Программирование RISC микроконтроллеров семейства AVR при создании устройств записи/воспроизведения звука

Теймураз Кивиладзе[1], Дмитрий Раквиашвили[2]

[1] ТГУ, Факультет Прикладной Математики и Компьютерных Наук.
[2] Грузинская Национальная Комиссия по Коммуникациям.

Аннотация:

В статье рассмотрены вопросы записи и воспроизведения звуковой информации. Разработано устройство и системное программное обеспечение для записи и воспризведения звука. Устройство разработано на базе микроконтроллера производства ATMEL CORP AT90S8535. Для хранения оцифрованных звуковых данных используется модуль энергонезависимой памяти DataFlash AT45DB161.

Ключевые слова: *программирование, микроконтроллер, AVR, Data Flash, запись звука, АЦП, оцифровка.*

Введение

Термин "микроконтроллер" обычно означает отдельную микросхему, содержащую процессорное ядро и все необходимые периферийные устройства на одном кристалле для того, чтобы реализовать специализированный микрокомпьютер для задач контроля / управления. Современные микроконтроллеры, как правило, содержат целый арсенал развитых цифровых и аналоговых периферийных блоков и модулей. Некоторые из них имеют обширный набор инструкций и относительно медленно выполняют помещенные в них программы (обычно с архитектурой **CISC** - Complex Instruction Set Computers). Другие работают очень быстро (обычно с архитектурой **RISC** - Reduced Instruction Set Computers), но имеют ограниченный набор исполняемых команд. Массовое производство микроконтроллеров привело к их значительному удешевлению и, как следствие, к широчайшему использованию в разнообразном промышленном и бытовом оборудовании, особенно в 4-разрядном и 8-разрядном исполнении. Ассортимент предлагаемых микроконтроллеров на мировом рынке постоянно растет, появляются новые, более совершенные и технологичные изделия повышенной степени интеграции, новые полупроводниковые структуры, новые идеологические решения. Количество фирм-производителей также неуклонно растет при одновременном повышении их уровня технической и технологической оснащенности[1].

Корпорация **ATMEL** (США) хорошо известна на мировом рынке электронных компонентов и является одним из признанных мировых лидеров в разработке и производстве сложных изделий современной микроэлектроники - устройств энергонезависимой памяти высокого быстродействия и минимального удельного энергопотребления, микроконтроллеров общего назначения и микросхем программируемой логики. Сейчас **ATMEL** удерживает первое место в мире по производству микросхем параллельной и последовательной EEPROM, лидирует в производстве Flash - микроконтроллеров общего назначения и входит в первую пятерку по производству EPROM, микросхем Flash – памяти[2].

В данной статье описывается электронная схема и программное обеспечение цифрового устройства записи/воспроизведения речи, рализованного на базе микроконтоллера **AVR** и Flash памяти производства **ATMEL CORP**. Затронуты такие аспекты как : использование Data Flash для хранения обработанного с помощью АЦП звука, использование последовательного периферийного интерфейса (SPI) для доступа к внешней памяти и широтно-импульсной модуляции (ШИМ) для воспроизведения.

Основные характеристики устройства

- 8-битная запись звука
- Частота дискретизации 8 кГц
- Частота звука до 4 кГц
- Максимальное время записи 4,25 минуты

1. Принцип действия

Аналоговый сигнал, поступивший с микрофона усиливается и нормализуется. Микроконтроллер AVR AT90S8535 используется для съёма выборок аналогового сигнала с усилителя и их последующей оцифровки. Встроенный SPI управляет передачей данных в/из DataFlash. Функция ШИМ используется для воспроизведения.

Полученные даннные хранятся в DataFlash типа AT45DB161, которая является флэш-памятью, объемом 16 Мбит. AT45DB161 использует последовательный периферийный интерфейс(SPI) для доступа к данным. Этот интерфейс облегчает подключение аппаратных средств, повышает надёжность системы, уменьшает помехи переключения, а также позволяет уменьшить габаритные размеры и количество задействованных выводов ИМС. Подобные DataFlash обычно применяется для хранения изображений, данных и оцифрованных голосовых сообщений.

Перед сохранением аналогового сигнала речи в DataFlash, он должен быть оцифрован. Это делается за несколько шагов.

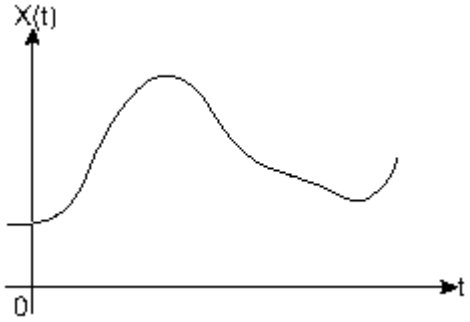

Рисунок 1. Пример аналогового сигнала

Во-первых, аналоговый сигнал (Рисунок 1) преобразуется в сигнал разделённый по времени, посредством периодической выборки (Рисунок 2). Временной интервал между двумя выборками называется «периодом выборки», а его обратная величина называется «частотой дискретизации». Согласно теореме о дискретном представлении, частота дискретизации должна быть, по крайней мере, в два раза больше частоты сигнала. В противном случае периодическое продолжение сигнала в частотной области приведёт к спектральному перекрытию, называемому «наложением спектров». Такой сигнал с наложением спектра не может быть однозначно восстановлен из выборки[3].

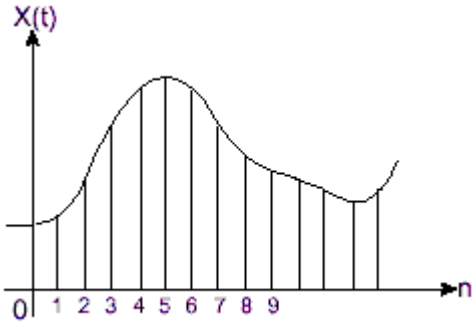

Рисунок 2. Временная дискретизация сигнала

Основную информацию речевого сигнала несут частоты до 3000 Гц. Поэтому для ограничения полосы (частот) сигнала может быть использован фильтр нижних частот.

Для идеального фильтра нижних частот с частотой отсечки 3000 Гц, частота выборки должна быть 6000 Гц. В зависимости от фильтра меняется его крутизна. Особенно важно выбрать наибольшую частоту дискретизации для фильтра первого порядка, такого как RC-фильтр, используемый в этом примере. Верхний предел ограничивается возможностями АЦП.

«Квантованием» называется определение цифровых значений, соответствующих аналоговым выборкам, взятым на частоте дискретизации. Аналоговый сигнал квантуется путём приписывания аналоговой величине ближайшего «допустимого» цифрового значения (Рисунок 3). Количество цифровых значений называется «разрешением» и всегда ограничивается. Например, 256 значений для 8-битного цифрового сигнала или 10 значений в этом примере. Поэтому квантование аналоговых сигналов всегда приводит к потере информации. Эта «ошибка квантования» обратно пропорциональна разрешению цифрового сигнала. Она также обратно пропорциональна «динамическому диапазону» сигнала, т.е. интервалу между минимальным и максимальным значениями [3]. АЦП микроконтроллера AT90S8535 может быть настроен на динамический диапазон сигнала, если на AGND и AREF подать соответственно минимальное и максимальное значение сигнала[4].

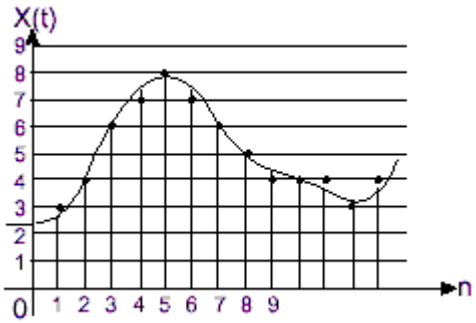

Рисунок 3. Квантованный сигнал

С другой стороны, усилитель микрофона может быть настроен так, что он будет перекрывать динамический диапазон АЦП, как показано ниже. Оба метода снижают ошибку квантования. К тому же, последний метод увеличивает отношение сигнал/шум (SNR) и может быть выделен хотя бы поэтому.

На рисунке 4 показаны цифровые значения, которые соответствуют аналоговому сигналу. Эти значения сняты с выхода АЦП.

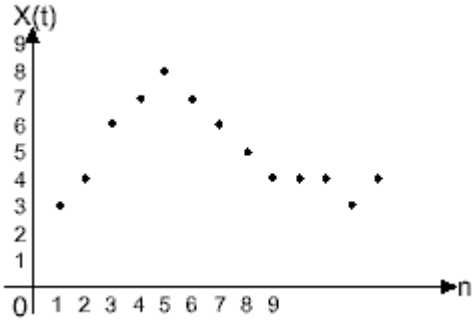

Рисунок 4. Цифровой сигнал

В этом примере сигнал имеет минимальное и максимальное значения, которые никогда не превышают предела. Части сигнала ниже минимального и выше максимального значений не содержат информации. Они должны быть удалены, чтобы не «забивать» память. Это делается путём сдвига вниз всего сигнала и отбраковкой частей превышающих максимальное значение «max» (Рисунок 5) [3].

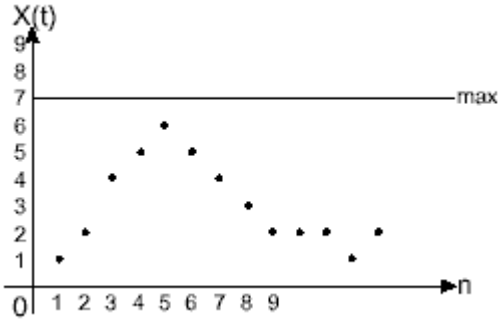

Рисунок 5. Сокращённый цифровой сигнал

Окончательный сигнала состоит из 8 бит. Теперь он может быть сохранён в DataFlash. DataFlash не требует отдельного цикла стирания перед программированием. При использовании команд «Буфер в основную страницу памяти программы с встроенным стиранием» и «Основная страница памяти программы сквозь буфер», DataFlash будет автоматически стирать определённую страницу в массиве памяти перед программированием действительных данных. Если система требует большую программную пропускную способность (больше 200 Kbps), то области массива основной памяти могут быть предварительно очищены, для уменьшения суммарного программного времени. Дополнительная команда «Очистка страницы» предназначена для стирания отдельной страницы памяти, в то время как команда «Очистка блока» позволяет очистить одновременно 8 страниц памяти. При предварительной очистке части главного массива памяти, для уменьшения общего времени, может использоваться команда «Буфер в основную страницу памяти программы без встроенного стирания».

Первый метод более эффективен в отношении записи программного кода, так как не применяются дополнительные циклы стирания. Однако в этом примере используется очистка блока для того, чтобы показать как большая часть памяти может быть очищена, если это потребуется. Очистка всей памяти может занять несколько секунд. После очистки памяти данные могут записываться до тех пор, пока не заполнятся все страницы[2].

Для записи в DataFlash используется буфер 1. Когда этот буфер заполнится (528 выборками), он записывается в память во время 529 преобразования. Данные записываются до тех пор, пока нажата кнопка «Запись» или память не заполнилась. Если вся память заполнена, то новые данные не могут быть записаны, пока не очищена DataFlash. Если память заполнена лишь частично, то при повторном нажатии кнопки «Запись» новые данные будут добавлены сразу за уже записанными данными[4].

Воспроизведение звука всегда начинается с начала DataFlash. Оно прекращается, если все записанные данные воспроизведены или когда кнопка «Воспроизведение» отпущена. DataFlash позволяет проигрывать данные либо напрямую из основной страницы памяти, либо путём копирования страницы в один из двух буферов и последующим чтением из буфера. Метод прямого доступа не подходит для этого примера, так как это метод двухадресный (один адрес для страницы, другой для позиции байта), и, следовательно, в DataFlash должна быть отправлена длинная загрузочная последовательность для каждого отдельного байта. Это занимает больше одного цикла ШИМ, который длится 510 тактовых импульсов для 8-битного ШИМ сигнала[2].

Поэтому, одна страница памяти копируется в один из двух буферов. Пока данные читаются из этого буфера, следующая страница памяти копируется в другой буфер. Когда все данные считаны из первого буфера, чтение продолжается из другого буфера, в это время первый буфер перезагружается. Чтение данных из буфера DataFlash синхронизируется частотой ШИМ.

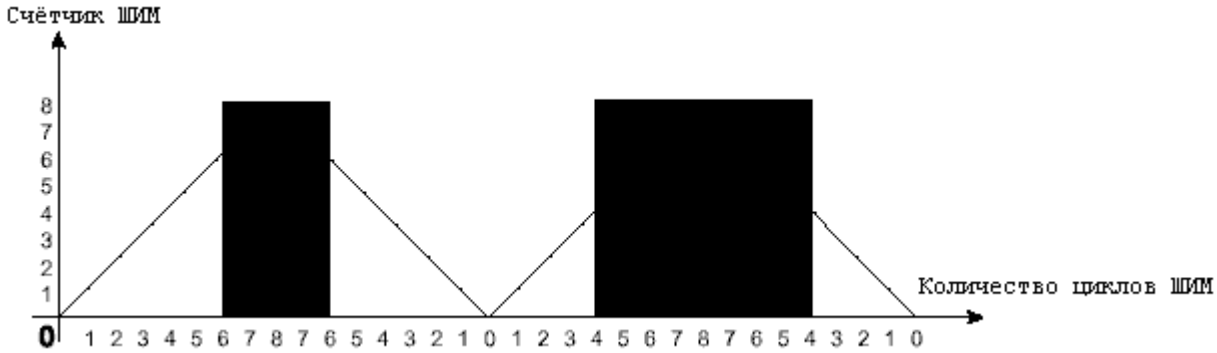

Рисунок 6. Два примера ШИМ циклов

Цифровое значение воспроизводится с использованием широтно-импульсной модуляции (ШИМ). На рисунке 6, показаны выборки 2 и 3 сигнала из примера. Один цикл ШИМ сигнала состоит из двух этапов: первый - счётчик считает до максимального значения, которое может быть представлено данным разрешением (8 в этом примере), и второй - счётчик досчитывает до нуля. Вывод разрешается, когда значение ШИМ счётчика совпадает со значением цифрового сигнала и запрещается, когда значение ШИМ счётчика становится меньше этого значения. В этом примере тёмная область представляет собой энергию сигнала.

Частота ШИМ должна быть, по крайней мере, в два раза выше, чем частота сигнала. Рекомендуется, чтобы частота ШИМ была, по меньшей мере, в четыре раза выше (в зависимости от выходного фильтра). Это может быть достигнуто или снижением частоты сигнала, или увеличением частоты тактовых импульсов, или снижением разрешения сигнала[3].

Рисунок 7. Фильтрованный выходной ШИМ сигнал

В этом примере частота отсечки выходного фильтра установлена равной 4000 Гц, что составляет приблизительно одну четверть частоты ШИМ (15.686 Гц). Частота системного таймера и разрешение ШИМ определяют частоту ШИМ. При частоте системного таймера 8 МГц, частота 10-битной ШИМ равна 3922 Гц, 7843 Гц для 9-битного разрешения, и 15.686 Гц для 8-битного разрешения. Только высокое значение частоты (15.686 Гц) достаточно, чтобы служить в качестве несущей частоты для 4000 Гц сигнала. Поэтому, первоначальная 10-битная цифровая выборка преобразуется в 8-битную.

Выходной фильтр сглаживает выходной сигнал и удаляет высокочастотную несущую ШИМ сигнала. Результирующий выходной сигнал для сигнала из примера похож на тот, что изображён на рисунке 8. Если исключить ошибку квантования (она очень большая в данном примере, т.к. используется только 8 цифровых значений) и отсутствующее усиление, то сигнал полностью похож на входной аналоговый сигнал (Рисунок 1)[3].

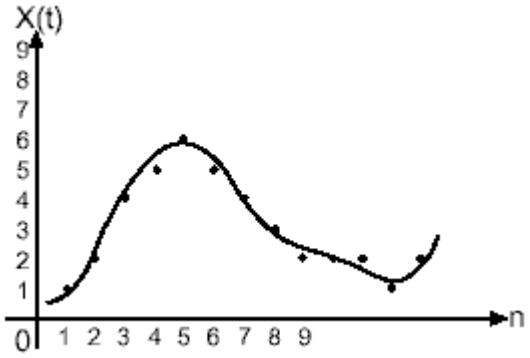

Рисунок 8. Выходной ШИМ сигнал

2.1 Электронная схема цифровой части устройства

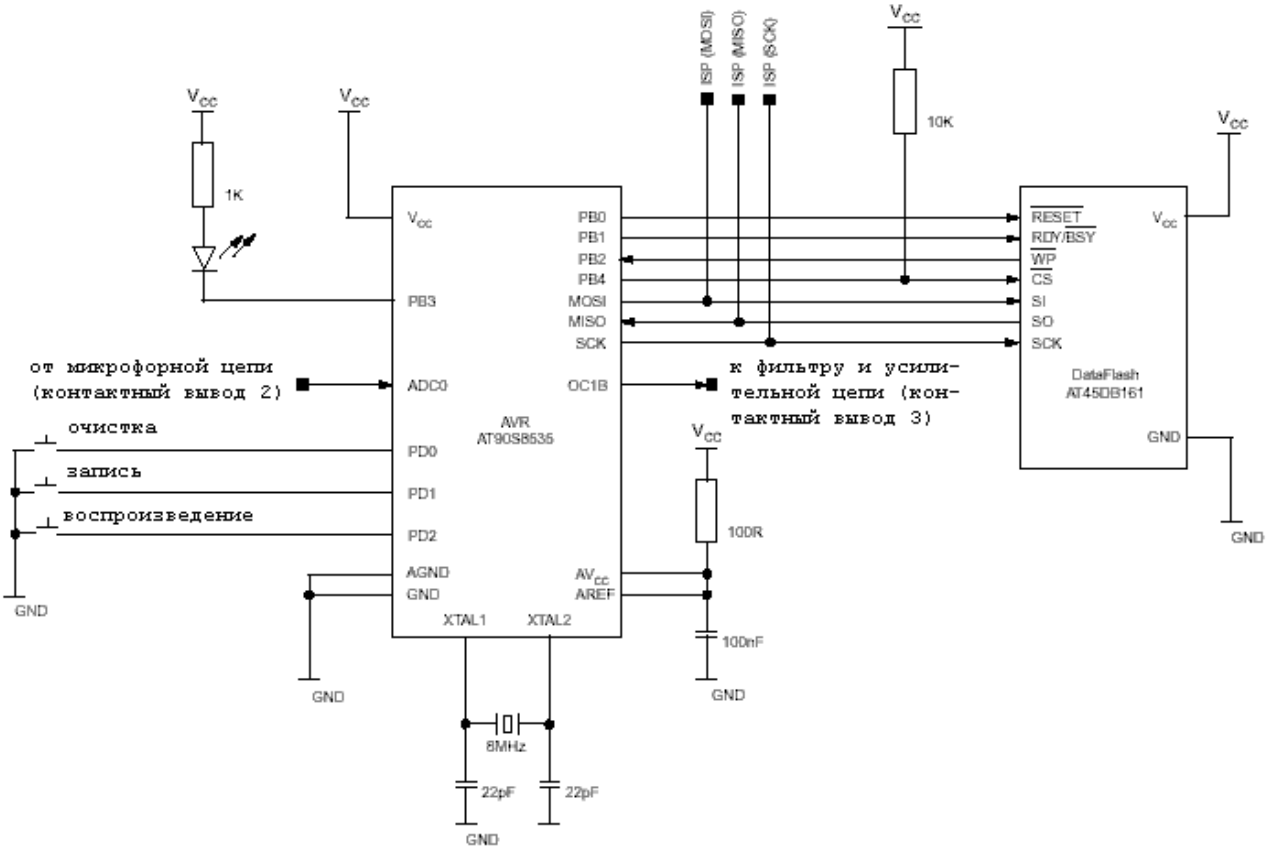

Рисунок 9. Микроконтроллер и память

Пользователь может управлять звуковой системой через три кнопки, которые называются: «Очистка», «Запись» и «Воспроизведение». Если кнопки не нажаты, то внутренний нагрузочный резистор обеспечивает VCC на PD0…PD2. Нажатие кнопки переключает входную линию на GND. В качестве обратной связи для пользователя выступает LED, отображающий состояние системы. DataFlash напрямую подключается к AVR микроконтроллеру через шину SPI. В случае использования опции ISP перепрограммирования AVR, нагрузочный резистор на линии Chip Select(#CS) предотвращает DataFlash от перехода в активное состояние. Если опция ISP не используется, то этот резистор может отсутствовать. Аналоговое напряжение, AVCC, подключается к VCC через RC-фильтр нижних частот. Опорное напряжение устанавливается равным AVCC.

Кварцевый резонатор с двумя развязывающими конденсаторами (22 пФ) генерирует системные тактовые импульсы[5].

2.2 Электронная схема аналоговой части устройства

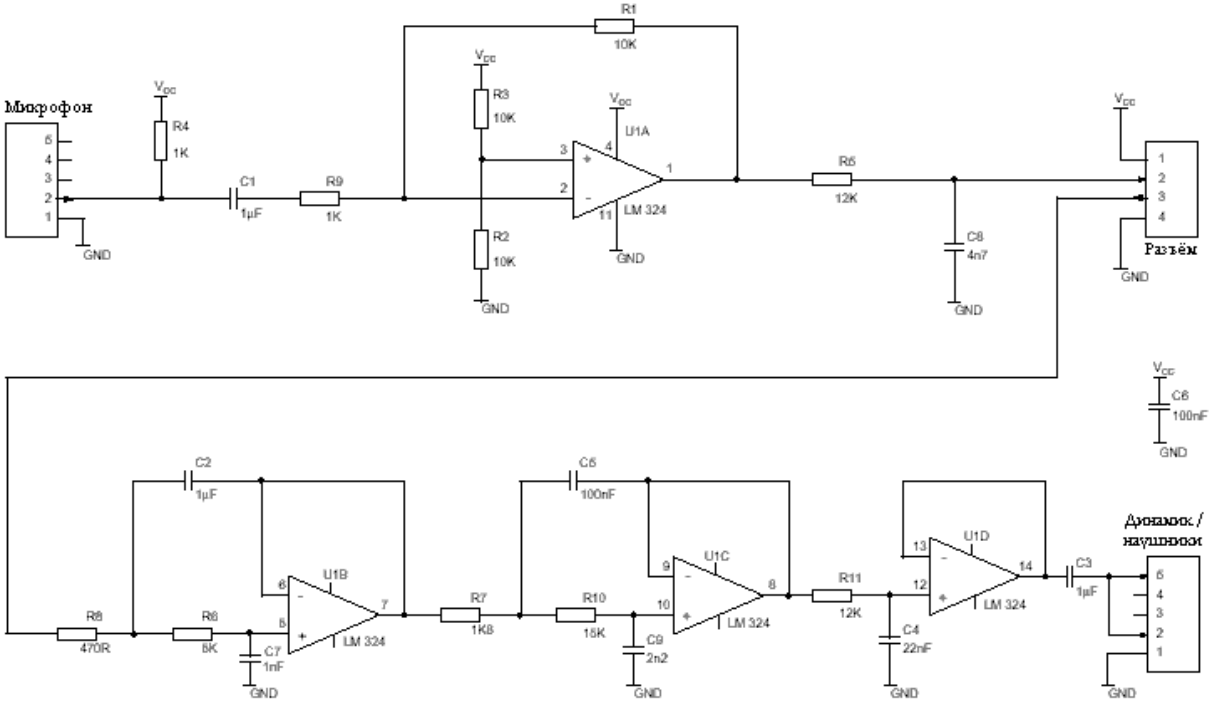

Рисунок 10. Аналоговая часть

Микрофонный усилитель является простым инвертирующим усилителем. Коэффициент усиления устанавливается через R1 и R9 (коэффициент усиления = R1/R9). R4 предназначен для питания микрофона, а C1 блокирует любые DC составляющие на входе усилителя. R2 и R3 устанавливают смещение. R5 и C8 формируют простой фильтр нижних частот первого порядка. Также R5 защищает усилитель от любых повреждений, если выходная цепь закорочена. Цепь динамика состоит из фильтра нижних частот Чебышева пятого порядка и усилителя с единичным коэффициентом усиления. Фильтр состоит из двух фильтров Чебышева второго порядка с расстроенными контурами (R6, R7, R8, C2, C7 и R7, R10, R11, C9, C5) и пассивного фильтра первого порядка (R11, C4). Частоты отсечки этих трёх фильтров немного сдвинуты относительно друг друга («расстроены») для ограничения неравномерности в полосе пропускания всей цепи фильтра. Суммарная частота отсечки установлена равной 4000 Гц, что приблизительно равно одной четверти частоты ШИМ(15.686 Гц). Усилитель с единичным коэффициентом усиления защищает цепь от возникновения обратной связи с выхода. C3 блокирует любую DC составляющую на входе динамика[5].

3. Программная реализация

3.1 Настройка

Когда программа запущена, порты должны быть настроены. Это делается в подпрограмме «setup» (установка). Протокол SPI определяет одно устройство как «ведущее», а другие устройства, подключенные к «ведущему», как «ведомые». В данном примере, микроконтроллер AVR выступает в роли «ведущего», а DataFlash в роли «ведомого». Так как AT90S8535, в данном примере, является только «ведущим», то в этом примере вывод SS может быть использован как вывод I/O.

SPI интерфейс AT90S8535 определён как альтернативная функция PortB (PB5…PB7). В данном устройстве, управляющие сигналы для DataFlash являются также настроечными на

PortB (PB0…PB2 и PB4). Свободный вывод (PB3) используется для управления состоянием LED. Для установок «ведущего», сигналы Serial Clock(SCK), Master Out/Slave In(MOSI), Chip Select(#CS), Write Protect(#WP) и Reset(#RST) являются выходами, тогда как Master In/Slave Out(MISO) и Ready/Busy(RDY/#BSY) являются входами. PB3 для LED также определяется как выход регистра направления данных для PortB, установленного как 0xBD.

В начальном положении все выходы PortB находятся в высоком состоянии, а на всех входах - внутренние нагрузочные резисторы. АЦП AT90S8535 подключено к PortA. Поэтому PortA определён как вход в высокоимпедансном состоянии. PortD служит в качестве входа для кнопок и выхода ШИМ сигнала. Здесь используется функция ШИМ Timer1 на выводе PD4. Прерывания разрешены. В данном устройстве, используются два прерывания («ADC» и «Timer1 Overflow»), которые разрешаются и запрещаются напрямую в подпрограмме, когда они требуются.

3.2 Главный цикл

В главном цикле, отслеживается состояние всех трёх кнопок. Если одна из них нажата, то LED загорается и показывает, что система занята, и вызывается соответствующая подпрограмма. Дополнительный цикл выполняется до тех пор, пока кнопка нажата, в качестве программной противодребезговой защиты для функций «Очистка» и «Воспроизведение». Во время главного цикла, LED погашен, это значит, что система работает в холостом режиме.

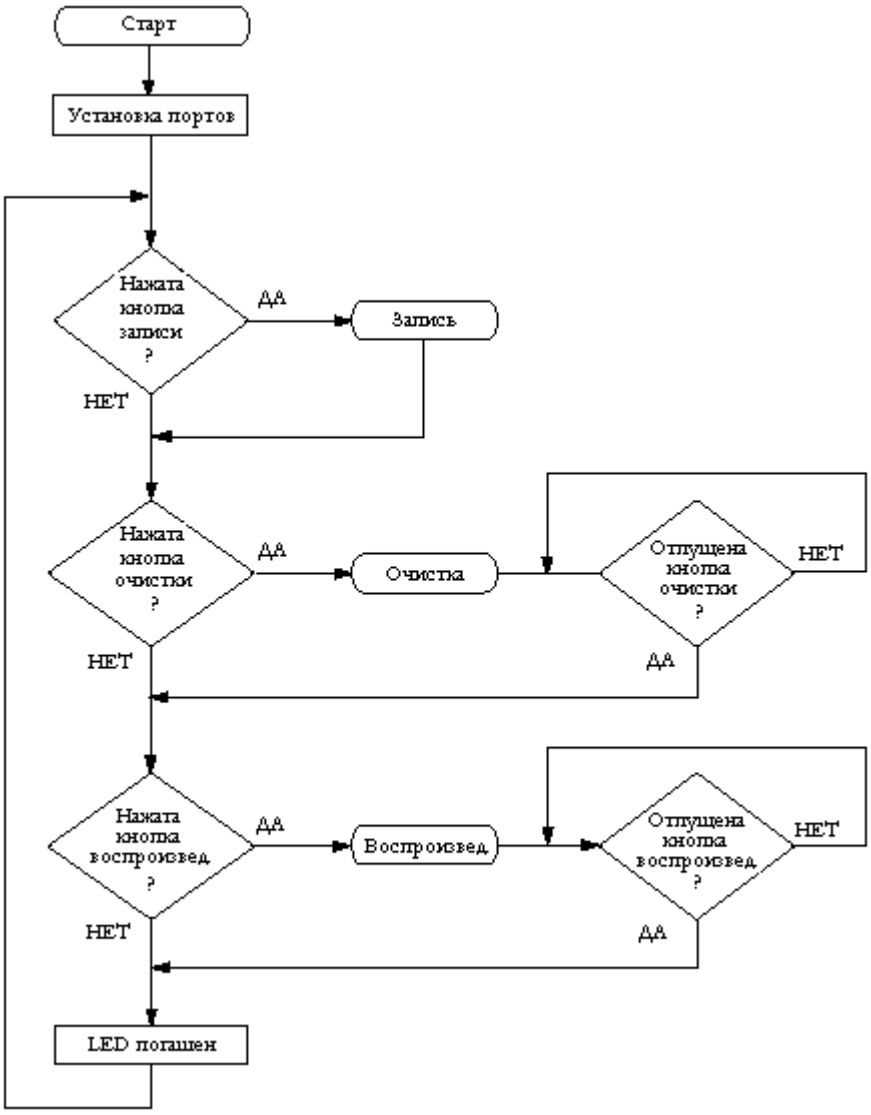

Рисунок 11. Главный цикл

3.3 Очистка

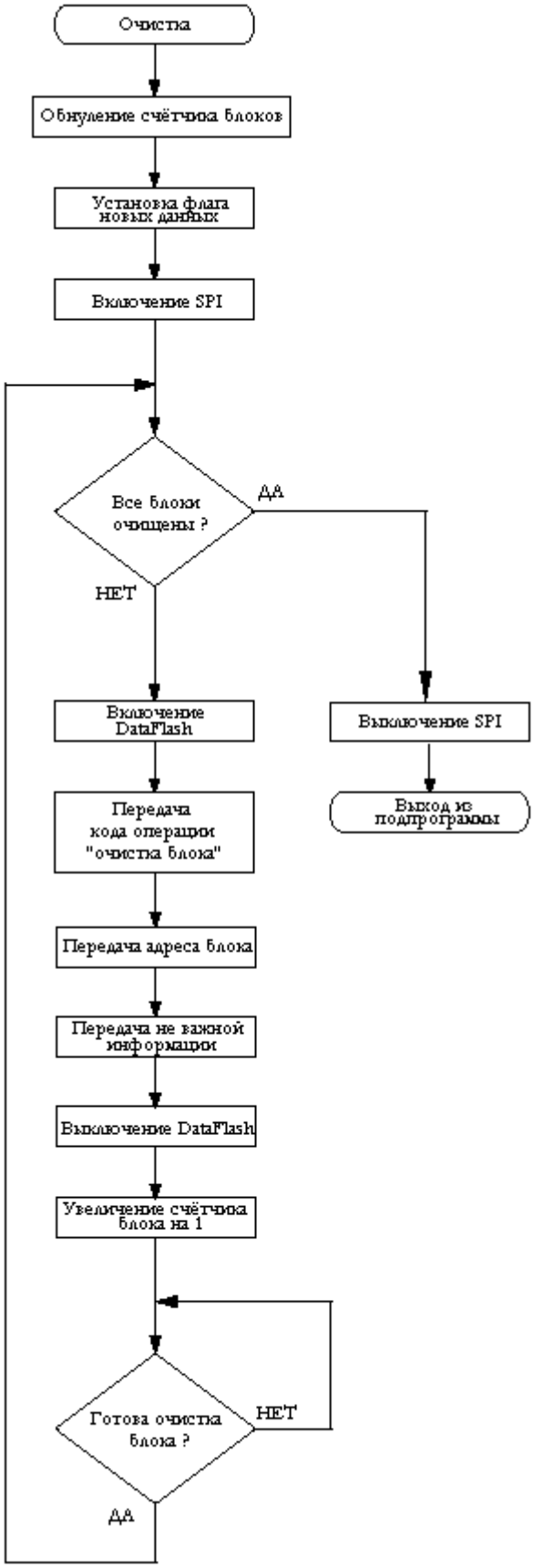

Рисунок 12. Очистка

При вызове подпрограммы «erase» (очистка), устанавливается флаг, который показывает, что в следующем цикле записи новые данные могут быть сохранены в начале DataFlash.

SPI должен быть установлен для доступа к DataFlash. Здесь не используются прерывания. Порядок данных для DataFlash следующий: MIB является первым, а AT90S8535 - «ведущим». DataFlash принимает либо сигнал SCK, который находится в низком состоянии, когда #CS переключается из высокого в низкое состояние (SPI режим 0), либо сигнал SCK, который находится в высоком состоянии, когда #CS переключается из низкого в высокое состояние (SPI режим 3), во время положительной фазы тактовых импульсов. В данном примере SPI установлен в режим 3. Для того чтобы получить наибольшую скорость передачи данных, выбирается наименьшее деление тактовой частоты, шина SPI запускается на частоте 2 МГц, если используется кварцевый генератор с частотой 8 МГц.

Для выполнения очистки блока, линия #CS переводится в низкое состояние и в DataFlash, следом за двумя зарезервированными битами (нулями), загружается код операции 0x50, затем 9-разрядный адрес блока и 13 не имеющих значения бит. Эта последовательность передаётся побайтно «ведомому». После каждого байта, регистр состояния SPI (SPSR) проверяется до тех пор, пока флаг прерываний SPI не покажет, что передача завершена. После записи всей последовательности, сразу после перевода линии #CS в высокое состояние, начинается очистка блока. Вывод Ready/Busy переводится памятью DataFlash в низкое состояние, до тех пор, пока блок не очиститься. Затем следующий блок будет очищен тем же самым способом, что и текущий. Очистка будет продолжаться, пока все 512 блоков не очистятся. Очищенные зоны читаются как 0xFF.

3.4 Запись

Подпрограмма записи состоит из установки АЦП и пустого цикла, который продолжается пока нажата кнопка «Запись». В данном примере используется вывод ADC0, для которого требуется, чтобы регистр выбора мультиплексора АЦП (ADMUX) был установлен в нуль. В регистре управления и состояния АЦП (ADCSR) разрешается работа с коэффициентом деления тактовой частоты 32, устанавливается режим одиночного преобразования, разрешаются прерывания, а также сбрасываются флаги прерываний. Аналого-цифровое преобразование начинается сразу. Первое преобразование занимает больше времени, чем последующие преобразования (832 тактовых импульса вместо 448). После этого времени, возникает прерывание АЦП, показывающее, что преобразование закончено, и результат может быть прочитан из регистра данных АЦП. Аналоговый сигнал из цепи микрофона выбирается на частоте 15.686 Гц. Это та же самая частота, что и выходная (ШИМ) частота.

Для достижения частоты выборки 15.686 Гц, выборка должна происходить каждые 510 циклов (15.686 Гц x 510 = 8 МГц). Для получения одного результата АЦП, нужно каждые 510 циклов запускать АЦП в режиме одиночного преобразования с коэффициентом деления частоты 32. Одиночное преобразование занимает 14 циклов АЦП. Поэтому преобразование будет готово после 14 x 32 = 448 циклов. Когда преобразование закончено, возникает прерывание. Процедура прерывания выполняет цикл для заполнения пустых 62 циклов (510–448), перед началом нового преобразования. Результатом 10-разрядного преобразования является величина на входе АЦП, которая появляется через 2 цикла после начала преобразования. Эти 10 бит перекрывают диапазон от AGND до AREF (в данном примере от 0 до 5В). Выходной сигнал цепи микрофона ограничен диапазоном 2.3В…3.5В. Поэтому из результата 10-разрядного преобразования вычитается минимальное входное напряжение. Это 0x1D5 для 2.3В. Часть данных, представляющих сигнал величиной выше 3.5В, убирается путём удаления двух MSB. Это делается автоматически, когда результат преобразования передаётся в подпрограмму «запись во флэш», так как эти переменные «flash_data» определяются типом «char» (8-бит). Последние 8-бит данных должны быть записаны в DataFlash перед следующим прерыванием преобразования.

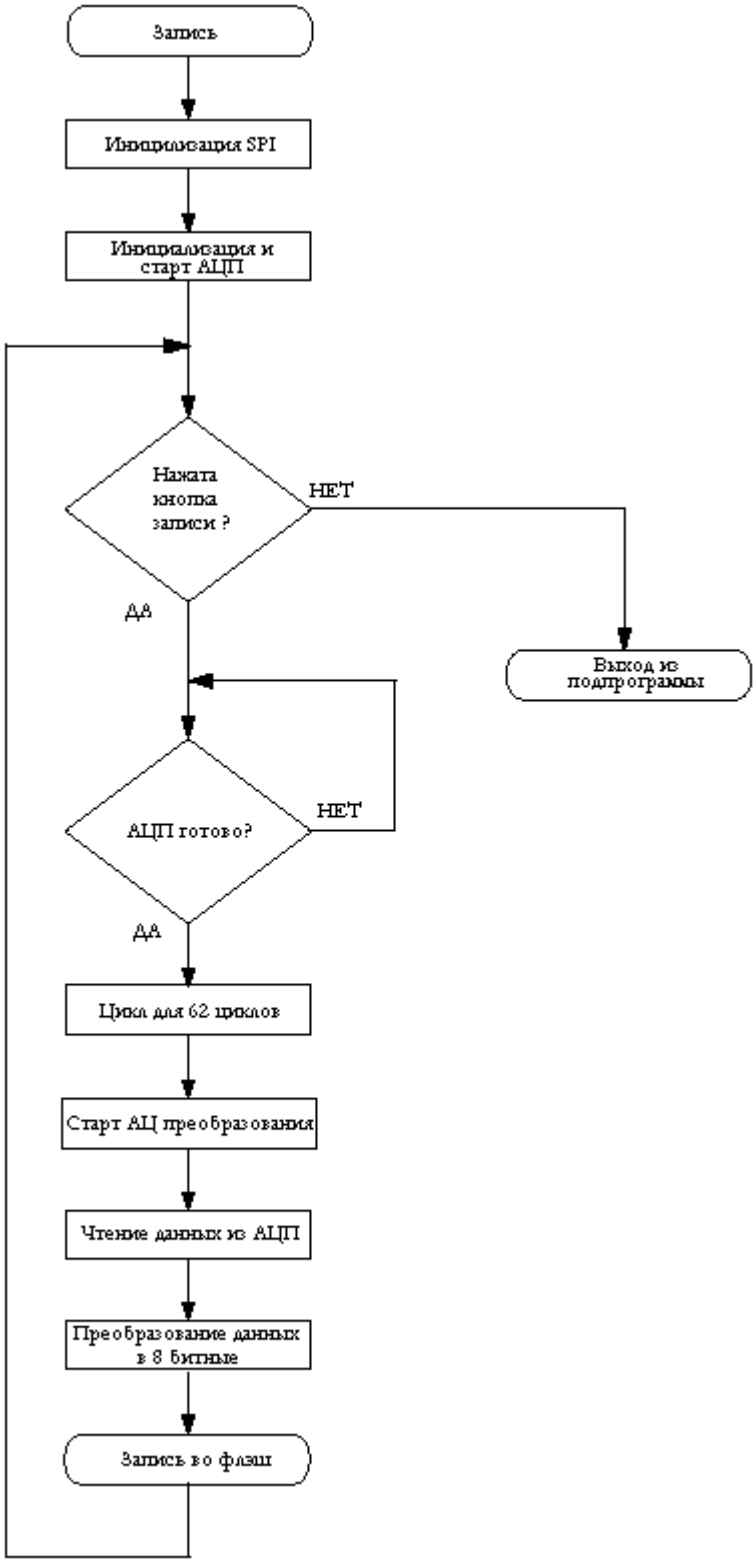

Рисунок 13. Запись

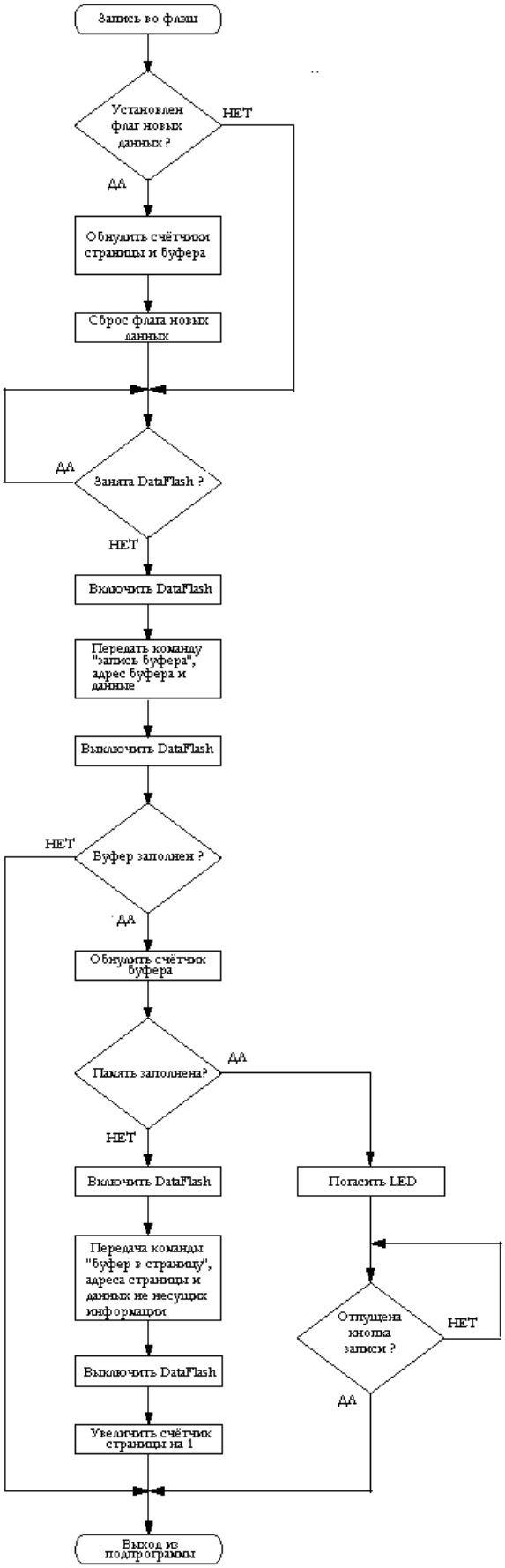

Рисунок 14. Запись в DataFlash

Запись данных в DataFlash производится путём записи сначала в буфер, а, когда этот буфер будет заполнен, его содержимое запишется в одну страницу главной памяти.

В подпрограмме «write_to_flash» переменная «j» соответствует номеру байта в буфере, а переменная «k» номеру страницы, в которую будет записываться содержимое буфера. Если флаг новых данных показывает, что DataFlash пуста, то оба счётчика устанавливаются в нуль. Если память уже содержит некоторые данные, то переменные показывают следующее свободное место в памяти, и гарантируют, что новые данные добавятся к содержимому памяти. Для того чтобы защитить содержимое этих переменных при двух вызовах функций, они объявляются статическими переменными. Для записи данных в буфер, линия #CS переводится в низкое состояние и в DataFlash загружается операционный код 0x84. Это следует за 14 не имеющими смысла битами и 10-битовым адресом положения внутри буфера. Затем вводятся 8-бит данных. Эта последовательность передаётся «ведомому» побайтно. После каждого байта проверяется регистр состояния SPI (SPSR), пока флаг прерывания SPI не покажет, что последовательная передача завершена. После записи всей последовательности линия #CS переводится в высокое состояние. Если буфер заполнен и остались пустые страницы, то буфер копируется на следующую страницу DataFlash. Так как память была очищена раньше, то данные могут быть записаны без дополнительного стирания. Если память заполнена, то цикл выполняется, пока нажата кнопка «запись». Любые данные, записанные в то время, когда память уже заполнена, будут потеряны.

3.5 Воспроизведение

В процедуре «воспроизведения», содержимое DataFlash считывается и модулируется как 8-разрядная ШИМ на частоте 15.686 Гц. Для достижения большей скорости, данные не читаются напрямую из основной памяти, а передаются в один из двух буферов и затем читаются из буфера. В это время копируется следующая страница памяти в другой буфер. Для ШИМ, 16-разрядный Таймер/Счётчик 1 используется с выходом ШИМ на OC1B. Это описывается в регистре управления Таймера/Счётчика A и B (TCCRA/TCCRB). Для запуска ШИМ с возможной наибольшей частотой, делитель тактовой частоты ШИМ устанавливается в 1.

Когда установка завершена, первая страница копируется в буфер 1, посредством перевода линии #CS в низкое состояние и передачей соответствующих команд в DataFlash. Передача страницы в буфер начинается, когда линия #CS переводится снова в высокое состояние. Когда состояние на выводе Ready/Busy меняется памятью DataFlash на высокое, то это означает, что буфер 1 содержит действительные данные. Затем начинается передача следующей страницы в буфер 2. Так как оба буфера независимы друг от друга, то данные могут всегда читаться из буфера 1, пока DataFlash остаётся занятой копированием данных из второй страницы в буфер 2.

Для чтения байта из буфера, в DataFlash должна быть записана фиктивная величина. Операция записи «ведущего» в SPI «ведомого» приводит к тому, что содержимое его регистра данных SPI (SPDR) будет изменено. После записи фиктивного байта в DataFlash, регистр SPDR микроконтроллера AVR содержит выходные данные из DataFlash.

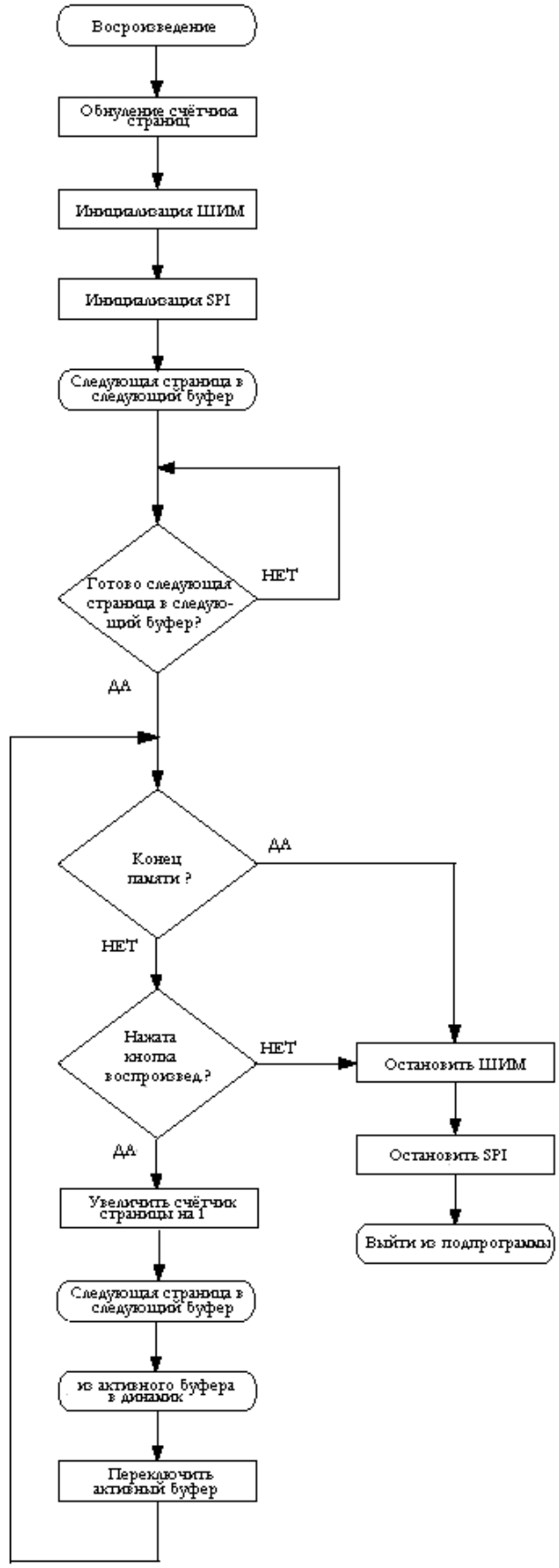

Рисунок 15. Воспроизведение

Когда значения ШИМ счётчика равно «0», Таймер 1 вызывает прерывание переполнения. Это прерывание используется для синхронизации выходных данных из DataFlash частотой ШИМ. Когда значение из буфера сдвигается в микроконтроллер AVR,

цикл выполняется до тех пор, пока Таймер 1 не вызовет прерывание переполнения. Затем данные записываются в выходной регистр сравнения Таймера/Счётчика 1 B (OCR1B), автоматически защёлкивая выход ШИМ, когда счётчик ШИМ достигнет максимального значения (255 для 8-разрядной ШИМ).

После того как считается последнее значение из буфера, активный буфер переключится. Если воспроизведена вся память, то все прерывания отключены и Таймер/Счётчик 1 остановлен.

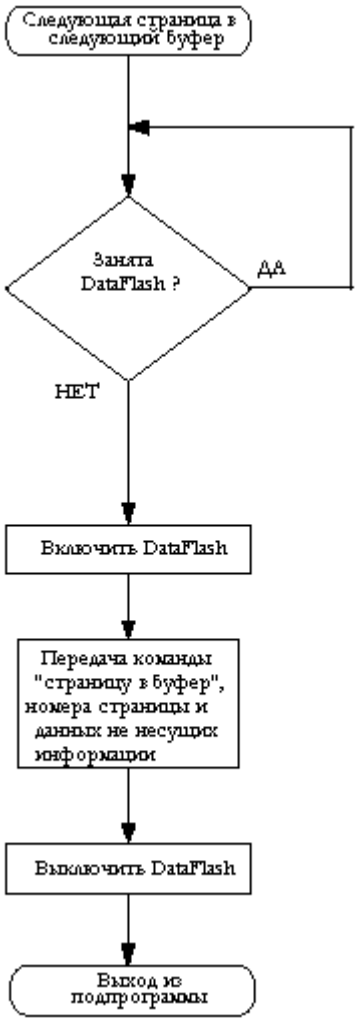

Рисунок 16. Следующая страница в следующий буфер

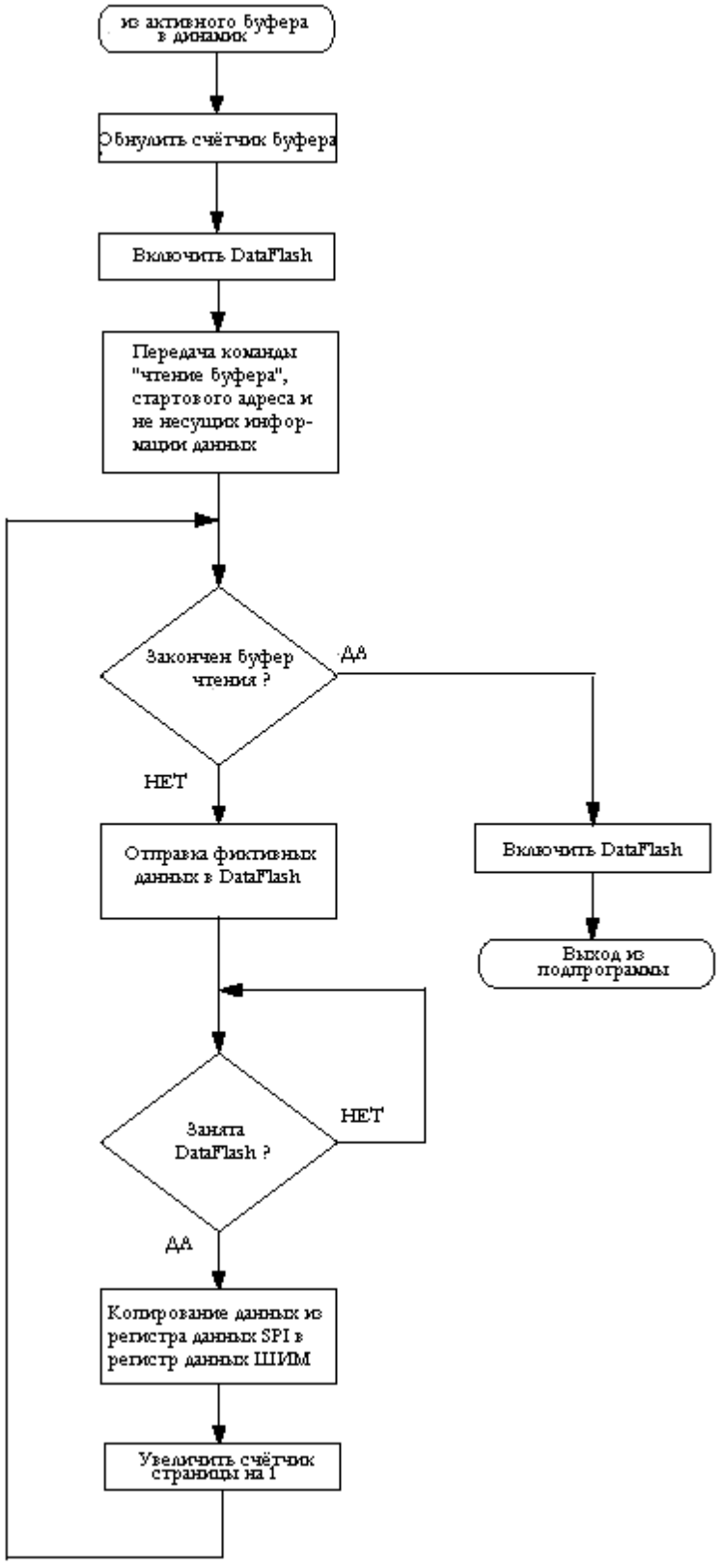

Рисунок 17. Активный буфер в динамик

3.6 Изменение и оптимизация

Сигнал с выхода микрофона может изменяться в зависимости от типа используемого микрофона. Для достижения лучших результатов важно выбрать такой коэффициент усиления микрофонного усилителя, который обеспечит максимальный сигнал, наиболее близкий к AREF. Данные, записанные в DataFlash, полностью соответствуют данным,

считанным с АЦП. В случае записи в течение большого промежутка времени или записи стерео сигнала может потребоваться упаковка этих данных, или увеличить объем используемой микросхемы памяти DataFlash.

Частота выборки равная 15.686 Гц (приблизительно 510 циклов), генерируется с помощью прерывания АЦП и цикла задержки. Она может быть заменена независимым таймером (Таймер/Счётчик 0 или Таймер/Счётчик 2), если он не используется для других целей.

4. Исходный код программы

Язык реализации : C

Среда разработки : IAR Embedded Workbench for Atmel AVR v.1.5

```
#include "io8535.h"
#include
#include "stdlib.h"
#include "dataflash.h"

// прототипы
void setup (void);
void erasing (void);
void recording (void);
void interrupt[ADC_vect] sample_ready (void);
void write_to_flash (unsigned char ad_data);
void playback (void);
void next_page_to_next_buffer (unsigned char active_buffer, unsigned int
page_counter);
void interrupt[TIMER1_OVF1_vect] out_now(void);
void active_buffer_to_speaker (unsigned char active_buffer);

// глобальные переменные
volatile unsigned char wait = 0;


void setup(void)
{
DDRB = 0xBD;            // Инициализация порта SPI
// SCK, MISO, MOSI, CS, LED, WP , RDYBSY, RST
// PB7, PB6, PB5, PB4, PB3, PB2 , PB1, PB0
//  O    I    O    O    O    O     I    O
//  1    0    1    1    1    1     0    1
PORTB = 0xFF;  // все выходы в высоком состоянии, на входах
   нагрузочные резисторы (LED погашен)
DDRA = 0x00;            // Port A определяется как вход
PORTA = 0x00;
DDRD = 0x10;            // Port D определяется как вход (D4: выход)

_SEI();                 // прерывания разрешены
}


void erasing(void)
{
unsigned int block_counter = 0;
unsigned char temp = 0x80;
ACSR |= 0x02;  // установка флага, показывающего, что следующим
   этапом должна быть запись данных

// прерывания запрещены, порт SPI включён, «ведущий» режим, первый MSB, 3 режим
SPI, Fcl/4
SPCR = 0x5C;
```

```
while (block_counter < 512)
{
PORTB &= ~DF_CHIP_SELECT; // включение DataFlash
SPDR = BLOCK_ERASE;
while (!(SPSR & temp));   // ожидание завершения передачи
SPDR = (char)(block_counter>>3);
while (!(SPSR & temp));   // ожидание завершения передачи
SPDR = (char)(block_counter<<5);
while (!(SPSR & temp));   // ожидание завершения передачи
SPDR = 0x00;                  // не важно
while (!(SPSR & temp));   // ожидание завершения передачи
PORTB |= DF_CHIP_SELECT;  // выключение DataFlash

block_counter++;
while(!(PINB & 0x02));         // ожидание очистки блока
}
SPCR = 0x00;                   //отключение SPI
}


void recording(void)
{
// прерывания запрещены, порт SPI включён, «ведущий» режим, первый MSB, 3 режим
SPI, Fcl/4

SPCR = 0x5C;
ADMUX = 0x00; // номер входного вывода АЦП = 0
ADCSR = 0xDD; // одиночное АЦ преобразование, fCK/32, старт
    преобразования
while (!(PIND & 8)); // цикл продолжается пока нажата кнопка записи
    (кнопка 3)

ADCSR = 0x00; // выключение АЦП
SPCR = 0x00;  // выключение SPI
}


void interrupt[ADC_vect] sample_ready(void)
{
unsigned char count = 0;

while (count < 6) count++; // ожидание в течение нескольких циклов
ADCSR |= 0x40;         // старт нового АЦ преобразования
write_to_flash(ADC-0x1D5); // чтение данных, преобразование 8 бит и
   сохранение во флэш
}


void write_to_flash(unsigned char flash_data)
{
static unsigned int buffer_counter;
static unsigned int page_counter;
unsigned char temp = 0x80;

if((ACSR & 0x02)) // если флаг установлен, то новые данные должны быть
 установлены
{
buffer_counter = 0;
page_counter = 0;   // сброс счётчика если должны быть записаны
   новые данные
ACSR &= 0xFD;  // очистка флага сигнала
}
```

```
while(!(PINB & 0x02));          // проверка занятости флэша

PORTB &= ~DF_CHIP_SELECT;       // включение DataFlash

SPDR = BUFFER_1_WRITE;
while (!(SPSR & temp));         // ожидание завершения передачи
SPDR = 0x00;                    // не важно
while (!(SPSR & temp));         // ожидание завершения передачи

SPDR = (char)(buffer_counter>>8); // не важно + первые два бита буфера
    адреса
while (!(SPSR & temp));                 // ожидание завершения передачи

SPDR = (char)buffer_counter;    // буфер адреса (макс. 2^8 = 256
    страниц)
while (!(SPSR & temp));                 // ожидание завершения передачи
SPDR = flash_data;              // запись данных в регистр данных SPI
while (!(SPSR & temp));         // ожидание завершения передачи


PORTB |= DF_CHIP_SELECT;        // выключение DataFlash

buffer_counter++;

if (buffer_counter > 528)       // если буфер заполнен, то его
  содержимое записывается в страницу памяти
{
buffer_counter = 0;
if (page_counter < 4096) // если память не заполнена
{
PORTB &= ~DF_CHIP_SELECT;       // включить DataFlash

SPDR = B1_TO_MM_PAGE_PROG_WITHOUT_ERASE; // записать
данные из буфера 1 в страницу
while (!(SPSR & temp));         // ожидание завершения
   передачи
SPDR = (char)(page_counter>>6);
while (!(SPSR & temp));         // ожидание завершения
   передачи
SPDR = (char)(page_counter<<2);
while (!(SPSR & temp));         // ожидание завершения
   передачи
SPDR = 0x00;                    // не важно
while (!(SPSR & temp));         // ожидание завершения
   передачи

PORTB |= DF_CHIP_SELECT;        // выключение DataFlash

page_counter++;
}
else
{
PORTB |= 0x08;  // погасить LED
while (!(PIND & 8)); // ждать пока кнопка записи
           не отпущена (кнопка 3)
}
}
}


void playback(void)
{
unsigned int page_counter = 0;
unsigned int buffer_counter = 0;
```

```
unsigned char active_buffer = 1; // активный буфер = буфер 1
unsigned char temp = 0x80;

TCCR1A = 0x21; // 8 бит ШИМ, используется COM1B
TCNT1 = 0x00;  // обнуление счётчика 1
TIFR = 0x04;           // сброс флага превышения  счётчика 1
TIMSK = 0x04;  // разрешение прерывания переполнения счётчика 1
TCCR1B = 0x01; // коэф. Пересчёта счётчика 1 = 1
OCR1B = 0x00;          // обнуление выходного регистра сравнения B

// прерывания запрещены, порт SPI включён, «ведущий» режим, первый MSB, 3 режим
SPI, Fcl/4

SPCR = 0x5C;

next_page_to_next_buffer (active_buffer, page_counter); // чтение страницы 0
      в буфер 1

while (!(PINB & 0x02)); // ожидание завершения передачи данных из страницы 0
в буфер 1
while ((page_counter < 4095)&(!(PIND & 2))) // пока кнопка воспроизведения
(кнопка 1) нажата
{
page_counter++; // теперь берём следующую страницу

next_page_to_next_buffer (active_buffer, page_counter);
active_buffer_to_speaker (active_buffer);

if (active_buffer == 1) // если буфер 1 является активным буфером
{
active_buffer++;       // то устанавливаем буфер 2 в качестве активного
}
else                   // иначе
{
active_buffer--;       // устанавливаем буфер 1 в качестве активного
}
}
TIMSK = 0x00;  // запрещаем все прерывания
TCCR1B = 0x00; // останавливаем счётчик 1
SPCR = 0x00;           // отключаем SPI
}


void next_page_to_next_buffer (unsigned char active_buffer, unsigned int
page_counter)
{
unsigned char temp = 0x80;
while(!(PINB & 0x02));         // ждём, пока флэш не освободится

PORTB &= ~DF_CHIP_SELECT; // включаем DataFlash

if (active_buffer == 1)        // если буфер 1 активный
{
SPDR = MM_PAGE_TO_B2_XFER; // то передаём следующую страницу в
   буфер 2
}
else                                   // иначе
{
SPDR = MM_PAGE_TO_B1_XFER; // передаём следующую страницу в
                                  буфер 1
       }
while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = (char)(page_counter >> 6);
while (!(SPSR & temp)); // ожидаем завершения передачи
```

```
SPDR = (char)(page_counter << 2);
while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = 0x00; // записываем не имеющий значения байт
while (!(SPSR & temp)); // ожидаем завершения передачи
PORTB |= DF_CHIP_SELECT; // выключаем DataFlash и начинаем передачу
}


void interrupt[TIMER1_OVF1_vect] out_now(void)
{
wait = 0; // возникновение прерывания
}


void active_buffer_to_speaker (unsigned char active_buffer)
{
// пока активный буфер не очистится воспроизводим его содержимое на динамике
unsigned int buffer_counter = 0;
unsigned char temp = 0x80;

PORTB &= ~DF_CHIP_SELECT; // включение DataFlash

if (active_buffer == 1)          // если буфер 1 активный буфер
{
SPDR = BUFFER_1_READ; // то читаем из буфера 1
}
else                             // иначе
{
SPDR = BUFFER_2_READ; // читаем из буфера 2
}

while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = 0x00; // запись не имеющего значения байта
while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = 0x00; // запись не имеющего значения байта
while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = 0x00; // начать с адреса 0 буфера
while (!(SPSR & temp)); // ожидаем завершения передачи
SPDR = 0x00; // запись не имеющего значения байта
while (!(SPSR & temp)); // ожидаем завершения передачи

while (buffer_counter < 528)
{
SPDR = 0xFF; // записываем фиктивное значение в начало
   сдвигового регистра
while (!(SPSR & temp)); // ожидаем завершения передачи
while(wait); // ожидаем прерывание переполнения таймера 1
OCR1B = SPDR; // воспроизводим данные из сдвигового регистра
wait = 1; // сброс флага сигнала

buffer_counter++;
}

PORTB |= DF_CHIP_SELECT; // выключение DataFlash
}


void main(void)
{
setup();

for(;;)
{
if (!(PIND & 8)) // если кнопка записи нажата (кнопка 3)
```

```
{
PORTB &= 0xF7; // зажигаем LED
recording();
}
if (!(PIND & 4)) // если нажата кнопка очистки (кнопка 2)
{
PORTB &= 0xF7; // зажигаем LED
erasing();
while (!(PIND & 4)); // ждём пока кнопка очистки не
    отпущена (кнопка 2)
}
if (!(PIND & 2)) //если нажата кнопка воспроизведения(кнопка 1)
{
PORTB &= 0xF7; // зажигаем LED
playback();
while (!(PIND & 2)); // ждём пока кнопка воспроизведения
    не отпущена (кнопка 1)

              }
PORTB |= 0x08; // гасим LED во время «холостой» работы
}
}
```

Литература

1. Кривченко И.В. Микроконтроллеры общего назначения для встраиваемых приложений производства Atmel Corp. // Электронные компоненты 5, 2002. с. 69-73.
2. Гребнев В. В. Микроконтроллеры семейства AVR фирмы Atmel. М.: ИП Радиософт. 2002.
3. Proakis, J.G. and Manolakis, D.G. // 1992 // Digital Signal Processing: Principles, Algorithms, and Applications. Second Edition
4. AVR Technical Training. Atmel Corp. Norway December 2003.
5. http://www.atmel.com/avr.

Статья получена: 2006-02-28