DOI: 10.18137/RNU.V9187.23.03.P.213 УДК 004.072

И.А. Теремов, М.Р. Проказин

СПОСОБ ОРГАНИЗАЦИИ СТРАНИЧНОЙ ПАМЯТИ В МИКРОЭЛЕКТРОННЫХ УСТРОЙСТВАХ

Аннотация. Предлагается возможный способ организации страничного перепрограммируемого постоянного запоминающего устройства в микроконтроллерной технике. Целью статьи является формирование нового подхода к организации хранения произвольных блоков данных в памяти страничного типа. Применимость данного способа показана экспериментальным методом. В результате исследования сформирован принцип разбиения пространства памяти на сегменты и алгоритмы дефрагментации сегментов.

Ключевые слова: страничная память, микроконтроллер, дефрагментация.

I.A. Teremov, M.R. Prokazin

METHOD OF ORGANIZING PAGE MEMORY IN MICROELECTRONIC DEVICES

Abstract. This article proposes a possible way to organize a page PROM in microcontroller technology. The purpose of the article is to form a new approach to organizing the storage of arbitrary data blocks in page-type memory. The applicability of this method is shown experimentally. As a result of the study, the principle of partitioning the memory space into segments and segment defragmentation algorithms were formed.

Keywords: storage device, microcontroller, defragmentation.

Введение

Окружение человека наполнено множеством микроэлектронных устройств, выполняющих различные функции и во многом облегчающих и вносящих разнообразие в повседневную жизнь и труд человека. Существует множество видов таких устройств: различные датчики показателей окружающей среды, компоненты умного дома, бытовые приборы и др. Основной проблемой данных устройств являются множественные ограничения по физическому размеру, потреблению электроэнергии в условиях портативности и мобильности [1]. Отдельно стоящей проблемой является ограничение по объему и функциональной сложности компонентов, отвечающих за хранение оперативных пользовательских данных, участвующих в процессе эксплуатации устройства. В настоящей статье предлагается возможная реализация организации памяти в подобных устройствах.

Общая организация памяти

Страничный доступ к памяти на чтение и запись является основным принципом практически в любых видах памяти, используемых в микроэлектронике или компактных устройствах [2]. По такому же принципу функционирует память EEPROM (англ. Electrically Erasable Programmable Read-Only Memory – электрически стираемое перепрограммируемое постоянное запоминающее устройство, далее – ППЗУ) [3]. Таким образом, при проектировании структуры памяти размер страницы являлся основной отправной точкой.

При формировании предлагаемой организации памяти в качестве предполагаемого целевого устройства был выбран абстрактный модуль EEPROM размером страницы

Теремов Иван Алексеевич
магистрант, МИРЭА – Российский технологический университет, Москва. Сфера научных интересов: информационные (digital-) технологии, микроэлектроника, цифровая микроэлектроника.
Электронный адрес: ivan.teremow2011@aol.com

Проказин Михаил Русланович
магистрант, МИРЭА – Российский технологический университет, Москва. Сфера научных интересов: информационные (digital-) технологии, микроэлектроника, цифровая микроэлектроника.
Электронный адрес: bearpro@outlook.com

64 байта, а вся память модуля состоит из 512 страниц, обозначенных условными номерами, начиная с нуля.

Для поддержания функционирования модели данных предлагается разбиение всего пространства памяти на сегменты со следующими типами:

• начальная страница – страница с адресом 0, используемая для адресации сегментов других типов страниц;

• страница метаданных – страница, содержащая метаданные блоков данных;

страница данных – страница, содержащая непосредственно хранимые блоки данных.

На Рисунке 1 приведена схема представления всего пространства памяти по типам страниц.

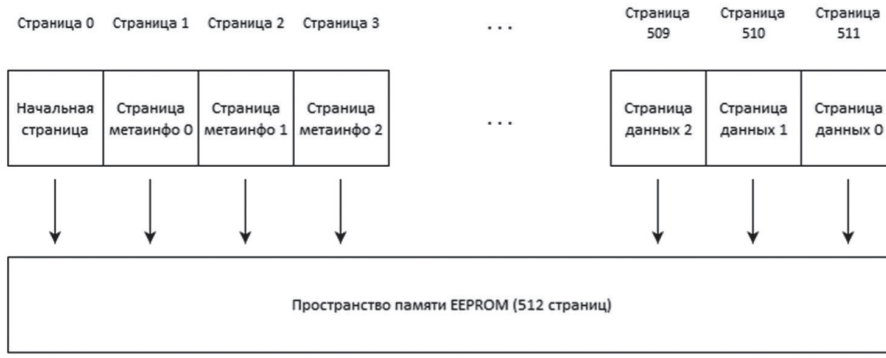

Рисунок 1. Структура памяти хранилища
Здесь и далее рисунки составлены авторами

При этом для любого типа страницы первые 4 байта используются для хранения контрольной суммы CRC32. Хранение подобной суммы необходимо для подтверждения предотвращения ошибок при чтении/записи памяти и подтверждения устойчивого поведения модуля памяти [4]. В общем случае для целей подтверждения сохранности данных в ячейках памяти избыточность составляет 6,25 %.

Операции над памятью в микроконтроллерах обычно выполняются не над отдельными байтами, а над регистрами, длина которых составляет обычно более одного байта [5]. Подобные участки памяти называются машинным словом. Здесь и далее будут применяться термины «полуслово», «слово», «двойное слово» которые означают участки памяти длиной 16, 32 и 64 бит соответственно.

В качестве области для хранения общей информации о состоянии памяти, адресах начала сегментов была использована страница с индексом 0. Побитовое представление ее составляющих представлено на Рисунке 2.

Рисунок 2. Структура начальной страницы

Биты с 0 по 15 слова 1 выделены для хранения количества страниц метаданных, а биты 16 по 31 – для хранения абсолютного сдвига первой страницы метаданных. Два этих поля позволяют полностью указать расположение всего сегмента метаданных.

Сегмент метаданных

Для хранения метаданных, содержащих идентификатор блока, его расположение, используется отдельный сегмент памяти, выделенный под метаданные. Данный сегмент начинается со страницы определенным абсолютным сдвигом, указанным на начальной странице памяти, как и количество страниц в данном сегменте.

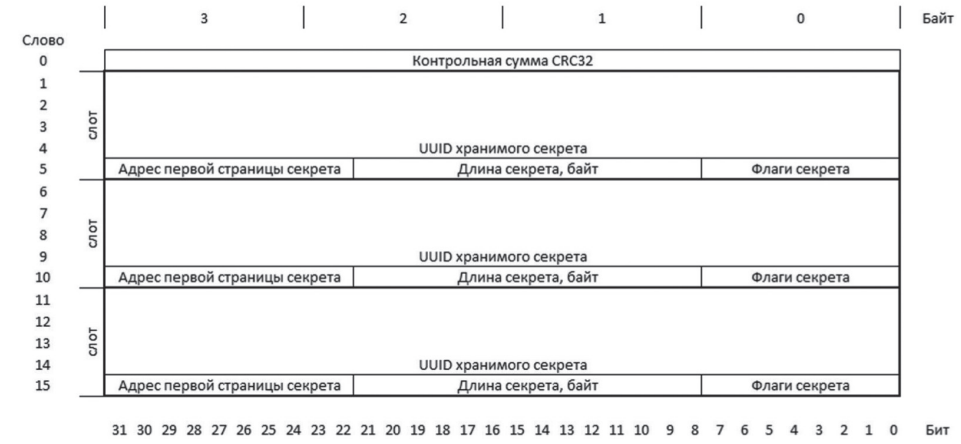

Рисунок 3. Структура страницы метаданных

Как следует из Рисунка 3, на одной странице памяти успешно удалось поместить информацию о трех блоках в основном сегменте. Каждый из участков страницы имеет длину

5 слов, 4 из которых занимает уникальный идентификатор блока. В качестве уникального идентификатора использован стандартизированный формат UUID (universally unique identifier) – популярный стандарт идентификации сущностей в сфере производства программного обеспечения [4]. Данный участок также называется слотом метаданных.

Для хранения адреса первой страницы блока, длины блока в байтах и флагов состояния используется последнее слово участка страницы. Флаги блока описывают его тип и состояние. Идентификатор с бинарным представлением, состоящим только из нулей, является зарезервированным и не участвует в процессе идентификации блока.

Сегмент данных

Страницы, выделенные под хранение непосредственно пользовательских данных, обладают простой структурой, пример которой представлен на Рисунке 4.

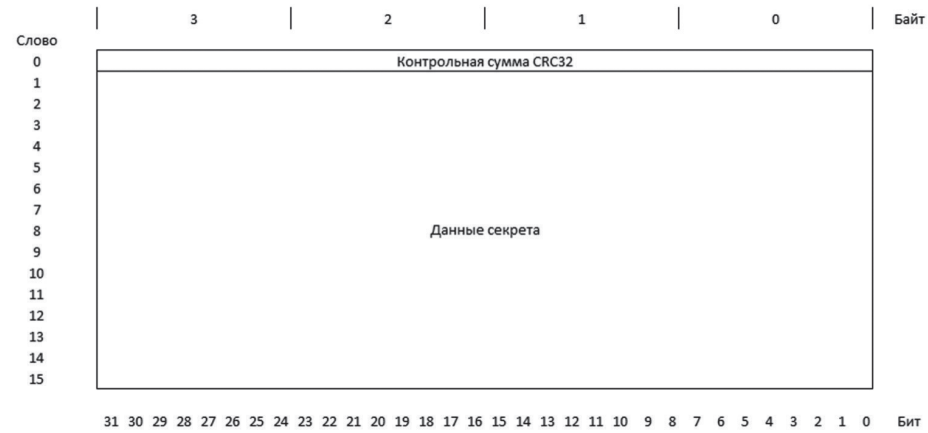

Рисунок 4. Структура страницы данных

Единственным отличием этого сегмента является обратная нумерация страниц, которая начинается с конца всего участка памяти. Это необходимо для более эффективного добавления и удаления блоков и отвечающих за них слотов в сегменте метаданных при динамическом изменении их количества.

Процесс дефрагментации при операциях перезаписи блоков

Процесс эксплуатации рассматриваемого хранилища данных предполагает операции чтения, записи и удаления блоков данных. В отличие от чтения запись и удаление блоков приводят к изменениям состояния хранилища. Так как блоки памяти используют страницы строго последовательно, при оперировании блоками различной длины хранилище будет накапливать дефрагментацию.

В процессе эксплуатации предлагаемой структуры данных предполагается также поддержание специальной битовой модели, описывающей текущее использование страниц памяти. Используемая структура для хранения метаданных соответствует размеру одной физической страницы, что не допускает дробного использования таких страниц, и даже если страница не заполнена блоком данных до конца, она не разделяется между другими блоками.

Обозначенные выше свойства структуры памяти позволяют выделить свойство атомарности физической страницы: она может быть либо использована, либо свободна. Частичного или вероятного использования не предусматривается.

Таким образом, для описания состояния каждой страницы памяти достаточно одного бита. Это позволяет уместить такую модель максимально компактно: для рассматриваемого прототипа с 512 страницами достаточно 64 байта.

На Рисунке 5 представлено битовое отображение такой модели, состоящей из массива 16 слов.

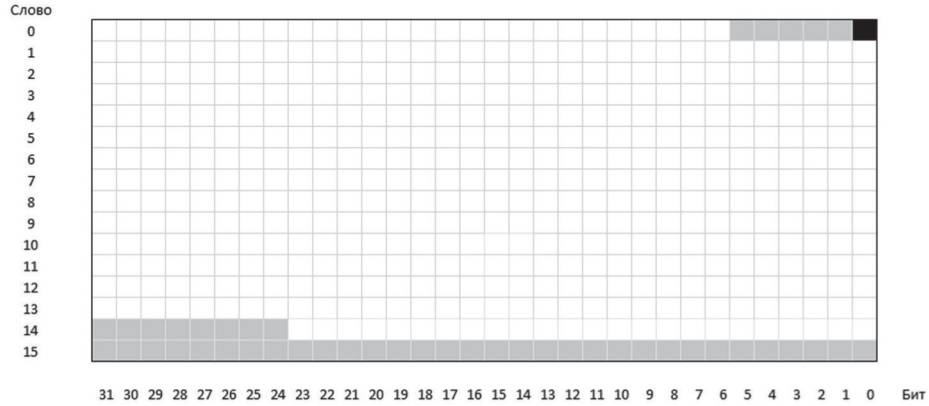

Рисунок 5. Битовое отображение модели

В приведенной иллюстрации использованная страница обозначена слабым выделением, пустая страница не выделена (белая заливка), а первая страница всего модуля, предназначенная для сохранения общих метаданных о структуре сегментов в модуле памяти, обозначена черной заливкой. Данная страница используется всегда, в связи с чем она исключена из процесса моделирования.

В приведенном примере в памяти уже записано некоторое количество блоков, а также соответствующих им страниц метаданных. Определить индекс слова N и номер бита M в слове, соответствующем определенной странице P, можно по формулам:

$$N = P \ div 32 ; \quad M = P \ mod 32 .$$

Обратная операция может быть выполнена по формуле

$$P = N \times 32 + M .$$

В случае получения команды на добавление нового блока микроконтроллер, поддерживающий модель, проверяет наличие достаточного места для размещения данных и метаданных этого блока. В случае отсутствия достаточного места команда будет отклонена. При наличии необходимого количества свободных страниц/слотов метаданных компонент выберет по модели первую доступную область памяти, запишет необходимые данные и обновит описанную модель.

На Рисунке 6 представлено целевое состояние модели памяти, которая претерпела несколько операций удаления блоков. Страницы с блоками были стерты, соответствующие слоты метаданных были также помечены как неактивные и освобождены от идентификаторов. В данном случае страницы метаданных, содержащие неиспользуемые слоты, на рисунке отмечены слабым выделением. Последовательность страниц данных содержит промежутки свободных страниц между заполненных. Для удобства восприятия обозначены также блоки с условными номерами, отображенные на этих страницах. Длина блоков в используемых страницах записана в скобках после условного номера блока.

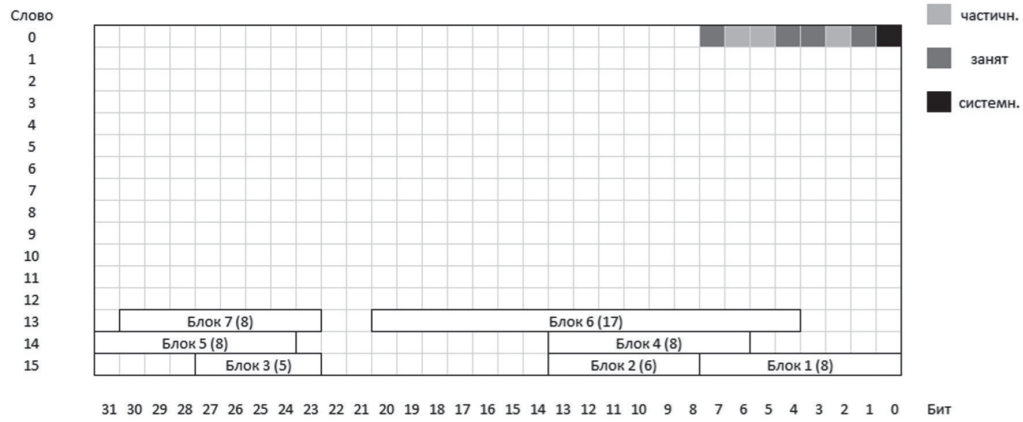

Рисунок 6. Состояние модели памяти после удаления нескольких блоков

При последующей эксплуатации системы остальная область памяти будет также заполняться разреженно за счет последовательного удаления и записи новых блоков. Это же актуально и для сегмента метаданных, однако в данном случае промежутки будут на уровне слотов, а не страниц.

Дефрагментация сегмента метаданных

Для обеспечения компактности хранимых метаданных на устройстве в компоненте управления блоков предлагается алгоритм дефрагментации данного сегмента. На Рисунке 7 проиллюстрирован пример состояния слотов метаданных участка метаданных из 6 страниц, а также процесс применения алгоритма дефрагментации. В данном случае выделением показаны используемые слоты, пустые ячейки обозначают свободные слоты.

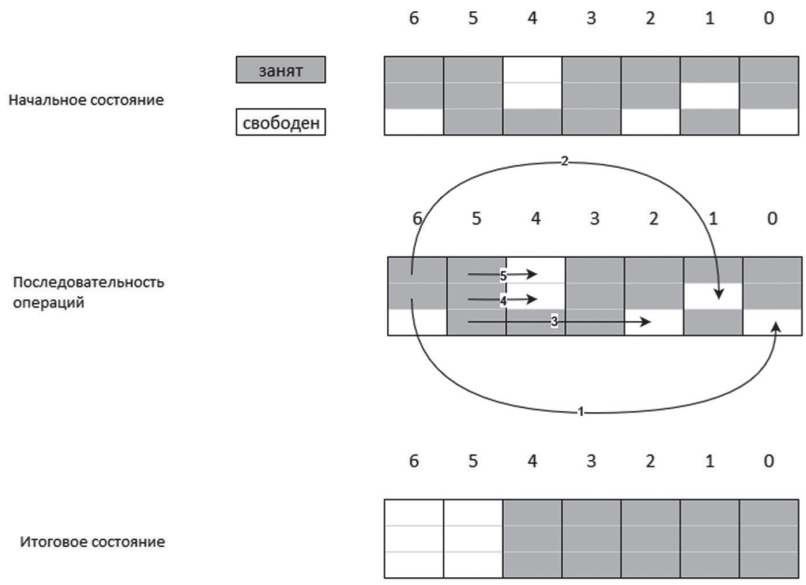

Рисунок 7. Иллюстрация процесса дефрагментации слотов метаданных

Алгоритм дефрагментации заключается в последовательном прохождении через каждый слот, начиная с начала, и перезаписи в пустые слоты метаданных – с конца сегмента. На Рисунке 7 проиллюстрирована последовательность и направление перемещения слотов метаданных для представленного ранее примера. Номер на стрелке указывает порядок и направление выполняемых перемещений.

В конечном итоге в процессе сжатия памяти было освобождено 2 страницы памяти, которые могут быть использованы для записи блока данных.

Дефрагментация сегмента данных

Ввиду того что последовательность страниц/слотов в сегменте метаданных не имеет значения, так как блок данных идентифицируется глобальным идентификатором UUID, их представление является коммутативным.

Для страниц памяти такое правило может быть применено исключительно в случае, если выполняется синхронизация также с соответствующими слотами метаданных. Таким образом, алгоритм дефрагментации сегмента данных схож с алгоритмом дефрагментации метаданных. Однако важной особенностью является переменная длина свободных/занятых участков дефрагментируемых данных.

Первым шагом реализованного алгоритма является формирование справочника длин всех известных участков данных. Данный справочник кандидатов будет использоваться для поиска фрагмента подходящего множества блоков данных для вставки в промежутки.

Следующим шагом является последовательный поиск пустых фрагментов в сегменте, начиная с конца. Когда такой сегмент найден, по составленному на предыдущем шаге справочнику выполняется поиск подходящего участка или комбинации участков, сумма длин которых в точности равна длине пустого промежутка. Если поиск увенчался успехом, выбранный участок перемещается в промежуток и исключается из справочника. Если подходящего участка не нашлось, выбирается замыкающий с противоположной стороны участок записанных данных и перемещается в промежуток вплотную с предыдущим участком. После этого перемещенный участок также исключается из справочника.

Процесс дефрагментации завершается, когда справочник кандидатов не содержит ни одного участка.

Рассмотрим пример дефрагментации области данных, указанный ранее на Рисунке 6. В результате применения алгоритма дефрагментации сегмент данных примет вид, показанный на Рисунке 8.

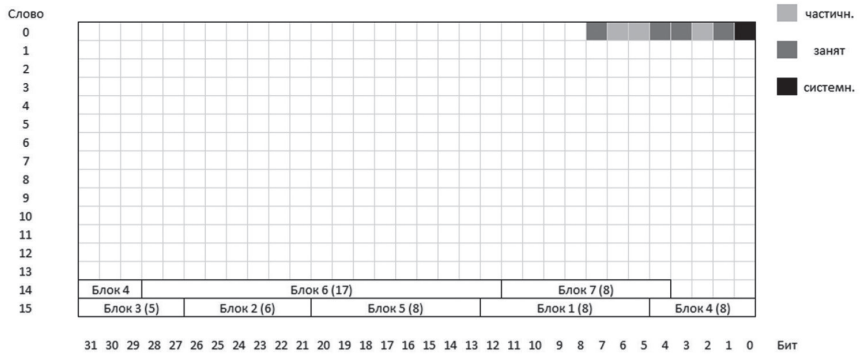

Рисунок 8. Дефрагментированный сегмент данных

Как было сказано выше, в процессе дефрагментации адреса блоков меняются, что требует обязательного обновления их в соответствующих слотах метаданных.

Оба этапа дефрагментации хранилища данных выполняются одновременно по требованию приложения, выполняемого на микроконтроллере.

Заключение

Предлагаемый подход к организации хранения данных и управления памятью позволяет использовать устройства хранения данных ограниченного объема во встраиваемой микропроцессорной технике, предоставляя оптимизации, максимизирующие долю эффективных данных. Предлагаемый алгоритм дефрагментации позволяет обслуживать хранилище, обеспечивая возможность записи новых блоков данных после удаления старых. Данная структура может быть также доработана для использования методов защиты данных в приложениях, к которым предъявляются требования обеспечения безопасности оперируемых данных.

Литература

1. *Симмондс К.* Встраиваемые системы на основе Linux. Бирмингем: Packt Publishing Ltd, 2014. 750 с.
2. *Берикашвили В.Ш.* Электроника и микроэлектроника: импульсная и цифровая электроника. М.: Юрайт, 2018. 242 с.
3. *Гусев В.Г., Гусев Ю.М.* Электроника и микропроцессорная техника: учебник для вузов. 5-е изд. М.: КноРус, 2013. 797 с.
4. *Угрюмов Е.П.* Цифровая схемотехника. Санкт-Петербург: БХВ-Петербург, 2007. 800 с.
5. *Денищенко Н.* Устройство и криптоанализ UUID-генератора в ОС Windows // RSDN Magazine: электронный журнал. URL: http://rsdn.ru/article/Crypto/UuidCrypto.xml (дата обращения: 20.08.2023).

Literature

1. Simmonds K. (2014) *Vstraivaemye sistemy na osnove Linux* [Embedded systems based on Linux]. Birmingham: Packt Publishing Ltd, 2014, 750 p. (in Russian).
2. Berikashvili V.Sh. (2018) *Jelektronika i mikrojelektronika: impul'snaja i cifrovaja jelektronika* [Electronics and microelectronics: pulsed and digital electronics]. Moscow: Yurayt Publishing, 2018, 242 p. (in Russian).
3. Gusev V.G., Gusev Yu.M. (2013) *Jelektronika i mikroprocessornaja tehnika: uchebnik dlja vuzov* [Electronics and microprocessor technology: textbook for universities]. Moscow: KnoRus Publishing, 2013, 797 p. (in Russian).
4. Ugryumov E.P. (2007) *Cifrovaja shemotehnika* [Digital circuitry]. St. Petersburg: BHV-Petersburg Publishing, 2007, 800 p. (in Russian).
5. Denishchenko N. (2008) Design and cryptanalysis of the UUID generator in Windows OS. *RSDN Magazine: electronic magazine*. Available at: http://rsdn.ru/article/Crypto/UuidCrypto.xml (accessed: 20.08.2023) (in Russian).