NeoGS
Руководство программиста
(fpgaD xx.yy.200z, txtrev.A)

Оглавление

1 Введение

1.1 Общая информация

NeoGS – высокопроизводительная звуковая карта для spectrum-совместимых компьютеров с шиной ZX-BUS, имеющая в своём составе 2 мегабайта ОЗУ, 512 килобайт ПЗУ (flash-память), процессор Z80, работающий на частоте до 24 мегагерц, 8 каналов семплированного звука, MP3-декодер и разъём SD-карты. Сердцем карты является программируемая логическая интегральная схема (ПЛИС или FPGA) типа acex ep1k30 ёмкостью 1728 логических элементов.

1.2 Структура NeoGS

Структурная схема NeoGS приведена на рис.1. Как следует из рисунка, сердцем карты является FPGA, которая управляет всеми аспектами функционирования карты, а именно:

- обеспечивает интерфейс с ZX-BUS
- обеспечивает функционирование процессора Z80, управляет памятью устройства
- выдаёт периодические прерывания на Z80
- выдаёт семплированный звук на аудиоЦАП
- управляет под контролем процессора SD-картой и MP3-декодером

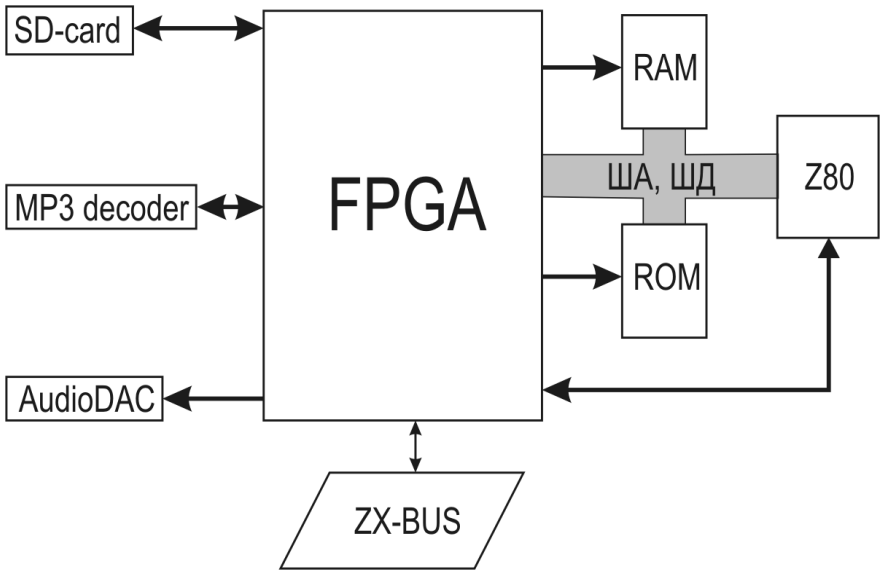

рис. 1: Структурная схема NeoGS

С точки зрения программиста, управление 'железом' NeoGS осуществляется процессором через массив портов ввода-вывода, физически расположенных в FPGA. Кроме того, являясь устройством ZX-BUS, карта имеет набор портов в пространстве портов компьютера с ZX-BUS, служащих для управления и обмена данными с устройством.

Подробное описание всей аппаратуры карты и соответствующих портов представлено в следующих главах.

1.3 Обозначения

На протяжении всего руководства адреса любых задействованных портов, константы и

биты обозначаются исключительно символически. Для использования этих обозначений в программах необходимо включить в исходный код файл `ports.inc`. Кроме того, этот файл может рассматриваться как сверхкраткая памятка по портам.

Биты обозначаются символическими именами вида `B_bitname`. Данное имя определено в `ports.inc` как номер бита (биты нумеруются начиная с младшего, чей номер 0). Кроме того, для каждого бита `B_bitname` определена его маска `M_bitname`, содержащая единицу в позиции бита. Например:

```
B_SETNCLR       equ  7
M_SETNCLR       equ  #80
```

Также в данном руководстве используется обозначение вида (1<<`B_bitname`), эквивалентное обозначению `M_bitname`. Обозначение (0<<`B_bitname`) тождественно нулю.

Константы, определяющие то или иное предопределённое значение для записи в порт, обозначаются именами вида `C_constantname`.

2 Взаимодействие с ZX

2.1 Интерфейс

Обмен данными, а также контроль и управление картой со стороны компьютера происходит через интерфейс ZX-BUS, который представляет собой два набора взаимосвязанных портов: со стороны компьютера и со стороны процессора NeoGS.

Со стороны ZX-BUS имеются следующие порты: `GSCTR`, `GSCOM`, `GSDAT` и `GSSTAT`.

Порт `GSCTR` (write-only) служит для общего контроля NeoGS. При записи в него константы `C_GRST` происходит 'тёплый' сброс процессора и железа NeoGS (без перезагрузки прошивки FPGA). При записи константы `C_GNMI` на процессор NeoGS выдаётся запрос немаскируемого прерывания /NMI. Наконец, при записи константы `C_GLED` светодиод на плате NeoGS меняет своё состояние (горит-не горит). Поведение NeoGS при записи любых других констант не определено и может измениться в новых релизах прошивок.

Порт `GSCOM` (write-only) предназначен для записи в NeoGS т.н. командного байта.

Порт `GSDAT` (read/write) предназначен для передачи в NeoGS или из NeoGS однонаправленного потока данных.

В порту `GSSTAT` (read-only) находятся т.н. командный бит (`B_CBIT`) и бит данных (`B_DBIT`).

Со стороны NeoGS имеются следующие порты: `ZXCMD`, `ZXDATRD`, `ZXDATWR`, `ZXSTAT`, `CLRCBIT`.

Порт `ZXCMD` (read-only) – чтение командного байта, записанного ранее спектрумом в порт `GSCMD`.

Порт `ZXDATRD` (read-only) – чтение байта данных, записанного ранее спектрумом в порт `GSDAT`.

Порт `ZXDATWR` (write-only) – запись байта данных, предназначенного для прочтения спектрумом из порта `GSDAT`. Примечание: байты, читаемые из порта `ZXDATRD`, с одной стороны, и из порта `GSDAT`, с другой, являются физически независимыми байтами. При записи байта в `ZXDATWR` байт, читаемый из `ZXDATRD` не меняется, равно как и не меняется читаемый из `GSDAT` байт при записи другого байта в тот же `GSDAT`.

Порт `ZXSTAT` (read-only) – то же самое, что и в порту `GSSTAT` со стороны спектрума

(B_CBIT и B_DBIT).

Порт CLRCBIT (read/write) – очистка командного бита. Очистка бита происходит по факту любого обращения в этот порт: при записи данные игнорируются, а при чтении данные не определены. Однако как чтение, так и запись обнулят командный бит.

Порт GSCOM предназначен для подачи NeoGS какой-либо однобайтовой команды. При записи данных в этот порт автоматически устанавливается в 1 командный бит. Со стороны NeoGS записанный байт может быть прочитан из порта ZXCMD, состояние командного бита — выяснено из порта ZXSTAT. Очистка командного бита происходит при обращении к порту CLRCBIT.

Бит данных используется для синхронизации обмена потоком данных через порты GSDAT (со стороны спектрума), а также ZXDATWR и ZXDATRD (со стороны NeoGS). Данный бит устанавливается в 1 при записи данных спектрумом в порт GSDAT, а также при записи данных NeoGS в порт ZXDATWR. Бит обнуляется при чтении спектрумом порта GSDAT, а также при чтении NeoGS порта ZXDATRD.

2.2 Примеры сценариев взаимодействия

Типичный сценарий посылки в NeoGS однобайтовой команды и ожидания её завершения:

1. Спектрум пишет байт команды в GSCOM
2. NeoGS видит установленный командный бит, считывает команду из ZXCMD и выполняет её.
3. Спектрум всё это время ожидает обнуления командного бита
4. NeoGS, закончив обработку команды, обращается к порту CLRCBIT, обнуляя таким образом командный бит.
5. Спектрум видит обнулённый командный бит и определяет завершение выполнения команды.

Типичный сценарий передачи потока данных со спектрума в NeoGS со стороны спектрума:

1. Спектрум пишет байт в GSDAT.
2. В момент записи бит данных устанавливается в 1, разрешая NeoGS подхватить только что записанный байт.
3. Перед посылкой следующего байта спектрум ожидает обнуления бита данных.

Со стороны NeoGS:

1. NeoGS ожидает установки бита данных в 1.
2. NeoGS читает записанный спектрумом байт из порта ZXDATRD.
3. При этом бит данных устанавливается в ноль, разрешая спектруму записать следующий байт.

Аналогично происходит и передача потока данных из NeoGS в спектрум.

Со стороны спектрума:

1. Спектрум ожидает установки бита данных.
2. Спектрум читает принятый байт из порта GSDAT.
3. При этом автоматически обнуляется бит данных, разрешая NeoGS записать новый байт.

Со стороны NeoGS:

1. NeoGS пишет данные в порт ZXDATWR
2. При этом автоматически устанавливается бит данных, разрешая спектруму прочитать только что записанные данные.

3. NeoGS ожидает обнуления бита данных, свидетельствующего о том, что спектрум подхватил только что переданный байт.

Данные сценарии обмена можно комбинировать и модернизировать в соответствии с требованиями программы и объёмом передаваемых данных. Например, если команда требует 1 байт в качестве аргумента, то он записывается в GSDAT до подачи командны. Если команда возвращает 1 байт результата, то он также может быть прочитан из порта GSDAT, будучи записанным в ZXDATWR до обнуления командного бита. В обоих случаях квитирование обмена через бит данных не применяется. Другие примеры можно найти в руководстве к прошивке gs105a.

Советы по использованию:

1. Перед началом обмена потоком байт установите бит данных в подходящее для начала обмена состояние (например в ноль, если планируется передача данных в NeoGS). Установка бита данных производится холостым обращением к порту ZXDATWR или ZXDATRD.
2. Если обмен потоком байт происходит в ответ на подачу команды, проводите действия, описанные в пункте 1 до очистки командного бита. Со стороны спектрума не приступайте к обмену до очистки этого бита.
3. Старайтесь предусмотреть в алгоритмах обмена все маловероятные ситуации. Типичной, хоть и маловероятной ситуацией является сброс спектрума пользователем в момент, когда тот находится в середине алгоритма обмена с NeoGS. Как правило, такое событие приводит к зависанию алгоритма со стороны NeoGS в цикле ожидания командного бита или бита данных. Для предотвращения таких ситуаций можно использовать выход из алгоритма по таймауту.

3 Управление процессором NeoGS

Частота процессора Z80 в карте NeoGS может программно изменяться от 10 до 24 МГц. Управление частотой осуществляется битами B_CKSEL0 и B_CKSEL1 в порту GSCFG0 в соответствии с таблицей:

Частота процессора, МГц	Биты {B_CKSEL1,B_CKSEL0}
10 МГц	{1,1}
12 МГц	{0,1}
20 МГц	{1,0}
24 МГц	{0,0}

Для установки выбранной частоты также могут быть использованы константы C_10MHZ, C_12MHZ, C_20MHZ и C_24MHZ. Пример:

```
IN   A,(GSCFG0)             ;читаем текущее состояние порта
                            ;GSCFG0
AND  #FF-M_CKSEL0-M_CKSEL1  ;удаляем старое значение частоты
OR   C_20MHZ                ;подготавливаем новое
OUT  (GSCFG0),A             ;включаем другую частоту
```

После сброса частота процессора устанавливается равной 10 МГц. Работа стандартной прошивки gs105a проходит на частоте 12 МГц, хотя она нормально работает и при бОльших частотах. При обращениях к ПЗУ (flash-памяти) NeoGS следует устанавливать частоту процессора 10 МГц.

Внимание! В карте NeoGS используется процессор Z84C0020, максимальная рабочая частота которого 20 МГц. Следовательно, не гарантируется устойчивая работа процессора на частоте 24 МГц.

4 Организация памяти

4.1 Типы памяти

На плате NeoGS установлено 2 типа памяти для процессора NeoGS: ПЗУ (flash-память) объёмом 512 килобайт и ОЗУ объёмом 2 либо 4 мегабайта. Размер ОЗУ зависит от ревизии платы (ревизии A и B имеют 2 мегабайта, ревизия C — 4 мегабайта). С программной точки зрения все ревизии отличаются только объёмом установленной памяти. В дальнейшем описывается вариант с 4 мегабайтами памяти. Для случая 2 мегабайт старший бит номера страниц игнорируется (т. е. физическая память имеет 2 образа в адресном пространстве).

ПЗУ применяется для хранения загрузчика FPGA, который исполняется при холодном старте NeoGS, а также для хранения прошивки gs105a, которая при тёплом сбросе копируется сервис-загрузчиком в ОЗУ, откуда и выполняется в дальнейшем. Flash-ПЗУ допускает перезапись содержимого областями по 64 килобайта. Используемые области показаны в таблице:

Номер области ПЗУ	Смещение	Функция
0	$00000-$0FFFF	Сервис-загрузчик gs105a
1	$10000-$1FFFF	Основная прошивка gs105a
2	$20000-$2FFFF	
3	$30000-$3FFFF	
4	$40000-$4FFFF	
5	$50000-$5FFFF	
6	$60000-$6FFFF	
7	$70000-$7FFFF	Загрузчик FPGA

Порядок использования пустых областей не определён.

4.2 Страничная адресация

Процессор Z80, как известно, имеет адресное пространство всего лишь 64 килобайта, что вынуждает применять страничную адресацию для доступа к бОльшим объёмам памяти.

В карте NeoGS каждый тип памяти имеет физическую адресацию в диапазонах $000000-$07FFFF для ПЗУ и $000000-$3FFFFF для ОЗУ. Для процессора память предстаёт в виде страниц по *32 килобайта*. Нумерация страниц начинается с нуля: нулевая страница соответствует физическим адресам в памяти $000000-$007FFF, первая - $008000-$00FFFF и т.д. В некоторых случаях речь идёт также о половинах страниц — первой (смещения внутри страницы $0000-$3FFF) и второй (смещения $4000-$7FFF).

4.3 Управление памятью

Базовое отображение физической памяти на адресное пространство Z80 указано в таблице:

Область адресного пространства Z80	Страницы физической памяти
$0000-$3FFF	первая половина нулевой страницы
$4000-$7FFF	вторая половина первой страницы
$8000-$FFFF	произвольная страница памяти

Номер страницы памяти в области $8000-$FFFF задаётся в порту `MPAG` числом от 0 до 127 (D7=0, D6..D0 — номер страницы).

При установленном бите `B_NOROM` в порту `GSCFG0` во всех вышеперечисленных областях адресного пространства включено ОЗУ и речь идёт о страницах ОЗУ. Если же бит `B_NOROM` сброшен, то в областях $0000-$3FFF и $8000-$FFFF включены страницы ПЗУ, в то время как в области $4000-$7FFF остаётся вышеуказанная страница ОЗУ. При тёплом сбросе бит `B_NOROM` очищается, таким образом процессор стартует из ПЗУ.

При установке бита `B_RAMRO` в порту `GSCFG0` нулевая страница ОЗУ защищается от записи в неё. Данный бит не влияет на возможность записи в нулевую страницу ПЗУ. Возможность защиты от записи используется при размещении в нулевой странице ОЗУ прошивки gs105a.

Установленный бит `B_EXPAG` в порту `GSCFG0` включает режим расширенной адресации страниц. В этом режиме область адресного пространства Z80 $8000-$FFFF разделяется на две подобласти: $8000-$BFFF и $C000-$FFFF, в каждую из которых может быть включена какая-либо половина произвольной страницы. Управление для подобласти $8000-$BFFF осуществляется портом `MPAG`, а для подобласти $C000-$FFFF – портом `MPAGEX`. Бит D7 записываемого в эти порты значения определяет половину страницы, отображаемую в подобласти (D7=0 – первая половина, D7=1 – вторая), в то время как биты D6-D0, как и прежде, определяют номер страницы, из которой выбирается половина. Данную схему можно представить, как адресацию страницами по 16 килобайт, однако номер такой страницы перед записью в порт `MPAG` или `MPAGEX` должен быть сдвинут вправо командной процессора `RRC` (`RRCA`).

При переключении бита `B_EXPAG` актуальная карта отображения страниц в области $8000-$FFFF не меняется; изменения происходят лишь при последующей записи в порты `MPAG`, `MPAGEX`.

Внимание! При обращениях к ПЗУ следует устанавливать частоту процессора 10 МГц.

5 Звук

5.1 Звуковые интерфейсы NeoGS

Карта NeoGS содержит два типа звуковых интерфейсов: интерфейс семплированного звука и интерфейс MP3-декодера. Семплированный звук выводится при помощи аудио-ЦАПа на частоте дискретизации 37500 Гц и может содержать до 8 каналов разрядностью 8 бит с индивидуальной 6-битной громкостью. MP3-декодер поддерживает формат MPEG audio layer 3 с битрейтом до 320 килобит в секунду и частотой дискретизации до 48 кГц.

5.2 Интерфейс семплированного звука

Карта NeoGS выводит семплированный звук при помощи 16-битного стереоаудиоЦАПа TDA1543, работающего с частотой дискретизации 37500 Гц. Однако с точки зрения программиста, NeoGS имеет 4 или 8 8-битных каналов, каждый с индивидуальной 6-битной громкостью. Масштабирование 8-битного отсчёта громкостью, сложение каналов и вывод их в ЦАП происходят автоматически и независимо от процессора NeoGS.

Громкостью является беззнаковое 6-битное число: 0 соответствует минимальной громкости (канал выключен), 63 — максимальной.

Отсчёт является 8-битным беззнаковым либо знаковым числом, изменяющимся от 0 до +255 или от -128 до +127 соответственно. Масштабирование отсчёта громкостью происходит согласно формуле chan_value = (ucount-$80)*volume либо chan_value = scount*volume, где ucount – беззнаковый 8-битный отсчёт, scount – знаковый 8-битный отсчёт, volume – беззнаковая 6-битная громкость, chan_value – знаковое число. Сумма chan_value для всех левых или правых каналов выводится в ЦАП, который работает со знаковыми 16-битными числами.

Как NeoGS воспринимает отсчёты (знаковые или беззнаковые), определяет бит B_INV7B порта GSCFG0. Если этот бит устанавливается в 1, то отсчёты воспринимаются, как знаковые числа, в противном случае — как беззнаковые. После сброса значение этого бита 0.

NeoGS имеет следующие режимы вывода семплированного звука: *4-канальный режим*, *8-канальный режим* и *4-канальный режим с паннингом*.

Для указания громкости в каналах служат порты ввода-вывода `VOL1`, `VOL2`, …, `VOL8`. Для указания отсчёта в канале процессор *читает* байт из области памяти $6000-$7FFF. Конкретный адрес определяет номер канала, в который пойдёт отсчёт, а прочитанное значение запоминается FPGA во внутреннем буфере и является отсчётом. Громкости при записи в порты `VOL1-VOL8` также запоминаются во внутреннем буфере FPGA. Из буфера с частотой 37500 Гц отсчёты и громкости передаются в ЦАП. Конкретное время, когда отсчёт из внутреннего буфера отправится в ЦАП не определено и является различным для разных каналов. При попадании нескольких отсчётов или громкостей для одного канала во внутренний буфер сохраняется лишь последнее значение. Для синхронизации поступления отсчётов в ЦАП следует использовать прерывание, следующее с частотой 37500 Гц.

Примечание: не следует выбирать отсчёты из области памяти $6000-$7FFF циклами выборки команд (M1). При необходимости исполнения кода из этой области, а также при хранении там данных, не относящихся к семплированному звуку, громкость всех каналов должна быть установлена на 0.

4-канальный режим выбирается битами в порту `GSCFG0`: `B_8CHANS`=0, `B_PAN4CH`=0. В этом режиме задействованы порты громкости `VOL1-VOL4`, соответствующие каналам 1-4. Распределение памяти для отсчётов следующее:

Область памяти	Канал
$6000-$60FF	1
$6100-$61FF	2
$6200-$62FF	3
$6300-$63FF	4
$6400-$64FF	1
...	...

$7F00-$7FFF	4

Левыми каналами являются каналы 1 и 2, правыми — 3 и 4.

8-канальный режим выбирается битами в порту `GSCFG0: B_8CHANS`=1, `B_PAN4CH`=0. В этом режиме задействованы порты громкости `VOL1-VOL8`, соответствующие каналам 1-8. Распределение памяти для отсчётов следующее:

Область памяти	Канал
$6000-$60FF	1
$6100-$61FF	2
$6200-$62FF	3
$6300-$63FF	4
$6400-$64FF	5
$6500-$65FF	6
$6600-$66FF	7
$6700-$67FF	8
$6800-$68FF	1
...	...
$7F00-$7FFF	8

Левыми каналами являются каналы 1, 2, 5 и 6, правыми — 3, 4, 7 и 8.

Внимание! В 8-канальном режиме относительная громкость отдельно взятого канала в 2 раза меньше, чем в 4-канальном.

4-канальный режим с паннингом выбирается битами в порту `GSCFG0: B_8CHANS`=0, `B_PAN4CH`=1. В этом режиме распределение памяти отсчётов по каналам такое же, как в обычном *4-канальном режиме*. Однако используются все порты `VOL1-VOL8` в соответствии с таблицей:

Левые каналы	Правые каналы
канал 1, громкость `VOL1`	канал 1, громкость `VOL3`
канал 2, громкость `VOL2`	канал 2, громкость `VOL4`
канал 3, громкость `VOL5`	канал 3, громкость `VOL7`
канал 4, громкость `VOL6`	канал 4, громкость `VOL8`

5.3 Интерфейс MP3-декодера

MP3-декодер подключён к FPGA через два SPI-интерфейса, один из которых управляющий, а по другому передаются MP3-данные. Обобщённое описание SPI-интерфейсов приводится в главе 7 , а описание самого MP3-декодера — в его даташите (datasheet) и аппнотах (application notes) на него. Данное руководство ни в коей мере не собирается каким-либо образом пересказывать документы на MP3-декодер, откуда следует, что для работы с декодером *необходимо* изучить эти документы и главу 7 .

В NeoGS ставятся микросхемы MP3-декодеров MA8201 и MA8201A, являющиеся

полными аналогами микросхем VS1001 и VS1011 фирмы VLSI Solutions Oy. Даташиты и аппноты следует скачивать с сайта http://www.vlsi.fi.

Внимание! Оба типа микросхем имеют существенные отличия друг от друга. Необходимо предусматривать определение типа микросхемы MP3-декодера и при работе учитывать различия между ними. В противном случае ваш плеер не будет работать у половины пользователей.

Для работы с MP3-декодером, помимо SPI-интерфейсов, предусмотрены следующие сигналы, доступные программисту:

- Выходной сигнал аппаратного сброса на MP3-декодер, управляемый битом `B_MPXRS` в порту `SCTRL`, активный уровень низкий.
- Входной сигнал готовности принимать данные, поступающий с декодера, состояние которого считывается из бита `B_MDDRQ` порта `SSTAT`.

Замечания по работе с декодером:

1. Декодеры тактируются кварцевым резонатором 14 МГц, что является небольшим превышением официальной максимальной частоты согласно даташитам. Для нормальной работы (декодирование MP3-файлов с любым битрейтом) следует *обязательно* использовать внутреннее удвоение тактовой частоты, предусмотренное в декодерах (рабочая частота 28 МГц).
2. Максимальные частоты F_{sck} сигнала SCK для обоих SPI-интерфейсов декодера ограничены следующими величинами. При передаче байтов по управляющему интерфейсу (содержимое `MC_READ` не используется), равно как и по интерфейсу MP3-данных частота сигнала SCK соответсвуюшего интерфейса $F_{sck} < F_{dec}/4$, где F_{dec} - частота, на которой работает декодер (F_{dec}=14 МГц, если внутреннее удвоение выключено, или F_{dec}=28 МГц в противном случае). При приёме байтов по управляющему интерфейсу (содержимое `MC_READ` используется) $F_{sck} < F_{dec}/6$. В обоих случаях следует учитывать текущую частоту процессора Z80, состояние битов, задающих F_{sck} каждого интерфейса (`B_MCSPD1`, `B_MCSPD0`, `B_MDHLF`) и то, включено ли внутреннее удвоение тактовой частоты декодера или выключено.
3. После аппаратного или программного сброса внутреннее удвоение оказывается выключенным. Следует его включить, процедура включения отличается для разных (VS1001 и VS1011) декодеров. В случае VS1001 после установки регистра CLOCKF следует записать константу 0x8008 в недокументированный регистр INT_FCNTLH (см. аппнот vs10XXan.pdf). В случае VS1011 на месте регистра INT_FCNTLH оказывается другой по функции регистр и запись в него производить не требуется, а для включения удвоителя требуется записать значение частоты дискретизации (любое применимое) в регистр AUDATA (см. аппнот vs10XXan.pdf). В дальнейшем актуальное значение частоты дискретизации установится автоматически. В обоих случаях константа, записываемая в регистр CLOCKF, остаётся неизменной (0x9B58)
4. Для определения типа декодера (VS1001 или VS1011) следует прочитать регистр STATUS и заглянуть в биты 7:4 этого регистра. Так как определять тип декодера требуется до включения удвоителя (пункт 3), следует особо внимательно отнестись к пункту 2: $F_{dec}/6$ в этом случае составит 2.3 МГц, и F_{sck} для корректного чтения не должна быть более этого значения.
5. При проигрывании некорректного или испорченного MP3-файла, вследствие электромагнитных помех и т.д. декодер может зависнуть: при этом, как правило, проигрывание звука прекращается, а счётчик времени проигрывания (регистр DECODE_TIME) не инкрементируется. Также признаком зависания является чересчур низкая или чересчур высокая скорость потребления декодером MP3-данных (см.

аппнот vs10XXan.pdf). При обнаружении зависания необходимо выполнить программный сброс, повторить процедуру инициализации и вновь обратить внимание на признаки зависания. В случае, если декодер не 'отвис', необходимо провести аппататный сброс (бит B_MPXRS в порту SCTRL). Грубый замер скорости потребления декодером MP3-данных можно провести, используя как репер звуковые прерывания NeoGS, следующие с частотой 37500 Гц независимо от частоты процессора Z80.

6. После окончания данных в MP3-файле рекомендуется выдать в декодер 2048 байт нулей, после чего для начала декодирования следующего файла необходимо провести программный сброс. Применять вместо него аппаратный сброс не рекомендуется, так как при этом возникают щелчки в звуковом тракте декодера.
7. Управление режимами bass/treble и surround в обоих типах декодеров (VS1001 и VS1011) существенно различается.
8. В декодере VS1011 предусмотрено несколько режимов приёма MP3-данных. Следует использовать лишь установленный по умолчанию после сброса режим совместимости с декодером VS1001 с участием сигнала битовой синхронизации BSYNC. Сигнал BSYNC генерируется аппаратно схемой SPI-интерфейса для MP3-данных в FPGA и внимания к себе со стороны программы Z80 не требует.

6 Интерфейс SD-карты

6.1 Введение

Для изучения принципов работы с SD-картой *необходимо* предварительно ознакомиться с главой 7 данного руководства. Также необходимо изучить описание стандарта на SD-карты. При необходимости обрабатывать файловую систему SD-карты (например, FAT16, FAT32) информацию по ним следует искать в документах от microsoft (для случая FATxx) или в других описаниях. В данной главе приводится лишь информация, не охваченная вышеназванными источниками.

Кроме SPI-интерфейса, к интерфейсу SD-карты относятся биты B_SDDET и B_SDWP в порту SSTAT, отражающие, соответственно, наличие SD-карты в разъёме и установку флажка защиты от записи на SD-карте. Бит B_SDDET установлен в 0 при отсутствии карты. Бит B_SDWP установлен в 1 при флажке в положении «защита записи». *Примечание:* флажок защиты записи не соединён электрически (или как ещё) с электроникой SD-карты. Соответственно, записывать на SD-карту можно при любом положении этого флажка. *Внимание!* Не рекомендуется использовать данные биты в программах. Контакты на разъёме SD-карты могут загрязниться или окислиться, из-за чего возможен долговременный дребезг, плавание сигнала или несрабатывание. Кроме того, на первой ревизии платы NeoGS бит B_SDDET из-за ошибки в схеме определяется состоянием наплатного светодиода вместо положения флажка на SD-карте. Наличие SD-карты в разъёме можно определить фоновой попыткой её инициализации (с обязательным выходом по таймауту).

Частота процессора в NeoGS варьируется от 10 до 24 МГц, следовательно, частота Fsck изменяется в пределах от 5 до 12 МГц, что лежит в пределах спецификаций SD-карт на эту частоту (не более 25 МГц).

Порт SD_RSTR, который выдаёт предыдущий принятый байт и инициирует обмен с передаваемым байтом 0xFF, может использоваться для всех сценариев обмена данными и командами с SD-картой по аналогии работы с интерфейсом SD-карты в Z-controller'e, следует лишь помнить про программные паузы 16 или 18 тактов.

6.2 Пример

Инициализация SD-карты:

```
     LD         A,M_SETNCLR+M_SDNCS
     OUT        (SCTRL),A            ;CS для карты в 1

     LD         B,74
     LD         A,#FF
sendones:
     OUT        (SD_SEND),A
     DJNZ       sendones     ;посылаем много единичек

     LD         A,M_SDNCS
     OUT        (SCTRL),A            ;CS для карты в 0

     LD         HL,cmd0
     CALL       sendcmd
wait1:
     LD         A,#FF
     OUT        (SD_SEND),A
     NOP
     NOP
     IN         A,(SD_READ)
     DEC        A
     JR         NZ,wait1     ;ждём выполнения команды CMD0
recmd1:
     LD         HL,cmd1
     CALL       sendcmd
wait2:
     LD         A,#FF
     OUT        (SD_SEND),A
     NOP
     NOP
     IN         A,(SD_READ)
     CP         #FF
     JR         Z,wait2
     RRA
     JR         C,recmd1 ;ждём выполнения команды CMD1

     ; SD-карта готова к работе

     …
sendcmd:
     LD         B,6
scmd:
     LD         A,(HL)
     OUT        (SD_SEND),A
     INC        HL
     DJNZ       scmd
     RET
cmd0:     DB   #40,0,0,0,0,#95
```

```
cmd1:     DB   #41,0,0,0,0,#FF
```

7 SPI-интерфейсы

7.1 Введение

В NeoGS существует три последовательных SPI-интерфейса для работы с SD-картой, для управления MP3-декодером и для выдачи данных в декодер. Каждый интерфейс обладает своими свойствами, кроме того, для управления MP3-декодером используются дополнительные (по отношению к SPI-интерфейсам) сигналы. Дополнительные сигналы, равно как и некоторые свойства интерфейсов контролируются через два порта: `SSTAT` и `SCTRL`.

Порт `SSTAT` (только чтение) служит для чтения состояния некоторых сигналов, имеющих отношение к устройствам и SPI-интерфейсам и содержит в себе биты `B_MDDRQ`, `B_SDDET`, `B_SDWP` и `B_MCRDY`. Чтение порта возвращает текущее состояние битов.

Порт `SCTRL` (чтение или запись) служит для управления некоторыми сигналами, имеющими отношение к устройствам и SPI-интерфейсам и содержит в себе биты `B_SDNCS`, `B_MCNCS`, `B_MPXRS`, `B_MCSPD0`, `B_MCSPD1` и `B_MDHLF`. Чтение этого порта возвращает текущее состояние битов. Для изменения же состояния битов используется следующая процедура.

Для установки какого-либо отдельного бита в 1 в порт `SCTRL` записывается число `(1<<B_SETNCLR)+(1<<B_bit)`, где `B_bit` - один из вышеперечисленных битов порта `SCTRL`, а `B_SETNCLR` - служебный бит этого порта. Для установки отдельного бита в 0 в порт записывается число `(0<<B_SETNCLR)+(1<<B_bit)`.

Данным способом одновременно перевести в 1 либо в 0 можно сразу несколько бит, для чего следует набрать их маску в записываемом числе, а состояние бита `B_SETNCLR` в записываемом числе определяет, в какое состояние (0 или 1) устанавливается выбранный набор.

Примеры:

```
ld   A,(0<<B_SETNCLR)+(1<<B_MPXRS) ;установить B_MPXRS
out  (SCTRL),A                     ;в ноль

ld   A,(1<<B_SETNCLR)+(1<<B_MDHLF) ;установить B_MDHLF в
out  (SCTRL),A                     ;единицу

ld   A,(1<<B_SETNCLR)+(1<<B_SDNCS)+(1<<B_MCNCS)
out  (SCTRL),A ;установить B_SDNCS и B_MCNCS в единицу

ld   A,(0<<B_SETNCLR)+(1<<B_MCSPD1)+(1<<B_MCSPD0)    ;сначала
out  (SCTRL),A             ;очистить биты B_MCSPD1 и B_MCSPD1
ld   A,(1<<B_SETNCLR)+(1<<B_MCSPD1)+(0<<B_MCSPD0)      ;затем
out  (SCTRL),A                     ;установить их в состояние
                                   ;{B_MCSPD1,B_MCSPD0}={1,0}
```

Данная схема управления битами позволяет управлять различными битами из различных задач: например основной код может управлять битом B_MCNCS, в то время как код в прерывании может управлять битом B_SDNCS. Так как каждое отдельное изменение битов в порту является атомарным и не влияет на другие биты порта, не требуется запрещать прерывания на моменты изменения битов, что потребовалось бы при традиционном подходе

(di:in A,(port):and MASK:or BITS:out (port),A:ei).

Конкретное назначение битов в портах `SCTRL` и `SSTAT` описано ниже, а также в разделах, касающихся работы с MP3-декодером и SD-картой.

7.2 Обобщённое описание SPI-интерфейсов

SPI-интерфейсы используются для управления и обмена данными с SD-картой и MP3-декодером. SPI-мастер (схема, осуществляющая тактирование на SPI-шине) находится в FPGA и управляется процессором Z80, SPI-слейв (схема, тактируемая мастером) находится в том или ином устройстве (SD-карта, MP3-декодер).

Далее приводится обобщённое описание SPI-интерфейсов с точки зрения программиста. В NeoGS используется три специализированных SPI-интерфейса, один из которых обменивается данными с SD-картой, другой - с управляющим каналом MP3-декодера, а последний - посылает MP3-данные в декодер. Каждый из них может не поддерживать полный набор свойств, приведённых в обобщённом описании.

Обмен данными через SPI-интерфейс производится побайтно. В процессе каждого обмена аппаратно производится передача байта от FPGA к устройству и одновременно приём байта от устройства по двунаправленной последовательной SPI-шине. Эта шина состоит из сигналов MOSI (выход SPI-мастера и вход SPI-слейва, последовательная передача данных от мастера слейву), MISO (вход SPI-мастера и выход SPI-слейва, последовательная передача данных от слейва мастеру) и SCK (выход SPI-мастера и вход SPI-слейва, сигнал синхронизации передачи). Управление вышеописанными сигналами осуществляется полностью аппаратно, без участия процессора Z80.

В зависимости от используемого протокола работы с устройством, значение могут иметь оба байта, переданных в разных направлениях, один из них или ни одного (значение имеет только сам факт обмена).

SPI-интерфейс содержит также сигнал выборки NCS. Процесс обмена, как правило, выглядит следующим образом: процессор Z80 переводит сигнал NCS из неактивного (=1) в активное (=0) состояние, далее следует серия обменов байтами, после окончания которой процессор возвращает сигнал NCS в неактивное состояние (=1). В некоторых случаях требуется начинать обмен при неактивном состоянии NCS, при этом, как правило, имеет значение сам факт обмена (наличие тактирования сигналом SCK на SPI-шине), в то время как передаваемые данные игнорируются обоими участниками обмена (мастером и слейвом).

Процесс обмена байтами по SPI-шине инициируется процессором Z80 путём записи байта в порт `xx_SEND` (`xx` соответствует префиксу, определяющему, к какому устройству относится данный порт, например `SD_SEND` – порт относится к SPI-интерфейсу SD-карты). Записанный байт передаётся устройству, в то время как принятый в обмене байт сохраняется в временном регистре SPI-интерфейса. После окончания обмена принятый байт из временного регистра можно считать из порта `xx_READ`. Из порта `xx_RSTR` считывается тот же байт из временного регистра, однако факт чтения этого порта также запускает обмен по SPI-шине с передаваемым байтом 0xFF.

Таким образом, команды

```
in   A,(xx_RSTR)
ld   B,A
```

эквивалентны с точки зрения передаваемых по SPI-шине данных командам

```
in   A,(xx_READ)
ld   B,A
```

```
ld   A,0xFF
out  (xx_SEND),A
```

Процесс байтового обмена, инициированный записью в порт xx_SEND или чтением из порта xx_RSTR, занимает конечное время. SPI-интерфейсы тактируются частотой процессора Z80 F_{cpu}. Однако сигнал SCK, являясь выходом синхронной схемы SPI-интерфейса, может иметь частоты $F_{cpu}/2$, $F_{cpu}/4$, $F_{cpu}/8$ или $F_{cpu}/16$. В NeoGS не предусмотрен сигнал /WAIT для процессора Z80, следовательно между последовательными инициированиями обмена (запись в xx_SEND или чтение xx_RSTR) должна программно формироваться пауза в определённое число тактов процессора Z80. Кроме того, пауза должна соблюдаться между инициированием обмена и чтением принятого байта (чтение xx_READ или xx_RSTR), а также между инициированием обмена и установкой NCS в неактивное состояние. В SPI-интерфейсе предусмотрен также бит готовности RDY, который устанавливается в 0 сразу же после инициирования обмена и возвращается в 1, как только обмен закончился. Данный бит используется, как правило, в случае низких частот сигнала SCK (например, $F_{cpu}/8$ и ниже) для упрощения программных процедур обмена.

Частота SCK, F_{sck}	Пауза между инициированиями обмена или между инициированием обмена и чтением принятого байта, такты Z80	Пауза между инициированием обмена и снятием активного уровня сигнала NCS, такты Z80
$F_{cpu}/2$	16 (использовать RDY неэффективно)	18 (использовать RDY неэфф.)
$F_{cpu}/4$	34 или используйте бит RDY	34 или используйте бит RDY
$F_{cpu}/8$	Используйте бит RDY	Используйте бит RDY
$F_{cpu}/16$	Используйте бит RDY	Используйте бит RDY

Под словами 'пауза между инициированием обмена и чем-либо' подразумевается количество тактов Z80 между началом циклов ввода-вывода, осуществляющих описанные действия. Примеры:

```
ld   C,xx_SEND
out  (C),D
nop
out  (C),E ;выдержана пауза 16 тактов,
           ;используйте для работы на Fsck=Fcpu/2

ld   C,xx_READ
out  (xx_SEND),A
nop
in   A,(C) ;выдержана пауза 16 тактов, используйте
           ;для работы на Fsck=Fcpu/2

ld   C,xx_SEND
ld   HL,buffer
outi
outi ;выдержана пауза 16 тактов,
     ;используйте для работы на Fsck=Fcpu/2

ld   C,xx_RSTR
```

```
ld   HL,buffer
ini
ini ;выдержана пауза 16 тактов,
    ;используйте для работы на Fsck=Fcpu/2

ld   C,xx_SEND
out  (C),A
nop
in   A,(xx_READ) ;НЕВЕРНО!
                 ;пауза между циклами ввода-вывода
                 ;составляет 15 тактов!
```

Различные SPI-интерфейсы являются полностью независимыми, соблюдение пауз относится к последовательным обращениям к одному и тому же интерфейсу, например:

```
ld   C,SD_RSTR  ;чтение данных с SD-карты и немедленная
                ;пересылка их в MP3-декодер
in   A,(C)      ;предполагается, что оба интерфейса работают
                ;на Fsck=Fcpu/2
out  (MD_SEND),A;интерфейсы независимы, немедленная пересылка
                ;байта
in   A,(C)      ;между инициированиями обмена на интерфейсе
                ;SD-карты пауза составляет 23 такта
out  (MD_SEND),A;между инициированиями обмена на интерфейсе
                ;MP3-декодера пауза составляет 23 такта
```

7.3 Интерфейс SD-карты

Как упоминалось выше, конкретная реализация SPI-интерфейса не обладает всеми свойствами обобщённого описания. SPI-интерфейс, используемый для обмена с SD-картой, обладает лишь следующими свойствами:

- Порт посылки байта и инициирования обмена: `SD_SEND`
- Порт чтения принятого байта: `SD_READ`
- Порт чтения принятого байта и инициирования следующего обмена с посылаемым байтом 0xFF: `SD_RSTR`
- Бит управления сигналом NCS для SD-карты, находящийся в порту `SCTRL`: `B_SDNCS`

Частота SCK фиксирована и составляет $F_{sck}=F_{cpu}/2$, бит RDY отсутствует, следовательно требуется программное соблюдение пауз 16 или 18 тактов.

7.4 Управляющий интерфейс MP3-декодера

Этот интерфейс обладает следующими свойствами.

- Порт посылки байта и инициирования обмена: `MC_SEND`
- Порт чтения принятого байта: `MC_READ`
- Порт чтения и инициирования обмена отсутствует.
- Бит управления сигналом NCS для MP3-декодера, находящийся в порту `SCTRL`: `B_MCNCS`

– Бит готовности SPI-интерфейса в порту SSTAT: B_MCRDY *Внимание*! Следует чётко различать биты B_MCRDY и B_MDDRQ в порту SSTAT. Бит B_MCRDY относится к управляющему SPI-интерфейсу MP3-декодера, отражает состояние готовности интерфейса и генерируется схемой этого SPI-интерфейса в FPGA. Бит B_MDDRQ относится к SPI-интерфейсу MP3-данных декодера, и отражает готовность декодера принять очередной блок данных по этому интерфейсу. Сигнал генерируется декодером и просто транслируется FPGA с выхода декодера в бит порта SSTAT.
– Биты установки частоты F_{sck} управляющего интерфейса MP3-декодера B_MCSPD0 и B_MCSPD1 в порту SCTRL.

Состояние битов {B_MCSPD1,B_MCSPD0}	Частота сигнала SCK, F_{sck}
{0,0}	$F_{cpu}/2$
{0,1}	$F_{cpu}/4$
{1,0}	$F_{cpu}/8$
{1,1}	$F_{cpu}/16$

7.5 Интерфейс данных MP3-декодера

– Порт посылки байта и инициирования обмена MD_SEND
– Порт чтения принятого байта — отсутствует.
– Порт чтения и инициирования — отсутствует.
– сигнал NCS и управление им — отсуствует.
– Бит готовности интерфейса - отсутствует (следует соблюдать паузы программно).
– Бит установки частоты F_{sck} интерфейса данных B_MDHLF в порту SCTRL:

B_MDHLF	F_{sck}
0	$F_{cpu}/2$
1	$F_{cpu}/4$

Частоты $F_{cpu}/8$ и $F_{cpu}/16$ не поддерживаются.

8 Контроллеры DMA

8.1 Введение

NeoGS предоставляет возможность переложить некоторые задачи по пересылке больших объёмов данных на контроллеры DMA, освобождая тем самым процессор. DMA-контроллеры реализуют обмен данными между каким-либо устройством (источником и приёмником данных) и локальным ОЗУ NeoGS. В настоящее время реализован только контроллер DMA для обмена с ZX-машиной.

8.2 Интерфейс управления контроллерами DMA

В каждый момент времени возможно управлять одним контроллером DMA через 4 порта:

Порт	Режим доступа	Функция
DMA_HAD	R/W	Биты 5:0 порта задают старшие бита адреса ОЗУ 21:16, биты 7:6 порта устанавливать в 0 (High ADdress)
DMA_MAD	R/W	Порт задаёт биты адреса ОЗУ 15:8 (Middle ADdress)
DMA_LAD	R/W	Порт задаёт биты адреса ОЗУ 7:0 (Low ADdress)
DMA_CST	R/W	Порт контроля и статуса, конкретные значения при чтении и записи см. в описании контроллеров (Control and STatus)

Управляемый контроллер задаётся портом DMA_MOD:

Порт	Режим доступа	Функция
DMA_MOD	R/W	Задаёт контроллер, доступный для управления в портах DMA_HAD, DMA_MAD, DMA_LAD, DMA_CST. Разрешено записывать в этот порт следующие величины: C_DMA_NONE — ни один контроллер не доступен, C_DMA_ZX — доступен для управления контроллер обмена с ZX-машиной

Все существующие контроллеры DMA способны работать одновременно, при этом пропускная способность памяти будет делиться между ними поровну.

Для включения контроллера DMA необходимо выполнить как минимум следующие действия:

- Задать начальный адрес в регистрах DMA_HAD, DMA_MAD, DMA_LAD.
- Произвести требуемые настройки контроллера (порядок настроек зависит от контроллера) и разрешить ему DMA-обращения в регистре DMA_CST.
- Произвести действия, необходимые для начала работы контроллера.

Адресные регистры каждого контроллера после окончания операции указывают на следующую необработанную ячейку памяти (адрес инкрементируется после каждого обработанного байта).

8.3 DMA-контроллер обмена с ZX-машиной

Данный контроллер предназначен для ускорения обмена данными с ZX-машиной (по сравнению с обменом через порты связи ZXDATRD/ZXDATWR/GSDAT). После включения контроллера со стороны ZX-машины данные передаются через область ПЗУ. Для запросов чтения ZX-машины NeoGS отключает ПЗУ при помощи сигнала $\overline{\text{RDROM}}$ шины ZXBus и выставляет на шине данных ZX-машины содержимое своего ОЗУ, считанное контроллером DMA. Для запросов записи в область ПЗУ NeoGS, аналогичн, пересылает записанные данные в своё ОЗУ.

Конкретное смещение в пределах области $0000-$3FFF со стороны ZX-машины не

имеет значения: последовательность считанных ZX-машиной байт определяется содержимым памяти NeoGS подряд с начального адреса DMA; любые записи в область ПЗУ ZX-машиной будут сохраняться в памяти NeoGS подряд с начального адреса DMA.

Порт `DMA_CST` при работе с данным контроллером ведёт себя следующим образом:

Биты порта `DMA_CST`	Режим доступа	Функция
7	write	При записи 1 – включает контроллер DMA-обмена с ZX-машиной и разрешает подмену ПЗУ
	read	Отражает ранее записанное состояние бита 7
6 : 0	-	Не используется: при чтении состояние не определено, при записи устанавливать в 0.

Порядок работы с контроллером:

1. Со стороны ZX-машины включается ПЗУ в область $0000-$3FFF.
2. Со стороны NeoGS устанавливается начальный адрес DMA, включается DMA-контроллер.
3. ZX-машина производит чтения или записи области ПЗУ, производя пересылку данных.
4. NeoGS отключает DMA-контроллер.

Примечания: с момента включения и до момента выключения контроллера допускается со стороны ZX-машины совершать либо только чтения, либо только записи. В случае смешивания типов обращений поведение NeoGS не определено (возможна порча ОЗУ и зависание, требующее «холодной» перезагрузки NeoGS).

Частота процессора NeoGS при работе данного DMA-контроллера не может быть менее частоты процессора ZX-машины, в противном случае поведение NeoGS не определено.

К неопределённому поведению также может привести изменение адресных портов контроллера, находящегося во включенном состоянии.

При *чтении* ZX-машиной следует игнорировать первый считанный (от момента включения контроллера) байт. Последующие считанные байты будут принадлежать последовательности с установленного в адресных портах начального адреса. Однако, адрес в данных портах инкрементируется после каждого чтения, т. е. количество инкрементов равно количеству чтений, но на 1 больше количества правильных байт.

Рекомендации по применению: при включенном DMA стоит выключить IM1 прерывания на ZX-машине, так как из адресного пространства ПЗУ будут читаться данные из NeoGS. По аналогичной причине не будет работать и сброс ZX-машины. Рекомендуется предусматривать в программе NeoGS отключение DMA после некоторого периода неактивности ZX-машины (например, 40мс или 2 периода кадрой синхронизации). Частоту процессора NeoGS при работе с DMA стоит ставить максимально возможной, что снизит количество wait-циклов для процессора ZX-машины. Для синхронизации ZX-машины и NeoGS, помимо обычного механизма обмена через порты, можно использовать /NMI для процессора NeoGS (см. главу 2).