Программная эмуляция EEPROM в микроконтроллерах семейства STM32F40x/STM32F41x

В статье описан способ программной эмуляции EEPROM в микроконтроллерах ARM Cortex-M4 производства STMicroelectronics. Рассматриваются преимущества и недостатки такого способа по сравнению с другими способами энергонезависимого хранения данных. Статья основана на документе Application note AN3969 "EEPROM emulation in STM32F40x/STM32F41x microcontrollers" [3]. Описанный способ программной эмуляции EEPROM может быть использован и для других микроконтроллеров, в которых отсутствует встроенная EEPROM-память.

Андрей КУРНИЦ
kurnits@stim.by

Введение

Ко многим электронным устройствам предъявляется требование сохранять свое состояние или какие-либо данные при выключенном питании. Для хранения таких данных служит класс энергонезависимой памяти. Этот класс очень обширен, тем не менее когда речь идет о встраиваемых системах и небольших объемах памяти, то одной из наиболее применяемых технологий является EEPROM-память.

Из преимуществ EEPROM следует отметить произвольный доступ и большой ресурс циклов запись/чтение — до миллиона раз. Память EEPROM может быть выполнена как в виде отдельной микросхемы (например, AT24C256 [1]), так и встроена в микроконтроллер (например, семейства микроконтроллеров AVR, PIC).

К сожалению, микроконтроллеры ARM Cortex-M4 компании STMicroelectronics (к которым относится семейство STM32F40x/STM32F41x) не содержат встроенной EEPROM. И если разработчик желает использовать EEPROM-память, то она должна быть представлена в виде внешней микросхемы, подключенной к микроконтроллеру по одному из последовательных или параллельных интерфейсов (I^2C, SPI и др.)

Отдельная микросхема EEPROM увеличивает себестоимость устройства, поэтому для решений, где важна минимальная стоимость, внешнюю EEPROM можно заменить благодаря одной из следующих возможностей семейства STM32F40x/STM32F41x:

- Расположенная на кристалле 4-кбайтная резервная память SRAM с отдельным батарейным питанием.
- Расположенная на кристалле flash-память плюс дополнительное программное обеспечение для эмуляции EEPROM.

В микроконтроллерах STM32F40x/STM32F41x можно задействовать 4 кбайт резервной памяти SRAM, которая может быть запитана от отдельного источника VBAT, когда основное питание VDD отключено. Резервная память SRAM может быть использована как внутренняя EEPROM без какого-либо дополнительного программного обеспечения так долго, пока на микроконтроллер подается напряжение VBAT (например, при питании от батарей). При этом достоинством является быстрый доступ к резервной памяти SRAM, так же, как и к обычной внутренней памяти SRAM.

Однако когда резервная SRAM используется для других целей и/или устройство не предусматривает применение автономного питания VBAT, то внутренняя flash-память может служить для эмуляции наличия внутренней EEPROM-памяти на кристалле (в этом случае необходимо реализовать алгоритм эмуляции EEPROM-памяти).

При эмуляции EEPROM задействуется как минимум два сектора flash-памяти. Алгоритм эмуляции записывает данные в один сектор, а другой остается чистым. Как только один из секторов заполняется данными, алгоритм обменивает содержимое этих секторов flash-памяти. Программное обеспечение для эмуляции EEPROM средствами встроенной flash-памяти будем называть драйвером EEPROM.

Основные черты драйвера EEPROM:

- Легковесная реализация с простым интерфейсом API, который содержит всего три функции для инициализации, чтения и записи данных.
- Минимальный износ flash-памяти: страница стирается только тогда, когда заполнена на 100%.
- Очистка памяти и другие внутренние операции с flash-памятью прозрачны для программиста.
- Очистка сектора в фоновом режиме: обработка прерываний не блокируется во время программирования/стирания сектора.
- Конфигурируемый пользователем размер эмулируемой EEPROM. Задействовано как минимум два сектора, хотя можно задействовать больше. В этом случае следует следить за равномерностью использования секторов (технология выравнивания износа Wear Leveling).

Размер эмулируемой EEPROM-памяти можно увеличить, используя сектора flash-памяти большего размера и в большем количестве.

Основные отличия между внешней и эмулируемой EEPROM

EEPROM — это ключевой компонент многих встроенных устройств, которые требуют наличия энергонезависимой памяти с возможностью обновления данных и доступом к данным в виде отдельных байтов или двухбайтовых слов.

Микроконтроллеры, используемые в таких системах, наиболее часто имеют встроенную flash-память. Для того чтобы уменьшить число внешних компонентов на печатной плате, размер самой платы, а также себестоимость устройства, flash-память микроконтроллеров STM32F40x/STM32F41x можно использовать одновременно для хранения кроме кода программы также энергонезависимых данных.

Основное отличие flash-памяти от памяти EEPROM в том, что flash-память не допускает стирания отдельных байтов или слов. Поэтому для повторной записи отдельного байта придется стереть весь сектор, размер которого составляет обычно несколько кбайт. EEPROM-память, напротив, позволяет индивидуально стирать и записывать каждый отдельный байт.

Особенность записи во flash-память и значительно меньший ресурс записи/стирания (10^3 против 10^6 у EEPROM) диктуют требование к алгоритму эмуляции, которое заключается в достижении минимально возможного количества стираний сектора. Поэтому для того чтобы сохранять данные во встроенной flash-памяти, необходимо наличие дополнительного программного обеспечения — драйвера для программной эмуляции EEPROM.

Реализация алгоритма эмуляции зависит от множества факторов, включая требования к надежности EEPROM, внутреннее устройство используемой flash-памяти и требования к конечному продукту.

Отличия между способами хранения энергонезависимых данных не зависят от конкретного семейства микроконтроллеров. Основные отличия сведены в таблицу 1.

Таблица 1. Сравнение способов энергонезависимого хранения данных

Параметр	Внешняя память EEPROM (например, M24C64 — микросхема EEPROM-памяти с интерфейсом I²C)	Эмуляция EEPROM во внутренней flash-памяти	Эмуляция EEPROM с помощью внутренней резервной SRAM
Время записи	Запись по произвольному адресу: запись байта — 5 мс; запись 32-битного слова — 20 мс Последовательная запись: запись страницы в 32 байта — 5 мс; запись 32-битного слова — 625 мкс	Время записи полуслова из 16 бит: от 30 мкс до 237,25 мс	Со скоростью работы ядра микроконтроллера
Время стирания	Не определено	Стирание сектора: от 1 до 3 с (зависит от размера сектора)	Не определено
Метод записи	Единожды инициированный, не требует участия процессора, необходимо только напряжение питания	Единожды инициированный, задействует процессор. Аппаратный сброс останавливает запись, программный — нет.	Можно записывать байты (8 бит), полуслова (16 бит) и слова (32 бит).
		Можно записывать байты (8 бит), полуслова (16 бит) и слова (32 бит)	Запись прерывается программным сбросом
Время чтения	Последовательное чтение: сотни мкс Случайное чтение слова: 92 мкс Чтение страницы: 22,5 мкс на 1 байт	Полуслово: от 0,68 до 251 мкс (для тактовой частоты 168 МГц)	Со скоростью работы ядра микроконтроллера
Количество циклов записи/чтения	1×10^6	1×10^3 на сектор. Использование нескольких секторов увеличивает количество циклов	Не ограничено, пока подается напряжение VBAT

Так как flash-память обладает меньшим временем доступа, то критические параметры могут быть сохранены быстрее в эмулированной EEPROM, чем во внешней EEPROM, подключенной по последовательному интерфейсу. Но это справедливо лишь тогда, когда не возникает необходимости в стирании сектора flash-памяти.

Разница во времени стирания — еще одно важное отличие внешней EEPROM по сравнению с эмулируемой. В отличие от flash-памяти технология EEPROM не требует операции стирания перед каждой операцией записи. Это означает, что для сохранения данных во flash-памяти необходимо дополнительное программное обслуживание процедур записи/чтения. Более того, процедура стирания страницы flash-памяти занимает продолжительное время, в течение которого не должно быть выключения питания, сброса процессора и других событий, которые могут прервать процесс стирания страницы. Эту особенность следует учитывать при проектировании программы, использующей эмуляцию EEPROM.

Что касается семейства STM32F40x/STM32F41x, то при программном сбросе процесс стирания страницы flash-памяти не прерывается.

Методы записи для эмулируемой и внешней EEPROM схожи. Для внешней EEPROM операция записи, будучи инициирована, не может быть остановлена сбросом микроконтроллера. Только спад напряжения питания может остановить процесс записи. Что касается эмулируемой EEPROM, лишь отсутствие питания может остановить процесс записи. Если при записи очередного слова во flash-память произойдет сброс микроконтроллера, то алгоритм эмуляции, естественно, остановится, но записываемое слово запишется во flash-память полностью.

Принцип работы алгоритма эмуляции EEPROM

Эмуляцию EEPROM можно выполнить множеством способов, учитывая ограничения flash-памяти и требования конкретного приложения. Для изложенного далее подхода нужно как минимум два сектора flash-памяти одинакового размера: один изначально стирается и позволяет побайтно записывать данные, другой находится в состоянии готовности заменить первый сектор, когда тот нуждается в очистке от накопившегося «мусора». Каждый из этих секторов будем называть страницами. Поле заголовка, которое занимает первое полуслово (16 бит) каждого сектора, показывает его статус. Поле заголовка располагается по базовому адресу каждой страницы.

Каждая переменная в эмулируемой EEPROM-памяти определяется ее виртуальным адресом и значением. В данной реализации и адрес переменной, и сама переменная заданы длиной 16 бит. Как адрес, так и значение переменной записывается во flash-память так, чтобы в дальнейшем по адресу переменной можно было получить ее значение, а также изменить значение переменной с заданным адресом.

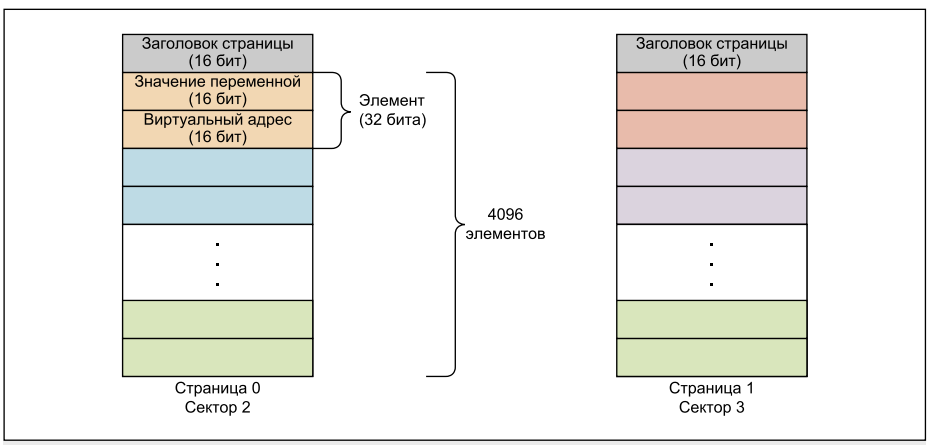

Рис. 1. Формат хранения переменной в эмулируемой EEPROM

Запись переменной

Когда необходимо присвоить переменной другое значение, новое значение переменной вместе с виртуальным адресом записываются в новое место flash-памяти (в новый элемент на рис. 1). Старое значение переменной никуда не пропадает: оно остается записано вместе с виртуальным адресом в элементе, расположенном по младшему физическому адресу (рис. 2).

Чтение переменной

При считывании переменной возвращается ее последнее записанное значение. Для этого перебираются все элементы, начиная с последнего на странице, до тех пор, пока не будет найден элемент с заданным виртуальным адресом (рис. 3).

Так как ячейки flash-памяти после стирания хранят значение 0xFFFF, то для переменной можно использовать диапазон виртуальных адресов от 0x0000 до 0xFFFE.

Смена страницы при ее заполнении

Очевидно, что по мере процесса модификации переменных в эмулируемой EEPROM свободная память внутри страницы рано или поздно закончится. При этом действительные значения переменных могут быть «разбросаны» по всей странице. Именно поэтому драйвер эмуляции EEPROM использует две страницы flash-памяти. Вторая страница

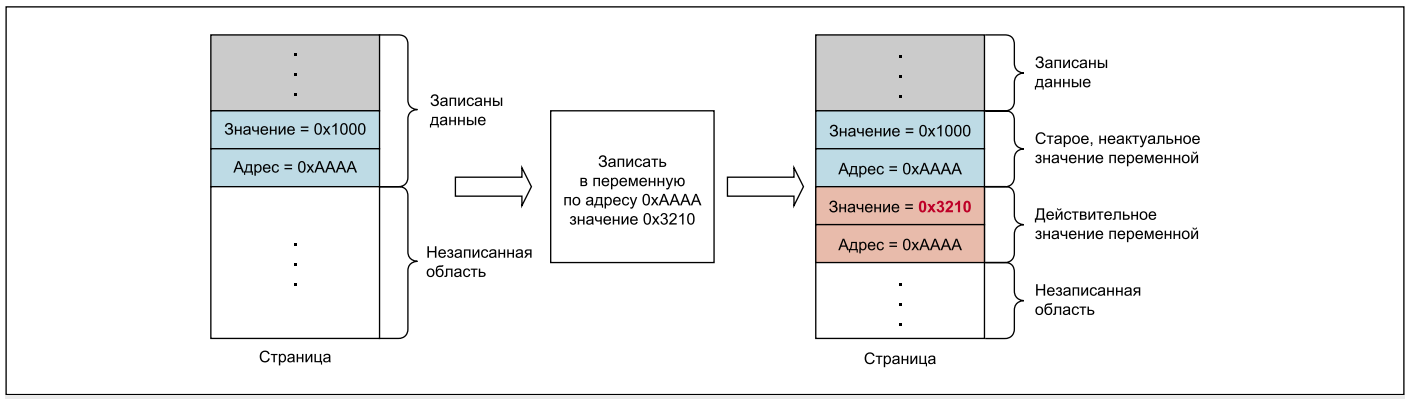

Рис. 2. Принцип модификации значения переменной

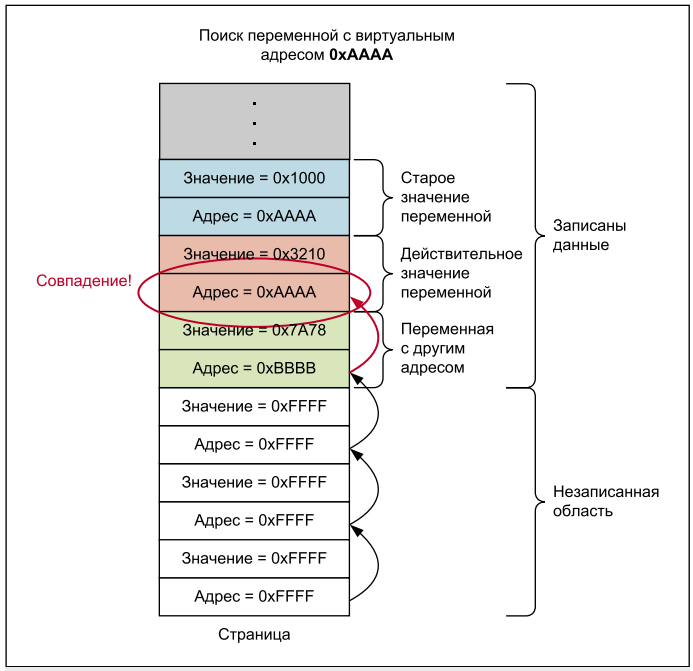

Рис. 3. Поиск переменной при ее чтении

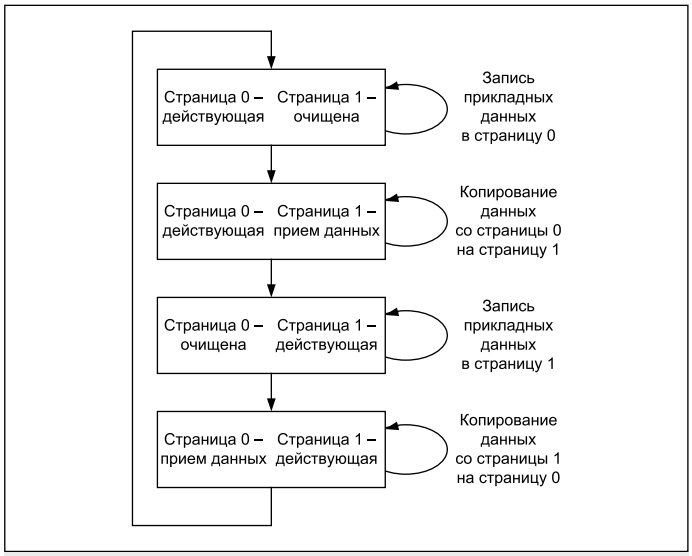

Рис. 4. Переходы между состояниями страницы 0 и страницы 1

нужна для того, чтобы переписать действительные значения всех переменных на нее перед тем, как очистить первую страницу.

Чтобы реализовать этот принцип, каждой странице присваивается состояние. Каждая страница может находиться в одном из трех состояний:

1. ERASED. Очищена. Страница пуста.
2. RECEIVE_DATA. Прием данных. Страница принимает данные от другой заполненной страницы.
3. VALID_PAGE. Действующая страница, на нее производится запись пользовательских данных. Это состояние не изменяется для страницы, пока все действующие данные не будут переписаны на только что очищенную страницу.

В процессе модификации и записи переменных в эмулируемую EEPROM страницы поочередно меняют свои состояния так, как показано на рис. 4.

Поле заголовка располагается по базовому адресу каждой страницы и предоставляет информацию о ее состоянии. Размер поля заголовка — 16 бит.

Следующий пример показывает процессы заполнения одной страницы и переход к записи на другую при работе с тремя переменными в эмулируемой EEPROM (Var1, Var2, Var3). Соответственно, каждая переменная имеет виртуальный адрес 0x5555, 0x6666, 0x7777 (рис. 5).

Реализация драйвера EEPROM

Здесь описывается реализация драйвера для программной эмуляции EEPROM с использованием flash-памяти микроконтроллеров семейства STM32F40x/STM32F41x. Этот драйвер компания STMicroelectronics предоставляет бесплатно, загрузить его можно по адресу [2]. Демонстрационная программа также включена в состав драйвера.

Непосредственно драйвер EEPROM реализован внутри файлов *eeprom.c* и *eeprom.h*. Демонстрационный проект содержит также файл *main.c*.

Назначение этих файлов:

- *eeprom.c* — содержит реализацию следующих API-функций: ***EE_Init()***; ***EE_Format()***; ***EE_FindValidPage()***; ***EE_VerifyPageFullWriteVariable()***; ***EE_ReadVariable()***; ***EE_PageTransfer()***; ***EE_WriteVariable()***.
- *eeprom.h* — содержит прототипы функций и конфигурационные макроопределения. Разработчику предлагается использовать этот файл, чтобы оптимизировать драйвер EEPROM для своего приложения.
- *main.c* — демонстрационная прикладная программа, где показано, как использовать функции драйвера EEPROM.

Программный интерфейс (API) драйвера EEPROM

Все функции драйвера EERPOM имеют имя, начинающееся с префикса «EE_». Список всех функций и их краткое описание сведены в таблицу 2.

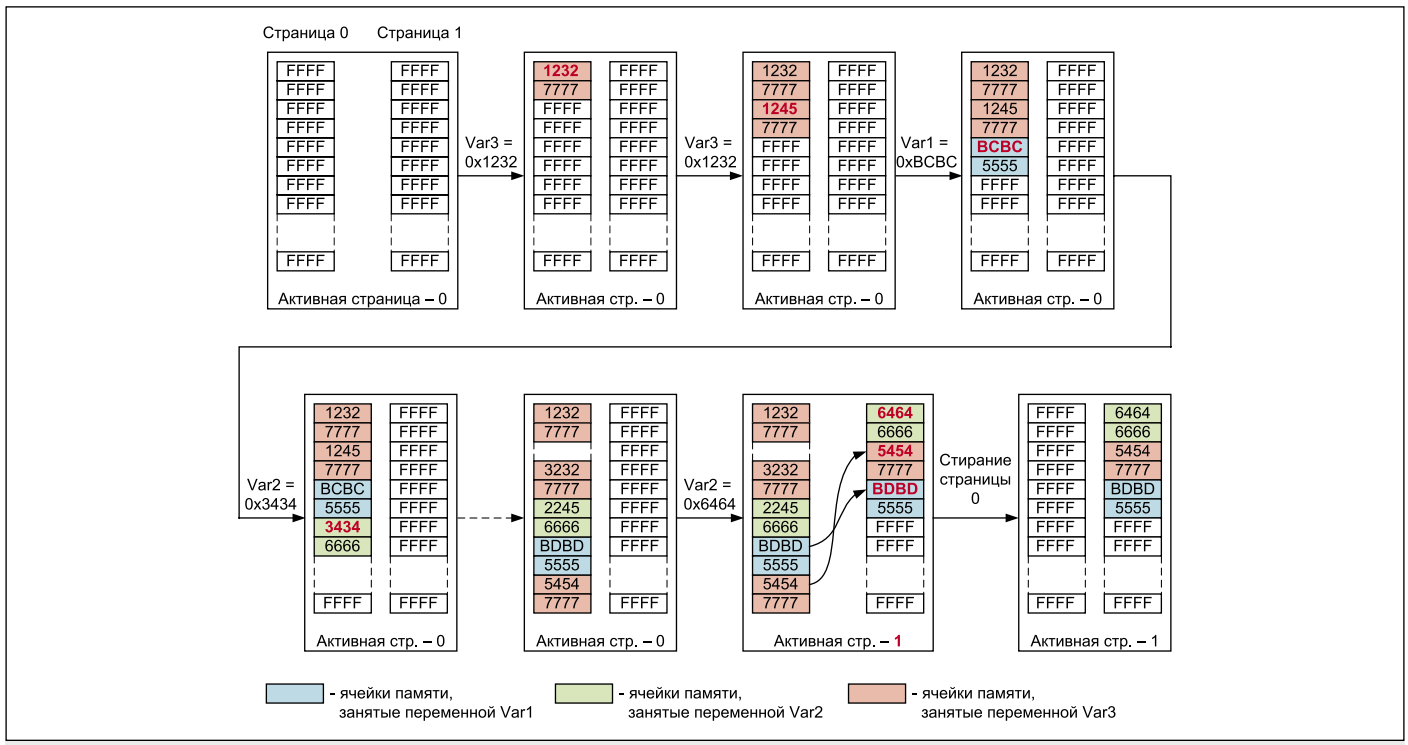

Рис. 5. Принцип обновления данных во flash-памяти при эмуляции EEPROM

Таблица 2. Список API-функций драйвера EEPROM

Имя API-функции	Краткое описание
uint16_t EE_Init(void)	Заголовок сектора может быть поврежден при пропадании напряжения питания во время обновления данных или очистки/копирования всего сектора. В этом случае функция EE_Init() может быть применена для попытки восстановления эмулируемой EEPROM до известного неповрежденного состояния. Эту функцию необходимо вызывать перед доступом к эмулируемой EEPROM после каждого выключения питания. Функция не получает параметров и возвращает FLASH_COMPLETE при успешной инициализации, в противном случае — код ошибки
FLASH_Status EE_Format(void);	Функция очищает страницу 0 и страницу 1 и записывает состояние «Действующая страница» (VALID_PAGE) в заголовок страницы 0
uint16_t EE_FindValidPage (uint8_t Operation);	Эта функция читает заголовки обеих страниц и возвращает номер страницы, имеющей состояние «Действующая» (VALID_PAGE). Передаваемый параметр определяет, для какой операции необходимо найти действующую страницу: для чтения (READ_FROM_VALID_PAGE) или записи (WRITE_IN_VALID_PAGE)
uint16_t EE_VerifyPageFullWriteVariable (uint16_t VirtAddress, uint16_t Data);	Эта функция реализует процесс записи, который должен также обновить или создать первый экземпляр переменной. Функция ищет первый пустой элемент в активной странице и заполняет его переданными в качестве аргументов функции виртуальным адресом переменной и ее значением. В случае если активная страница заполнена, возвращается значение PAGE_FULL. Аргументы функции: – VirtAddress — виртуальный адрес. Может быть любой из ранее определенных виртуальных адресов. – Data — данные, значение записываемой в EEPROM переменной. Функция возвращает FLASH_COMPLETE, если запись прошла успешно, PAGE_FULL, если недостаточно памяти или произошла ошибка, или код ошибки flash-памяти
uint16_t EE_ReadVariable (uint16_t VirtAddress, uint16_t* Data);	Функция возвращает значение переменной с переданным в качестве аргумента виртуальным адресом. Только последний, самый «свежий» экземпляр переменной считывается. Функция входит в цикл, внутри которого считываются все элементы на странице, начиная с ее конца, пока не будет найден элемент, содержащий искомый виртуальный адрес. Этот элемент и содержит действительное значение переменной. Если ни одного экземпляра переменной не найдено, аргумент ReadStatus возвращается со значением 1, в противном случае он сброшен в 0. Это означает, что переменная успешно прочитана и ее значение возвращено в аргументе Read_data
uint16_t EE_PageTransfer (uint16_t VirtAddress, uint16_t Data);	Функция переносит последние записанные экземпляры всех переменных (значения и виртуальные адреса) из текущей в новую активную страницу. Прежде всего, функция определяет активную страницу, из которой будет считываться информация. В заголовок новой страницы записывается состояние «Прием данных» (RECEIVE_DATA). После того как данные переписаны, состояние новой страницы сменяется на «Действующая страница» (VALID_PAGE). Старая же страница стирается, и в ее заголовок записывается состояние «Очищена» (ERASED)
uint16_t EE_WriteVariable (uint16_t VirtAddress, uint16_t Data);	Эта функция вызывается из программы пользователя для обновления значения переменной. Функция использует вызовы EE_VerifyPageFullWriteVariable() и EE_PageTransfer()

Заметьте, что для работы с эмулируемой EEPROM пользовательской программе достаточно вызвать всего лишь три функции: ***EE_Init()***; ***EE_ReadVariable()***; ***EE_WriteVariable()***. Пример использования этих функций приведен в демонстрационной программе, которая включена в состав драйвера EEPROM [2]. Остальные функции являются служебными и вызываются из приведенных выше трех функций. Например, функция ***EE_WriteVariable()***, которая модифицирует переменную в эмулируемой EEPROM, сводится к действиям (и соответственно, вызовам API-функций), показанным на рис. 6.

Настройка драйвера EEPROM

Для того чтобы использовать драйвер EEPROM в своей прикладной программе, необходимо настроить его в соответствии с требованиями к разрабатываемому приложению.

Настройка осуществляется изменением значений макроопределений, содержащихся в файле *eeprom.h*. Кроме того, программист должен определить в своей программе массив, содержащий виртуальные адреса всех переменных в следующем виде:

```
extern uint16_t VirtAddVarTab[NB_OF_VAR];
```

Заметьте, что виртуальный адрес 0xFFFF применять нельзя, так как это число используется как признак пустой ячейки. (Дело в том, что после стирания сектора flash-памяти все двухбайтовые слова в нем будут содержать число 0xFFFF.)

Список конфигурационных макроопределений:

1. ***PAGE0_ID*** и ***PAGE1_ID*** — номера используемых секторов (по умолчанию это сектора 2 и 3).
2. ***PAGE_SIZE*** — размер сектора flash-памяти в байтах (по умолчанию 16 кбайт).
3. ***EEPROM_START_ADDRESS*** — адрес начала страницы 0 во flash-памяти. Адрес начала страницы 1 получается как EEPROM_START_ADDRESS + PAGE_SIZE (по умолчанию адрес начала сектора 2 — 0x08008000).
4. ***VOLTAGE_RANGE*** — диапазон питающих напряжений (по умолчанию VoltageRange_3). В демонстрационном проекте есть функция ***FLASH_ProgramHalfWord()***. Она может быть использована, только если питание микроконтроллера составляет от 2,1 до 3,6 В. Если же диапазон от 1,8 до 2,1 В, то следует использовать

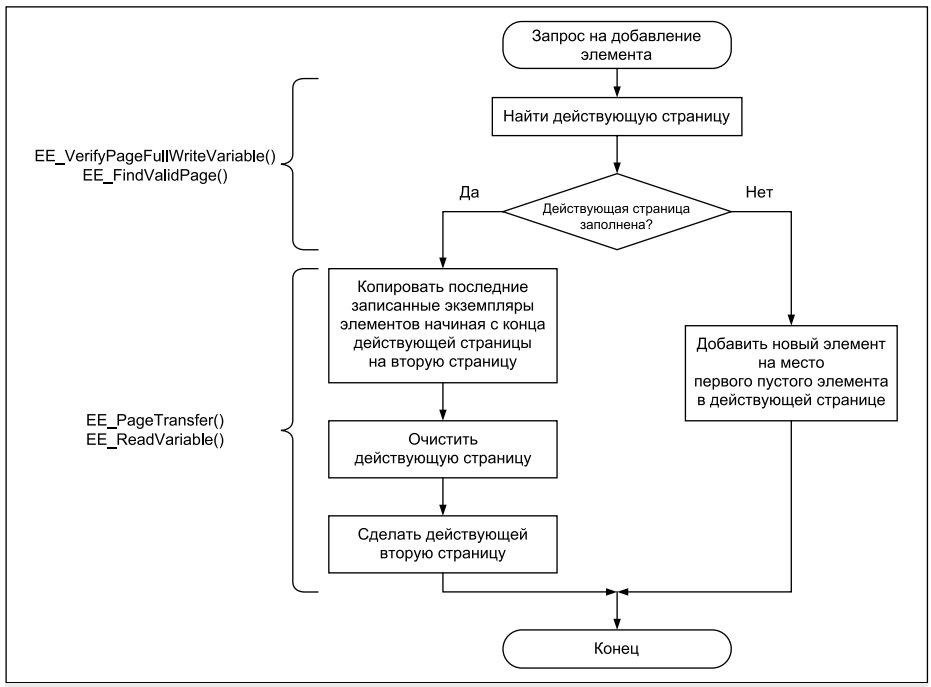

Рис. 6. Алгоритм обновления значения переменной в эмулируемой EEPROM

функцию ***FLASH_ProgramByte()***. (Далее будет подробнее описана эта особенность семейства STM32F40x/STM32F41x.)

5. ***NB_OF_VAR*** — количество используемых переменных в эмулируемой EEPROM и, соответственно, размер массива VirtAddVarTab (по умолчанию 3).

Таблица 3. Размеры секторов flash-памяти в семействе микроконтроллеров STM32F40x/STM32F41x

Сектор	Размер сектора, кбайт
Сектора с 0 по 3	16
Сектор 4	64
Сектора с 5 по 11	128

Драйвер эмуляции EEPROM и его реализация, описанные здесь, используют два сектора flash-памяти по 16 кбайт каждый — сектора 2 и 3. Выбор в пользу секторов 2 и 3 сделан исходя из их наименьшего объема относительно других секторов flash-памяти в семействе STM32F40x/STM32F41x. Сектора большого размера можно использовать в зависимости от требований к приложению. Размеры других секторов flash-памяти для семейства STM32F40x/STM32F41x приведены в таблице 3.

Расход памяти и время чтения/записи при эмуляции EEPROM

В таблице 4 представлены минимально возможные объемы памяти программ (flash) и памяти данных (SRAM), которые задействуются алгоритмом программной эмуляции памяти EEPROM путем использования flash-памяти.

Данные из таблицы 4 получены со средой разработки и компилятором IAR EWARM 6.10 с уровнем оптимизации High Size. При этом использовалась всего одна переменная, причем размер элемента составлял 32 бита (16 бит виртуальный адрес и 16 бит — значение переменной). Использование большего числа переменных повлечет за собой увеличение объемов потребляемой памяти.

Таблица 4. Минимальное потребление памяти драйвером эмуляции EEPROM

Механизм	Минимальный объем занимаемой памяти, байт	
	flash	SRAM
Механизм программной эмуляции EEPROM-памяти с помощью flash-памяти	984	6

Рис. 7 позволяет оценить объем потребляемой flash-памяти относительно общего объема для микроконтроллеров с 1 Мбайт flash-памяти.

Что касается времени чтения/записи, то временные характеристики были получены для следующих условий:

- Эмуляция EEPROM с использованием двух секторов flash-памяти по 16 кбайт каждый.
- Размер элемента — 32 бита (16 бит занимает виртуальный адрес и 16 бит — значение переменной).
- Микроконтроллер STM32F407IGT6 Revision A.
- Диапазон питающих напряжений — 2,7–3,6 В.
- Тактовая частота 168 МГц, предвыборка из flash отключена, кэш задействован.
- Выполнение программы из flash-памяти.
- Комнатная температура.

В таблицу 5 собраны полученные экспериментальным путем результаты.

Разница между максимальным и минимальным временем записи переменной без копирования и стирания страницы объясняется тем, что первый свободный элемент может быть первым в странице (в этом случае время минимально), а может быть и послед-

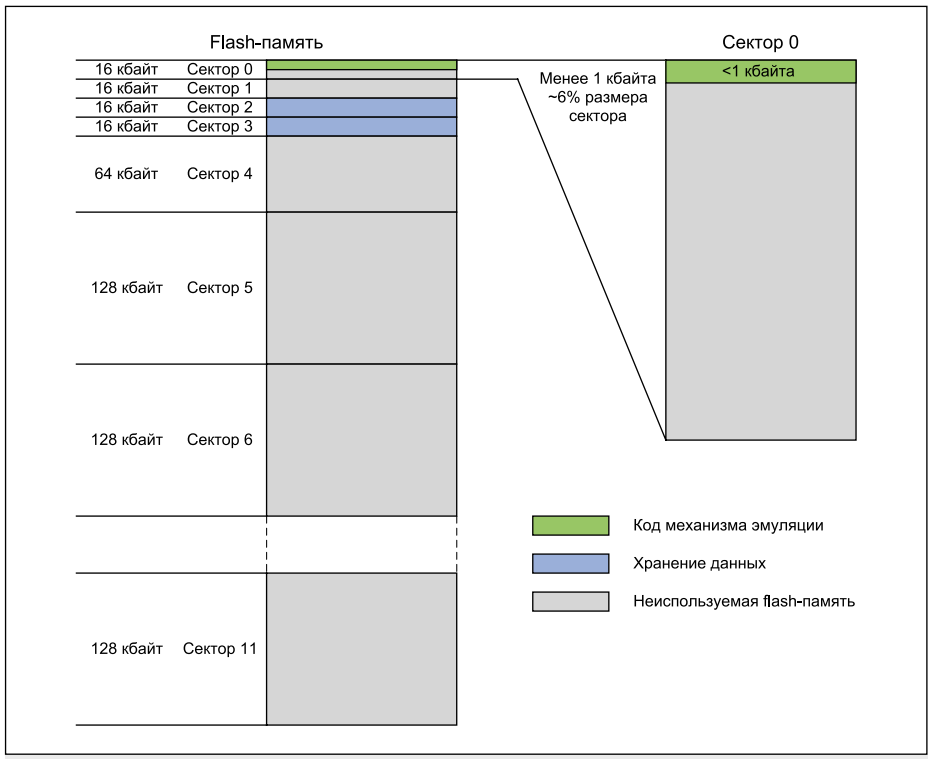

Рис. 7. Расход flash-памяти для эмуляции EEPROM (реализация драйвера и хранение данных)

Таблица 5. Время операций при эмуляции EEPROM

Операция	Время, затраченное на операцию		
	min	тип.	max
Запись переменной без копирования на новую страницу	28 мкс	–	255 мкс
Запись переменной с копированием на новую страницу	–	227 мс	–
Чтение переменной	0,68 мкс	–	251 мкс
Инициализация со стиранием двух страниц	–	473 мс	–
Типичная инициализация (стирание одной страницы)	–	237 мс	–

ним, когда страница почти заполнена (время максимально).

Операция копирования на новую страницу (с последующим стиранием) выполняется тогда, когда вся действующая страница заполнена и требуется записать новое значение переменной (или новую переменную).

Разница между максимальным и минимальным временем чтения переменной также обусловлена различным временем поиска расположения переменной на странице flash-памяти. Минимальное время соответствует расположению переменной в первой ячейке, максимальное — в последней.

Когда механизм EEPROM начинает работать в первый раз или когда обнаружена ошибочная комбинация состояний страниц, то обе страницы очищаются. Типичная инициализация механизма эмуляции EEPROM происходит, когда только одна из страниц имеет состояние «действующая», при этом стирается только одна страница.

Режим реального времени

Предлагая способ эмуляции EEPROM, мы подразумеваем, что доступ к flash-памяти во время записи переменной и (что особенно важно) во время стирания страницы будет приостановлен. Как следствие, во время вышеперечисленных операций с flash-памятью не может выполняться никакая другая программа, в том числе и обработка прерываний.

Такие ограничения могут быть допустимы для множества приложений, но при жестких требованиях по времени реакции устройства следует организовать выполнение критических участков кода с внутренней памяти RAM. В этом случае:

- Разместите таблицу векторов прерываний во внутренней RAM.
- Выполняйте все критические процессы и обработку прерываний из внутренней RAM. Обычно компилятор позволяет с помощью ключевого слова задать расположение той или иной функции в памяти RAM. При этом компилятор создаст такой код, который скопирует код функции из flash-памяти в RAM в начале выполнения программы сразу после подачи напряжения питания. Важно, что все вызываемые функции и все переменные, используемые из функции, размещенной в памяти RAM, также должны быть размещены в памяти RAM.

Управление разрядностью переменной

Механизм эмуляции EEPROM может использоваться для записи 8-, 16- и 32-битных переменных. В общем случае это зависит от требований к приложению и архитектуры flash-памяти.

Микроконтроллеры семейства STM32F40x/STM32F41x позволяют записывать flash-память побайтно (8 бит), по полуслову (16 бит) и пословно (32 бит). Однако семейство STM32F40x/STM32F41x обладает важной особенностью, которая заключается в том, что разрядность записываемых данных (8, 16, 32 бита) ограничивается напряжением питания микроконтроллера. Чем ниже напряжение питания, тем меньшего размера блоки допускается записывать во flash-память. Эта зависимость приведена в таблице 6.

Таблица 6. Зависимость разрядности данных, записываемых во flash-память, от напряжения питания

Размер данных	Имя функции	Допустимый диапазон питающего напряжения, В
Слово 32 бита	FLASH_ProgramWord()	2,7–3,6
Полуслово 16 бит	FLASH_ProgramHalfWord()	2,1–3,6
Байт 8 бит	FLASH_ProgramByte()	Весь возможный диапазон

При программировании flash-памяти побайтно можно записать большее количество переменных, однако производительность будет ниже, чем при записи пословно или по полуслову.

Выравнивание износа

Для семейства STM32F40x/STM32F41x каждый сектор встроенной flash-памяти гарантированно, без потерь данных, может быть записан и стерт 10 000 раз. Это накладывает определенные ограничения на количество операций записи переменной в эмулируемую EEPROM.

Для приложений, где предусматривается большáя интенсивность записи переменных в энергонезависимую память, может быть использовано более двух секторов flash-памяти для эмуляции EEPROM. В таких случаях рекомендуется реализовать алгоритм выравнивания износа, который позволяет отслеживать количество стираний каждой из страниц flash-памяти и управлять им.

Если не использовать алгоритм выравнивания износа, то страницы могут стираться с различной частотой. Страницы, содержащие «долго живущие» данные, не испытывают такого частого стирания, как те, что содержат часто изменяемые данные.

Алгоритм выравнивания износа гарантирует, что каждый из секторов задействованной для эмуляции EEPROM flash-памяти будет записан и стерт одинаковое количество раз.

Пример реализации алгоритма выравнивания износа

В этом примере в целях увеличения объема эмулируемой EEPROM задействуется четыре сектора flash-памяти и, соответственно, четыре страницы.

Алгоритм выравнивания износа сводится к следующему: когда страница N заполнена, происходит переключение на очищенную ранее страницу N+1. Со страницы N считываются действительные значения переменных, они переписываются на страницу N+1, затем очищается страница N. Когда последняя страница заполнена, следующей за ней используется страница 0. Графически этот алгоритм представлен на рис. 8 (в случае использования четырех страниц).

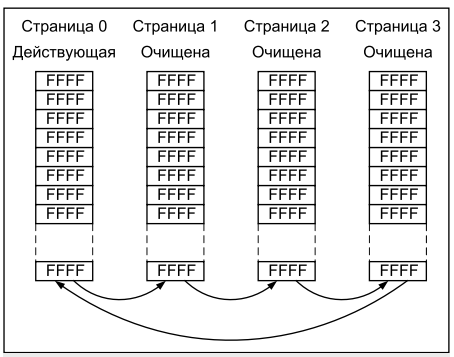

Рис. 8. Принцип работы алгоритма выравнивания износа

Программно алгоритм выравнивания износа может быть реализован внутри функции ***EE_FindValidPage()***.

Восстановление заголовка страницы

Потеря данных или случайное изменение заголовка страницы возможно при внезапном выключении напряжения питания микроконтроллера во время операции обновления переменной, стирания страницы или ее копирования. Для определения факта повреждения информации и ее восстановления служит API-функция ***EE_Init()***. Она должна вызываться в прикладной программе сразу после подачи питания на микроконтроллер. Функция проверяет состояния страниц flash-памяти для определения целостности данных и их восстановления, если это необходимо.

В зависимости от состояния страниц API-функция ***EE_Init()*** выполняет действия, описанные в таблице 7.

Функция ***EE_Init()*** проверяет состояние страниц, записанное в их заголовках. Для двух страниц и трех возможных состояний каждой страницы возможно девять комбинаций. Таблица 7 показывает, какие действия выполняет функция ***EE_Init()*** в зависимости от комбинации состояний страниц.

Ресурс памяти

Цикл записи/стирания содержит одну или больше операций записи и одну опера-

Таблица 7. Возможные состояния страниц flash-памяти при эмуляции EEPROM и соответствующие им действия

Страница 1	Страница 0		
	Очищена	Прием данных	Действующая страница
Очищена	Ошибочное состояние! Очистить обе страницы и отформатировать страницу 0	Очистить страницу 1, пометить страницу 0 как действующую	Использовать страницу 0 как действующую и очистить страницу 1
Прием данных	Очистить страницу 0, пометить страницу 1 как действующую	Ошибочное состояние!	Использовать страницу 0 как действующую. Переместить последние измененные переменные со страницы 0 на страницу 1
		Очистить обе страницы и отформатировать страницу 0	Пометить страницу 1 как действующую
			Очистить страницу 0
Действующая страница	Использовать страницу 1 как действующую и очистить страницу 0	Использовать страницу 1 как действующую. Переместить последние измененные переменные со страницы 1 на страницу 0	Ошибочное состояние!
		Пометить страницу 0 как действующую	Очистить обе страницы и отформатировать страницу 0
		Очистить страницу 1	

цию стирания страницы. При использовании истинной технологии EEPROM каждый байт памяти может быть записан и стерт конечное число раз, обычно ресурс составляет от 10 000 до 1 000 000 раз для «обычной» EEPROM-памяти.

Однако для встроенной flash-памяти минимальный объем, который может быть очищен за один раз, равен одному сектору. При этом число циклов записи/стирания, применимое к сектору flash-памяти, есть число возможных стираний. Для семейства STM32F40x/STM32F41x гарантированный ресурс записи/стирания одного сектора составляет 10 000 циклов. Максимальное время жизни эмулируемой EEPROM ограничено частотой обновления самой часто обновляемой переменной.

Ресурс эмулируемой EEPROM зависит от количества и размера переменных, которые требуется хранить и обновлять в энергонезависимой памяти. Например, два сектора по 16 кбайт каждый используются для хранения 16-битных переменных. Каждая переменная определяется ее 16-битным виртуальным адресом. Таким образом, каждая переменная хранится в виде элемента (значение переменной + адрес) и занимает одно слово (32 бита) flash-памяти. Размер сектора в 16 кбайт, умноженный на ресурс flash-памяти в 10 000 циклов, дает общий объем данных в 160 000 кбайт, которые могут быть гарантированно записаны и считаны из одной страницы эмулируемой EEPROM. Так как используется две страницы, то общий объем данных, который может быть записан в эмулируемую EEPROM, составляет 320 000 кбайт. Если используется большее число страниц flash-памяти, то объем в 160 000 кбайт умножается на количество задействованных страниц.

Исходя из размера переменной и ее виртуального адреса (то есть размера одного элемента), можно вычислить допустимое количество записей переменной в эмулируемую flash-память. Например, для 16-битного виртуального адреса и 16-битной переменной для одной страницы получаем 160 000 кбайт/4 байт = 4×10^7 записей.

Выбор размера и количества секторов

Размер сектора и количество секторов, требуемые для механизма эмуляции EEPROM, можно выбрать исходя из необходимого количества записей одной переменной на протяжении времени жизни проектируемого устройства.

Например, страница размером в 16 кбайт и ресурсом в 10 000 циклов может быть использована для записи максимум 160 Мбайт информации на протяжении жизненного цикла устройства, в то время как страница размером 128 кбайт позволяет записать уже 1280 Мбайт.

Свободный объем памяти для размещения переменных можно вычислить по формуле:

$$V_{своб} = V_{стр}/V_{эл}-(N_{пер}+1),$$

где $V_{стр}$ — размер страницы в байтах (например, 16 384 байта); $V_{эл}$ — размер элемента (число байтов, используемых для хранения одной переменной, которое может принимать следующие значения: 8 байтов — для 32-битной переменной и ее адреса, 4 байта — для 16-битной переменной и ее адреса, 2 байта — для 8-битной); $N_{пер}$ — число используемых переменных (например, десять).

Требуемое количество страниц flash-памяти, необходимых для неполного исчерпания ресурса в течение заданного времени жизни устройства, можно вычислить по формуле:

$$N_{страниц} = N_{операций\_записи}/$$
$$/(N_{стираний\_страницы} \times V_{своб}),$$

где $N_{операций\_записи}$ — количество операций записи переменной в эмулируемую EEPROM; $N_{стираний\_страницы}$ — ресурс стираний одной страницы.

Если используются переменные размером в 16 бит, то для хранения такой переменной требуется 32 бита (16 бит переменная плюс 16 бит виртуальный адрес), значит, каждая операция записи такой переменной расходует 4 байта flash-памяти. Следовательно, в страницу размером 16 кбайт (128 кбайт) можно записать 4096 (32 768) переменных к моменту, когда она вся заполнится.

Пример расчета

Вычислим необходимое количество секторов flash-памяти и их размер, исходя из требований к разрабатываемому устройству. Расчет носит оценочный характер.

Нужно разработать устройство, в котором необходимо хранить в энергонезависимой памяти 20 различных переменных ($N_{пер}$), причем обновление значения каждой переменной происходит раз в 2 мин. Пусть расчетный жизненный цикл устройства составляет 10 лет. Тогда:

$$N_{операций\_записи} = 10 \text{ лет} \times 365 \text{ дней} \times$$
$$\times 24 \text{ ч} \times (60 \text{ с}/2 \text{ раза в секунду}) \times N_{пер} \approx$$
$$\approx 52\times10^6 \text{ операций «запись»};$$

$$N_{стираний\_страницы} = 10\,000.$$

Сведем результаты расчета в таблицу 8. Расчет произведен для различной разрядности переменных и разного размера страницы flash-памяти.

Заметьте, что если расчет дает результат меньше одной страницы, все равно используется как минимум две страницы (требование алгоритма эмуляции EEPROM). Так, если переменная и ее виртуальный адрес занимают по 8 бит, тогда необходимое количество страниц — две страницы по 16 кбайт или две страницы по 128 кбайт. Если переменная и ее виртуальный адрес занимают по 16 бит, тогда необходимое количество страниц — две страницы по 16 кбайт или две страницы по 128 кбайт. Если переменная и ее виртуальный адрес занимают по 32 бит, тогда необходимое количество страниц — три страницы по 16 кбайт или две страницы по 128 кбайт.

Таблица 8. Расчет необходимого количества секторов flash-памяти

Размер переменной, бит	Количество операций «запись» (Nопераций_записи)	Общий объем данных, байт	Размер страницы, кбайт	Ресурс стирания страницы	Требуемое количество страниц	Требуемое количество страниц с учетом специфики алгоритма эмуляции
8	52 560 000	105 120 000	16	10 000	0,6	2
			128		0,1	
16	52 560 000	210 240 000	16	10 000	1,3	2
			128		0,2	
32	52 560 000	420 480 000	16	10 000	2,6	3
			128		0,3	2

Выводы

Итак, эмуляция EEPROM средствами встроенной flash-памяти представляет собой действенный способ хранить энергонезависимые данные, когда по тем или иным причинам невозможно использовать внешнюю микросхему EEPROM или внешнее батарейное питание для хранения данных во встроенной резервной SRAM.

Тем не менее этому способу свойственны существенные недостатки:

- Большое время выполнения операции записи переменной, если при этом происходит копирование всех действующих переменных на другую страницу,— порядка 200 мс, что по меркам современной цифровой электроники очень много.
- Большая дисперсия (большой разброс) времени чтения и особенно записи переменной. Для операции записи переменной время, затрачиваемое на эту операцию, может отличаться почти на 4 (!) порядка.
- Во время записи переменной и особенно во время стирания страницы приостанавливается доступ к flash-памяти, то есть в это время невозможно выполнять обработку прерываний из flash-памяти.

Эти недостатки не позволяют применять драйвер эмуляции EEPROM для тех устройств, которые работают в режиме реального времени и для которых время реакции микроконтроллера на внешнее событие — небольшая жестко заданная величина.

Кроме того, большая продолжительность операций переноса информации между страницами и стирания страницы подразумевает возможные проблемы при выключении питания микроконтроллера во время этих операций или требует использования дополнительных схемных решений, задачей которых является поддержание напряжения питания во время продолжительных операций.

Литература

1. http://www.atmel.com/devices/at24c256.aspx
2. http://www.st.com/st-web-ui/static/active/en/st_prod_software_internet/resource/technical/software/firmware/stsw-stm32066.zip
3. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/DM00036065.pdf