Работа с внешней последовательной энергонезависимой памятью

Денис Ягов (г. Екатеринбург)

В статье рассматривается возможность быстрой и эффективной интеграции последовательной памяти в проект с микроконтроллером STM32. Производится анализ существующего решения инженеров STMicroelectronics. Автор статьи вносит свои изменения в программу контроллера, которые позволяют снизить участие ядра при обмене с последовательной памятью, и оценивает загрузку системы в тестовом приложении.

Сегодня трудно найти электронное устройство, которое не использовало бы цифровую обработку данных. Цифровая обработка данных, как правило, требует использования памяти. Материал посвящён освоению методик использования внешней последовательной памяти.

Увеличение скорости работы приложения наращиванием объёма памяти

Очень часто можно построить простой алгоритм приложения на использовании памяти большого объёма. Классическим примером можно назвать использование таблицы конечного результата. Например, можно хранить звуковую запись в сжатом формате, таком как mp3, wma, speex и других, а можно в формате, готовом для воспроизведения без декомпрессии. Это будет формат wav (или похожий, описывающий отклонение диффузора от среднего значения в каждый момент времени). В первом случае для воспроизведения звука потребуются значительные ресурсы ядра контроллера, потому что до момента воспроизведения звуковые данные на лету будут преобразованы в формат wav. Во втором – большой объём памяти хранения.

Есть оптимальный баланс между производительностью микроконтроллера и памятью, имеющейся у него. Оптимальность этого баланса измеряется ценой решения. Можно поставить процессор с мощным ядром, высокой тактовой частотой и небольшим объёмом памяти, а можно, наоборот, поставить более простой процессор, но с памятью большего объёма. Алгоритмы решения задачи будут разными. С точки зрения автора, первое решение будет дешевле, но потребует больше времени на разработку (или более высокой квалификации разработчика), что также измеряется деньгами. Использование недорогой внешней памяти в ряде случаев меняет баланс в пользу памяти большого объёма. Поэтому выбор подходящего пути разработки не является очевидным.

Чем обосновано применение внешней энергонезависимой памяти

Действительно, можно взять микроконтроллер с большим объёмом флэш-памяти и писать в неё всевозможные таблицы данных. Тогда решение будет однокристальным и внешняя энергонезависимая память не понадобится. Ввод данных – вместе с программой одним программатором, в процессе одного подключения. Красота!

Если так было бы на самом деле, то микросхем последовательной памяти флэш уже давно бы не существовало. В чём причина? Если посмотреть на рынок микроконтроллеров, то легко заметить, что процессор со встроенной флэш-памятью 1 Мб или больше является скорее экзотикой, чем массовым решением. Цена такого контроллера существенно выше, чем у аналогичного с меньшим объёмом флэш-памяти. Рост цены контроллера находится в линейной зависимости от роста объёма памяти. Дело в том, что процесс размещения флэш-памяти на кристалле не является оптимальным. В первую очередь из-за того, что кроме самой памяти требуется вырастить остальные сегменты контроллера и технологический процесс выращивания кристаллов имеет ограничения. Из-за ограничений процесса производства флэш-память, даже небольшого объёма, занимает большую площадь на кристалле. Соответственно, наибольший вклад в рост площади кристалла микроконтроллера вносит увеличение флэш-памяти (а не добавление периферии или дополнительных возможностей ядра). Чем больше площадь кристалла, тем меньше их умещается на кремниевой пластине, и как результат – рост цены. Сколько стоит микроконтроллер с памятью 1 Мб? Думаю, цена начнётся с 300 руб. Сколько стоит последовательная память такого же объёма? Вполне возможно, 10...20 руб. (например, микросхема M25P80). Более того, если взять память ещё большего объёма, то легко заметить, что цена её растёт нелинейно. Например, цена SD-карты памяти, которую можно подключить через SPI-интерфейс к контроллеру объёмом 1 Гб, вряд ли превзойдёт 200 руб.

Таким образом, существующая технология производства микроконтроллеров не позволяет нам выполнить однокристальное решение с большим объёмом памяти, оптимальное по цене. Именно это обстоятельство является обоснованием применения внешней флэш-памяти для хранения данных.

Решения на внешней последовательной памяти

На существующий момент на рынке представлено множество предложений последовательной флэш-памяти. Принцип их построения, по большому счёту, одинаков. Различия касаются интерфейса подключения, системы команд и некоторых несущественных особенностей. Мы будем рассматривать применение внешней памяти на примере последовательной M25 в сочетании с контроллерами семейства STM32. Для того чтобы подключить последовательную память M25 к микроконтроллеру STM32, можно (и нужно) прочитать её описание. Кроме того, есть более быстрый путь. Возьмём схему отладочной платы, на которой имеется память M25. С сайта www.st.com берём описание и софт отладочной платы STM3210B-EVAL. Данная плата построена на базе микроконтроллера

STM32F103VBT6. Далее на рис. 1 показана электрическая принципиальная схема подключения. Производитель отладочной платы учёл воздействие помех и исключил возможность несанкционированного возникновения команды модификации данных на переходном процессе подачи питания. Эту схему можно взять за основу в своём проекте.

Таким образом, подключить аппаратно микросхему внешней памяти не составит труда. Вероятнее всего, сложность будет при подключении памяти программно. Семейство микросхем памяти M25 имеет набор команд:

- снять защиту записи;
- установить защиту записи;
- считать идентификатор микросхемы памяти;
- считать состояние микросхемы памяти;
- установить состояние микросхемы;
- считать данные;
- быстрое чтение данных;
- запись страницы;
- стереть сектор;
- стереть все данные;
- уход в энергосберегающий режим;
- выход из энергосберегающего режима.

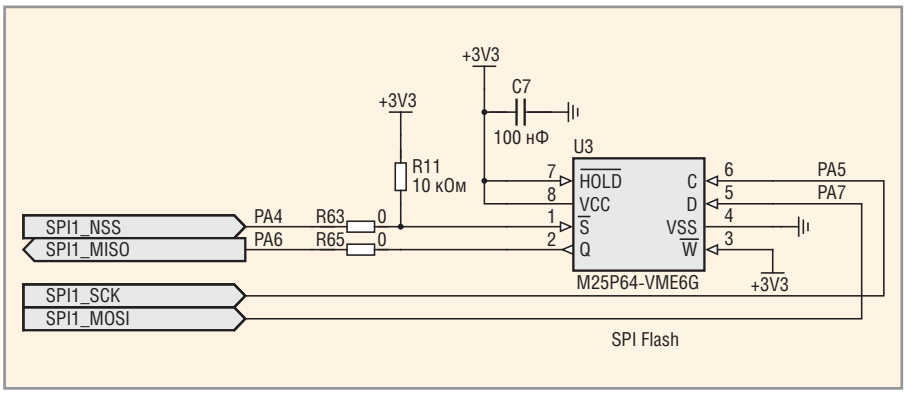

Рис. 1. Подключение памяти M25 к микроконтроллеру

Очевидно, что для полноценной работы с памятью потребуется описать минимум половину этих процедур в теле программы микроконтроллера (кроме того, каждая команда из перечня имеет свой формат, что, безусловно, добавит работы программисту). Более того, потребуется правильно настроить периферию контроллера. Обратимся к материалам производителя контроллеров. STMicroelectronics все указанные проблемы уже решила за разработчика и предоставила стек для работы с памятью M25. Чтобы нам было ещё проще, инженеры ST предоставили нам исходный код.

Программное решение инженеров STMicroelectronics

Для того чтобы начать работу с микроконтроллером STM32, вы можете использовать традиционный подход: прочитать описание регистров периферии, а затем пробовать писать программы, напрямую управляя этими регистрами. Для работы с периферией инженеры STMicroelectronics предложили под-

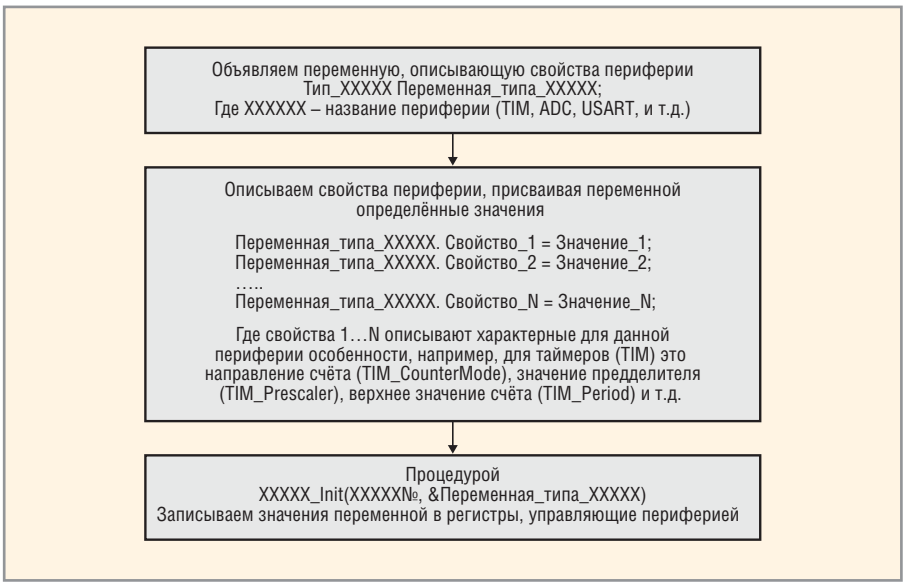

Рис. 2. Инициализация любой периферии микроконтроллера STM стандартной библиотекой

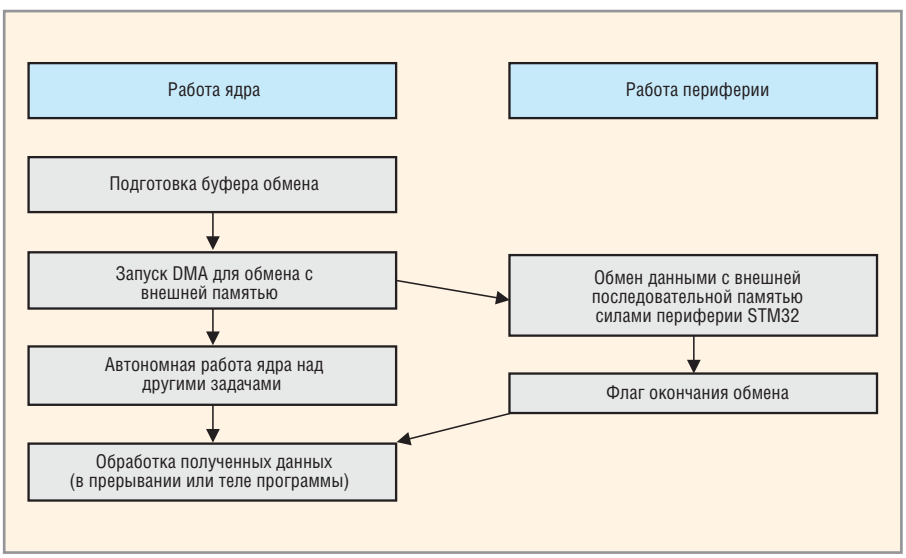

Рис. 3. Оптимальная работа с внешней последовательной памятью

ход,связанный с использованием стандартной библиотеки периферии для соответствующего контроллера.

Изучение расположения управляющих регистров и назначения их битов не требуется. Всю эту информацию знает процедура XXXXX_Init (рис. 2). Для подключения памяти типа M25 требуется инициализация интерфейса SPI. Листинг 1 (см. дополнительные материалы к статье на сайте журнала) показывает, как это сделали инженеры STMicroelectronics, используя собственную библиотеку.

Итак, настроить периферию мы можем. Рассмотрим в качестве примера одну из команд, приведенную в Листинге 2 (см. дополнительные материалы к статье на сайте журнала).

Если описать процедуру в двух словах, то ядро последовательно отправляет несколько байт – код команды стирания и адреса стираемого сектора – через SPI в последовательную память и ожидает окончания записи. Никакими другими операциями в этот момент ядро не может заниматься, за исключением обработки прерываний. Это ещё лучше видно при рассмотрении процедуры sFLASH_SendByte (вызываемой рассматриваемой функцией sFLASH_EraseSector), в которой ядро дожидается окончания передачи каждого байта информации. Таким образом, у нас нет возможности выполнять основную программу параллельно с процессом чтения внешней памяти.

Очевидно, что такой подход нерационально использует ресурс времени микроконтроллера. Данный алгоритм понятен, но далеко не оптимален с точки зрения производительности процессора. Семейство STM32 имеет мощную гибко настраиваемую периферию. Далее мы рассмотрим более совершенный алгоритм обращения к внешней последовательной памяти.

Оптимальное программное решение

Семейство STM32 позволяет буферизировать обмен данными через любой интерфейс подключением контроллера прямого доступа к памяти (DMA). В этом смысле SPI не стал исключением. Более того, мы можем настроить сразу два буфера: приёма и отправки. Сделать это несложно, т.к. инженеры STMicroelectronics уже сделали это средствами стандартной библиотеки. В ней имеется пример, где задействованы два интерфейса SPI, один из которых «ведущий», другой – «ведомый», а с помощью DMA осуществляется отправка и приём данных между ними через внешнее соединение. Ядро только настраивает периферии и запускает их, но не участвует в процессе обмена. На базе этого примера мы построим своё приложение. Цель – заставить периферию работать с внешней последовательной памятью без участия ядра. Схема работы приложения должна выглядеть как на рис. 3.

В проекте использован SPI2 – в качестве периферии, к которой подключена внешняя память типа M25 и два канала обмена контроллера прямого доступа к памяти DMA1.

Проверка боем

Создаём приложение на базе полученного стека и оцениваем результат. Задача приложения – периодически считывать аудиоданные из внешней последовательной памяти типа M25 и выводить их на динамик. Структура приложения показана на рис. 4. В качестве источника звука применяем ШИМ таймера 2. Таймер выдаёт импульсы частотой 22 кГц, их ширина модулируется звуковым сигналом.

В процессе отладки приложения стало очевидно, что для правильной работы приложения требуется двойная буферизация принимаемых аудиоданных. То есть в один буфер считываются данные из внешней памяти, в то время как воспроизведение аудио происходит из другого буфера. По окончании воспроизведения аудиоданных из буфера вызывается прерывание, которое меняет эти два буфера местами. Данная схема позволяет воспроизводить звук без потери качества. Работа периферии в нашем приложении выглядит следующим образом (рис. 5):

- таймер 2 выдаёт ШИМ-сигнал;
- при переполнении таймера 2 вызывается копирование DMA новых звуковых данных в регистр сравнения ШИМ таймера 2;
- по окончании копирования буфера аудиоданных в ШИМ таймера возникает прерывание от DMA. В нём указывается новый буфер аудиоданных, которые будут копироваться в ШИМ. Кроме того, запускается процесс изъятия аудиоданных из внешней памяти.

Теперь нам необходимо оценить получившийся проект с точки зрения загрузки ядра. Ядро подключается к работе только в случае окончания буфера аудиоданных. Его задача заключается в перенастройке периферии и её запуске. Как часто возникают прерывания? Частота дискретизации звука составляет 22 кГц, объём одного буфера 256 байт, соответственно, ядро подключается к работе воспроизведения с частотой 22 000/256 = 86 Гц. Размер обработчика прерывания составил 130 байт (хотя при отказе от использования библиотек можно объём или время выполнения кода существенно снизить), работа – практически линейная, так что на частоте 36 МГц время выполнения составит 3,6 мкс. Таким образом, общая загрузка ядра составляет менее 0,05%. Куда потратить остальные 99,95% мощности ядра – решает разработчик приложения. С указанной задачей справится любой STM32.

Код самого приложения, а также дополнительную информацию проекта можно взять здесь: http://forum.promelec.ru/index.php/topic,2231.0.html.

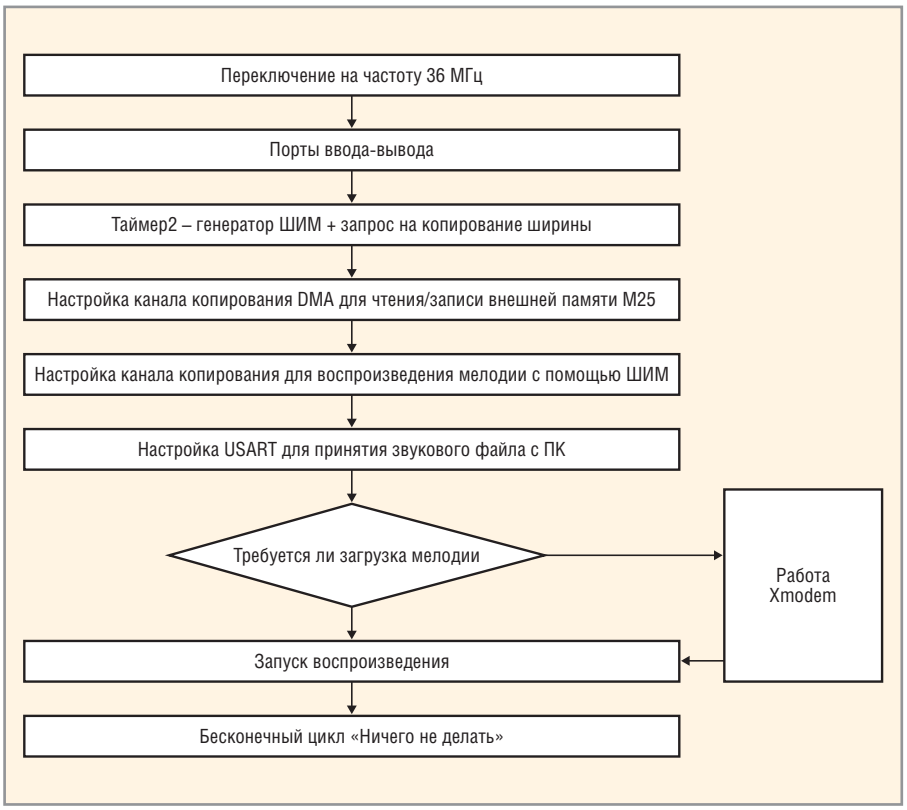

Рис. 4. Блок-схема приложения

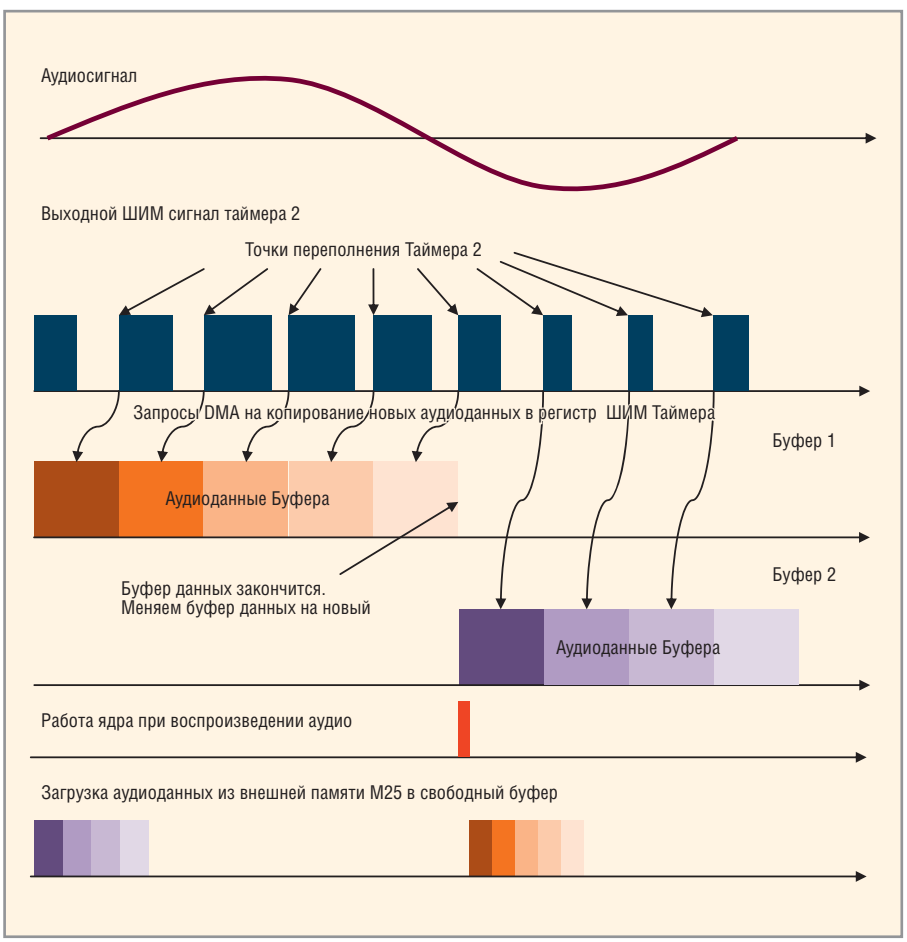

Рис. 5. Программно-аппаратный алгоритм работы

Заключение

Мы рассмотрели работу связки микроконтроллера с внешней последовательной памятью на примере микросхемы M25. Если возникнет задача подключить другой тип внешней памяти, то это можно достаточно легко сделать на основе приведённого материала. Вероятное отличие будет в размере сектора данных, интерфейсе подключения и системе команд. Кроме того, инженеры компании STMicroelectronics выложили в открытый доступ исходные файлы работы с картами памяти SD через интерфейс SPI и SDIO, а также работу с памятью M24 (eeprom), подключенной через интерфейс I²C. Задача программиста в этом случае состоит в правильном соединении готового программного стека и своего проекта.

Благодаря лёгкой интеграции микросхем внешней памяти в новые проекты, разработчики получили дополнительную возможность оптимизации своих изделий без потери времени на их освоение.