AN2594

Указания по применению

Эмуляция EEPROM в микроконтроллерах STM32F10x

Введение

Для многих приложений требуется EEPROM (электрически стираемая программируемая постоянная память) для энергонезависимого хранения данных. С целью снижения цены, устройства STM32F10x не содержат EEPROM. Вместо этого они поддерживают эмуляцию EEPROM с использование встроенной Flash-памяти.

Эти указания раскрывают отличия между внешней EEPROM и встроенной Flash-памятью, а также описывают программный метод эмуляции EEPROM на основе встроенной в микроконтроллер Flash-памяти в STM32F10x.

Этот документ также освещает некоторые встроенные аспекты в эмулированных хранилищах данных EEPROM, о которых читатель, предположительно, знает.

Глоссарий

Устройства с низкой ёмкостью – это микроконтроллеры STM32F101xx, STM32F102xx, STM32F103xx с ёмкостью Flash-памяти от 16 до 32 кБайт.

Устройства с средней ёмкостью – это микроконтроллеры STM32F101xx, STM32F102xx, STM32F103xx с ёмкостью Flash-памяти от 32 до 128 кБайт.

Устройства с высокой ёмкостью – это микроконтроллеры STM32F101xx, STM32F102xx, STM32F103xx с ёмкостью Flash-памяти от 256 до 512 кБайт.

Устройства из серии подключаемых – это микроконтроллеры STM32F105xx and STM32F107xx.

Август 2009

www.st.com

Содержание

Перечень таблиц

Перечень рисунков

1 Встроенная Flash-память против EEPROM: главные отличия

Электрически стираемая программируемая постоянная память (EEPROM) является ключевым компонентом многих встроенных приложений, которым необходима энергонезависимое хранилище данных, обновляемых с дискретностью в байт или пословно во время выполнения.

С другой стороны, микроконтроллеры, используемые в тех приложениях, всё чаще и чаще основываются на встроенной Flash-памяти. Чтобы уменьшить число компонентов, сохранив площадь кремния и уменьшить стоимость системы, STM32F10xxx Flash-память может использоваться вместо EEPROM для синхронного шифрования и сохранения данных.

В отличие от Flash-памяти, внешняя EEPROM не требует какой-либо операции стирания для освобождения места перед перезаписыванием данных. Поэтому необходимо специальное программное управление для хранения данных во встроенной Flash-памяти.

Очевидно, структура программного обеспечения эмуляции зависит от многих факторов, включая надежность EEPROM, архитектуру используемой Flash-памяти, а также требований продукта.

Главные отличия между встроенной Flash-памятью и внешней последовательной EEPROM универсальны для любых микроконтроллеров, использующих идентичные технологии Flash-памяти (это не является спецификой продукции семейства STM32F10xxx). Главные отличия суммированы в *Таблице 1.*

Таблица 1. Отличия между встроенной Flash-памятью и EEPROM

Характеристика	Внешняя EEPROM	Эмулированная EEPROM с использованием встроенной Flash-памяти
Время записи	– единицы ms; – случайный байт: от 5 до 10 ms – страница: 100 µs для слова (от 5 до 10 ms для страницы)	Время записи слова: 20 µs
Время стирания	N/A	Серийное время стирания страницы: 20 ms
Метод записи	– после запуска независима от CPU, лишь требует надлежащего питания.	– после запуска зависима от CPU: сброс CPU остановит процесс записи даже если питание остаётся в пределах технических условий
Доступ для чтения	– последовательный: 100 µs – случайное слово: 92 µs – страница: 22.5 µs / байт	– параллельный: 100 ns – очень мало циклов CPU для чтения слова. – время доступа: 35 ns
Циклы записи/стирания	– от 10 000 циклов до 1 000 000 циклов	– от 10 000 циклов до 100 000 циклов (использование многих встроенных страниц Flash-памяти эквивалентно увеличению числа циклов записи) см. *Раздел 3.4: Возможности цикличности*

1.1 Отличие во времени доступа для записи

Поскольку Flash-память более короткое время доступа для записи, критические параметры могут быть сохранены в эмулированной EEPROM быстрее, чем во внешней последовательной EEPROM, улучшая таким образом сохранность данных.

1.2 Отличие в методике записи

Одно из важнейших отличий для встроенных приложений между внешней EEPROM и эмулированной состоит в методике записи.

- Автономная внешняя EEPROM: запись слова, запущенная CPU, не может быть прервана его сбросом. Только нарушение питания прервёт процесс записи, поэтому верный подбор разделяющих конденсаторов может обеспечить завершение процесса записи в автономную EEPROM

- Эмулированная EEPROM, использующая встроенную Flash-память: процесс записи, запущенный CPU, может быть прерван как нарушением питания, так и сбросом CPU. Это отличие должно быть проанализировано разработчиками системы для понимания возможных влияний на их приложения и определения приемлемого метода записи.

1.3 Отличие во времени стирания

Отличие во времени стирания является другим важным отличием между автономной внешней и эмулированной EEPROM, использующей встроенную Flash-память. В отличие от Flash-памяти, EEPROM не требует операции стирания для освобождения места перед записью. Это означает, что требуется некоторая форма управления программным обеспечением для хранения данных во Flash-памяти. Более того, поскольку процесс стирания страницы Flash-памяти занимает несколько миллисекунд, отключение памяти и другие нежелательные события могут прервать процесс стирания (к примеру, сброс CPU), что должно учитываться при разработке программного обеспечения, управляющего памятью. Для разработки надёжного программного обеспечения, управляющего Flash-памятью, необходимо глубокое понимание процесса стирания Flash-памяти.

2 Реализация эмуляции EEPROM

2.1 Принцип

Эмуляция EEPROM реализуется различными способами, в зависимости от пределов Flash-памяти и требований продукта. Подход, детализированный ниже, требует минимум две свободные страницы Flash-памяти идентичного размера, выделенного для энергонезависимых данных. Одна из них первоначально стёрта и предлагается для побайтной записи, другая готова занять её место, когда первая будет переполнена устаревшими данными. Поле заголовка, которое занимает первое 16-битное полуслово каждой страницы, отображает статус страницы.

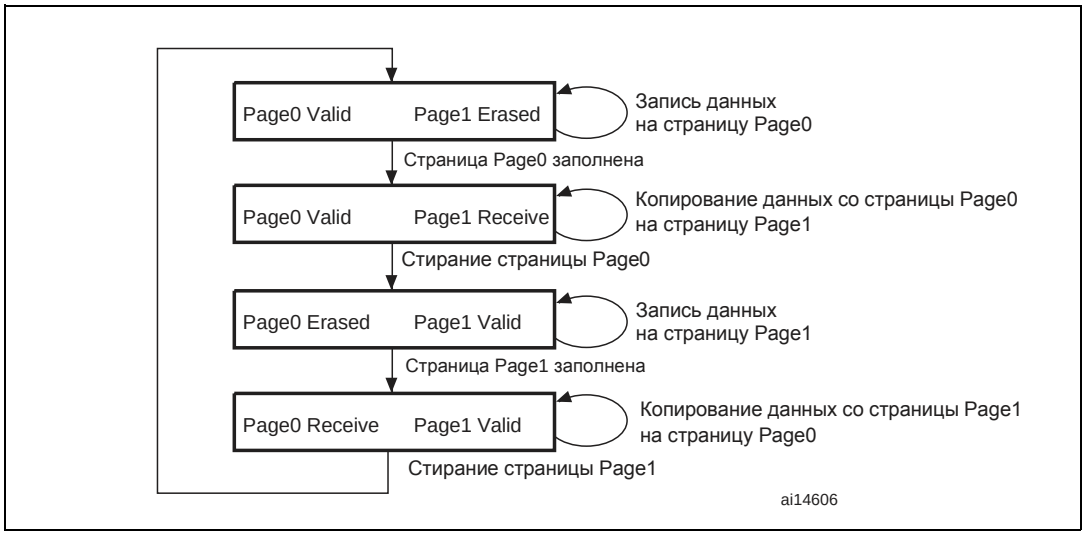
Поле заголовка расположено на начальном адресе каждой страницы и даёт информацию о статусе страницы.

Каждая страница может иметь три возможных статуса:

- **ERASED**: страница пуста.
- **RECEIVE_DATA**: страница получает данные с другой заполненной страницы.
- **VALID_PAGE**: страница содержит валидные данные и это состояние будет неизменным, пока валидные данные не будут перенесены на стёртую страницу.

Рисунок 1 показывает, как статусы страниц изменяются относительно друг друга.

Рисунок 1. Статус заголовка при переключении между страницами page0 и page1

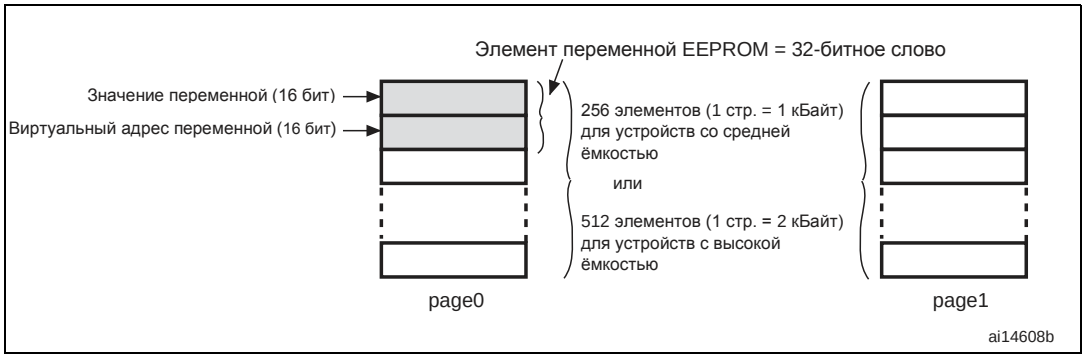


Обычно, при использовании этого метода, пользователь не знает заранее частоту обновления переменной.

Программное обеспечение и реализация, рассмотренная в этом документе, использует две страницы Flash-памяти для эмуляции EEPROM.

Каждый переменный элемент определён виртуальным адресом и значением, которое будет сохранено во Flash-памяти для последующего извлечения или обновления (в реализованном программном обеспечении и виртуальный адрес и данные 16 бит длиной). Когда данные модифицированы, они ассоциируются с новым виртуальным адресом, сохранённым на новое место во Flash-памяти. Поиск данных возвращает модифицированные данные из последнего расположения во Flash-памяти.

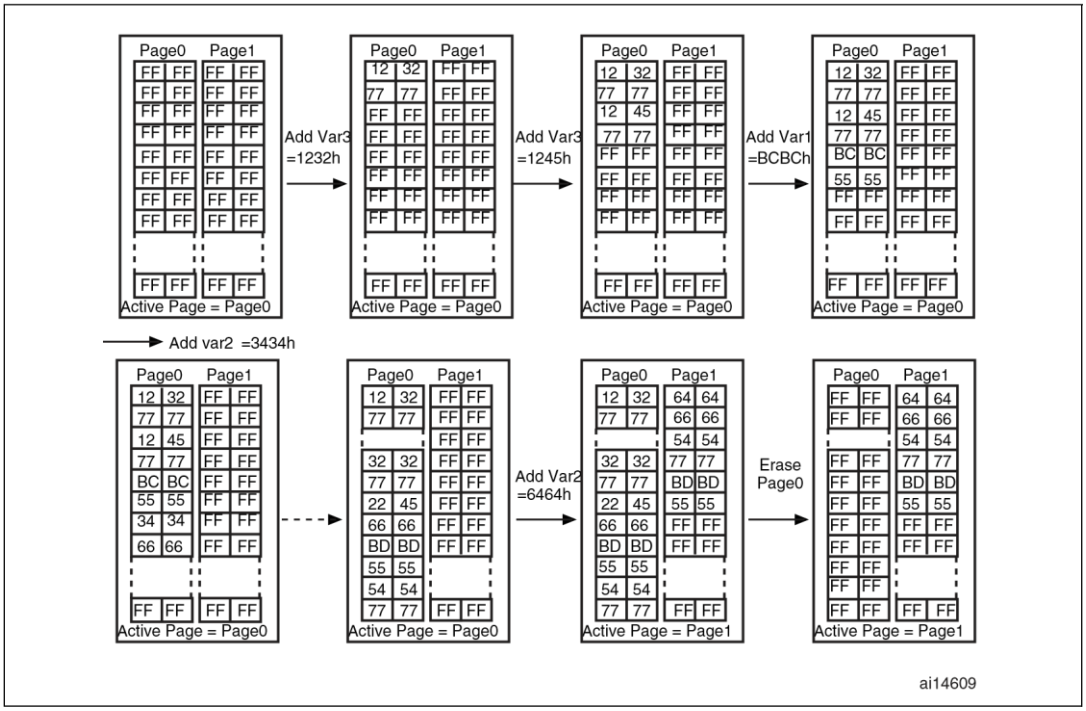
Рисунок 2. Формат переменных EEPROM



2.1.1 Пример применения

Следующий пример иллюстрирует программное управление тремя переменными EEPROM (Var1, Var2 и Var3) со следующими виртуальными адресами: Var1: 5555h, Var2: 6666h and Var3: 7777h

Рисунок 3. Поток обновления данных



2.1.2 Описание программного обеспечения EEPROM

Этот раздел описывает драйвер, разработанный для эмуляции EEPROM с использованием драйвера Flash-памяти семейства STM32F10xxx, предоставляемого STMicroelectronics

Образец демонстрационной программы также представлен для рассмотрения и тестирования драйвера эмуляции EEPROM с использованием трёх переменных Var1, Var2 и Var3, определённых в таблице VirtAddVarTab, объявленной в файле приложения *main.c.*

Проект содержит три исходных файла в дополнение к исходным файлам библиотеки Flash-памяти:

- *eeprom.c*: содержит код на языке C для следующих структур программы:

`EE_Init()`

`EE_Format()`

`EE_FindValidPage()`

`EE_VerifyPageFullWriteVariable()`

`EE_ReadVariable()` `EE_PageTransfer()`

`EE_WriteVariable()`

- *eeprom.h*: содержит прототипы структур и некоторые объявления.
- *main.c*: это программное приложение является примером использования описанных структур для записи и чтения EEPROM.

Пользовательское определение API

Набор функций, содержащийся в файле *eeprom.c,* которые используются для эмуляции EEPROM, описаны ниже:

- `EE_Init()`

В случае пропадания питания при обновлении данных или стирании/переносе сектора возможно повреждение сектора заголовка. В этом случае функция `EE_Init()` будет пытаться восстановить базу данных к последнему рабочему состоянию. Эта функция должна быть вызвана перед доступом к базе данных после каждого пропадания питания. Она не принимает параметров. Процесс описан в *Таблице 2*.

- `EE_Format()`

Эта функция стирает страницы page0 и page1 и записывает значение VALID_PAGE в заголовок страницы page0.

- `EE_FindValidPage()`

Эта функция читает заголовки обеих страниц и возвращает номер страницы с валидными данными. Переданный параметр указывает, если валидная страница занята операцией чтения либо записи

(READ_FROM_VALID_PAGE или WRITE_IN_VALID_PAGE).

- `EE_VerifyPageFullWriteVariable()`

Здесь реализован процесс записи, который также должен обновить либо создать первое значение переменной. Процесс заключается в поиске первого пустого места на активной странице, и заполнение его (заполнение начинается с конца) полученным виртуальным адресом и значением переменной. В случае, если активная страница переполнена, возвращается значение PAGE_FULL. Эта структура использует следующие параметры:

– Виртуальный адрес: может быть любой из виртуальных адресов трех заявленных переменных (Var1, Var2 or Var3)

– Данные: значение переменной, которое должно быть сохранено

Эта функция возвращает FLASH_COMPLETE при успехе, PAGE_FULL в случае нехватки памяти для обновления переменной, или код ошибки Flash-памяти для индикации сбоя операции (стирания либо записи).

- `EE_ReadVariable()`

Эта функция возвращает данные, соответствующие виртуальному адресу, переданному ей в качестве параметра. Считывается только последнее обновлённое значение. Функция входит в цикл, в котором она читает все значения переменной до тех пор, пока не дойдёт до последнего. Если не найдено ни одного значения, переменная ReadStatus возвращается со значением “1”, в противном случае она сбрасывается в нуль для индикации того, что переменная была найдена и её значение возвращено в качестве переменной Read_data.

- `EE_PageTransfer()`

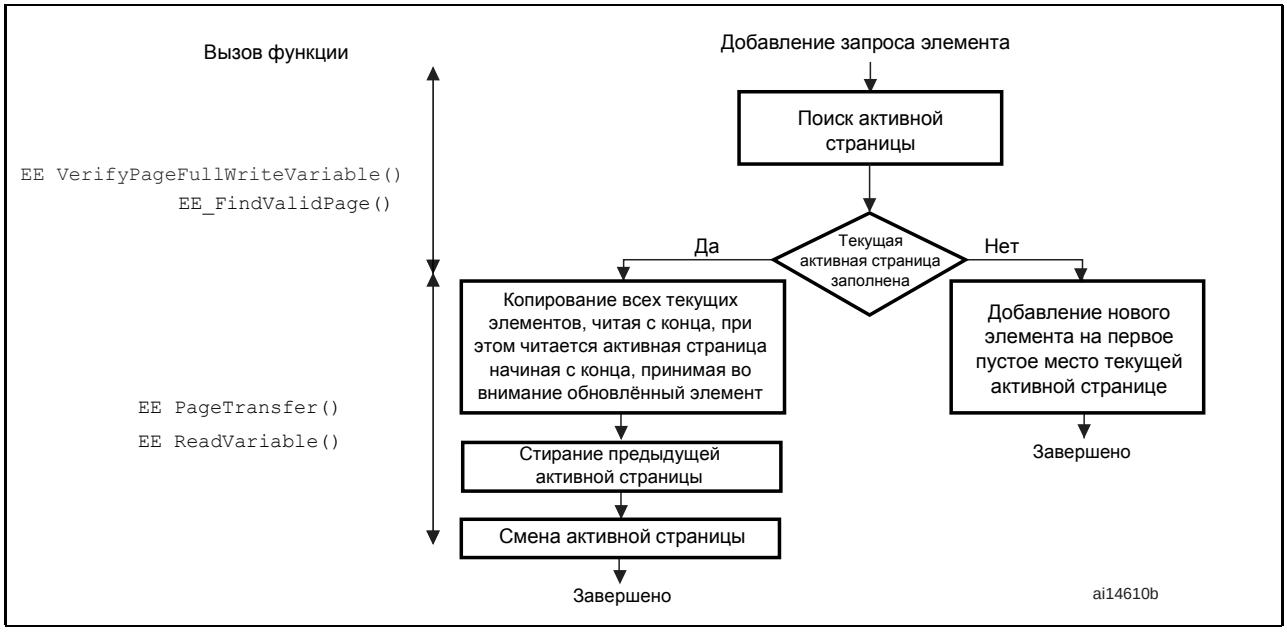
Эта функция перемещает наиболее новые данные (последние обновления переменных) с переполненной страницы на пустую. Для начала происходит определение активной страницы, той, откуда будут перемещаться. Поле заголовка новой страницы определяется и записывается (присваивается новый статус страницы RECEIVE_DATA, поскольку происходит процесс получения данных). Когда перемещение данных закончено, в статус заголовка записывается значение VALID_PAGE, старая страница стирается и её заголовок приобретает статус ERASED.

- `EE_WriteVariable(..)`

Эта функция вызывается пользовательским приложением для обновления переменной. Она использует структуры `EE_VerifyPageFullWriteVariable()` и `EE_PageTransfer()`, которые уже описывались выше.

Рисунок 4 демонстрирует процедуру обновления значения переменной в EEPROM.

Рисунок 4. Блок-схема записи переменной



Ключевые особенности:

- Конфигурируемый пользователем объём эмулированной EEPROM
- Уменьшение износа Flash-памяти: страница стирается только при переполнении
- Энергонезависимые переменные могут обновляться нерегулярно
- Возможно обслуживание прерывания во время записи/стирания

3 Аспекты встраиваемого приложения

Этот раздел посвящён некоторым рекомендациям относительно того, как преодолеть программные ограничения во встраиваемых приложениях и выполнить потребности различных приложений.

3.1 Управление дискретностью обновления данных

Эмулированная EEPROM может использоваться во встроенных приложениях, где энергонезависимые данные должны обновляться с дискретностью в байт, полуслово или слово. Обычно это зависит от требований пользователя и архитектуры Flash-памяти, таких, как длина сохранённых данных, доступ к записи и т.д.

Flash-память, встроенная в микроконтроллеры семейства STM32F10xxx поддерживает 16-ти битную, запись, запись полуслова. Однако данные могут записываться побайтно или пословно с использованием некоторых программных приёмов.

3.1.1 Основы пословной записи

Драйвер Flash-памяти предоставляет функцию `FLASH_ProgramWord(VarAddress, VarData)`, которая будет записывать 32-х разрядные данные “VarData” в желаемый адрес Flash-памяти “VarAddress”. С помощью этой функции возможна пословная запись в определённую область Flash-памяти.

3.1.2 Основы побайтной записи

Побайтная запись позволяет пользователю сохранять большее число переменных. Однако производительность при этом может быть уменьшена.

Используя функцию `FLASH_ProgramHalfWord()`, можно записать в одно полуслово одновременно и значение переменной, и её адрес.

3.2 Выравнивание степени износа как улучшение износостойкости Flash-памяти

Каждая страница Flash-памяти, встроенной в микроконтроллеры семейства STM32F10xxx, может быть надёжно стёрта либо записана 10 000 раз.

Для активно перезаписывающих приложений, которые используют более двух страниц (3 или 4) для эмулированной EEPROM, рекомендуется реализовать снижающий износ алгоритм для мониторинга и распределения числа циклов записи между страницами.

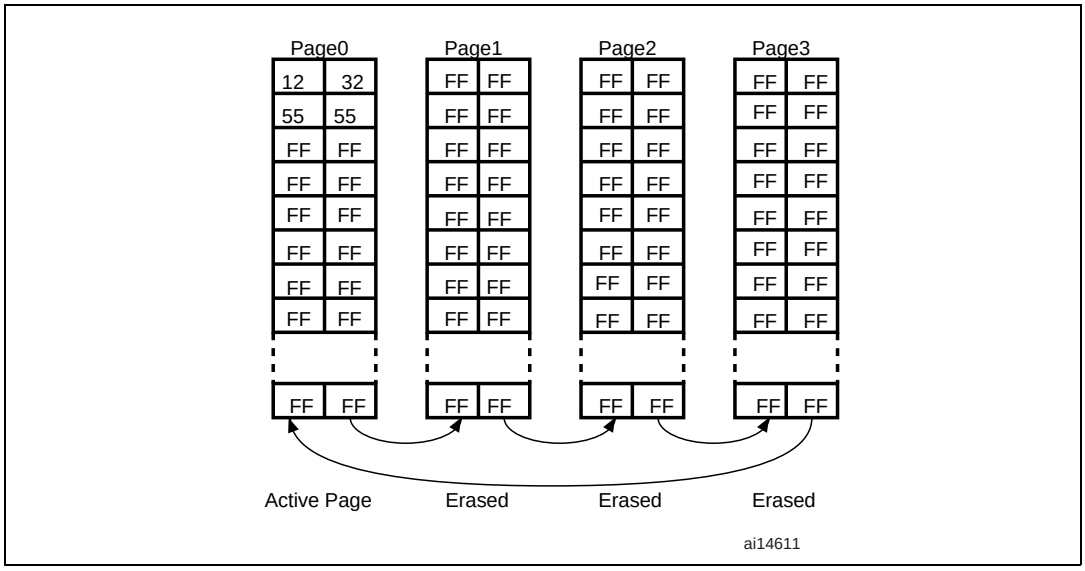
Когда не используется алгоритм, страницы не используются с равной интенсивностью. Страницы с долгоживущими данными не проходят через такое же число циклов перезаписи, как страницы с часто обновляемыми данными. Уравнивающий износ алгоритм обеспечивает, что каждый сектор используется одинаково на протяжении всех циклов перезаписи.

3.2.1 Пример реализации уравнивания износа

В этом примере с целью увеличения ёмкости эмулированной EEPROM будет использовано 4 страницы Flash-памяти (Page0, Page1, Page2 and Page3).

Уравнивающий износ алгоритм реализуется следующим образом: когда страница n заполнена, устройство переключается на страницу n+1. Страница n, как хранящая устаревшие данные, стирается. Когда процесс доходит до того, что заполняется страница Page3, устройство возвращается к странице Page0, а страница Page3, как хранящая устаревшие данные, стирается и т.д. (см. *Рисунок 5*).

Рисунок 5. Схема чередования для четырёх страниц (уравнивание износа)



В приложении алгоритм уравнивания износа может быть реализован с использованием функции `EE_FindValidPage()`.

3.3 Восстановление заголовка страницы в случае пропадания питания

При пропадании питания в момент обновления переменной, стирания или перемещения страницы возможна потеря данных или повреждение заголовка страницы.

Для определения такого повреждения и восстановления после него реализована структура *`EE_Init()`*. Она должна быть вызвана незамедлительно после включения питания. Принцип структуры описан в этом руководстве. Структура использует статус страницы для проверки на целостность и, при необходимости, для восстановления.

После потери питания структура *`EE_Init()`* используется для проверки статуса страницы. Возможны 9 комбинаций статусов страниц, три из них ошибочны. *Таблица 2* иллюстрирует действия, которые должны быть предприняты на основании статусов страниц после восстановления питания.

Таблица 2. Комбинации статусов и необходимые действия

Статус страницы Page1	Статус страницы Page0		
	ERASED	**RECEIVE_DATA**	**VALID_PAGE**
ERASED	Ошибочное состояние. Стереть обе страницы и отформатировать страницу Page0	Стереть страницу Page1 и отметить страницу Page0 как VALID_PAGE	Использовать страницу Page0 как активную, стереть страницу Page1
RECEIVE_DATA	Стереть страницу Page0 и отметить страницу Page1 как VALID_PAGE	Ошибочное состояние. Стереть обе страницы и отформатировать страницу Page0	Использовать страницу Page0 как активную, переместить последние обновлённые переменные со страницы Page0 на страницу Page1, отметить страницу Page1 как активную, стереть страницу Page0
VALID_PAGE	Использовать страницу Page1 как активную, стереть страницу Page0	Использовать страницу Page1 как активную, переместить последние обновлённые переменные со страницы Page1 на страницу Page0, отметить страницу Page0 как активную, стереть страницу Page1	Ошибочное состояние. Стереть обе страницы и отформатировать страницу Page0

3.4 Возможное число циклов

Цикл записи/стирания состоит из одной или более операций записи и одного стирания страницы.

Когда используется технология EEPROM, каждый байт может быть записан и стёрт конечное число раз, обычно в пределах от 10 000 до 100 000.

Однако во встроенной Flash-памяти минимальный стираемый объём – это страница, и количество циклов применительно к странице это возможное число циклов стирания. Электрические характеристики семейства STM32F10xxx гарантируют 10 000 циклов записи/стирания для страницы. Таким образом, максимальное время жизни эмулированного EEPROM ограничено частотой обновления наиболее часто перезаписываемого параметра.

Возможное число циклов обусловлено суммой/размером данных, которыми пользователь хочет управлять.

В этом примере две страницы (1 кБайт для устройств средней ёмкости или 2 кБайт для подключаемых устройств или устройств высокой ёмкости) используются для записи 16-битных данных. Каждой переменной соответствует 16-битный виртуальный адрес. Таким образом, каждая переменная занимает в памяти место в 1 слово. Страница может хранить 1 кБайт (для устройств средней ёмкости) или 2 кБайт (для подключаемых устройств или устройств высокой ёмкости), что при умножении на износостойкость Flash-памяти 10 000 циклов даст 10 000 кБайт для устройств средней ёмкости) или 20 000 кБайт (для подключаемых устройств или устройств высокой ёмкости) ёмкости записанных данных за время жизни одной страницы встроенной Flash-памяти. Следовательно, при использовании в процессе эмуляции двух страниц в эмулированной EEPROM может быть сохранено 20 000 кБайт (для устройств средней ёмкости) или 40 000 кБайт (для подключаемых устройств или устройств высокой ёмкости). Если используется более чем две страницы, соответственно умножается и объём данных.

Зная размерность сохранённой переменной, возможно рассчитать общее число переменных, которые могут быть сохранены в пределах эмулированной EEPROM за время её службы.

Таблица 3 даёт общее представление о количестве переменных, которые могут быть сохранены в эмулированном EEPROM согласно виртуальному адресу и размерам данных.

Таблица 3. Максимальное число переменных, сохранённых в EEPROM (из расчёта на 10 000 циклов перезаписи) (1) (2)

Размерность переменных	Две страницы по 1 кБайт	Две страницы по 2 кБайт	Единица
8-битная переменная (с 8-битным виртуальным адресом)	10 000 × (2^{10} – 2)	10 000 × (2^{11} – 2)	Переменная
16-битная переменная (с 16-битным виртуальным адресом)	5000 × (2^{10} – 4)	5000 × (2^{11} – 4)	Переменная
32-битная переменная (с 32-битным виртуальным адресом)	2500 × (2^{10} – 8)	2500 × (2^{11} – 8)	Переменная

1. Здесь максимальное число переменных не включает соответствующий им виртуальный адрес.

2. Значение рассчитано исходя из максимального количества переменных, которые могут быть записаны, используя две страницы, учитывая расположенное в начале страницы поле статуса. В зависимости от размерности переменных и, чтобы сохранить выравнивание, некоторые пустые байты добавлены к статусу страницы. Эти байты также вычтены из этого максимального количества.

4 История версий

Таблица 4. История версий документа

Дата	Версия	Изменения
05.10.2007	1	Первоначальное издание.
24.06.2008	2	Документ обновлён для поддержки также устройств высокой ёмкости семейства MCU STM32F10xxx. Небольшие текстовые изменения. Циклы стирания/записи добавлены в *Таблица 1: отличия между встроенной Flash-памятью и EEPROM*. Добавлена функция `EE_Init()` под *Пользовательское определение API а странице 9.* *Рисунок 4: Блок-схема записи переменной на странице 9* модифицирована. *Ключевые особенности на странице 10* обновлены. *Раздел 3.3: Восстановление заголовка странице в случае пропадания питания* обновлено. *Таблица 2: Комбинации статусов и необходимые действия* обновлена *Таблица 3. Максимальное число переменных, сохранённых в EEPROM (из расчёта на 10 000 циклов перезаписи)* обновлена, добавлены примечания.
04.08.2009	3	Обновлено для подключаемых устройств.