле используется внешняя функция перезапуска таймера WDT, которая объявлена в основном программном модуле:

```
void WDT (void)
{WDTCN=0xA5;}
```

Теперь рассмотрим собственно программный модуль интерфейса.

```
//*****************************//
// DataFlash.C
//*****************************//
#include <Main.h>

// Определение флага условного компилирования
#define                     FL_EXTEND   0

// Массив текущего адреса операции во встроенной
// дополнительной оперативной памяти
xdata byte        ADDR[64];

//Массив данных во встроенной
// дополнительной оперативной памяти
xdata byte        DBuff[1065];

//Байт состояния DataFlash памяти
xdata byte        FL_ERROR;
//************************************************
// Макроопределение включения защиты записи
#define FL_Write_Protect_ON()          FL_WP=0;
// Макроопределение выключения защиты записи
#define FL_Write_Protect_OFF()         FL_WP=1;
// Макроопределение активации выборки кристалла
#define FL_CS_0()                      FL_CS=0;WDT();
// Макроопределение дезактивации выборки кристалла
#define FL_CS_1()                      WDT();FL_CS=1;
//************************************************

// Подпрограмма генерации тактового импульса с
// длительностью примерно 100 нс и паузой после
// выключения примерно 50 нс при тактовой частоте
// 22,1184 МГц

void FL_Pulse (void)
{
        FL_CLK=1; WDT();
        FL_CLK=0; WDT();
}
//************************************************

// Подпрограмма записи байта в микросхему AT45DB642

void FL_SetB (byte DB)
{
        DATABUS_CF=0;      // Порт 3 с открытым стоком
        DATABUS=DB;        // Установить данные
        FL_Pulse();        // Выдать тактовый импульс
        DATABUS=0xFF;      // Порт 3 в режим ввода
}
//************************************************

// Подпрограмма чтения байта из микросхемы AT45DB642

byte FL_GetB (void)
{
```

```
        DATABUS_CF=0;       // Порт 3 с открытым стоком
        DATABUS=0xFF;       // Порт 3 в режим ввода
        FL_Pulse();         // Выдать тактовый импульс
        return DATABUS;     // Вернуть данные порта 3
}
//***********************************************

// Подпрограмма ожидания готовности микросхемы

void FL_WaitReady (void)
{
byte DB;

        FL_CS_0();          // Активировать CS
        FL_SetB (0x57);     // Опкод операции
        while (1)
        {
                DB=FL_GetB (); // Читать статус
                if (DB&0x80)  // Если Готов
                    break;    // Прервать цикл
                WDT();                  // Перезапустить WDT
        }
        FL_CS_1();          // Деактивировать CS
}
//***********************************************

// Подпрограмма формирования трех байт адреса из
// номера страницы и номера байта в странице

void FL_AddressCalc (int PageAddr, int ByteAddr)
{
        // Если номер страницы > 8192 ошибка 2
        if (PageAddr>8192) FL_ERROR|=0x02;
        // Если номер байта > 1056 ошибка 1
        if (ByteAddr>1056) FL_ERROR|=0x01;
        // Очистить массив адреса
        memset (ADDR,0,sizeof(ADDR));
        // Вычислить старший байт текущего адреса FAD2
        ADDR[0]     =(byte)((PageAddr>>5)&0x0FF);
        // Вычислить средний байт текущего адреса FAD1
        ADDR[1]     =(byte)((ByteAddr>>8)&0x07|(PageAddr&0x1F)<<3);
        // Вычислить младший байт текущего адреса FAD0
        ADDR[2]     =(byte)(ByteAddr&0x0FF);
        FL_CS_0();  // Активировать CS
}

//***********************************************

// Подпрограмма записи трех байтов адреса и DCB
// незначащих байтов в микросхему AT45DB642
// Первые три байта содержат адрес, а остальные –
// незначащие

void FL_AddressWrite (int DCB)
{
        for( int i=0; i<DCB+3; i++) FL_SetB(ADDR[i]);
}
```

```
//*********************************************

// Подпрограмма записи Len байтов из внешнего буфера
// данных Buff в страницу BuffNum встроенной сверх-
// оперативной памяти начиная с адреса ByteAddr

void FL_BufferWrite (int BuffNum, int ByteAddr, char *Buff, int Len)
{
        FL_AddressCalc (0,ByteAddr);          // Вычислить адрес
        FL_SetB ((BuffNum)?0x87:0x84);  // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        for(int i=0; i<Len; i++) // Записать Len
        {FL_SetB (Buff[i]);}    // Байтов данных
        FL_CS_1();              // Деактивировать CS
}

//*********************************************

// Подпрограмма чтения страницы Flash памяти с номером
// PageAddr в оперативную память DBuff

void FL_MainPageRead (int PageAddr)
{
        FL_AddressCalc (PageAddr,0);  // Вычислить адрес
        FL_SetB (0x52);                 // Записать Опкод
        FL_AddressWrite (60); // Записать 3 байта адреса
        // и 60 незначащих байтов синхронизации чтения
        for (int i=0; i<1056; i++)          // Считать 1056
        {DBuff[i]=FL_GetB();}                   // Байтов данных
        FL_CS_1();              // Деактивировать CS
}
//*********************************************

// Подпрограмма записи данных из встроенного буфера с
// номером BuffNum в страницу Flash памяти с номером
// PageAddr с автоматическим стиранием страницы

void FL_BufferToMainPageWE (int BuffNum, int PageAddr)
{
        FL_AddressCalc (PageAddr,0);  // Вычислить адрес
        FL_SetB ((BuffNum)?0x86:0x83); // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();              // Деактивировать CS
}

//*********************************************
//   ДОПОЛНИТЕЛЬНЫЕ КОМАНДЫ
// УСЛОВНОГО КОМПИЛИРОВАНИЯ
//*********************************************
#if FL_EXTEND                   // Проверка флага условного
                                // компилирования

//*********************************************

// Подпрограмма чтения статусного регистра микросхемы
```

```
// Подпрограмма возвращает код состояния,
// а не статусный байт

int  FL_StatusRead (void)
{
byte DB;

        FL_CS_0();              // Активировать CS
        FL_SetB (0x57);         // Записать Опкод
        DB=FL_GetB ();          // Прочитать регистр
        FL_CS_1();              // Деактивировать CS
        switch (DB)             // Расшифровать статус
        {
                case 0x80:  return 1;  // Готов и равен
                case 0xC0: return 2;  // Готов и неравен
                default:      return 0;  // Не готов
        }
}

//***********************************************

// Подпрограмма записи байта CH в встроенный буфер
// номер BuffNum по адресу ByteAddr

void FL_BufferByteWrite (int BuffNum, int ByteAddr, byte CH)
{
        FL_AddressCalc (0,ByteAddr);            // Вычислить адрес
        FL_SetB ((BuffNum)?0x87:0x84); // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_SetB (CH);                   // Записать байт данных
        FL_CS_1();              // Деактивировать CS
}

//***********************************************

// Подпрограмма стирания страницы Flash памяти

void FL_PageErase (int PageAddr)
{
        FL_AddressCalc (PageAddr,0); // Вычислить адрес
        FL_SetB (0x81);                 // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();              // Деактивировать CS
}
//***********************************************

// Подпрограмма стирания блока Flash памяти

void FL_BlockErase (int BlockAddr)
{
        // Вычислить адрес
        FL_AddressCalc ((BlockAddr<<3)&0x1FF8,0);
        FL_SetB (0x50);         // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();              // Деактивировать CS
}
//***********************************************
```

```
// Подпрограмма передачи содержимого страницы PageAddr
// Flash памяти во встроенный буфер оперативной памяти
// номер BuffNum

void FL_MainPageToBufferTransfer (int BuffNum, int PageAddr)
{
        FL_AddressCalc (PageAddr,0); // Вычислить адрес
        FL_SetB ((BuffNum)?0x55:0x53); // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();             // Деактивировать CS
}
//***********************************************

// Подпрограмма сравнения содержимого встроенного
// буфера оперативной памяти номер BuffNum со страницей
// Flash памяти номер PageAddr. Результат сравнения влияет
// на регистр статуса и доступен с помощью функции
// FL_StatusRead

void FL_MainPageToBufferCompare (int BuffNum, int PageAddr)
{
        FL_AddressCalc (PageAddr,0);          // Вычислить адрес
        FL_SetB ((BuffNum)?0x61:0x60);// Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();             // Деактивировать CS
}
//***********************************************

// Подпрограмма записи страницы Flash памяти номер
// PageAddr через встроенный буфер оперативной памяти
// номер BuffNum, в который записываются данные длиной
// Len байтов  из буфера внешней оперативной памяти Buff
// начиная с адреса ByteAddr

void FL_MainPageProgThBuffer (int BuffNum, int PageAddr,
                              int ByteAddr, char *Buff, int Len)
{
        FL_AddressCalc (PageAddr,ByteAddr); // Вычислить адрес
        FL_SetB ((BuffNum)?0x85:0x82);      // Записать Опкод
        FL_AddressWrite (0);            // Записать 3 байта адреса
        for (int i=0; i<Len; i++)       // Записать Len байтов
        {FL_SetB (Buff[i]);}            // данных из буфера Buff
        FL_CS_1();                      // Деактивировать CS
}
//***********************************************

// Подпрограмма начала режима непрерывного чтения
// данных их Flash памяти начиная с байта номер ByteAddr
// страницы номер PageAddr.
// Подразумевается, что за этой командой следует серия команд
// FL_GetB – чтения байтов, которые будут продолжать читать
// Flash память байт за байтом и страница за страницей до
// деактивации CS

void FL_ContinuousArrayReadBegin (int PageAddr, int ByteAddr)
{
```

```
        FL_AddressCalc (PageAddr,ByteAddr); // Вычислить адрес
        FL_SetB (0x68);         // Записать Опкод буфера 2
        FL_AddressWrite (60); // Записать 3 байта адреса и 60
                                // незначащих байтов синхронизации
}
//***********************************************

// Подпрограмма начала режима непрерывного чтения
// данных их Flash памяти начиная с байта номер ByteAddr
// страницы номер PageAddr.
// Подразумевается, что за этой командой следует серия команд
// FL_GetB – чтения байтов, которые будут продолжать читать
// Flash память байт за байтом и страница за страницей до
// деактивации CS, причем после окончания каждой страницы
// будут читаться 32 незначащих байта

void FL_ContinuousArrayReadBeginWDelay (int PageAddr, int ByteAddr)
{
        FL_AddressCalc (PageAddr,ByteAddr); // Вычислить адрес
        FL_SetB (0x69);         // Записать Опкод буфера 2
        FL_AddressWrite (60); // Записать 3 байта адреса и 60
                                // незначащих байтов синхронизации
}
//***********************************************

// Подпрограмма идентификации микросхемы AT45DB642
// Если установлена именно эта микросхема, возвращается
// значение, равное 1

int FL_642 (void)
{
byte DB;

        FL_CS_0();              // Активировать CS
        FL_SetB (0x57);         // Записать Опкод
        DB=FL_GetB ();          // Читать регистр статуса
        FL_CS_1();              // Деактивировать CS
        return (DB&0x3C==0x3C)?1:0; // Вернуть значение
}
//***********************************************

// Подпрограмма сброса встроенного цифрового автомата

void FL_Reset (void)
{
        FL_CLK=0;               // Сбросить тактовый уровень
        FL_RST=0;               // Активировать сброс
        WDT();                  // Задержка 80 нс
        FL_RST=1;               // Деактивировать сброс
}
//***********************************************

// Подпрограмма записи данных из встроенного буфера с
// номером BuffNum в страницу Flash памяти с номером
// PageAddr без автоматического стирания страницы
```

```
void FL_BufferToMainPageWOE (int BuffNum, int PageAddr)
{
        FL_AddressCalc (PageAddr,0); // Активировать CS
        FL_SetB ((BuffNum)?0x89:0x88); // Записать Опкод
        FL_AddressWrite (0);   // Записать 3 байта адреса
        FL_CS_1();             // Деактивировать CS
}
//************************************************

// Подпрограмма чтения данных из встроенного буфера с
// номером BuffNum

void FL_BufferRead (int BuffNum)
{
        FL_AddressCalc (0,0);          // Активировать CS
        FL_SetB ((BuffNum)?0x56:0x54); // Записать Опкод
        FL_AddressWrite (1);   // Записать 3 байта адреса и
                               // один незначащий байт
        for (int i=0; i<1056; i++) // Читать 1056 байт данных
        {DBuff[i]=FL_GetB ();} // во внешний буфер DBuff
        FL_CS_1();             // Деактивировать CS
}

//************************************************
#endif
```

Следует обратить внимание читателей на то, что в состав безусловно компилируемых программ могут и не входить подпрограммы определения готовности FL_StatusRead или FL_WaitReady. Это возможно в том случае, если при записи процесс заполнения одного из встроенных буферных регистров идет с фиксированной определенной скоростью и суммарное время записи 1056 байт по заведомо превышает максимальное время записи данных из второго заполненного буфера (примерно 18 мс). Иными словами, если период поступления байтов в заполняемый буфер не менее 18 мкс.

Кроме собственно программного модуля, приведенного выше, приведем еще включаемый файл, содержащий объявления всех функций.

```
//*****************************//
// DataFlash.H
//*****************************//

// Подпрограммы группы A
void FL_ContinuousArrayReadBegin (int PageAddr, int ByteAddr);
void FL_ContinuousArrayReadBeginWDelay (int PageAddr, int ByteAddr);
void FL_MainPageRead (int PageAddr);
void FL_BufferToMainPageWE (int BuffNum, int PageAddr);
void FL_BufferToMainPageWOE (int BuffNum, int PageAddr);
void FL_PageErase (int PageAddr);
void FL_BlockErase (int BlockAddr);
void FL_MainPageProgThBuffer (int BuffNum, int PageAddr, int ByteAddr, char *Buff, int Len);
void FL_MainPageToBufferTransfer (int BuffNum, int PageAddr);
void FL_MainPageToBufferCompare     (int BuffNum, int PageAddr);

// Подпрограммы группы B
void FL_BufferRead (int BuffNum);
void FL_BufferWrite (int BuffNum, int ByteAddr, char *Buff, int Len);
int  FL_StatusRead (void);
```

```
// Общие функции
void FL_Reset (void);
void FL_Pulse (void);
void FL_SetB  (byte DB);
byte FL_GetB  (void);
void FL_WaitReady (void);
int  FL_642 (void);
void FL_AddressCalc (int PageAddr, int ByteAddr);
void FL_AddressWrite (int DCB);
void FL_BufferByteWrite (int BuffNum, int ByteAddr, byte CH);
```

Приведенный в данной статье набор подпрограмм позволяет реализовать все возможности, заложенные в микросхему AT45DB642 производителем. Естественно, что возможны и другие варианты написания приведенных подпрограмм. Даже описанный вариант подпрограмм может быть значительно модифицирован. Например, возможно объединение трех наиболее часто используемых функций FL_AddressCalc, FL_SetB и FL_AddressWrite в одну функцию, что позволит несколько увеличить скорость выполнения программы, но несколько усложнит читаемость программы. Можно значительно уменьшить длину массива ADDR до трех байт и модифицировать функцию FL_AddressWrite. Возможно объединить функции FL_StatusRead, FL_WaitReady и FL_642 в одну более сложную функцию. Однако эти и многие другие возможные модификации читатель легко может проделать и сам, базируясь на полученных в ходе чтения настоящей статьи знаниях.

**Литература:**


1. http://www.atmel.com/products/
2. http://www.atmel.com/dyn/resources/prod_documents/DOC1638.PDF
3. http://www.silabs.com
4. https://www.mysilabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Precision_Mixed-Signal/en/C8051F02x.pdf
5. http://www.keil.com