

*Recoded by Evgeny Muchkin 06.10.1998
SysOp of PALLY_STATION tel: 176-74-19
Last update and correxion by Cyrax, Inc. 18.04.2002
Former GS programmer... ;) email: reptyle@mail.ru*

**(c) STINGER &
(c) Cyrax, Inc. - (*)**

Руководство по программированию General Sound.

Версия v1.04. Редакция 006.

(отредактировал в документ CHRV)

Оставлен полностью авторский текст за исключением
форматирования и исправления ошибок в тексте.
Добавлено описание исправлений в GS ROM v1.05.

19.07.2009

Оглавление

1.Краткие технические характеристики GS.....	4
2.Краткое описание GS, или много всякой лабуды.....	4
3.Интерфейс со Спектрумом.....	5
4.Система команд GS.....	5
4.1 Команды GS:.....	7
4.1.1 #00 Reset flags.....	7
4.1.2 #01 Set silence (*)......	7
4.1.3 #02 Set low volume (*)......	7
4.1.4 #03 Set high volume (*)......	7
4.1.5 #04 Set 'E' 3bits (*)......	7
4.1.6 #05 Out volume port (*)......	7
4.1.7 #06 Send to DAC (*)......	7
4.1.8 #07 Send to DAC and to volume port (*)......	8
4.1.9 #08 - то же что и команда #00.....	8
4.1.10 #09 Sets one's byte volume. (*)......	8
4.1.11 #0A DAC output (*)......	8
4.1.12 #0B DAC and Volume output (*)......	8
4.1.13 #0C Call SounDrive Covox mode (*)......	8
4.1.14 #0D Call Ultravox mode (*)......	9
4.1.15 #0E Go to LPT Covox mode.....	9
4.1.16 #0F Go in Profi Covox mode (*)......	9
4.1.17 #10 Out to any port (*)......	10
4.1.18 #11 In from any port (*)......	10
4.1.19 #12 OUT to 0 port (*)......	10
4.1.20 #13 Jump to Address (*)......	10
4.1.21 #14 Load memory block (*)......	10
4.1.22 #15 Get memory block (*)......	11
4.1.23 #16 Poke to address (*)......	11
4.1.24 #17 Peek from address (*)......	11
4.1.25 #18 Load DE Pair (*)......	11
4.1.26 #19 Poke to (DE) address (*)......	12
4.1.27 #1A Peek from (DE) address (*)......	12
4.1.28 #1B Increment of DE Pair (*)......	12
4.1.29 #1C Poke to (#20XX) address (*)......	12
4.1.30 #1D Peek from (#20XX) address (*)......	12
4.1.31 #1E - #1F Зарезервированы.....	13
4.1.32 #F1 - #F2 Зарезервированы.....	13
4.1.33 #F3 Warm restart.....	13
4.1.34 #F4 Cold restart.....	13
4.1.35 #F5 Busy on.....	13
4.1.36 #F6 Busy off.....	13
4.1.37 #F7 Get HX Register (*)......	13
4.1.38 #F8 - #F9 Зарезервированы.....	14
4.1.39 #FA Out zero_to_zero.....	14
4.1.40 #FB - #FF Зарезервированы.....	14
4.1.41 #20 Get total RAM.....	14
4.1.42 #21 Get free RAM.....	14
4.1.43 #22 Get GS Variable (*)......	14
4.1.44 #23 Get number of RAM Pages.....	14
4.1.45 #24 - #29 Зарезервированы.....	15

4.1.46 #2A Set Module Master Volume.....	15
4.1.47 #2B Set FX Master Volume.....	15
4.1.48 #2E Set Current FX.....	15
4.1.49 #30 Load Module.....	16
4.1.50 #31 Play module.....	16
4.1.51 #32 Stop module.....	17
4.1.52 #33 Continue module.....	17
4.1.53 #35 Set Module Volume.....	17
4.1.54 #36 Data on (*).....	17
4.1.55 #37 Reinitialisation (*).....	17
4.1.56 #38 Load FX.....	17
4.1.57 #39 Play FX.....	19
4.1.58 #3A Stop FX in channels.....	19
4.1.59 #3D Set FX Volume.....	20
4.1.60 #3E Load FX (Extended version).....	20
4.1.61 #40 Set FX Sample Playing Note.....	20
4.1.62 #41 Set FX Sample Volume.....	21
4.1.63 #42 Set FX Sample Finetune.....	21
4.1.64 #43 - #44 Зарезервированы.....	21
4.1.65 #45 Set FX Sample Priority.....	21
4.1.66 #46 Set FX Sample Seek First parameter.....	21
4.1.67 #47 Set FX Sample Seek Last parameter.....	21
4.1.68 #48 Set FX Sample Loop Begin (*).....	21
4.1.69 #49 Set FX Sample Loop End (*).....	22
4.1.70 #4A - #4F Зарезервированы.....	22
4.1.71 #51 - #5F Зарезервированы.....	22
4.1.72 #60 Get Song Position.....	22
4.1.73 #61 Get Pattern Position.....	22
4.1.74 #62 Get Mixed Position.....	22
4.1.75 #63 Get Channel Notes.....	23
4.1.76 #64 Get Channel Volumes.....	23
4.1.77 #65 Jump to position (*).....	23
4.1.78 #66 Set speed/tempo (*).....	24
4.1.79 #67 Get speed value (*).....	24
4.1.80 #68 Get tempo value (*).....	24
4.1.81 #69 Process Sound (*).....	24
4.1.82 #6A - #7F зарезервированы.....	24
4.1.83 #80 Direct Play FX Sample (#80..#83).....	24
4.1.84 #88 Direct Play FX Sample (#88..#8B).....	24
4.1.85 #90 Direct Play FX Sample (#90..#93).....	25
4.1.86 #98 Direct Play FX Sample (#98..#9B).....	25
4.1.87 #A0 Change Channel Note (#A0..#A3) (*).....	25
4.1.88 #A8 Change Channel Volume (#A8..#AB) (*).....	25
4.1.89 #B0 - #F0 Зарезервированы.....	25
4.2 Примечания	26
5. Немного лирики.....	27
5.1 Авторы GS : (2 штуки ;).....	27
5.1.1 Слава Dangerous (X-Trade).....	27
5.1.2 Stinger.....	27
5.2 Sanx 4 moral support:.....	27
6. General Sound ROM v1.05a.....	28

1. Краткие технические характеристики GS.

Процессор: Z80, 12MHz, без циклов wait

ROM: 32k, 27256

RAM: Static Ram 128k всего, 112k доступно для модулей и сэмплов в базовой версии

INT: 37.5 KHz Каналы: 4 независимых 8-и битных канала, каждый с 6-и битным контролем громкости.

2. Краткое описание GS, или много всякой лабуды.

GS - музыкальная карточка, предназначенная для проигрывания музыкальных модулей и отдельных сэмплов (эффектов).

Модули для GS - это стандартные Амижные и РСшные 4-х каналные MOD файлы, а сэмплы - как Амижные signed sample, так и РСшные unsigned sample.

Проигрыватель MOD файлов в GS является практически полным аналогом ProTracker'a на Амиге и создавался при интенсивном использовании исходников ProTracker'a. (Исходники были из Protracker'a v2.1A by Lars "ZAP" Hamre - Amiga Freelancers)

MOD Player поддерживает все команды Pro Tracker'a, за исключением двух:

- E01 Filter On Амига-специфичная команда, включает фильтр высоких частот.
- EFX Invert Loop я еще не видел плейера, который бы поддерживал эту команду. Возможно, она поддерживается на каких-то старых плейерах.

GS представляет из себя, по-сути, микропроцессорный комплекс со своим процессором, ПЗУ, ОЗУ и портами, и абсолютно не зависит от главного процессора Спектрума, что позволяет, например, загрузить свой любимый модуль, сбросить Спектрум, загрузить ассемблер и творить под любимую музыку. Soft внутри GS полностью берет на себя задачи проигрывания звука, интерпретации модуля и т.д. Программирование GS'a сводится к передаче байт за байтом модуля и/или сэмплов, а затем требуется только подавать команды типа: запустить модуль, установить глобальную громкость проигрывания модуля, запустить сэмпл #09 в канале #02 и т.д.

Если предполагается загрузить модуль вместе с сэмплами, то в GENERAL требуется загружать вначале модуль, а затем сэмплы.

При загрузке модуля очень рекомендуется оставить свободными 2к памяти, т.е. загружать модули длиной максимум 110К. Это условие не является необходимым, но его исполнение очень желательно в целях совместимости с последующими версиями.

Аналогично очень рекомендуется оставлять по 80 байт для каждого сэмпла, например, если требуется загрузить 63-х килобайтный модуль и 18 сэмплов, то имеем:

$$\text{Total_Sample_Length} = 112 * 1024 - 63 * 1024 - 2 * 1024 - 18 * 80 = 46688 \text{ байт}$$

Это суммарная длина сэмплов, которые при таком положении вещей могут быть загружены.

Если же, например, требуется вычислить, сколько поместится в память GS'a 2-х килобайтных сэмплов, то это вычисляется следующим образом:

$$112 * 1024 / (2048 + 80) = 53 \text{ сэмпла.}$$

В GS'e имеются 4 физических канала, которые и проигрывают звук.

Каналы 0 и 1 - левые, а 2 и 3 - правые.

3. Интерфейс со Спектрумом.

На мир GS смотрит при помощи 4 регистров:

1. **Command register** - регистр команд, доступный для записи порт по адресу 187 (#BB). В этот регистр записываются команды.
2. **Status register** - регистр состояния, доступный для чтения порт по адресу 187 (#BB).
Биты регистра:
7- Data bit, флаг данных
6 - Неопределен
5 - Неопределен
4 - Неопределен
3 - Неопределен
2 - Неопределен
1 - Неопределен
0 - Command bit, флаг команд
Этот регистр позволяет определить состояние GS, в частности можно ли прочесть или записать очередной байт данных, или подать очередную команду, и т.п.
3. **Data register** - регистр данных, доступный для записи порт по адресу 179 (#B3). В этот регистр Спектрум записывает данные, например, это могут быть аргументы команд.
4. **Output register** - регистр вывода, доступный для чтения порт по адресу 179 (#B3). Из этого регистра Спектрум читает данные, идущие от GS.

Command bit в регистре состояний устанавливается аппаратно после записи команды в регистр команд. Сбрасываться в 0 он может только из GS, что сигнализирует об определенном этапе исполнения команды.

Data bit в регистре состояний может быть установлен или сброшен как по желанию Спектрума, так и по желанию GS: при записи Спектрумом в регистр данных он аппаратно устанавливается в 1, а после чтения GS'ом из этого регистра сбрасывается в 0. При записи GS в регистр вывода он (все тот же Databit) аппаратно устанавливается в 1, а после чтения из этого порта Спектрумом сбрасывается аппаратно в 0.

Несмотря на то, что регистр данных и регистр вывода расположены в пространстве адресов портов по одному и тому же адресу и воздействуют на один и тот же бит данных, они являются двумя независимыми регистрами. Значение, один раз записанное в один из этих регистров, остается неизменным в нем до новой записи.

Состояние бита данных очень часто неопределено, и если в спецификации команд не определены значения этого бита на определенных этапах исполнения команды, недопустимо делать какие-либо предположения относительно значения этого бита.

4. Система команд GS.

Вначале позволю себе небольшое отступление от собственно системы команд. GS, как известно, предназначен в основном для проигрывания модулей и сэмплов. В данной версии (1.04) GS ROM допускается загрузка одного модуля и/или до 32 сэмплов.

Каждый сэмпл при загрузке его в память получает свой уникальный идентификатор, который однозначно определяет обращение к данному сэмплу в командах, которые требуют номер сэмпла. Самый первый загруженный сэмпл получает номер (handle) = 1, следующий - номер 2, и т.д.

То же самое применимо и к модулям, и этот единственный загруженный модуль будет

иметь handle=1 после загрузки.

Особенностью данной версии является также то, что вначале требуется загружать модуль, а затем уже сэмплы.

Особенности описания команд:

Команды описываются следующим образом:

1. Нех код команды
2. Название команды
3. Выполняемые действия при исполнении команды
4. Формат команды
5. Комментарии к команде

Формат команды описывается следующим образом:

GSCOM EQU 187

GSDAT EQU 179

SC #NN : Послать код команды в регистр команд
LD A, #NN
OUT (GSCOM), A

WC : Ожидание сброса Command bit
WCLP IN A, (GSCOM)
RRCA
JR C, WCLP

SD Data : Послать данные в регистр данных
LD A, Data
OUT (GSDAT), A

WD : Ожидание сброса Data bit, по сути, ожидание, пока GS не примет
посланные ему данные
WDLP IN A, (GSCOM)
RLCA
JR C, WDLP

GD Data : Принять данные из регистра данных
IN A, (GSDAT)

WN : Ожидание установки Data bit, по сути, ожидание очередных данных
от GS
WNLP IN A, (GSCOM)
RLCA
JR NC, WNLP

(*) :

<PAUSE> - Просто небольшая задержка кадра два-три.

И напоследок — небольшая противовисовая последовательность (иногда помогает)

XOR A
OUT (#B3), A
OUT (#BB), A
IN A, (#BB)

для верности можно и продублировать ;).

4.1 Команды GS:

4.1.1 #00 Reset flags

Сбрасывает флаги Data bit и Command bit.

```
SC #00
WC
(Data bit=0, Command bit=0)
```

4.1.2 #01 Set silence (*)

Выводит в ЦАПы всех каналов #80. По сути устанавливает тишину.

```
SC #01
WC
```

4.1.3 #02 Set low volume (*)

Устанавливает громкость ЦАПов всех каналов в ноль.

```
SC #02
WC
```

4.1.4 #03 Set high volume (*)

Устанавливает громкость ЦАПов всех каналов в максимум.

```
SC #03
WC
```

4.1.5 #04 Set 'E' 3bits (*)

Устанавливает в 'E' регистре GS 3 младших бита в соответствии с заданным значением (2 младших бита в сущности являются номером канала #00-#03).

```
SD Chan (#00-#07)
SC #04
WC
```

4.1.6 #05 Out volume port (*)

Устанавливает громкость канала, номер которого содержится в 'E', в указанное значение. (Команда срабатывает при условии, что 'E' находится в пределах #00-#03)

```
SD Volume (#00-#3F)
SC #05
WC
```

4.1.7 #06 Send to DAC (*)

Выводит байт в ЦАП канала, указываемого по 'E'.

```
SD Byte
SC #06
WC
```

4.1.8 #07 Send to DAC and to volume port (*)

Выводит байт в ЦАП ('E') с заданной громкостью.

```
SD Byte
```

```
SC #07
WC
SD Volume
WD
```

4.1.9 #08 - то же что и команда #00

4.1.10 #09 Sets one's byte volume. (*)

Установка громкости канала, номер которого задан в 2х старших битах.

```
SD Byte (ccvvvvvvv)
SC #09
WC
```

cc - Номер канала

vvvvvv - Его громкость

4.1.11 #0A DAC output (*)

Еще один непосредственный вывод в ЦАП.

```
SD Byte
SC #0A
WC
SD Chan (#00-#03)
WD
```

4.1.12 #0B DAC and Volume output (*)

И наконец последний вывод в ЦАП с установкой громкости.

```
SD Fbyte
SC #0B
WC
SD Sbyte (ccvvvvvvv)
WD
```

Назначение битов Sbyte как и у к.#09

Команды #01- #0B служат в основном для построения различных Covox'ов и проигрывателей, при этом не слишком углубляясь во внутреннюю структуру GS.

vvvvvv - Его громкость

4.1.13 #0C Call SounDrive Covox mode (*)

Вызывает режим четырехканального Ковокса, последовательно копирует регистр данных по каналам. Выход из режима автоматически после вывода четвертого байта.

```
SD CH1
SC #0C
WC
SD CH2
WD
SD CH3
WD
SD CH4
WD
```


4.1.14 #0D Call Ultravox mode (*)

Вызывает режим универсального Ковокса, последовательно копирует регистр данных по каналам, число которых регулируется (1-4). В отличие от предыдущего варианта синхронизация не производится. Выход также производится автоматически по записи последнего байта.

```
SD CHANS
SC #0D
WC
```

```
SD CH1
SD CH2
SD CH3
SD CH4
```

CHANS (4-е младших бита) указывает какие каналы будут задействованы - для включения канала соответствующий бит нужно установить. Если канал выключен, то поступивший байт попадает на следующий включенный канал (если успеет :)

4.1.15 #0E Go to LPT Covox mode

Переходит в режим одноканального Ковокса, напрямую копирует регистр данных в ЦАПы двух (правого и левого) каналов. Выход из этого режима - запись #00 в регистр команд.

```
SC #0E
WC
```

```
SD \
SD \
...   Это вывод в ЦАПы
SD /
```

```
SC #00
WC
```

4.1.16 #0F Go in Profi Covox mode (*)

Переходит в режим двухканального Ковокса, напрямую копирует регистр данных в ЦАПы одного канала, а регистр команд в ЦАПы второго канала. Выход из этого режима - запись #4E в регистр данных, затем последовательно #0F и #AA в регистр команд.

```
SD #59
SC #0F
WC
```

```
SD \
SC \
SD \
SC \
...   Это вывод в ЦАПы
SD /
SC /
```

```
SD #4E
WD
SC #0F
WC
SC #AA
WC
```

4.1.17 #10 Out to any port (*)

Выводит байт во внутренний порт GS (#00-#09).

```
SD Port
SC #10
WC
SD Data
WD
```

4.1.18 #11 In from any port (*)

Читает байт из внутреннего порта GS (#00-#09).

```
SD Port
SC #11
WC
GD Data
WN
```

4.1.19 #12 OUT to 0 port (*)

Выводит байт в порт конфигурации GS (#00).

```
SD Data
SC #12
WC
```

4.1.20 #13 Jump to Address (*)

Передаёт управление по заданному адресу.

```
SD ADR.L
SC #13
WC
SD ADR.H
WD
```

4.1.21 #14 Load memory block (*)

Загрузка блока кодов по указанному адресу с заданной длиной.

```
SD LEN.L
SC #14
<PAUSE> (мб WD)
SD LEN.H
WD
SD ADR.L
WD
SD ADR.H
<PAUSE>

SD \
WD \
SD \
WD \ Блок данных длиной LEN
... /
SD /
WD /
```

4.1.22 #15 Get memory block (*)

Выгрузка блока кодов по указанному адресу с заданной длиной.

```
SD LEN.L
```

```

SC #15
<PAUSE> (мб WD)
SD LEN.H
WD
SD ADR.L
WD
SD ADR.H
<PAUSE>

GD \
WN \
GD \
WN      Блок данных длиной LEN
... /
GD /
WN /
GD

```

4.1.23 #16 Poke to address (*)

Записывает единичный байт по указанному адресу.

```

SD Byte
SC #16
WC
SD ADR.L
WD
SD ADR.H
WD

```

4.1.24 #17 Peek from address (*)

Считывает единичный байт из указанного адреса.

```

SD ADR.L
SC #17
WD
SD ADR.H
GD Byte

```

4.1.25 #18 Load DE Pair (*)

Загружает регистровую пару DE (относящуюся к GS, не путать с одноименной парой Main CPU) указанным словом.

```

SD Byte.E
SC #18
WC
SD Byte.D
WD

```

4.1.26 #19 Poke to (DE) address (*)

Записывает байт по адресу указанному в DE.

```

SD Byte
SC #19
WC

```

4.1.27 #1A Peek from (DE) address (*)

Считывает содержимое адреса, указываемого по DE.

```

SC #1A

```

WC
GD Byte
WN

4.1.28 #1B Increment of DE Pair (*)

Увеличивает пару DE на единичку.

SC #1B
WC

4.1.29 #1C Poke to (#20XX) address (*)

Записывает байт по адресу, старший байт которого равен #20. Смысла команда никакого не имеет, так как по этим адресам находится ПЗУ карты.

SD ADR.L
SC #1C
WC
SD Byte
WD

4.1.30 #1D Peek from (#20XX) address (*)

Читает байт с адреса, старший байт которого равен #20.

SD ADR.L
SC #1D
WC
GD Byte
WN

4.1.31 #1E - #1F Зарезервированы.

4.1.32 #F1 - #F2 Зарезервированы.

4.1.33 #F3 Warm restart

Сбрасывает полностью GS, но пропускает этапы определения количества страниц памяти и их проверки, что очень сильно ускоряет процесс инициализации.

SC #F3
WC

4.1.34 #F4 Cold restart

Полный перезапуск GS со всеми проверками. По сути, JP #0000.

SC #F4
WC

4.1.35 #F5 Busy on

Устанавливает флаг занятости в #FF

SC #F5
WC

4.1.36 #F6 Busy off

Устанавливает флаг занятости в #00

SC #F6

WC

Изначально Busy = #00. Исполнение всех команд в GS выполняется в главном цикле командного интерпретатора. Этот цикл в условном виде можно представить так:

```
1 if Command bit=0 then go to 1
2 Execute Command
3 if Command bit=1 then go to 2
4 if Playing=0 then go to 1
5 if Busy=#FF then go to 1
6 Process Sound
7 go to 1
```

Используя команды Busy можно например инициировать проигрывание сэмплов во всех каналах, потом скажем изменить параметры проигрывания в каналах, а потом запустить это все одновременно. Если же их не использовать, то возможна такая ситуация: иницируется первый (сэмпл станет проигрываться, а только потом иницируется второй сэмпл и т.д.).

4.1.37 #F7 Get HX Register (*)

Получить содержимое регистра HX (GS).

HX участвует в обработке флага Busy (bit 7 0/1 – Busy On/Off).

```
SC #F7
WC
GD HX
WN
```

4.1.38 #F8 - #F9 Зарезервированы.

4.1.39 #FA Out zero_to_zero

Вывод нуля в нулевой (конфигурационный) порт GS. Делает приостановку звучания музыки до следующего чтения из к.л. порта.

```
SC #FA
WC
```

4.1.40 #FB - #FF Зарезервированы.

4.1.41 #20 Get total RAM

Получить общий объем доступной памяти на GS. (В базовой версии это 112к)

```
SC #20
WC
GD RAM.L (Младшая часть)
WN
GD RAM.M (Средняя часть)
WN
GD RAM.H (Старшая часть)
```

Total RAM=65536*RAM.H+256*RAM.M+RAM.L

4.1.42 #21 Get free RAM

Получить общий объем свободной памяти на GS.

```
SC #20
WC
GD RAM.L (Младшая часть)
```

```

WN
GD RAM.M(Средняя часть)
WN
GD RAM.H(Старшая часть)

```

```
Free_RAM=65536*RAM.H+256*RAM.M+RAM.L
```

4.1.43 #22 Get GS Variable (*)

Получить значение переменной GS номер которой задан по Num. Переменные описывают текущее состояние карты, например номер паттерна, темп и т.п. Соответственно номеров переменных не приводится (в следствие несистематизированности и отрывочности данных, а так же по иным причинам).

```

SD Num
SC #22
WC
GD Variable
WN

```

4.1.44 #23 Get number of RAM Pages

Получить число страниц на GS. (В базовой версии 3 страницы)

```

SC #23
WC
GD Number_RAM_Pages

```

4.1.45 #24 - #29 Зарезервированы.

4.1.46 #2A Set Module Master Volume

Установить громкость проигрывания модулей.

```

SD Module_Master_Volume [#00..#40]
SC #2A
WC
[GD Old_Master_Volume] - Старая громк.

```

Маленький пример использования данной команды:
(Предполагается, что играет модуль)

```

LD B, #40

LOOP: LD A,B
      OUT (GSDAT),A
      LD A,#2A
      OUT (GSCOM),A
      EI
      HALT
      DJNZ LOOP

LD A,#32
OUT (GSCOM),A

```

Вышеописанное плавно снижает громкость играющего модуля, а затем останавливает его.

4.1.47 #2B Set FX Master Volume

Установить громкость проигрывания эффектов.

```
SD FX_Master_Volume [#00..#40]
```

```

SC #2B
WC
[GD Old_FX_Volume] - Старая громкость

```

Аналогично предыдущей команде, но действует на сэмплы.

С помощью этих двух команд можно регулировать баланс громкостей модуля и сэмплов, и т.п.

4.1.48 #2E Set Current FX

Установить текущий эффект. Просто присваивает переменной CURFX это значение. Если какая-либо команда требует номер сэмпла (sample handle), то можно вместо этого номера подать ей #00 и интерпретатор подставит вместо этого нуля значение переменной CURFX. (См. команды #38, #39, #40-#4F для понимания вышеизложенного.)

```

SD Cur_FX
SC #2E
WC

```

4.1.49 #30 Load Module

Загрузка модуля в память.

```

SC #30
WC
[GD Module_Handle]-номер модуля
(Command bit=0, Data bit=0)
SC #D1 (Open Stream-открыть поток)
WC

SD \
WD \
... Байты модуля
SD /
WD /

SC #D2 (Close Stream-закрыть поток)
WC

```

Пример:

```

LD HL,Mod_adress
LD DE,0-Mod_length
LD C,GSCOM

LD A,#30
CALL SENDCOM
LD A,#D1
CALL SENDCOM

LOOP: LD A,(HL)
      IN B,(C)
      JP P,READY
      IN B,(C)
      JP M,LOOP
READY: OUT (GSDAT),A
      INC HL
      LD A,(HL)
      INC E
      JP NZ,LOOP
      INC D
      JP NZ,LOOP

```

```

WAIT:    IN B, (C)    ;Ждем принятия
          JP M, WAIT  ;последнего байта
          LD A, #D2
          CALL SENDCOM
          IN A, (GSDAT) ; Номер модуля
          OUT (GSDAT), A
          LD A, #31

SENDCOM: OUT (GSCOM), A
WAITCOM: IN A, (GSCOM)
          RRCA
          JR C, WAITCOM
          RET

```

4.1.50 #31 Play module

Проигрывание модуля.

```

SD Module_Handle - номер модуля
SC #31
WC

```

4.1.51 #32 Stop module

Остановить проигрывание модуля.

```

SC #32
WC

```

4.1.52 #33 Continue module

Продолжить проигрывание модуля после остановки.

```

SC #33
WC

```

4.1.53 #35 Set Module Volume

Установить громкость проигрывания модулей.

```

SD Module_Master_Volume [#00..#40]
SC #35
WC
[GD Old_Master_Volume] - Старая громк.

```

4.1.54 #36 Data on (*)

Устанавливает регистр данных в #FF.

```

SC #36
WC
[GD Data (#FF) ]

```

4.1.55 #37 Reinitialisation (*)

Переустанавливает внутренние переменные в исходное состояние.

```

SC #37
WC

```

4.1.56 #38 Load FX

Загрузка сэмпла эффекта в память. Загружает беззнаковые сэмплы (PC type)

```

SC #38

```



```

WC
[GD FX_Handle]-номер сэмпла
(Command bit=0, Data bit=0)
SC #D1 (Open Stream-открыть поток)
WC

SD \
WD \
... Байты сэмпла
SD /
WD /

SC #D2 (Close Stream-закрыть поток)
WC

```

При загрузке каждого сэмпла, в памяти GS создается для этого сэмпла заголовок, в котором описываются различные параметры сэмпла. После загрузки эти параметры устанавливаются в определенные значения, как то: Note=60, Volume=#40, FineTune=0, SeekFirst=#0F, SeekLast=#0F, Priority=#80, No Loop и внутренняя переменная CurFX устанавливается равной FX_Handle.

Затем командами #40, #41, #42, #45, #46 и #47 можно эти значения по умолчанию сменить на свои. Это требуется потому что команда #39 для инициации проигрывания сэмпла использует значения параметров из заголовка сэмпла.

В своем естественном виде сэмплы обычно плохо пакуются компрессорами, но сжимаемость обычно можно поднять, если перевести сэмпл в Delta-вид, т.е. хранить не абсолютные значения сэмпла, а относительное смещение относительно предыдущего байта. Примерно вот так вот можно перевести сэмпл в Delta-вид:

```

LD HL,Start_of_sample
LD DE,0-Length_of_sample
LD C,#00

LOOP: LD A,(HL)
SUB C
LD C,(HL)
LD (HL),A
INC E
JP NZ,LOOP
INC D
JP NZ,LOOP

```

А вот как можно закатать сэмпл:

```

LD IX,Parameters
LD HL,Sample_adress
LD DE,0-Sample_length
LD C,GSCOM

LD A,#38
CALL SENDCOM
LD A,#D1
CALL SENDCOM

LOOP: LD A,(HL)
IN B,(C)
JP P,READY
IN B,(C)
JP M,LOOP
READY: OUT (GSDAT),A
INC HL

```

```

        ADD A, (HL)
        INC E
        JP NZ, LOOP
        INC D
        JP NZ, LOOP
WAIT:   IN B, (C) ;ждем принятия
        JP M, WAIT ;последнего байта
        LD A, #D2
        CALL SENDCOM

; Теперь переопределяем параметры
; сэмпла по умолчанию своими
; значениями

        LD A, (IX+#00)
        OUT (GSDAT), A ; Нота
        LD A, #40
        CALL SENDCOM
        LD A, (IX+#01)
        OUT (GSDAT), A ; Громкость
        LD A, #41

SENDCOM: OUT (GSCOM), A
WAITCOM: IN A, (GSCOM)
        RRCA
        JR C, WAITCOM
        RET

```

4.1.57 #39 Play FX

Проигрывание эффекта.

```

SD FX Handle - номер сэмпла
SC #39
WC

```

При исполнении этой команды происходит следующее: смотрятся каналы, указанные в SeekFirst параметре нашего сэмпла, и если хотя-бы один из них свободен, в нем и проигрывается сэмпл, в противном случае смотрятся каналы, указанные в SeekLast и если один из них свободен, в нем и играется сэмпл, если свободных нет, то просматриваются все каналы, указанные SeekLast, из них выбирается канал с наименьшим приоритетом и сравнивается с приоритетом нашего сэмпла (имеется в виду сэмпл, который мы хотим проиграть), если у этого сэмпла будет больший приоритет, чем у сэмпла, уже играющего в канале, то играющий в канале сэмпл будет остановлен, а наш сэмпл будет запущен в этом канале вместо старого сэмпла. Вот такая вот приоритетная схема...

Тогда сэмпл запускается в канале, то его нота, громкость и т.п. параметры записываются в область данных канала из заголовка сэмпла.

В общем случае, чтобы проиграть сэмпл с нужными параметрами, вы можете установить эти параметры после загрузки сэмпла и смело использовать команду #39. Если же параметры должны меняться, то можно поступать следующим образом: командой #2E сделать текущим требуемый сэмпл, командами #4x изменить его параметры, а затем уже запускать его командой #39.

Альтернативный метод запуска сэмплов предоставляют команды #80..#9F, при исполнении этих команд вы прямо в коде команды указываете, в каком канале требуется запустить сэмпл, и кроме этого, вы можете также указать с какой нотой и/или громкостью требуется запустить сэмпл.

4.1.58 #3A Stop FX in channels

Установка проигрывания эффектов в заданных каналах, которые указываются в маске каналов (Channel Mask). В ней единица в n-ном бите указывает на то, что эффект в n-ном канале требуется остановить

```
SD Channel_Mask
SC #3A
WC
```

Описанное выше есть идеальный вариант работы данной команды, но к сожалению не все так просто в этом мире, и эта команда действует не так, а именно: единица в бите 7 останавливает сэмплы в нулевом канале, и т.п. В следующих версиях это будет исправлено, а пока я могу порекомендовать останавливать вообще все сэмплы маской #FF.

4.1.59 #3D Set FX Volume

Установить громкость проигрывания эффектов.

```
SD FX Volume [#00..#40]
SC #3D
WC
[GD Old_FX_Volume] - Старая громкость
```

4.1.60 #3E Load FX (Extended version)

Загрузка сэмпла эффекта в память. Позволяет загружать сэмплы со знаком. (Amiga type)

```
SD #01 (Signed sample)
SC #3E
WC
[GD FX_Handle]-номер сэмпла
(Command bit=0, Data bit=0)
SC #D1 (Open Stream-открыть поток)
WC

SD \
WD \
... Байты сэмпла
SD /
WD /

SC #D2 (Close Stream-закрыть поток)
WC
```

4.1.61 #40 Set FX Sample Playing Note

Установка ноты по умолчанию для текущего эффекта.

```
SD Note [0..95]
SC #40
WC
```

```
Note=
0 C-0
1 C#0
12 C-1
24 C-2
36 C-3 (C-1 в Амиге)
48 C-4 (C-2 в Амиге)
60 C-5 (C-3 в Амиге)
72 C-6
84 C-7
```

В данной версии Sound Generators Wave 2, 3 могут воспроизвести октавы 3, 4 и 5, поэтому допустимым значением параметра Note является диапазон от 36 до 71.

4.1.62 #41 Set FX Sample Volume

Установка громкости по умолчанию для текущего эффекта.

```
SD FX_Volume [#00..#40]  
SC #41  
WC
```

4.1.63 #42 Set FX Sample Finetune

Установка Finetune по умолчанию для текущего эффекта.

```
SD FX_Finetune [#00..#40]  
SC #42  
WC
```

4.1.64 #43 - #44 Зарезервированы.

4.1.65 #45 Set FX Sample Priority

Установка приоритета для текущего эффекта. (См. Команду #39)

```
SD FX_Priority [#01..#fe]  
SC #45  
WC
```

4.1.66 #46 Set FX Sample Seek First parameter

Установка параметра Seek First для текущего эффекта. (См. команду #39)

```
SD FX_SeekFirst  
SC #46  
WC
```

4.1.67 #47 Set FX Sample Seek Last parameter

Установка параметра Seek Last для текущего эффекта. (См. команду #39)

```
SD FX_SeekLast  
SC #47  
WC
```

4.1.68 #48 Set FX Sample Loop Begin (*)

Установка начала цикла для текущего эффекта.

```
SD LEN.L  
SC #48  
WC  
SD LEN.M  
WD  
SD LEN.H  
WD
```

При равенстве LEN.H - #FF зацикливание не производится

4.1.69 #49 Set FX Sample Loop End (*)

Установка конца цикла для текущего эффекта.

```

SD LEN.L
SC #49
WC
SD LEN.M
WD
SD LEN.H
WD

```

4.1.70 #4A - #4F Зарезервированы.

4.1.71 #51 - #5F Зарезервированы.

4.1.72 #60 Get Song Position

Получение значения переменной Song_Position в текущем модуле.

```

SC #60
WC
GD Song_Position [#00..#FF]

```

Можно интерпретировать как количество проигранных паттернов модуля. После старта модуля принимает значение 0 и увеличивается на единицу после проигрывания очередного паттерна. Эта переменная может использоваться для синхронизирования процессов в Спектруме с проигрыванием модуля. Для этого можно, например, в начале процедуры обработки прерывания сделать SC #60, затем выполнить процедуры различных операций с экраном, скроллинга строчек и т.п. (т.е. чтобы была достаточная для выполнения команды задержка), а затем прочитать значение порта 179 (GD Song_Position), и сравнить его с требуемым и, в случае равенства, перейти на следующую часть демы, т.е.

```

if (Song_Position==My_Position)
then goto Next_Part_Of_Demo

```

4.1.73 #61 Get Pattern Position

Получение значения переменной Pattern_Position в текущем модуле.

```

SC #61
WC
GD Pattern_Position [#00..#3F]

```

Получить значение смещения в паттерне (текущий ROW), использование - аналогично предыдущей команде, однако требуется заметить, что эта величина изменяется довольно быстро, и поэтому

```

if (Pattern_Position>=My_Position) then goto Next_Part_Of_Demo

```

4.1.74 #62 Get Mixed Position

Получить значение Pattern_Position, немного смешанной с Song_Position.

```

SC #62
WC
GD Mixed_Position

Mixed_Position: (по битам)

7-Song_Position.1
6-Song_Position.0
5-Pattern_Position.5
4-Pattern_Position.4
3-Pattern_Position.3

```

```
2-Pattern_Position.2  
1-Pattern_Position.1  
0-Pattern_Position.0
```

То есть если получить `Mixed_Position` и сделать с ним `AND #3F`, то получится вылитый `Pattern_Position`, а если после получения его немного `RLCA, RLCA, AND #02` - то это будут младшие два бита `Song_Position`. См. примечания к командам #60 и #61.

4.1.75 #63 Get Channel Notes

Получить ноты всех каналов модуля.

```
SC #63  
WC  
GD Note_of_channel_0  
WN  
GD Note_of_channel_1  
WN  
GD Note_of_channel_2  
WN  
GD Note_of_channel_3
```

Если в каком-либо канале значение ноты изменилось с последнего исполнения команды #63, то бит 7 полученного значения `Note_of_channel_N` будет в нуле, если же это значение то же самое, что и было раньше, то этот бит будет в единице. Младшие семь битов и есть собственно нота от 0 до 95. Если это значение равно 127, то это означает, что никакие сэмплы в канале не играют. Данная команда предназначена в основном для построения на ее основе различных анализаторов.

4.1.76 #64 Get Channel Volumes

Получить громкости всех каналов модуля.

```
SC #64  
WC  
GD Volume_of_channel_0  
WN  
GD Volume_of_channel_1  
WN  
GD Volume_of_channel_2  
WN  
GD Volume_of_channel_3
```

См. описание команды #63

4.1.77 #65 Jump to position (*)

Делает переход на заданную позицию.

```
SD Position  
SC #65  
WC
```

4.1.78 #66 Set speed/tempo (*)

Установка скорости в пределах #01-#1F. При значениях #20-#FF устанавливается темп проигрывания. Значения темпа соответствуют оригинальным при скорости равной #06.

```
SD Speed/Tempo  
SC #66  
WC
```

4.1.79 #67 Get speed value (*)

Чтение текущей скорости.

SC #67
WC
GD Speed
WD

4.1.80 #68 Get tempo value (*)

Чтение текущего темпа.

SC #68
WC
GD Tempo
WD

4.1.81 #69 Process Sound (*)

Переход на следующий кварк (или тик) в процессе проигрывания звука. Может, в частности, использоваться для синхронизации с выводом звука. Для этого нужно установить флаг занятости (Busy On — что вызовет остановку звука), а затем с нужной периодичностью выдавать команду #69 для дальнейшего проигрывания.

SC #69
WC

4.1.82 #6A - #7F зарезервированы.

4.1.83 #80 Direct Play FX Sample (#80..#83)

Проигрывание сэмпла в заданном канале.

SD Sample_Number
SC #80..#83 (Младшие биты определяют непосредственно номер канала, в котором требуется играть сэмпл)
WC

4.1.84 #88 Direct Play FX Sample (#88..#8B)

Проигрывание сэмпла в заданном канале с заданной нотой.

SD Sample_Number
SC #88..#8B (Младшие биты определяют непосредственно номер канала, в котором требуется играть сэмпл)
WC
SD Note [0..95]
WD

4.1.85 #90 Direct Play FX Sample (#90..#93)

Проигрывание сэмпла в заданном канале с заданной громкостью.

SD Sample_Number
SC #90..#93 (Младшие биты определяют непосредственно номер канала, в котором требуется играть сэмпл)
WC
SD Volume [#00..#40]
WD

4.1.86 #98 Direct Play FX Sample (#98..#9B)

Проигрывание сэмпла в заданном канале с заданной нотой и громкостью.

```
SD Sample_Number
SC #98..#9B (Младшие биты определяют непосредственно номер канала, в
              котором требуется играть сэмпл)
WC
SD Note [0..95]
WD
SD Volume [#00..#40]
WD
```

4.1.87 #A0 Change Channel Note (#A0..#A3) (*)

Смена текущей ноты в заданном канале. Производится «на лету».

```
SD Note
SC #A0..#A3
WC
```

4.1.88 #A8 Change Channel Volume (#A8..#AB) (*)

Подобно предыдущей команде «на лету» меняет громкость канала.

```
SD Volume
SC #A8..AB
WC
```

Предыдущие две команды работают вне зависимости от того, что проигрывается в данном канале - семпл или модуль. Появлении в канале нового звука — от модуля либо семпла вернет все в исходное состояние — то есть громкость либо нота будут те, которые указаны вновь поступившего звука. В силу данной темпированности для получения требуемого эффекта данные команды следует вызывать с некоей периодичностью (устанавливается экспериментально).

4.1.89 #B0 - #F0 Зарезервированы.

4.2 Примечания

Команды, отмеченные как (*), являются недокументированными и в полной мере относятся только к версии 1.04. На работоспособность этих команд в последующих версиях автор описания (2) ответственности не несет.

2. Напоминаю, что регистры (их имена), упомянутые в этом описании, относятся только и только к внутренним регистрам GS и никакого отношения к регистрам основного процессора не имеют.

3. Данная редакция (006) является наиболее полной и точной на данный момент — 18.04.2002. Так же она является последней и несет скорее познавательный характер (по крайней мере автор не видит иного применения данной сводки команд, кроме как для высокоуровневой эмуляции GS).

На дополнение сего текста меня сподвиг тот факт, что после порядка 4х лет я узрел свой труд в сети, да еще в составе тех описаний карточки. Поэтому мне хотелось бы, чтобы дока была по возможности наиболее полной и точной...

5. Немного лирики...

5.1 Авторы GS : (2 штуки ;)

5.1.1 Слава Dangerous (X-Trade)

Ему принадлежит идея создания GS'a, аппаратная реализация одного, некоторые пожелания относительно Soft'a GS'a, а также Amiga 1200, на которой мною производились всяческие эксперименты. Он единственный и неповторимый производитель General Sound'a и именно он заведует производством и продажей GS.

5.1.2 Stinger

Это я, автор сего опуса и по совместительству душа и сердце GeneralSound'a. Я являюсь разработчиком всего встроенного Soft'a в GS'e и предполагаю и дальше заниматься сим делом. (Да, я также являюсь автором некоторых хитрых наворотов в аппаратной части GS'a, и был бы автором еще многих, если бы не был все время сдерживаем Славой, постоянно озабоченным проблемами понижения цены.)

Написав около 20 кб кода за пол-года, признаться, я немного устал, но имею довольно большие планы относительно следующих версий GS'a, как-то:

- Wave 4 Sound Generators воспроизводящих все октавы;
- Ускорение за счет оных процентов на 30-40 генерации звука;
- Очень хотелось бы проигрывание STM'ов от PC;
- Развитая система команд;
- Различные спецэффекты над сэмплами;
- Хранение паттернов в закомпрессованном виде (остается около 15% от изначального объема);
- И многое другое.

Все программное обеспечение должно работать и на последующих версиях прошивки, если оно написано в соответствии с моими вышеизложенными пожеланиями и требованиями. Кроме описанных команд в GS'e существует еще большое количество команд, которые не документированы, и я оставляю за собой право изменять их каким-угодно образом и только относительно документированных команд приемлю законные претензии типа: "В документации написано так, а в прошивке это работает по другому..."

Я планирую значительное расширение системы команд, и буду рад конструктивным (желательно конкретным) предложениям.

Так что, если вы озвучиваете игрушку или пишете музыкальный редактор для GS и обнаруживаете, что вам очень не хватает какой-либо команды, то звоните мне и высказывайте предложения. (Телефон, я думаю, особого труда узнать не составит ;)

5.2 Sanx 4 moral support:

Дима (X-Trade)

SParker (XLD)

6. General Sound ROM v1.05a

(c) Stinger, 1997,

bugfixed by psb & Evgeny Muchkin, 2007.

В данной версии прошивки исправлены глюки версии 1.04 Beta.

1. Глюк с модулями, в которых ≥ 63 паттерна (klisje.mod, tranceillusion.mod).
2. Глюк со скоростью проигрывания ПОСЛЕДНЕЙ ноты модуля, её скорость выставлялась стандартной, во многих модулях при зацикливании была заметна задержка (напр., technostyle(z).mod). Более того, при зацикливании не на 1-ю позицию, скорость все равно выставлялась стандартной!
3. Пофиксена неправильная скорость проигрывания сэмплов. На некоторых модулях было заметно, что сэмплы играли немного быстрее чем надо (например, EightMayDay.mod).
4. При начале проигрывания модуля GS сообщал, что играет какая-то нота, даже если в канале ничего не играло (команда #64 возвращала не 127).
5. Добавлена команда для плееров: #6A - Set player mode. После этой команды GS перестанет обращать внимание на команду останова в модуле (ком. F00). Полезно для некоторых модулей (bst.mod).

Формат команды:

```
SD #01 ; #01 - On, #00 - Off
SC #6A
WC
```

6. Встроен релупер для модулей. Раньше, если в модуле играл сэмпл, длина лупа которого была слишком маленькой (десятки-сотни байт), GS тормозил или зависал. После этой команды сэмплы в загружаемом модуле фиксируются и GS не тормозит.

Формат команды:

```
SD MinLoopLen_Low
SC #6B
WC
SD MinLoopLen_High
```

Параметр MinLoopLen задается в СЛОВАХ и может быть в диапазоне от 0 до 16384 (0 - релупер выключен).

Возможен короткий формат команды:

```
SC #6B
WC
SC ... ; следующая команда GS
```

В этом случае длина по умолчанию будет 512 слов.

ВНИМАНИЕ! Настройки команд #6A и #6B сбрасываются только аппаратным RESET или командой #F4 (командой #F3 не сбрасываются!).

P.S. В прошивке по смещению #0004 находится номер версии в BCD формате; по смещению #0100 находятся оригинальные копирайты (3 строки по 24 символа); по смещению #0080 находится информация о патче, строка заканчивается 0.

P.P.S. Для работы старых плееров в новых режимах (п.5 и 6), достаточно перед их запуском

дать из бейсика команды:

```
OUT 179,1  
OUT 187,106  
OUT 187,107
```

P.P.P.S. Хочется выразить особую благодарность следующим людям:

- Stinger: за прошивку и доступные исходники,
- Aprisobal: без SjASMPlus не было бы ничего этого,
- Evgeny Muchkin: за всяческое содействие при создании патча,
- Caro: за IDA и моральную поддержку,
- SMT & Alone Coder: за UnrealSpessy (и за исправление глюков в нем!;),
- Half Elf: за плагины к FAR'у,
- n1k-o & Manwe: за консультации по mod'ам.