

Recoded by Evgeny Muchkin 10/06/1998
SysOp of PALLY_STATION tel: [176-74-19](tel:176-74-19)
Last update and correxion by Cyrax, Inc. April 18, 2002
Former GS programmer ...;) email: reptyle@mail.ru

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

Programming Guide General Sound.

Version v1.04. Revision 006.

(edited into CHRV doc)

The entire author's text is left with the exception of
formatting and correction of errors in the text.
Added description of fixes in GS ROM v1.05.

24.02.2012

Table of contents

1. Brief technical characteristics of GS	five
2. Brief description of GS, or a lot of any crap	five
3. Interface with the Spectrum	6
4. GS Command System	6
4.1 GS Commands:	eight
4.1.1 # 00 Reset flags	eight
4.1.2 # 01 Set silence (*)	eight
4.1.3 # 02 Set low volume (*)	eight
4.1.4 # 03 Set high volume (*)	eight
4.1.5 # 04 Set 'E' 3bits (*)	eight
4.1.6 # 05 Out volume port (*)	eight
4.1.7 # 06 Send to DAC (*)	eight
4.1.8 # 07 Send to DAC and to volume port (*)	eight
4.1.9 # 08 - the same as command # 00	nine
4.1.10 # 09 Sets one's byte volume. (*)	nine
4.1.11 # 0A DAC output (*)	nine
4.1.12 # 0B DAC and Volume output (*)	nine
4.1.13 # 0C Call SounDrive Covox mode (*)	nine
4.1.14 # 0D Call Ultravox mode (*)	ten
4.1.15 # 0E Go to LPT Covox mode	ten
4.1.16 # 0F Go in Profi Covox mode (*)	ten
4.1.17 # 10 Out to any port (*)	eleven
4.1.18 # 11 In from any port (*)	eleven
4.1.19 # 12 OUT to 0 port (*)	eleven
4.1.20 # 13 Jump to Address (*)	eleven
4.1.21 # 14 Load memory block (*)	eleven
4.1.22 # 15 Get memory block (*)	eleven
4.1.23 # 16 Poke to address (*)	12
4.1.24 # 17 Peek from address (*)	12
4.1.25 # 18 Load DE Pair (*)	12

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 (*)	13
4.1.29 # 1C Poke to (# 20XX) address (*)	13
4.1.30 # 1D Peek from (# 20XX) address (*)	13
4.1.31 # 1E - # 1F Reserved	13
4.1.32 # F1 - # F2 Reserved	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 (*)	fourteen
4.1.38 # F8 - # F9 Reserved	fourteen
4.1.39 #FA Out zero_to_zero	fourteen
4.1.40 #FB - #FF Reserved	fourteen
4.1.41 # 20 Get total RAM	fourteen
4.1.42 # 21 Get free RAM	fourteen
4.1.43 # 22 Get GS Variable (*)	fifteen
4.1.44 # 23 Get number of RAM Pages	fifteen
4.1.45 # 24 - # 29 Reserved	fifteen

4.1.46 # 2A Set Module Master Volume	fifteen
4.1.47 # 2B Set FX Master Volume	fifteen
4.1.48 # 2E Set Current FX	sixteen
4.1.49 # 30 Load Module	sixteen
4.1.50 # 31 Play module	17
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	nineteen
4.1.58 # 3A Stop FX in channels	20
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 Reserved	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 (*)	22
4.1.69 # 49 Set FX Sample Loop End (*)	22
4.1.70 # 4A - # 4F Reserved	22

4.1.71 # 51 - # 5F Reserved	22
4.1.72 # 60 Get Song Position	22
4.1.73 # 61 Get Pattern Position	22
4.1.74 # 62 Get Mixed Position	23
4.1.75 # 63 Get Channel Notes	23
4.1.76 # 64 Get Channel Volumes	23
4.1.77 # 65 Jump to position (*)	24
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 Reserved	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 Reserved	25
4.1.90 #FA enable test mode	26
4.2 Notes	27
4.3 Correspondence of Amiga and General Sound channel numbering	27
5.Human lyrics	28
5.1 Autops GS: (2 pieces;)	28
5.1.1 Word Dangerous (X-Trade)	28
5.1.2 Stinger	28
5.2 Sanx 4 moral support:	28

6.General Sound ROM v1.05a	29
7.Version History	31

1. Brief technical characteristics of GS.

Processor: Z80, 12MHz, no wait cycles

ROM: 32k, 27256

RAM: Static Ram 128k total, 112k available for modules and samples in basic version

INT: 37.5 KHz Channels: 4 independent 8-bit channels, each with 6-bit volume control.

2. A short description of GS, or a lot of crap.

GS - a music card designed for playing music modules and individual samples (effects).

Modules for GS are standard Amiga and PC 4-channel MOD files, and Samples - both Amiga signed sample and PC unsigned sample.

The MOD file player in GS is almost a complete analogue of ProTracker on Amiga and was created with intensive use of ProTracker sources. (The sources were from Protracker v2.1A by Lars "ZAP" Hamre - Amiga Freelancers)

MOD Player supports all Pro Tracker commands except for two:

- E01 Filter On Amiga-specific command, turns on the high-pass filter.
- EFX Invert Loop I have not yet seen a player that would support this command. Perhaps it is supported on some old players.

GS is, in fact, a microprocessor complex with its own processor, ROM, RAM and ports, and is absolutely independent of the Spectrum main processor, which allows, for example, load your favorite module, reset Spectrum, load assembler and create to your favorite music. Soft inside GS completely takes over the tasks playing sound, interpreting the module, etc. GS programming is reduced to transfer byte by byte of the module and / or samples, and then only need to send commands type: start the module, set the global volume for playing the module, start sample # 09 in channel # 02, etc.

If you intend to load the module along with the samples, then GENERAL requires load the module first, and then the samples.

When loading a module, it is highly recommended to leave 2k of memory free, i.e. upload modules with a maximum length of 110K. This condition is not necessary, but its fulfillment highly desirable for compatibility with future versions.

Similarly, it is highly recommended to leave 80 bytes for each sample, for example, if you need to load a 63 kilobyte module and 18 samples, then we have:

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

This is the total length of the samples, which in this state of affairs can be loaded.

If, for example, you need to calculate how much will fit in GS's memory 2 kilobyte samples, it is calculated as follows:

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

In GS'e there are 4 physical channels that play sound.

Channels 0 and 1 are left, and 2 and 3 are right.

3. Interface with the Spectrum.

GS looks at the world using 4 registers:

1. **Command register** - command register, writeable port at address 187 (#BB).

Commands are written to this register.

2. **Status register** - a status register, readable port at address 187 (#BB).

Register bits:

7— Data bit, data flag

6 - Not defined

5 - Not defined

4 - Not defined

3 - Not defined

2 - Not defined

1 - Not defined

0 - Command bit, command flag

This register allows you to determine the state of the GS, in particular whether it can be or write the next data byte, or send the next command, etc.

3. **Data register** - data register available for writing port at address 179 (# B3). IN this register Spectrum writes data, for example, it can be arguments teams.

4. **Output register** - output register, readable port at address 179 (# B3). Of this register the Spectrum reads the data coming from the GS.

The command bit in the status register is set by hardware after the command is written to register of commands. It can be reset to 0 only from GS, which signals about a certain stage of command execution.

The data bit in the status register can be set or cleared as desired.

Spectrum, and at the request of GS: when the Spectrum writes to the data register, it is hardware is set to 1, and after reading by GS from this register it is reset to 0. When writing GS in the output register, it (all the same Databit) is hardware set to 1, and after reading from this port by the Spectrum is reset by hardware to 0.

Although the data register and the output register are located in space port addresses at the same address and affect the same data bit, they are two independent registers. A value once written to one of these registers remains unchanged in it until a new record.

The state of the data bit is very often undefined, and if the command specification is not the values of this bit are determined at certain stages of command execution, it is unacceptable make any assumptions about the meaning of this bit.

4. System of commands GS.

First, I will allow myself a small digression from the actual command system. GS like known to be mainly intended for playing modules and samples. In this version (1.04) GS ROM allows loading one module and / or up to 32 samples.

Each sample, when loaded into memory, gets its own unique identifier, which uniquely determines the reference to this sample in commands that require sample number. The very first loaded sample gets a handle = 1, the next one - number 2, etc.

The same applies to modules and this single loaded module will be

have handle = 1 after loading.

A feature of this version is also that you first need to load the module, and then samples.

Features of the command description:

The commands are described as follows:

1. Hex command code
2. Team name
3. Actions performed during command execution
4. Command format
5. Comments to the team

The command format is described as follows:

GSCOM EQU 187

GSDAT EQU 179

SC #NN: Send command code to command register

```
LD A, # NN
OUT (GSCOM), A
```

WC: Waiting for Reset Command bit

```
WCLP IN A, (GSCOM)
RRCA
JR C, WCLP
```

SD Data: Send data to data register

```
LD A, Data
OUT (GSDAT), A
```

WD: Waiting for Data bit to be reset, essentially waiting until GS accepts data sent to him

```
WDLP IN A, (GSCOM)
RLCA
JR C, WDLP
```

GD Data: Receive data from data register

```
IN A, (GSDAT)
```

WN: Waiting for the Data bit to be set, in fact, waiting for the next data from GS

```
WNLP IN A, (GSCOM)
RLCA
JR NC, WNLP
```

(*):

<PAUSE> - Just a little frame delay of two or three.

And finally - a small sequence against freezing (sometimes it helps)

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


for fidelity, you can also duplicate;).

4.1 GS Commands:

4.1.1 # 00 Reset flags

Clears the Data bit and Command bit flags.

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

4.1.2 # 01 Set silence (*)

Outputs to DACs of all channels # 80. Essentially sets up silence.

SC # 01
WC

4.1.3 # 02 Set low volume (*)

Sets the volume of the DACs of all channels to zero.

SC # 02
WC

4.1.4 # 03 Set high volume (*)

Sets the volume of the DACs of all channels to maximum.

SC # 03
WC

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

Sets the GS 'E' register to the 3 least significant bits according to the specified value (2 the least significant bits are essentially the channel number # 00- # 03).

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

4.1.6 # 05 Out volume port (*)

Sets the volume of the channel contained in 'E' to the specified value.
(The command works if 'E' is between # 00- # 03)

SD Volume (# 00- # 3F)

4.1.7 # 06 Send to DAC (*)

Outputs a byte to the DAC of the channel indicated by 'E'.

SD Byte
SC # 06
WC

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

Outputs a byte to the DAC ('E') at the specified volume.

SD Byte

SC # 07
WC
SD Volume
WD

4.1.9 # 08 - the same as command # 00

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

Setting the volume of the channel, the number of which is specified in the 2x most significant bits.

SD Byte (ccvvvvvv)
SC # 09
WC

cc - Channel number

vvvvvv - Its volume

4.1.11 # 0A DAC output (*)

Another direct output to the DAC.

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

4.1.12 # 0B DAC and Volume output (*)

And finally, the last output to the DAC with volume setting.

SD Fbyte
SC # 0B
WC
SD Sbyte (ccvvvvvv)
WD

Sbyte bit assignment as in # 09

Teams # 01- # 0B are mainly used to build various Covoxes and players without going too deep into the internal structure of the GS.

vvvvvv - Its volume

4.1.13 # 0C Call SounDrive Covox mode (*)

Calls the four-channel Kovoks mode, sequentially copies the data register through the channels. Exit automatically after the fourth byte is output.

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

4.1.14 # 0D Call Ultravox mode (*)

Calls the universal Kovoks mode, sequentially copies the data register by channels, the number of which is adjustable (1-4). Unlike the previous option synchronization is not performed. Logout is also done automatically by recording last byte.

```
SD CHANS
SC # 0D
WC

SD CH1
SD CH2
SD CH3
SD CH4
```

CHANS (4th least significant bit) indicates which channels will be used - to enable the corresponding bit must be set. If the channel is off, then the received byte goes to the next switched on channel (if it has time :)

4.1.15 # 0E Go to LPT Covox mode

Goes into single-channel Kovoks mode, directly copies the data register to the DACs two (right and left) channels. Exit from this mode - writing # 00 to the command register.

```
SC # 0E
WC
```

```
SD \
SD \
... This is the output to the DACs
/
SD /

SC # 00
WC
```

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

Goes into two-channel Kovoks mode, directly copies the data register to the DACs one channel, and the command register in the DACs of the second channel. Exit from this mode - record # 4E into the data register, then sequentially # 0F and #AA into the command register.

```
SD # 59
SC # 0F
WC

SD \
SC \
SD \
SC This is the output to the DACs
... /
SD /
SC /

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

4.1.17 # 10 Out to any port (*)

Outputs bytes to internal GS port (# 00- # 09).

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

4.1.18 # 11 In from any port (*)

Reads a byte from the internal GS port (# 00- # 09).

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

4.1.19 # 12 OUT to 0 port (*)

Outputs a byte to the GS configuration port (# 00).

SD Data
SC # 12
WC

4.1.20 # 13 Jump to Address (*)

Transfers control to the specified address.

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

4.1.21 # 14 Load memory block (*)

Loading a block of codes at a specified address with a specified length.

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

SD \
WD \
SD \
WD Data block length LEN
... /
SD /
WD /

4.1.22 # 15 Get memory block (*)

Unloading a block of codes at a specified address with a specified length.

SD LEN.L

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

GD \
WN \

GD \
WN Data block LEN length
... /
GD /
WN /
GD

4.1.23 # 16 Poke to address (*)

Writes a single byte to the specified address.

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

4.1.24 # 17 Peek from address (*)

Reads a single byte from the specified address.

SD ADR.L
SC # 17
WD
SD ADR.H
Gd byte

4.1.25 # 18 Load DE Pair (*)

Loads the register pair DE (related to GS, not to be confused with the pair of the same name Main CPU) with the specified word.

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

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

Writes a byte to the address specified in DE.

SD Byte
SC # 19
WC

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

Reads the contents of the DE address.

SC # 1A

Gd byte
WN

4.1.28 # 1B Increment of DE Pair (*)

Increases the DE pair by one.

SC # 1B
WC

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

Writes a byte to the address whose high byte is # 20. The command does not make any sense has, since the ROM of the card is located at these addresses.

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

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

Reads a byte from the address, the high byte of which is # 20.

SD ADR.L
SC # 1D
WC
Gd byte
WN

4.1.31 # 1E - # 1F Reserved.

4.1.32 # F1 - # F2 Reserved.

4.1.33 # F3 Warm restart

Clears the entire GS, but skips the steps of determining the number of memory pages and checking them, which greatly speeds up the initialization process.

SC # F3
WC

4.1.34 # F4 Cold restart

Complete restart of GS with all checks. Essentially JP # 0000.

SC # F4
WC

4.1.35 # F5 Busy on

Sets busy flag to #FF

SC # F5
WC

4.1.36 # F6 Busy off

Sets busy flag to # 00

SC # F6

WC

Initially Busy = # 00. All commands in the GS are executed in the main loop command interpreter. This loop can be conditionally represented as follows:

```

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

```

Using the Busy commands, you can, for example, initiate the playback of samples in all channels, then let's say change the playback parameters in the channels, and then run this all at the same time. If they are not used, then the following situation is possible: the first one (the sample will start playing, and only then the second sample is initiated, etc.).

4.1.37 # F7 Get HX Register (*)

Get the contents of the HX (GS) register.

HX participates in the processing of the Busy flag (bit 7 0/1 - Busy On / Off).

```

SC # F7
WC
GD HX
WN

```

4.1.38 # F8 - # F9 Reserved.

4.1.39 #FA Out zero_to_zero

Zero output to zero (configuration) GS port. Pauses the sound music until the next reading from K.L. port.

```

SC #FA
WC

```

4.1.40 #FB - #FF Reserved.

4.1.41 # 20 Get total RAM

Get the total available memory on the GS. (In the basic version it is 112k)

```

SC # 20
WC
GD RAM.L (Junior part)
WN
GD RAM.M (Middle)
WN
GD RAM.H (Major part)

```

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

4.1.42 # 21 Get free RAM

Get the total free memory on the GS.

SC # 20
WC
GD RAM.L (Junior part)

Page 15

WN
GD RAM.M (Middle)
WN
GD RAM.H (Major part)

$\text{Free_RAM} = 65536 * \text{RAM.H} + 256 * \text{RAM.M} + \text{RAM.L}$

4.1.43 # 22 Get GS Variable (*)

Get the value of the GS variable whose number is specified by Num. Variables describe the current state of the card, such as pattern number, tempo, etc. Accordingly numbers variables are not given (due to the unsystematization and fragmentary data, and also for other reasons).

SD Num
SC # 22
WC
GD Variable
WN

4.1.44 # 23 Get number of RAM Pages

Get the number of pages per GS. (Basic version has 3 pages)

SC # 23
WC
GD Number_RAM_Pages

4.1.45 # 24 - # 29 Reserved.

4.1.46 # 2A Set Module Master Volume

Set the volume of playing modules.

SD Module_Master_Volume [# 00 .. # 40]
SC # 2A
WC
[GD Old_Master_Volume] - Old volume

A small example of using this command:
(Assumes module being played)

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
```

The above will smoothly reduce the volume of the playing module and then stop it.

4.1.47 # 2B Set FX Master Volume

Set the volume for playing the effects.

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

```
SC # 2B
WC
[GD Old_FX_Volume] - Old volume
```

Similar to the previous command, but affects samples.

With these two commands, you can adjust the volume balance of the module and samples, and etc.

4.1.48 # 2E Set Current FX

Set the current effect. It just assigns this value to the CURFX variable. If a any command requires a sample handle, you can use this number instead give it # 00 and the interpreter will substitute the value of the CURFX variable instead of zero. (See commands # 38, # 39, # 40- # 4F for an understanding of the above.)

```
SD Cur_FX
SC # 2E
WC
```

4.1.49 # 30 Load Module

Loading the module into memory.

```
SC # 30
WC
[GD Module_Handle] - module number
(Command bit = 0, Data bit = 0)
SC # D1 (Open Stream - open stream)
WC

SD \
WD \
... Module Bytes
SD /
WD /
```

Example:

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

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

LD A, (HL)
LOOP: 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); Waiting for acceptance
JP M, WAIT; last byte
LD A, # D2
CALL SENDCOM
IN A, (GSDAT); Module number
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

Playing a module.

```
SD Module_Handle - module number
SC # 31
WC
```

4.1.51 # 32 Stop module

Stop playing the module.

```
SC # 32
WC
```

4.1.52 # 33 Continue module

Continue playing the module after stopping.

SC # 33
WC

4.1.53 # 35 Set Module Volume

Set the volume of playing modules.

SD Module_Master_Volume [# 00 .. # 40]
SC # 35
WC
[GD Old_Master_Volume] - Old volume

4.1.54 # 36 Data on (*)

Sets the data register to # FF.

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

4.1.55 # 37 Reinitialisation (*)

Resets internal variables to their original state.

SC # 37
WC

4.1.56 # 38 Load FX

Loading an effect sample into memory. Loads unsigned samples (PC type)

SC # 38

WC
[GD FX_Handle] -sample number
(Command bit = 0, Data bit = 0)
SC # D1 (Open Stream - open stream)
WC

SD \
WD \
... Sample Bytes
SD /
WD /

SC # D2 (Close Stream-close stream)
WC

When loading each sample, a header is created in GS memory for this sample, in which describes the various parameters of the sample. After loading these parameters are set to certain values, such as: Note = 60, Volume = # 40, FineTune = 0, SeekFirst = # 0F,

SeekLast = # 0F, Priority = # 80, No Loop and CurFX internal variable is set equal to FX_Handle.

Then with commands # 40, # 41, # 42, # 45, # 46 and # 47 you can change these default values on their own. This is required because command # 39 to initiate playback of the sample uses the parameter values from the sample header.

In their natural form, samples are usually poorly packed with compressors, but compressibility can usually be increased by converting the sample to Delta-view, i.e. keep not the absolute values of the sample, and the relative offset from the previous byte.

This is how you can translate a sample into a Delta-view:

```
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
```

And here's how you can upload a sample:

```
LD IX, Parameters
LD HL, Sample_address
LD DE, 0-Sample_length
LD C, GSCOM

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

LD A, (HL)
LOOP: 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); Waiting for acceptance
JP M, WAIT; last byte
LD A, # D2
CALL SENDCOM
```

```
; Now override the parameters
; samples by default
; values
```

```
LD A, (IX + # 00)
OUT (GSDAT), A; Note
LD A, # 40
CALL SENDCOM
LD A, (IX + # 01)
OUT (GSDAT), A; Volume
LD A, # 41
```

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

4.1.57 # 39 Play FX

Playing the effect.

```
SD FX_Handle - sample number
SC # 39
WC
```

When this command is executed, the following occurs: the channels specified in SeekFirst parameter of our sample, and if at least one of them is free, in it and the sample is played, otherwise the channels specified in SeekLast are watched and if one of them is free, the sample is played in it, if there are no free ones, then all are viewed channels specified by SeekLast, of which the channel with the lowest priority is selected and is compared to the priority of our sample (meaning the sample we want play) if this sample has a higher priority than a sample already playing in channel, then the sample playing in the channel will be stopped, and our sample will be started in this channel instead of the old sample. Here is such a priority scheme ...

Then the sample starts in the channel, then its note, volume, etc. parameters are written to the channel data area from the sample header.

In general, to play the sample with the desired parameters, you can set these parameters after loading the sample and feel free to use command # 39. If the parameters should change, then you can proceed as follows: with command # 2E make the current required sample, change its parameters with commands # 4x, and then start it team # 39.

An alternative method of launching samples is provided by commands # 80 .. # 9F, when executed these commands, you directly in the command code indicate in which channel you want to run the sample, and besides this, you can also specify with what note and / or volume you want to start sample.

4.1.58 # 3A Stop FX in channels

Setting the playback of effects in the specified channels, which are indicated in the mask channels (Channel Mask). In it, a one in the nth bit indicates that the effect in the nth bit the channel needs to be stopped

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

The above is the ideal way for this command to work, but unfortunately not all so simple in this world, and this command does not act like that, namely: one in bit 7 stops the sample at channel zero, etc. In future versions this will be fixed, and for now I can recommend stopping all samples with the # FF mask.

4.1.59 # 3D Set FX Volume

Set the volume for playing the effects.

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

4.1.60 # 3E Load FX (Extended version)

Loading an effect sample into memory. Allows to load signed samples. (Amiga type)

```
SD # 01 (Signed sample)
SC # 3E
WC
[GD FX_Handle] -sample number
(Command bit = 0, Data bit = 0)
SC # D1 (Open Stream - open stream)
WC

SD \
WD \
... Sample Bytes
SD /
WD /

SC # D2 (Close Stream-close stream)
WC
```

4.1.61 # 40 Set FX Sample Playing Note

Sets the default note for the current effect.

```
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 in Amiga)
48 C-4 (C-2 in Amiga)
60 C-5 (C-3 in Amiga)
72 C-6
84 C-7
```

In this version Sound Generators Wave 2, 3 can reproduce octaves 3, 4 and 5, so valid value for the Note parameter is 36 to 71.

4.1.62 # 41 Set FX Sample Volume

Sets the default volume for the current effect.

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

4.1.63 # 42 Set FX Sample Finetune

The default Finetune setting for the current effect.

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

4.1.64 # 43 - # 44 Reserved.

4.1.65 # 45 Set FX Sample Priority

Sets the priority for the current effect. (See Command # 39)

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

After uploading, each effect is set by default priority # 80. Effects, playable in modules have priority # 40.

4.1.66 # 46 Set FX Sample Seek First parameter

Sets the Seek First parameter for the current effect. (See Command # 39)

SD FX_SeekFirst
SC # 46
WC

In the FX_SeekFirst parameter, the 4 least significant bits are used, GS channel numbers are arranged in ascending order.

bit 0 - channel 0
bit 1 - channel 1
bit 2 - channel 2
bit 3 - channel 3

For example, byte [00001010](#) will enable the first and third General Sound channels.

4.1.67 # 47 Set FX Sample Seek Last parameter

Sets the Seek Last parameter for the current effect. (See command # 39)

SD FX_SeekLast
SC # 47
WC
FX_SeekLast format corresponds to FX_SeekFirst format (See command # 46)

4.1.68 # 48 Set FX Sample Loop Begin (*)

Sets the start of the loop for the current effect.

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

If LEN.H - #FF is equal, no looping is performed

4.1.69 # 49 Set FX Sample Loop End (*)

Sets the end of the loop for the current effect.

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

4.1.70 # 4A - # 4F Reserved.

4.1.71 # 51 - # 5F Reserved.

4.1.72 # 60 Get Song Position

Getting the value of the Song_Position variable in the current module.

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

Can be interpreted as the number of patterns played in the module. After the start module takes on the value 0 and increases by one after playing the next pattern. This variable can be used to synchronize processes in Spectrum with playing the module. To do this, for example, at the beginning of the procedure interrupt handling, make SC # 60, then perform various operations procedures with

screen, scrolling lines, etc. (i.e. so that there is enough to execute the command delay), and then read the value of port 179 (GD Song_Position), and compare it with required and, in case of equality, go to the next part of the demos, i.e.

```
if (Song_Position == My_Position)
then goto Next_Part_Of_Demo
```

4.1.73 # 61 Get Pattern Position

Getting the value of the Pattern_Position variable in the current module.

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

Get the offset value in the pattern (current ROW), use is similar the previous command, however, it is required to note that this value changes quite

fast, and therefore

```
if (Pattern_Position >= My_Position) then goto Next_Part_Of_Demo
```

4.1.74 # 62 Get Mixed Position

Get the Pattern_Position value mixed a little with Song_Position.

```
SC # 62
WC
GD Mixed_Position
```

Mixed_Position: (bit by bit)

```
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
```

That is, if you get Mixed_Position and do AND # 3F with it, you get a Pattern_Position, and if after receiving it there are some RLCA, RLCA, AND # 02 - then it will be the lower two bits of Song_Position. See the notes for commands # 60 and # 61.

4.1.75 # 63 Get Channel Notes

Get notes of all channels of the module.

```
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

If in any channel the note value has changed since the last command execution # 63, then bit 7 of the received Note_of_channel_N value will be zero, if this value then the same as before, then this bit will be in one. The lower seven bits are the actual note from 0 to 95. If this value is 127, it means that no samples do not play in the channel. This command is mainly intended for building on its basis various analyzers.

4.1.76 # 64 Get Channel Volumes

Get the volume of all channels of the module.

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

See command # 63 description

4.1.77 # 65 Jump to position (*)

Makes a transition to a given position.

SD Position
SC # 65
WC

4.1.78 # 66 Set speed / tempo (*)

Speed setting within # 01- # 1F. With values # 20- # FF, the tempo is set playing. The tempo values correspond to the original ones at a speed of # 06.

SD Speed / Tempo
SC # 66
WC

4.1.79 # 67 Get speed value (*)

Reading the current speed.

SC # 67
WC
GD Speed
WD

4.1.80 # 68 Get tempo value (*)

Reading the current tempo.

SC # 68
WC
GD Tempo
WD

4.1.81 # 69 Process Sound (*)

Moving to the next quark (or tick) while playing a sound. Maybe in particular, used for synchronization with audio output. To do this, you need to install busy flag (Busy On - which will cause the sound to stop), and then at the desired frequency issue command # 69 for further playback.

SC # 69
WC

4.1.82 # 6A - # 7F are reserved.

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

Playing a sample in the specified channel.

SD Sample_Number
SC # 80 .. # 83 (The least significant bits determine directly the channel number, in which you want to play the sample)
WC

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

Plays a sample in the specified channel with the specified note.

SD Sample_Number

SC # 88 .. # 8B (The least significant bits determine directly the channel number, in which you want to play the sample)
WC
SD Note [0..95]
WD

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

Playing a sample in a specified channel with a specified volume.

SD Sample_Number
SC # 90 .. # 93 (The least significant bits determine directly the channel number, in which you want to play the sample)
WC
SD Volume [# 00 .. # 40]
WD

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

Plays a sample in a specified channel with a specified note and volume.

SD Sample_Number

SC # 98 .. # 9B (The least significant bits determine directly the channel number, in which you want to play the sample)

WC

SD Note [0..95]

WD

SD Volume [# 00 .. # 40]

WD

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

Change the current note in the specified channel. Manufactured on the fly.

SD Note

SC # A0 .. # A3

WC

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

Like the previous command, it changes the channel volume on the fly.

SD Volume

SC # A8..AB

WC

The previous two commands work regardless of what is played in this channel - sample or module. The appearance of a new sound in the channel - from a module or a sample will return everything to its original state - that is, the volume or the note will be those specified newly arrived sound. Due to this tempering ability to obtain the required effect, these commands should be called with a certain frequency (set experimentally).

4.1.89 # B0 - # F0 Reserved

4.1.90 #FA enable test mode

Special command for testing General Sound: 250 (#FA). First into register commands, the #FA command is thrown, after which General Sound switches to the test command mode. Further, in the same command register, we throw the following commands:

Command	Act
----------------	------------

2 - 5	recording in volume 63 and then in sound alternately 0 and 255, until no new command submitted
6	writes to the data register then 0, then 255 (to check the reading of data from side of the spectrum)
7 - 10	writes 255 to sound, then 0 and 255 to volume
11 - 14	sets the maximum volume, beeps to channel 0 and 255, and the volume reduces
fifteen	in all channels sets the maximum volume and gives the saw to sound
sixteen	fetches data from the data register and resets the arrival flag commands
17	takes data and does not clear the command receipt flag
18 - 21	at maximum volume, writes to the sound then 0, then a byte from the register data

4.2 Notes

Commands marked with (*) are undocumented and fully only apply to version 1.04. The operability of these commands in subsequent versions the author of the description (2) is not responsible.

2.I remind you that the registers (their names) mentioned in this description refer only to only to internal registers of GS and nothing to do with registers of the main processor Dont Have.

3.This edition (006) is the most complete and accurate at the moment - April 18, 2002. It is also the last and is more of a cognitive character (according to at least the author sees no other use of this summary of commands, except for high-level emulation GS).

To supplement this text, I was prompted by the fact that after about 4 years I saw my work in the network, and even as part of those descriptions of the card. So I would like to see the dock was as complete and accurate as possible ...

4.3 Correspondence of Amiga and General Sound channel numbering

Despite the fact that the channel layout in stereo in General Sound is similar Amiga Protracker, channel numbering used in trackers does not match the numbering channels in General Sound control commands.

Layout of channels in stereo in Amiga Protracker:

AM1 - left channel
AM2 - right channel
AM3 - right channel
AM4 - left channel

Channel layout in stereo in General Sound:

GS0 - left channel
GS1 - left channel
GS2 - right channel
GS3 - Right Channel

Thus, the channels correspond to each other as follows:

GS0-> AM1
GS1-> AM4
GS2-> AM2
GS3-> AM3

This information is useful when setting SeekFirst / SeekLast parameters for alignment playing modules and effects.

If we translate the channel numbering of the SeekFirst parameter from GS to Amiga numbering, then the channel layout by SeekFirst / SeekLast bits will be as follows:

00001111
||||
3241

For example, if in the Mod-tracker you can see that the module has free channels 1 and 3, then in in most cases it would be wise to first try to play the sample in these channels, and then in any others. To do this, the parameter should be set to% 00001001 in SeekFirst, using GS-channels 0 and 3, which correspond to tracker channels 1 and 3.

5. A few lyrics ...

5.1 Autops GS: (2 pieces;)

5.1.1 Slava Dangerous (X-Trade)

It has the idea of creating a GS'a, the hardware implementation of it, some wishes relatively Soft'a GS'a, as well as the Amiga 1200, on which many have been produced experiments. He is the only and irreplaceable producer of General Sound, and he is The name of production and sales is GS.

5.1.2 Stinger

This is me, the author of this opus and by the joint soul and heart of General Sound. I am the developer of all the built-in Soft'a in GS'e and I presume to continue to engage in sim action. (Yes, I am also the author of some tricky habits in the hardware part GS'a, and would have been the author of many others, if I had not been kept by Glory all the time, constantly preoccupied with price reductions.)

Having written about 20 kb of the code for a half-year, I must say, I am a little tired, but I have enough big plans for the following versions of the GS, for example:

- Wave 4 Sound Generators that reproduce all octaves;
- Acceleration at the expense of these percentages for 30-40 sound generation;
- I would very much like to play STM'ov from PC;
- Advanced system of teams;
- Various special effects over samples;
- Storing the patters in a pre-printed form (about 15% of the original volume);
- And much more.

ALL SOFTWARE MUST WORK AND ON THE FOLLOWING VERSIONS OF if it is written in accordance with my above wishes and requirements. In addition to the described teams, there are an even larger number of teams in GS which are not documented, and I leave it to myself to change them as I please in the manner and only on the basis of the well-documented commands, I accept the legal requirements type: "The document is written like this, but in the firmware it works in a different way ..."

I am planning a significant expansion of the command system, and will be glad to be constructive (desirably specific) offers.

So if you are playing a game or writing a music editor for the GS and find that you really do not have enough of any team, then call me and ask for an offer. (Phone, I think it's not a special job to find out;)

6. General Sound ROM v1.05a

(c) *Stinger, 1997,*

bugfixed by psb & Evgeny Muchkin, 2007.

This firmware version fixes 1.04 Beta glitches.

1. A glitch with modules in which ≥ 63 patterns (klisje.mod, tranceillusion.mod).
2. Glitch with the speed of playing the LAST note of the module, its speed was set to standard, in many modules during looping it was latency is noticeable (e.g. technostyle (z) .mod). Moreover, when looping not on the 1st position, the speed was still set to standard!
3. Incorrect sample playback speed fixed. On some modules it was noticeable that the samples were playing a little faster than necessary (for example, EightMayDay.mod).
4. When the module started playing, GS reported that a note was being played, even if nothing was playing on the channel (command # 64 returned not 127).
5. Added a command for players: # 6A - Set player mode. After this command GS will stop paying attention to the stop command in the module (com. F00). Useful for some modules (bst.mod).

Command format:

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

6. Built-in module reluper. Previously, if a sample was played in the module, the loop length which was too small (tens-hundreds of bytes), GS slowed down or freezes. After this command, the samples in the loaded module are fixed and GS does not slow down.

Command format:

SD MinLoopLen_Low
SC # 6B
WC
SD MinLoopLen_High

The MinLoopLen parameter is set in WORDS and can range from 0 to 16384

(0 - the reducer is off).

Possible short command format:

SC # 6B
WC
SC ...; next GS command

In this case, the default length is 512 words.

ATTENTION! The settings of commands # 6A and # 6B are only reset by hardware RESET or command # F4 (command # F3 does not reset!).

PS The firmware at offset # 0004 contains the version number in BCD format; by offset # 0100 contains the original copyrights (3 lines of 24 characters each); by offset # 0080 contains patch information, line ends with 0.

PPS For old players to work in new modes (items 5 and 6), it is enough before starting them

give commands from BASIC:

OUT 179.1
OUT 187.106
OUT 187.107

PPPS Special thanks to the following people:

- Stinger: for firmware and available sources,
- Aprisobal: without SjASMPlus there would be none of this,
- Evgeny Muchkin: for any assistance in creating the patch,
- Caro: for IDA and moral support,
- SMT & Alone Coder: for UnrealSpeccy (and for fixing glitches in it !:),
- Half Elf: for plugins to FAR,
- n1k-o & Manwe: for advice on mods.

7. Version history

24.02.2012: thx Evgeny Muchkin

- added clause 4.1.90 #FA test mode activation.

01/18/2011: thx moroz1999

- added description 4.1.65 # 45 Set FX Sample Priority;
- added description 4.1.66 # 46 Set FX Sample Seek First parameter;
- added description 4.1.67 # 47 Set FX Sample Seek Last parameter;
- added item 4.3 Correspondence of Amiga and General Sound channel numbering.

07/19/2009: basic version.