# Проектирование с использованием процессоров Analog Devices. Первый проект для EZ-KIT Lite

Александр СОТНИКОВ

**В этой статье мы перейдем от моделирования к работе с реальными аппаратными средствами и создадим простой проект для платы EZ-KIT Lite, который будет использоваться в качестве каркаса для последующих проектов.**

Оценочные платы EZ-KIT Lite и EZ-KIT Board выпускаются компанией Analog Devices для типичных представителей каждого из подсемейств процессоров и предназначены для освоения навыков работы с процессором, а также могут использоваться для создания прототипов полнофункциональных систем. В качестве аппаратной платформы для экспериментов мы будем использовать одну из новых оценочных плат компании Analog Devices — ADSP-BF527 EZ-KIT Lite, предназначенную для освоения навыков работы с процессором Blackfin ADSP-BF527(C). Структурная схема платы изображена на рис. 1.

Плата содержит все необходимые для работы процессора вспомогательные цепи, включая стабилизаторы напряжения и генераторы тактовых сигналов. Подсистема памяти на плате EZ-KIT Lite помимо внутренней статической памяти самого процессора включает в себя следующие компоненты:

- 64 Мбайт (32M×16 бит) синхронной динамической памяти с максимальной частотой доступа 133 МГц.
- 4 Мбайт (2M×16 бит) асинхронной флэш-памяти. Асинхронная флэш-память отображена в банках ASYNC0–ASYNC3 пространства памяти (диапазон адресов 0x20000000–0x203FFFFF) и содержит в себе предварительно записанные программы мигания светодиодом, вывода изображений на ЖК-дисплей и автоматического теста POST (power-on self test).
- 4 Гбит (512M×8 бит) NAND флэш-памяти, которые подключены к выводам внутреннего контроллера NAND флэш-памяти процессора.
- 16 Мбит последовательной флэш-памяти, подключенной к процессору по шине SPI. В SPI-память также предварительно записаны программы мигания светодиодом и POST.

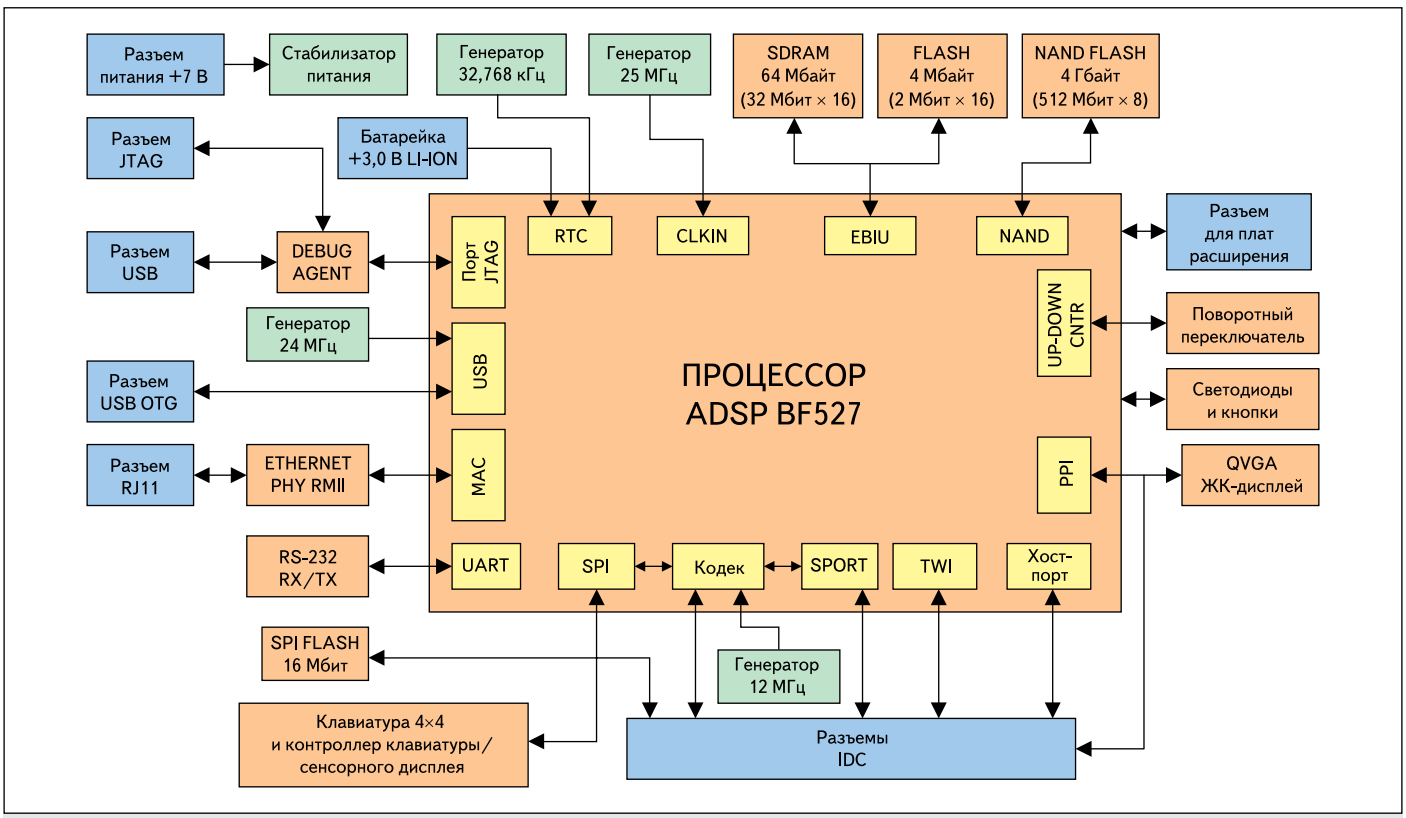


**Рис. 1.** Структурная схема платы ADSP-BF527 EZ-KIT Lite

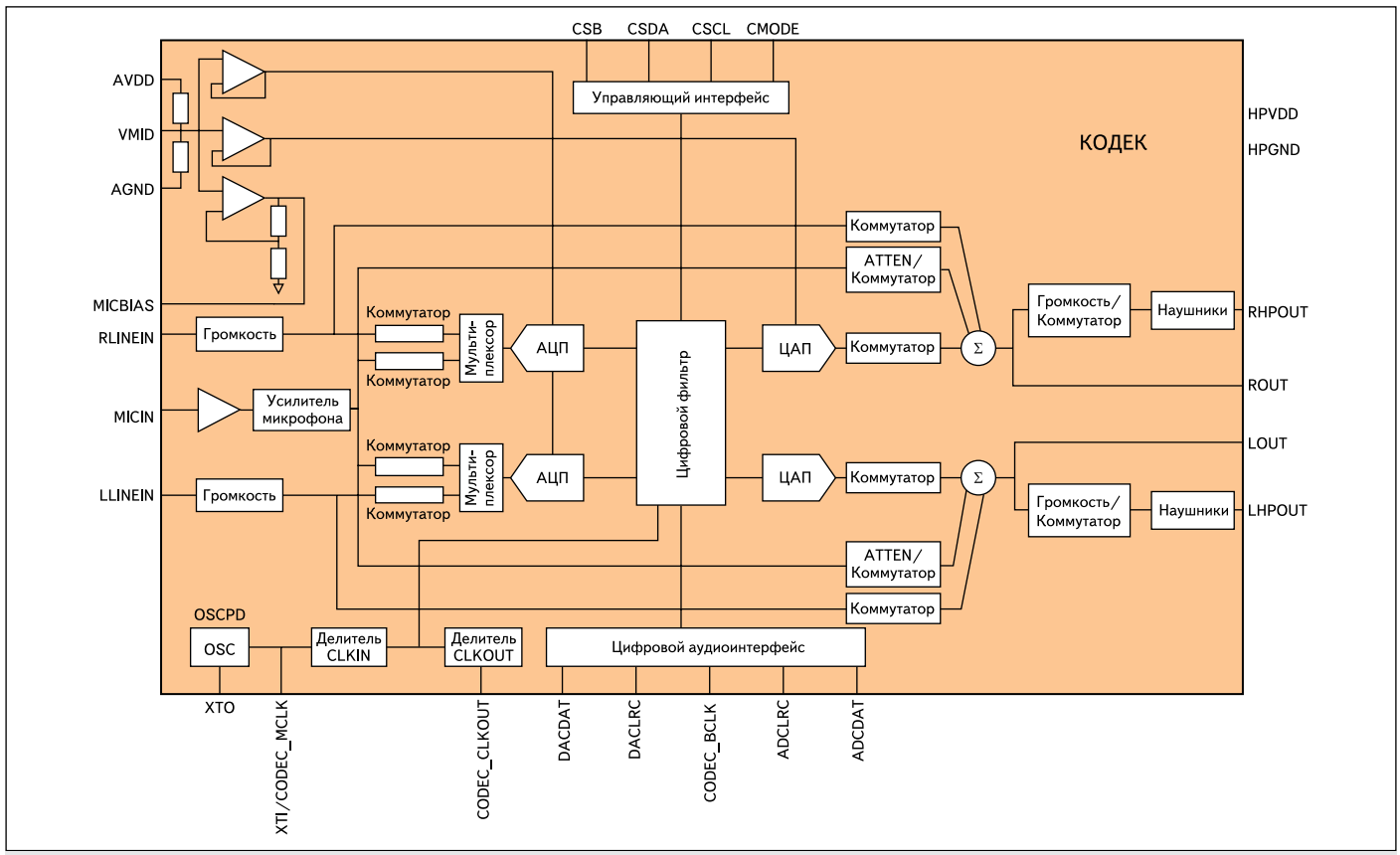


**Рис. 2.** Структурная схема кодека BF52xC

Помимо внешней памяти, на плате имеется также ряд внешних периферийных компонентов:

- Модуль TFT ЖК-дисплея Varitronix VL_PS_COG_T350MCQB. Этот дисплей с диагональю 3,5 дюйма и разрешением 320×240 имеет сенсорную панель и последовательно-параллельный интерфейс RGB-888, работающий от PPI-порта процессора.
- Контроллер клавиатуры и сенсорного экрана MAX1233, подключенный к SPI-порту процессора и обеспечивающий работу с сенсорной панелью ЖК-дисплея и расположенной на плате клавиатурой 4×4.
- Поворотный переключатель, который подключен к интегрированному реверсивному счетчику процессора через специальный трехпроводный интерфейс.
- Контроллер физического уровня (PHY) Ethernet MAC 10/100 LAN8700, соединенный с интегрированным контроллером Ethernet MAC процессора. MAC-адрес платы хранится в однократно программируемой памяти (OTP) процессора и указан на наклейке на обратной стороне платы.
- Аудиоинтерфейс для внутреннего кодека процессора, состоящий из двух входов (линейный стереофонический и микрофонный монофонический) и двух выходов (линейный стереофонический и микрофонный).
- Драйвер/приемник RS-232 ADM3203, подключенный к порту UART1 процессора.
- Литиевая батарейка 3 В на 280 мА/ч и генератор 32,768 кГц, необходимые для работы модуля часов реального времени (RTC).
- Кнопки и светодиоды, позволяющие подавать внешние воздействия и осуществлять индикацию с использованием линий ввода/вывода общего назначения (GPIO).
- Интерфейсная схема USB для внутреннего контроллера USB On-The-Go (OTG).
- Разъемы для подключения к плате внешних устройств.

Загрузку программ в процессор и их отладку в среде VisualDSP++ можно выполнять через интегрированный на плате контроллер Debug Agent или внешний эмулятор, приобретаемый отдельно.

## Внутренний кодек процессора ADSP-BF527C

Особенностью процессора ADSP-BF527C является наличие в одном корпусе с ним внутреннего стереокодека, что делает его очень интересным с точки зрения создания встраиваемых систем с обработкой звука. Этот кодек, структурная схема которого показана на рис. 2, аналогичен серийно выпускаемому кодеку SSM2603 и отличается от последнего только отсутствием функции автоматической регулировки уровня. Как следует из приведенной структурной схемы, кодек имеет два канала АЦП, подключаемых к линейному или микрофонному входам, и два канала ЦАП, сигнал с которых выдается на линейный выход или наушники. Сигналы с частотой дискретизации ЦАП и АЦП формируются от внешнего или внутреннего генератора. При работе с внешним генератором опорного тактового сигнала кодек поддерживает два режима: нормальный (для работы с тактовыми генераторами, имеющими одну из стандартных частот, например, 12,288 или 24,576 МГц) и режим USB (для работы с тактовым генератором с частотой 12 МГц, используемой в стандарте USB). Частоты дискретизации формируются путем деления тактовой частоты генератора опорного тактового сигнала и могут принимать значения в диапазоне от 8 до 96 кГц.

Для ввода/вывода оцифрованных звуковых сигналов кодек имеет 5-проводный интерфейс с программно настраиваемым форматом данных, который без дополнительных компонентов подключается к последовательному порту SPORT.

Настройка всех параметров кодека, включая конфигурацию аналогового тракта, задание частот дискретизации и выбор формата цифрового звукового потока, осуществляется через 3-проводный интерфейс, который может быть подключен к порту SPI или TWI. Выбрать тип подключения можно при помощи внешнего вывода CMODE. Пример соединения внешних выходов процессора и кодека при управлении через порт SPI показан на рис. 3.

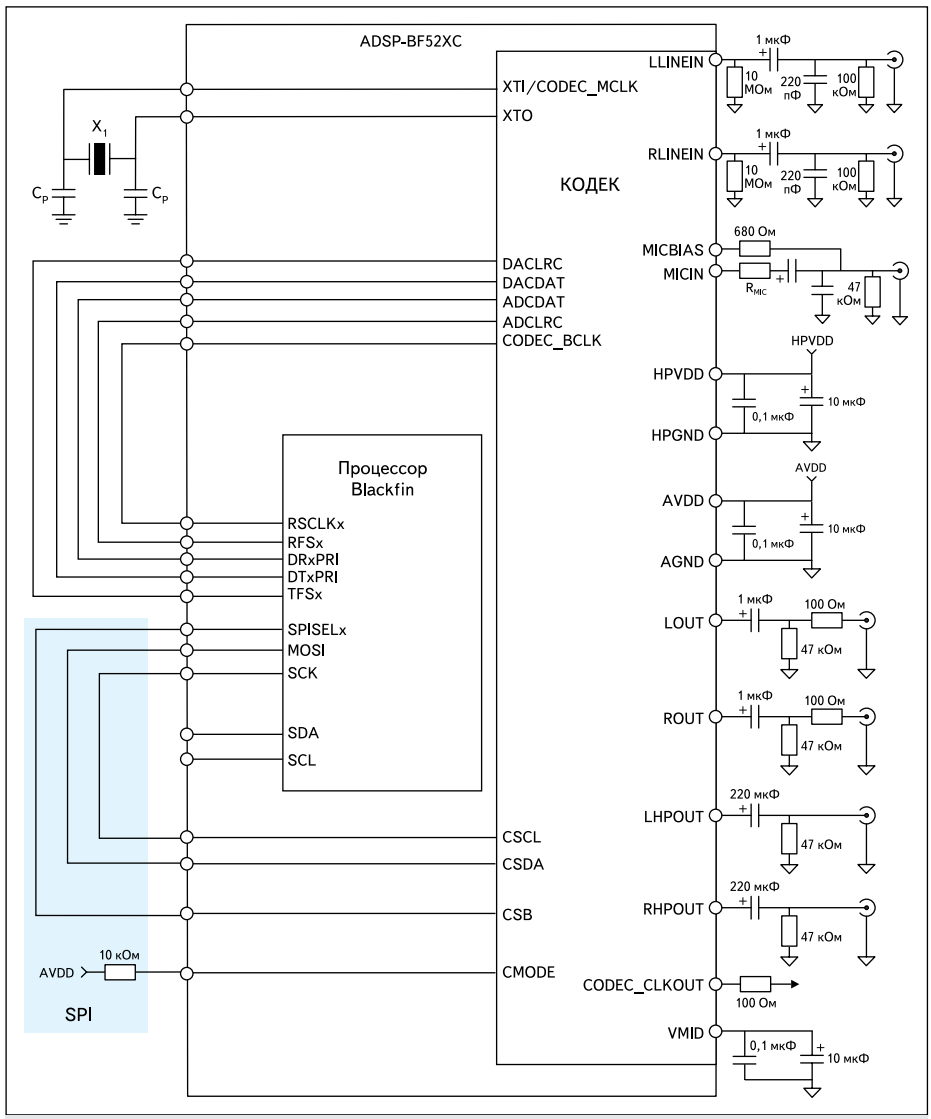


**Рис. 3.** Соединение кодека с процессором при управлении через порт SPI

В состав пакета VisualDSP++ 5.0 входит пример, иллюстрирующий настройку и обмен данными с внутренним кодеком. Как и большинство других примеров, поставляемых с современными платами EZ-KIT и EZ-BRD, он основан на использовании готовых драйверов устройств и библиотек системных служб. Подобные примеры, несомненно, весьма полезны и позволяют быстро собрать работоспособное приложение на основе готовых решений, однако не очень подходят для начального освоения принципов работы с процессором. Поэтому мы создадим с нуля собственный альтернативный проект, просто транслирующий аудиосигнал с входа кодека на его выход, который будет использоваться в дальнейшем в качестве каркаса при разработке более сложных приложений.

Для создания проекта в VisualDSP++ запустите «мастер» ***Project Wizard***, задайте в нем путь к каталогу и название нового проекта, укажите в качестве платформы процессор ADSP-BF527 и выберите автоматическое создание шаблона исходного файла на языке C без комментариев и поддержки аргументов командной строки. В последующих разделах мы рассмотрим основные операции, которые необходимо будет добавить в исходный файл программы.

## Управление кодеком

Для настройки параметров аналогового тракта и цифрового интерфейса кодек имеет 10 внутренних регистров, которые перечислены в таблице 1. Подробное описание назначения отдельных битов в регистрах кодека можно найти в техническом описании процессоров ADSP-BF52xC [1]. Как уже отмечалось выше, доступ к регистрам кодека может, в зависимости от заданной аппаратной конфигурации, осуществляться с использованием порта SPI или TWI. На плате ADSP-BF527 EZ-KIT Lite доступны оба варианта (выбираются при помощи переключателей), и в нашем проекте будет использоваться первый из них.

Интерфейс SPI (Synchronous Peripheral Interface) представляет собой синхронный последовательный канал передачи данных, который широко используется для задач управления в микропроцессорных системах. Он включает в себя четыре линии: две линии данных (***MOSI*** и ***MISO***), линию выбора устройства (***SPISS***) и линию стробированного (то есть активного только в процессе обмена данными) сигнала тактовой синхронизации (***SCK***). Процессор, в зависимости от выбранной программно конфигурации порта, может быть на шине ***SPI*** ведущим или ведомым. Наличие двух выводов данных позволяет обеспечивать полнодуплексное взаимодействие между SPI-совместимыми устройствами. В режиме ведущего процессор передает данные по линии ***MOSI***, осуществляет прием данных по линии ***MISO*** и является источником сигнала ***SCK***. Для возможности взаимодействия сразу с несколькими ведомыми устройствами процессор в режиме ведущего может использовать до семи внешних линий выбора ведомого ***SPISSEL1-7***. В режиме ведомого назначение линий ***MOSI*** и ***MISO*** меняется, линия ***SCK*** работает на вход, а логика порта активируется сигналом ***SPISS***.

**Таблица 1.** Регистры кодека ADSP-BF52xC

| Адрес | Назначение |
|---|---|
| 0x00 | Управление громкостью на входе АЦП левого канала |
| 0x01 | Управление громкостью на входе АЦП правого канала |
| 0x02 | Управление громкостью ЦАП левого канала |
| 0x03 | Управление громкостью ЦАП правого канала |
| 0x04 | Управление аналоговым трактом звукового сигнала |
| 0x05 | Управление цифровым трактом звукового сигнала |
| 0x06 | Управление энергопотреблением |
| 0x07 | Управление форматом цифрового аудиоинтерфейса |
| 0x08 | Выбор частоты дискретизации |
| 0x09 | Запуск кодека |
| 0x0F | Программный сброс кодека |

Структурная схема порта ***SPI*** процессора Blackfin изображена на рис. 4. Сердцем порта является сдвиговый регистр, через который синхронно с сигналом SCK осуществляется побитный ввод/вывод данных. Доступ к нему производится через 16-битные регистры приема (***SPI_RDBR***) и передачи (***SPI_TDBR***) либо непосредственно под управлением ядра, либо в режиме DMA. Во втором случае обмен данными между контроллером DMA и регистрами приема/передачи происходит через четырехэлементный буфер FIFO.

Когда процессор является ведущим устройством на шине SPI, частота сигнала тактовой синхронизации определяется содержимым регистра ***SPI_BAUD*** в соответствии с формулой:

$$F_{SCK} = F_{SCLK}(2\times\text{SPI\_BAUD}),$$

где $F_{SCLK}$ — частота тактового сигнала системы (частота, на которой работают синхронные периферийные модули процессора). Минимальное значение, которое может быть записано в регистр ***SPI_BAUD***, равно двум, и, следовательно, максимальная частота ***SPI_CLK*** равна одной четвертой от $F_{SCLK}$.

В режиме ведомого сигнал ***SCK*** является входным, и содержимое регистра ***SPI_BAUD*** игнорируется. При этом сигнал ***SCK*** будет иметь значение только при активном (низком) уровне сигнала ***SPISS***.

Полярность и фаза сигнала ***SCK*** по отношению к битам данных, наряду с другими параметрами интерфейса, задаются при помощи регистра управления ***SPI_CTL***, структура которого показана на рис. 5.

Регистр ***SPI_FLG*** необходим при работе в режиме ведущего. С его помощью осуществляется активация любой комбинации из восьми возможных выводов программируемых флагов для выбора ведомых устройств и программное управление ими. Запись единицы в один из битов младшего байта 16-битного регистра ***SPI_FLG*** разрешает использование соответствующего вывода флага для выбора ведомого, а биты старшего байта регистра управляют состоянием сигналов на этих выводах. Для работы с кодеком на плате ADSP-BF527 EZ-KIT Lite используется вывод SPISSEL5. Чтобы разрешить модулю SPI работать с сигналом ***SPISSEL5***, необходимо установить бит 5 ***SPI_FLG***, а поскольку данный сигнал имеет активный низкий уровень, во избежание ложных срабатываний при инициализации порта в бит 12 ***SPI_FLG*** должна быть также записана единица:

```
*pSPI_FLG=0xFF20;
```

При управлении кодеком порт SPI процессора является ведущим. Поскольку кодек имеет небольшое число регистров, которые в рассматриваемом примере достаточно проинициализировать один раз в начале программы, большого быстродействия не требуется, и пересылка данных через порт SPI будет выполняться под управлением ядра. Временная диаграмма записи в кодек по интерфейсу SPI изображена на рис. 6 (обозначения на рисунке соответствуют названиям сигналов кодека). Стоит особо отметить, что сигнал ***CSB*** кодека, соединяемый с сигналом ***SPISSEL5***, на самом деле используется не для активации логики кодека, а для копирования данных из сдвигового регистра в выбранный внутренний регистр кодека (защелкивание данных происходит по переднему фронту ***CSB***).

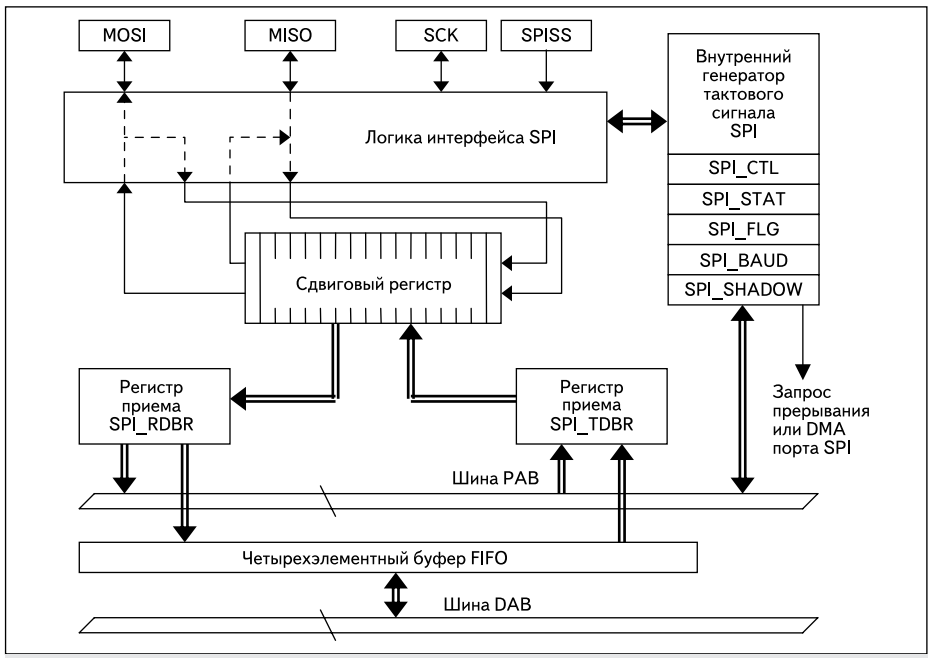


**Рис. 4.** Структурная схема порта SPI процессора Blackfin

В соответствии с этой временной диаграммой в регистре ***SPI_CTL*** необходимо выбрать следующие параметры интерфейса SPI:

- Формат данных — 16 бит (***SIZE*** = 1), передача начинается с MSB (***LSBF*** = 0).
- Активный уровень SCK — низкий (***CPOL*** = 0).

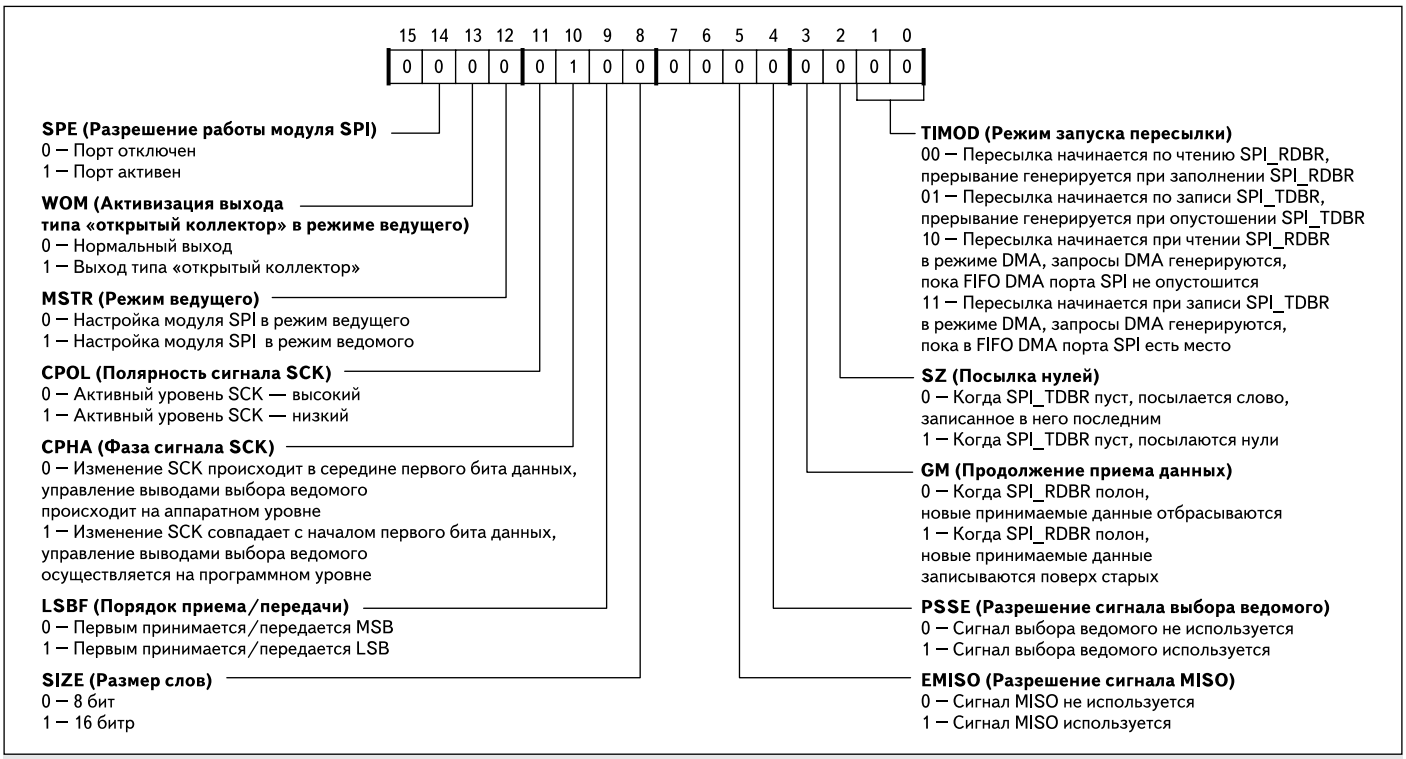


**Рис. 5.** Назначение полей регистра SPI_CTL

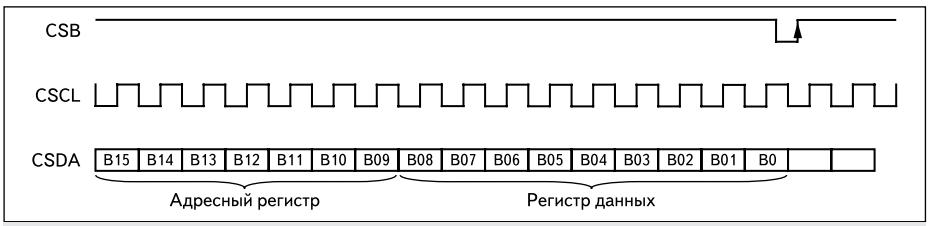


**Рис. 6.** Временная диаграмма записи в кодек по шине SPI

- Первое переключение SCK совпадает с началом первого бита данных, сигнал выбора ведомого управляется программно (***CPHA*** = 1).

Поскольку порт ***SPI*** процессора в рассматриваемом примере работает в режиме ведущего, бит ***MSTR*** необходимо установить в единицу, а биты ***EMISO*** и ***PSSE*** должны быть очищены. Состояние битов ***SZ*** и ***GM*** в этой конфигурации не имеет значения. Для инициирования каждой новой пересылки слова данных будет использоваться запись в регистр ***SPI_TDBR*** порта процессора (***TIMOD*** = 01).

Таким образом, для инициализации порта ***SPI*** на запись данных в кодек и активации порта необходимо добавить в проект строку вида:

```
*pSPI_CTL=0x5D01;
```

Так как параллельно с процедурой настройки кодека других действий в программе выполняться не будет, нет необходимости работать с портом по прерываниям. Для проверки готовности порта к передаче нового слова в нашем примере можно использовать программный опрос флагов регистра состояния ***SPI_STAT***, а именно флага ***TXS***, который указывает состояние буфера данных SPI (бит 4), и флага ***SPIF***, который сигнализирует об окончании пересылки слова данных (бит 1).

Код функции на языке С, которая осуществляет запись в регистр кодека:

```
void write_codec_reg(short addr, short data)
{
    *pSPI_TDBR=((addr&0xf)<<9)|(data&0x1ff);
    while(*pSPI_STAT&0x08||(!(*pSPI_STAT&0x01)));
    *pSPI_FLG=0xDF20;
    *pSPI_FLG=0xFF20;
}
```

Последние две строчки в функции формируют короткий импульс активного низкого уровня на линии ***SPISSEL5***, который используется для защелкивания данных во внутреннем регистре кодека.

## Прием/передача данных

Как уже отмечалось выше, для обмена оцифрованными звуковыми сигналами с процессором в кодеке имеется пятивыводный цифровой интерфейс, который без применения дополнительной внешней логики может взаимодействовать с портом SPORT. Форматы передачи данных выбираются пользователем программно. Всего возможны четыре варианта формата: стандартный формат последовательного порта с сигналом кадровой синхронизации (с опережением или запаздыванием относительно первого бита в слове данных), а также три характерных для цифровых стереофонических аудиосистем формата (left-justified, right-justified и I²S). В форматах второго типа вместо сигнала кадровой синхронизации применяется периодический сигнал, фронты которого выделяют данные левого и правого каналов стереосигнала. Примеры временных диаграмм формата с кадровой синхронизацией и формата I²S показаны на рис. 7.

Чтобы максимально упростить себе задачу, мы будем использовать в проекте формат данных с кадровой синхронизацией, показанный на рис. 7а, с 16 битами данных на каждый канал. При этом, после настройки SPORT0 на работу с 32-битными данными, каждое принимаемое/передаваемое слово будет содержать в старших двух байтах отсчет, соответствующий левому каналу, а в младших двух байтах — отсчет, соответствующий правому каналу. Источником сигналов кадровой и тактовой синхронизации будет кодек. Для реализации временной диаграммы в соответствии с рис. 7а необходимо настроить следующие параметры в регистрах управления SPORT0 (***SPORT0_RCR1,2*** и ***SPORT0_TCR1,2***):

- Режим кадровой синхронизации с запаздыванием (***LARFS*** = ***LATFS*** = 1).
- Импульс кадровой синхронизации необходим для каждого передаваемого/принимаемого слова (***RFSR*** = ***TFSR*** = 1).
- Внешняя кадровая синхронизация (***IRFS*** = ***ITFS*** = 0).
- Внешняя тактовая синхронизация (***IRCLK*** = ***ITCLK*** = 0).
- Разрядность слова — 32 бита (***SLEN*** = 31).
- Передача/прием, начиная со старшего бита (***RLSBIT*** = ***TLSBIT*** = 1).
- Активный фронт сигнала тактовой синхронизации — задний (***RCKFE*** = ***TCKFE*** = 1).

Таким образом, для инициализации порта SPORT0 на работу с кодеком в проект нужно добавить следующие строки:

```
*pSPORT0_RCR1=0x6410;
*pSPORT0_RCR2=0x001f;
*pSPORT0_TCR1=0x6410;
*pSPORT0_TCR2=0x001f;
```

Так как создаваемый нами проект не будет делать ничего, кроме копирования данных из регистра приема SPORT0 в регистр передачи SPORT0, для этой задачи целесообразно использовать прерывание приема порта по каждому слову данных. Процедура настройки прерывания SPORT обсуждалась в предыдущей статье, поэтому подробно описывать ее здесь не будем. Подпрограмма обслуживания прерывания крайне проста и имеет следующий вид:

```
EX_INTERRUPT_HANDLER(Sport0_RX_ISR):

{
    int x=0;
    x=*pSPORT0_RX;
    *pSPORT0_TX=x;
}
```

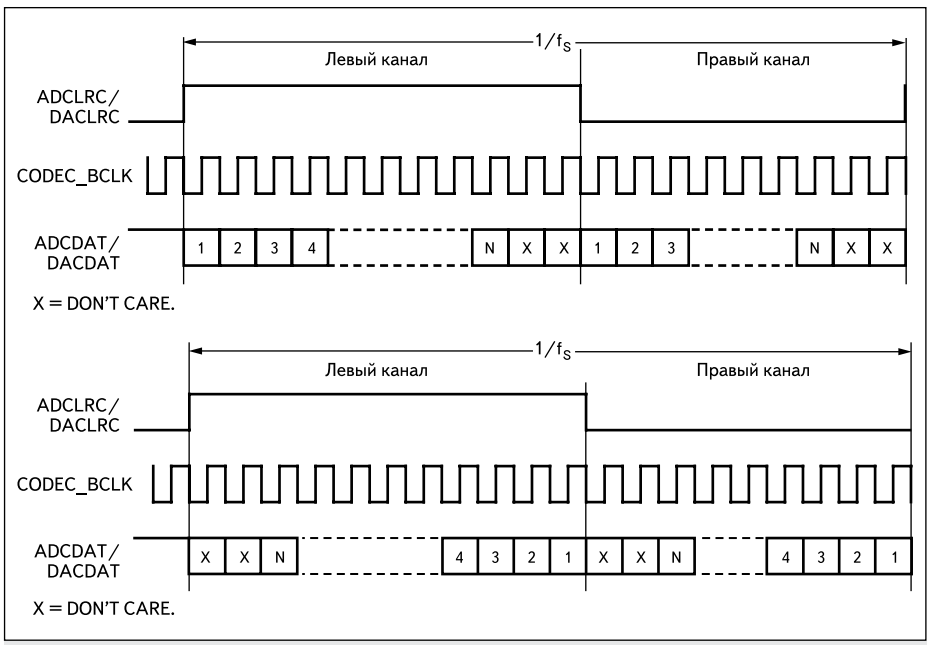


**Рис. 7.** Временные диаграммы: а) формата с кадровой синхронизацией (с запаздыванием) и б) формата I²S

## Система мультиплексирования сигналов периферийных модулей

Процессоры Blackfin обладают богатым набором интегрированных периферийных модулей, и для сохранения приемлемого количества внешних выводов на корпусе (и, следовательно, уменьшения его размера и стоимости) во всех процессорах семейства, начиная с ADSP-BF537, применена гибкая схема мультиплексирования. Большинство выводов корпуса, не относящихся к сигналам питания, управления и интерфейса памяти, сгруппированы в несколько портов общего назначения (в процессорах ADSP-BF52x таких портов 4 — F, G, H и J). Выводы портов, за редким исключением, могут работать в одном из двух режимов: режиме ввода/вывода общего назначения (General Purpose Input/Output, GPIO) или в режиме альтернативной функции (сигнала периферийного модуля).

По умолчанию все выводы портов работают в режиме GPIO. Чтобы подключить к внешнему выводу порта интерфейсный сигнал периферийного модуля, необходимо установить определенный бит в регистре ***PORTx_FER*** (***x*** — имя порта). Некоторые из выводов портов мультиплексированы между несколькими модулями периферии. В таких случаях периферийный модуль, подключаемый к выводу процессора, выбирается при помощи регистра управления мультиплексором (***PORTx_MUX***).

Сигналы порта SPI процессора доступны через порт G, а порта SPORT0 — через порт G (этот вариант используется на плате ADSP-BF527 EZ-KIT Lite) или порт F. Структуру порта G поясняет таблица 2. Единственный используемый в проекте сигнал, который доступен не через порт G, — это сигнал выбора ведомого ***SPISSEL5***, работа с которым осуществляется через порт H.

В соответствии с таблицей 2 для мультиплексирования сигналов SPI и SPORT0 на выводы порта G необходимо записать в биты [1:0] регистра ***PORTG_MUX*** значение “10”, в биты [5:4], [7:6] и [9:8] значение “01”, а в биты [13:12] значение “00”. Для активации альтернативной функции соответствующих выводов порта G также необходимо записать в регистр ***PORTG_FER*** значение 0x87DC. Для мультиплексирования и активации сигнала ***SPISSEL5*** достаточно установить бит 9 в регистре ***PORTH_FER*** (отвечающие за выбор данного сигнала биты [5:4] регистра ***PORTH_MUX*** по умолчанию имеют требуемое нулевое значение).

**Таблица 2.** Схема мультиплексирования порта G

| PORTG_MUX | 00 | 01 | 10 | 11 | |
|---|---|---|---|---|---|
| | 1-я функция | 2-я функция | 3-я функция | 4-я функция | Вывод GPIO |
| | | | | | PG0 |
| Биты [1:0] | SPI SS | | SPI SSEL1 | – | PG1 |
| | SPI SCK | | SPI SCK | – | PG2 |
| | SPI MISO | DR0SEC | SPI MISO | – | PG3 |
| | SPI MOSI | DT0SEC | SPI MOSI | – | PG4 |
| Биты [3:2] | TMR1/ PPI FS2 | TMR2 | TMR1/ PPI FS2 | – | PG5 |
| | DT0PRI | | PPI FS3 | – | PG6 |
| Биты [5:4] | TMR3 | DR0PRI | UART0 TX | – | PG7 |
| | TMR4 | RFS0 | UART0 RX | – | PG8 |
| Биты [7:6] | TMR5 | RSCLK0 | – | – | PG9 |
| Биты [9:8] | TMR6 | TSCLK0 | – | – | PG10 |
| Биты [11:10] | TMR7 | TMR7 | HOST WR | – | PG11 |
| | DMAR1 | UART1 TX | HOST ACK | – | PG12 |
| | DMAR0 | UART1 RX | HOST ADDR | – | PG13 |
| Биты [13:12] | TSCLK0 | RMII MDC | HOST RD | – | PG14 |
| | TFS0 | RMII PHYINT | HOST_CE | – | PG15 |

Таким образом, исходный код программы принимает окончательный вид:

```
#include <cdefBF52x_base.h>
#include <ccblkfn.h>
#include "sys\exception.h"

void write_codec_reg(short addr, short data);
volatile int x;

EX_INTERRUPT_HANDLER(Sport0_RX_ISR)
{
    x=*pSPORT0_RX;
    *pSPORT0_TX=x;
}

void main()
{
    //Ports setup
    *pPORTG_MUX=0x0152;
    *pPORTG_FER=0x87DC;
    *pPORTH_FER=0x0200;

    //SPI setup
    *pSPI_BAUD=0x07FF;
    *pSPI_FLG=0xFF20;
    *pSPI_CTL=0x5D01;

    //Codec initialization
    //Connect microphone input to the ADC, connect DAC to the chip output
    //Digital interface parameters: 16-bit processor mode, clock master,
    //codec BCLK not inverted
    //Sampling rate 48 kHz
    write_codec_reg(0xf,0);
    write_codec_reg(0x6,0xfd);
    write_codec_reg(0x4,0x0d);
    write_codec_reg(0x0,0x17);
    write_codec_reg(0x1,0x17);
    write_codec_reg(0x2,0x79);
    write_codec_reg(0x3,0x79);
    write_codec_reg(0x6,0x41);
    write_codec_reg(0x4,0x15);
    write_codec_reg(0x5,0);
    write_codec_reg(0x7,0x43);
    write_codec_reg(0x8,0x01);

    /* Sport set up for frame sync mode */
    *pSPORT0_RCR1=0x6410;
    *pSPORT0_RCR2=0x001f;
    *pSPORT0_TCR1=0x6410;
    *pSPORT0_TCR2=0x001f;

    //Interrupt initialization
    register_handler(ik_ivg9, Sport0_RX_ISR);
    *pSIC_IMASK0 = 0x00010000;
    //SPORT enable
    *pSPORT0_TCR1 |= 1;
    *pSPORT0_RCR1 |= 1;
    //Codec enable
    write_codec_reg(0x9,0x1);

    while(1);
}

void write_codec_reg(short addr, short data)
{
    *pSPI_TDBR=((addr&0xf)<<9)|(data&0x1ff);
    while(*pSPI_STAT&0x08||(!(*pSPI_STAT&0x01)));
    *pSPI_FLG=0xDF20;
    *pSPI_FLG=0xFF20;
}
```

## Загрузка и отладка проекта

Для загрузки программы в процессор и ее отладки необходимо соединить плату EZ-KIT Lite с ПК при помощи кабеля USB и настроить в среде VisualDSP++ соответствующую сессию. Настройка сессии отладки через ***Debug Agent*** осуществляется через «мастер» создания сессий (***Session Wizard***), который доступен через меню ***Session → New Session*** или по нажатию кнопки ***New Session*** в диалоговом окне Session List, вызываемом при помощи меню ***Session → Session List***. В «мастере» создания сессий следует выбрать процессор ADSP-BF527, тип подключения EZ-KIT Lite и платформу ADSP-BF527 EZ-KIT Lite via Debug Agent.

Активация сессии, которая доступна через меню ***Session → Select Session***, возможна только при включенном питании платы. Если на вкладке General диалогового окна Preferences (меню ***Settings → Preferences***) установлен флажок ***Load executable after build***, то в случае успешной компоновки проекта исполняемый файл будет автоматически загружен в память процессора. При этом, если на той же вкладке установлен флажок ***Run to main after load***, то программа автоматически запустится на исполнение и остановится на первой команде функции main, минуя процедуру начальной настройки среды окружения. Дальнейший процесс отладки похож на отладку в симуляторе: пользователь может устанавливать точки останова, работать в режиме пошагового исполнения, просматривать содержимое регистров и т. д.

Перед тем как запускать проект, необходимо установить на плате EZ-KIT (ревизии 2.1 и старше) следующую конфигурацию джамперов и переключателей:

```
JP6: 1/2
SW4: 1(OFF), 2(ON), 3(OFF), 4(OFF)
SW8: 1(OFF), 2(OFF), 3(ON), 4(OFF)
SW13: 1(ON), 2(ON), положение остальных переключателей не имеет значения
SW19: 1(ON), 2(OFF), 3(ON), 4(OFF)
SW20: 1(ON), 2(ON), 3(ON), 4(ON)
SW23: 1(ON), 2(ON), 3(ON), 4(ON)
```

После того как выбрана нужная конфигурация, можно подключить к разъемам на плате наушники и микрофон и запустить программу на исполнение. Результатом работы программы будет просто то, что вы услышите в наушниках сигнал с микрофона.

Созданный нами проект сам по себе не несет большой практической ценности, однако он может использоваться в качестве каркаса для создания более сложных приложений с обработкой звука при помощи внутреннего кодека процессора ADSP-BF527C. В последующих статьях мы модифицируем проект и добавим в него новые возможности.

## Литература

1. ADSP-BF522C/ADSP-BF523C/ADSP-BF524C/ADSP-BF525C/ADSP-BF526C/ADSP-BF527C Blackfin Embedded Processor with Codec Datasheet — www.analog.com