

---

**GR8NET** internetworking adapter with multimedia

# Technical Data Book

# Programmer's Guide

For firmware version 0.9  
Date of 30 May 2019 / Datecode 20190530

AGE Labs / Eugeny Brychkov



---

This page is intentionally left blank

---

## GR8NET adapter facts and features

GR8NET is designed as a standalone cartridge to be used with MSX-compatible personal computers. It allows PC accessing internet and intranet, provides comprehensive programming capabilities to programmers. Main features of the adapter are:

RJ-45 UTP Ethernet connection, 10/100 Mbit auto-negotiation / auto MDI, IPv4

TCP, UDP, IPRAW and HTTP communication protocols

Built-in simple web browser to browse web server directory tree

DHCP configuration, DNS auto-resolution

3 user programmable sockets

MSX-BASIC extended command support

Built-in terminal and graphical telnet application

Built-in HTTP-based binary load application

Operation as 1 Megabyte mapped RAM

ROM mapper emulation (plain, Konami, ASCII)

Built-in PSG and SCC/SCC+ for standalone and Konami implementations

Hardware accelerated functions:

- Interrupt generator using time period or frequency, with watchdog
- PCM buffered 8- and 16-bit mono and stereo performance
- Data prefetch from onboard RAM, ROM and W5100
- Micro-SD card interface
- MathPack for FAT16/FAT32 and 32-bit multiplication and division

Native stereo output since serial number 6C

Composite mappers GR8NET + Mapped RAM +Nextor/SDC support + game mapper

Video player for SCREEN 2 (incl. MSX1 w/TMS VDP), 8 and 12 modes

RAM-disk 2KB to 720KB, user programmable size

Operates at standard 3.579545 MHz as well as at overclocked 7.11 MHz bus speeds

Built-in OPLL (YM2413, MSX-Music) with its ROM BIOS (in mapper mode 8)

Built-in OPL-1 with ADPCM (Y8950, MSX-Audio)

MP3 audio streaming from network and SD-card into additional MP3 cartridge

TCP/IP UNAPI version 1.0 implementation

GR8cloud virtual volume through Nextor in composite mapper modes



---

Special thanks to the whole MSX community supporting the development of the device and software, and in particular:

- **Kazuhiko Nishi, Ph.D.** for continuous support of the MSX development activities;
- **Wouter Vermaelen** for suggestions in functionality of the hardware accelerated functions and development of the openMSX model of the device, for supervision and guidance of the OPLL/Y8950 hardware development, and FIR filter design;
- **Albert Beevendorp** for providing his knowledge and experience on the Konami SCC chip;
- **Daisuke Shiraishi (白石大介)** for advising in FPGA configuration development;
- **Nestor Soriano** for allowing and supporting implementation of Nextor for GR8NET, and for support in implementation of the TCP/IP UNAPI;
- **Edo van Zanten** and **Tristan Zondag** for providing measurement and testing for overclocked MSX machines; Edo for testing of MP3 streaming playback;
- **Moonsoft** (Remco Schrijvers) for MoonBlaster v1.4 software ([Freeware license](#));
- **Jos van den Biggelaar** for providing information on and firmware source code of MP3 cartridge for implementation of the MP3 cartridge's firmware fix;
- **Martin de Vries** and **Alex Mena** for testing and troubleshooting MSX-Audio implementation, including MSX-Audio BIOS compatibility;
- **Victor Martinez Sanz** for testing UNAPI implementation;
- **Timo Soilamaa** for concise ID3 tag size [decoding algorithm](#);
- **Ho Fai Ko** and **Nicola Nicolici** of McMaster University of Canada for publishing their MP3 decoder under *McMaster University Open Source Software License* (attached at the end of the manual). Source code [found here](#) is used without modifications, and executable portions of the code are provided free of charge;
- **SymbOS team** (Jörn Proda and Edo van Zanten) for hosting GR8cloud service and volume images on their SymbOS network server;
- **Fabio Roncolato** for testing terminal application and helping revealing the deficiencies in documentation;
- **Carlos Milán Figueredo, Carlos De Santa-Ana Garcia** and **Tobias Keizer** for sharing information on and sources of their implementations of telnet and graphical driver, which were used as source and starting point for the GR8NET built-in TELNET application;
- **David Heremans** for suggesting using Pletter as data compression software;
- **Volodymyr Bezobiuk** for designing and developing code for mapper detection, and for audio subsystem testing.



---

This page is intentionally left blank

---

# CONTENTS

<b>DEFINITIONS .....</b>	<b>10</b>
<b>0. Introduction .....</b>	<b>14</b>
<b>1. Quick user guide .....</b>	<b>15</b>
1.1. How do I ...? .....	20
1.2. BASIC command reference .....	22
<b>2. GR8NET adapter design .....</b>	<b>25</b>
2.1. Physical design .....	25
2.2. Initialization messages and sequence .....	29
2.3. Boot-up menu .....	32
<b>3. Using GR8NET in BASIC .....</b>	<b>34</b>
3.1. Operating in multi-adapter environment .....	34
3.2. Built-in helper.....	36
3.3. Diagnostic commands and mode.....	38
3.4. Overall subsystem status.....	41
3.5. Memory manager .....	42
3.6. Setting up the adapter .....	42
3.6.1. Getting effective configuration information .....	53
3.6.2. DHCP mode.....	55
3.6.3. Fixed IP address configuration mode .....	56
3.6.4. Managing configurations.....	59
3.6.5. Managing system time.....	60
3.6.6. Full configuration with NETCFG command .....	62
3.6.7. Exporting and importing GR8NET configuration .....	65
3.7. Using built-in HTTP-related commands .....	68
3.8. Using NETBLOAD command.....	72
3.9. Built-in communication tools.....	74
3.9.1. Terminal application .....	74
3.9.2. TELNET application .....	76
3.10. Built-in media player .....	80
3.10.1. Playing WAV or MP3 file from network source.....	80
3.10.2. Listening to internet radio.....	81
3.10.3. Playing WAV or MP3 files from SD-card.....	81
3.10.4. Playing wave from GR8NET RAM .....	82
3.10.5. Playing video from SD-card .....	84
3.11. Setting memory mapper .....	89
3.12. Managing audio mixer .....	92
3.13. Getting state of the resource (SD-card or network).....	95
3.14. Serial flash chip and configuration image management .....	97
3.15. Other utilities.....	91

---

<b>4. Using GR8NET as BASIC I/O device</b>	<b>109</b>
4.1. Using network device names in BASIC I/O	110
4.2. Identification of the network resource	111
4.3. BASIC operators and functions to use for network access	111
4.4. Sending datagrams or delayed TCP data	112
4.5. BASIC I/O using TCP devices	113
4.6. BASIC I/O using UDP device	115
4.7. BASIC I/O using IPRAW device	116
4.8. BASIC I/O using HTTP/HTTR devices	117
4.9. The URL parser	118
<b>5. Built-in web browser</b>	<b>119</b>
5.1. Opening SD-card located file in the browser	125
<b>6. Using integrated MSX-DOS</b>	<b>126</b>
6.1. GR8NET Disk subsystem (DOS1)	126
6.1.1. Initialization of the DOS1 disk subsystem	127
6.1.2. Using GR8NET DOS1 disk subsystem	129
6.2. Nextor disk subsystem	133
<b>7. GR8cloud virtual volume</b>	<b>135</b>
7.1. Setting up GR8cloud virtual volume	136
7.2. Precautions and disclaimers	137
7.3. Acknowledgements	138
<b>8. Built-in MSX-Audio, MSX-Music and PSG</b>	<b>139</b>
8.1. Starting with built-in OPLL and Y8950	142
8.2. Playing games with built-in OPLL	143
8.3. Using built-in MSX-Audio	143
8.4. Alternative interface to Y8950 and OPLL registers	145
8.5. Using built-in PSG	146
<b>9. GR8NET functionality extensions</b>	<b>148</b>
9.1. Regular GR8NET image	149
9.2. Embedded MP3 player	149
9.2.1. Limitations of GR8NET in MP3 player mode	150
9.3. Catalogs	151
9.3.1. Making the catalog	152
9.3.2. Putting the catalog into the GR8NET serial flash chip	152
<b>10. Firmware upgrade</b>	<b>153</b>
10.1. The order of the firmware update	155
10.2. Diagnosing firmware faults	156
10.3. Location of the firmware update files	156
10.4. Online method of firmware update	157
10.4.1. Online flash chip update	157
10.4.2. Online FPGA chip firmware update	158

---

---

10.5. Offline method of firmware update.....	158
10.5.1. Offline update of the onboard flash chip.....	159
10.5.2. Offline update of the FPGA firmware.....	160
10.6. Online migration from version 0.7 to version 0.8 .....	162
<b>11. GR8NET technical reference.....</b>	<b>164</b>
11.1. Identification and detection .....	164
11.2. Setting operating mode and mapper type .....	166
11.3. Logical page assignment .....	169
11.4. Special control registers.....	171
11.5. Hardware-accelerated functions .....	174
11.5.1. Controlled interrupt generator with watchdog .....	174
11.5.2. PCM function .....	178
11.5.3. Prefetch function.....	181
11.5.4. Combining functionalities of generator, prefetch and PCM .....	184
11.5.5. Sound custom chip (SCC/SCC+) .....	184
11.5.6. Digital waveform input and Music Module DAC .....	187
11.5.7. Volume registers .....	187
11.5.8. Micro-SD card interface .....	189
11.5.9. Math-Pack .....	192
11.5.10. Mixer and DAC (digital to analog converter).....	196
11.5.11. System registers.....	196
11.5.12. FPGA flash chip interface .....	199
11.5.12.1. FPGA flash chip access control/status register .....	199
11.5.12.2. Data read.....	200
11.5.12.3. Data write.....	201
11.5.12.4. Sector erase.....	202
11.5.12.5. Serial flash chip information.....	204
11.5.13. Current FPGA image properties.....	204
11.5.14. MP3 player interface.....	206
11.5.14.1. Identification of the MP3 GR8NET in the system.....	208
11.5.14.2. Issues with MP3 decoder.....	209
11.6. Mapper modes.....	198
11.6.1. Mode 0: GR8NET internetworking adapter.....	212
11.6.2. Mode 1: plain 32kByte write-protected memory chunk .....	213
11.6.3. Mode 2/3: Konami memory mappers .....	213
11.6.4. Mode 4: ASCII-8 memory mapper .....	213
11.6.5. Mode 5: ASCII-16 memory mapper.....	214
11.6.6. Mode 6: Mirrored ROM .....	214
11.6.7. Mode 7: 1 Megabyte mapped memory .....	214
11.6.8. Modes 8-14: Composite mappers.....	215
11.6.8.1. RAM allocation conflicts in composite mappers.....	216
11.6.8.2. Limitations of setting target mapper to composite mappers.....	218

---

<b>12. Programming API.....</b>	<b>219</b>
12.1. Identification of the adapter .....	219
12.2. Direct firmware calls .....	220
12.3. URI structure.....	228
12.4. TCP/IP UNAPI implementation .....	230
12.5. Video file formats.....	232
12.6. Networking libraries for Fusion-C (SDCC-based) .....	235
<b>13. Applications .....</b>	<b>240</b>
13.1. MSX webserver .....	240
13.2. FTP client.....	241
13.3. Video player .....	243
13.3.1. Making videos for MSX .....	244
13.3.2. Converting .SC2 file from version 0 to version 1 format .....	244
13.4. Heroes of Might and Magic III demo .....	245
13.5. Card game "DURAK" .....	247
13.6. GR8cloud server .....	248
<b>14. Troubleshooting .....</b>	<b>252</b>
<b>15. Examples of the code .....</b>	<b>257</b>
<b>16. References .....</b>	<b>262</b>
<b>17. Legal statements.....</b>	<b>263</b>
17.1. MP3 audio decoder legal statements.....	263
17.1.1. Citation of the work being used .....	263
17.1.2. McMaster University Open Source Software License.....	263
17.2. Pletter legal statement .....	264
<b>18. Document revision history .....</b>	<b>265</b>

---

## DEFINITIONS

### **Octet**

The set of 8 bits – the byte. This term is usually used to denote the part of IP address. IPv4 address consists of four octets, and thus 32 bits

### **Variable**

The entity in programming language (e.g. MSX-BASIC) which can get various values within pre-defined range. For example, integer variable may be of any value from -32758 to 32767 if it is considered as signed, and from 0 to 65535 if it is considered unsigned. String variable is a pointer to the memory location where string's characters (bytes) are stored. String variable has size which equals to number of characters in it. Useful characters in the string can be terminated with special control character (usually 0 or carriage return)

### **Configured value**

Value which is currently in effect for the system or subsystem. In order to configure specific value, you will have to issue control command putting set of values into effect

### **Gateway**

Device on the subnetwork which provides routing to the wider network (e.g. internet). Gateway can also serve additional functionality acting as DNS or DHCP server

### **DNS**

Stands for domain name server – the device which translates human-readable conventional internet names like [www.gr8bit.ru](http://www.gr8bit.ru) into IP address. For successful translation this server needs access to other name servers, that's why Gateway is a very good place to implement DNS for subnetwork

### **DHCP**

Stands for dynamic host configuration protocol, which is used to automatically obtain configuration information without having IP addresses and mask configured manually. For successful automatic configuration, subnetwork should have DHCP server configured and running. Usually one physical subnetwork has only one DHCP server

### **Mask**

Bitmap of 4 octets (32 bits) identifying membership of specific host with specific IP address in the subnet. If wrong mask is set up, you may not get access to subnet resources and the internet

### **CRLF**

Two characters, CR and LF, with CR having decimal code 13 and LF having decimal code 10. CR stands for "carriage return" which returns cursor to the beginning of current line, and LF stands for "line feed" which moves one line down. These terms come from dot matrix printing era, when CR was physical move of the printing head to the position 0, and LF was move of the shaft, and thus paper, one character position forward

---

## PCM

Pulse-code modulation, the array of 8-bit or 16-bit samples representing the waveform on the time dimension

## DAC

Digital-to-analog converter, the software, firmware and hardware implementation which converts digital code which represents specific voltage level to this analog voltage level

## Default adapter

The adapter which is used by default to service BASIC CALL statement and device I/O when user types CALLNET commands without numeric identification of the adapter, or uses BASIC I/O device name without numeric identification of the adapter

## Default URI structure

This structure is used as a template for generating actual URI structures. For example, if default structure contains [www.gr8bit.ru](http://www.gr8bit.ru) as host name, */software/* as a path, and *myfile.dat* as a name, calling URI parser with URI string */software/roms/mg2.rom* will cause output URI structure with same host name, same destination and source ports, but with path */software/roms/* and file name *mg2.rom*

## Dynamic source port number

Each TCP session is identified by the port number host uses at its end (source port) and another host's end (destination port). For example, for HTTP communication usual destination port number is 80. If same *source* port number is used for several sessions to the same remote host's remote port, it will confuse remote host and whole communication. Thus GR8NET implements dynamic port numbers in range 0c000h-0ffffh, every connection request dynamic port number is increased by 1 thus explicitly removing possible issues related to confusion of the sessions in complex communication. To enable dynamic source port usage set source port number by CALLNETSETPORT command to 0.

## Y8950 or MSX-Audio

Y8950 is a chip incorporating OPL version 1 frequency modulation sound generator, and ADPCM (adaptive differential pulse-code modulation) engine. It can produce FM tones, and play specially coded and compressed samples giving realistic audio output. MSX-Audio is MSX device based on the Y8950 chip. This chip and related circuits built on it is detected by port I/O commands. Standard port numbers for first MSX-Audio device are 0C0h/0C1h, for second MSX-Audio in the system are 0C2h/0C3h.

## OPLL or MSX-Music

OPLL is abbreviation of FM **O**perator Type-**L** Light, a chip, produced by Yamaha under part number YM2413, represents simplified version of Y8950 without ADPCM. Accessing this chip is easier, but it also has limitations having only one custom instrument and 18 preset instruments. This chip does not respond to CPU reads, and to identify its presence software usually looks for its ROM containing special APRLOPLL signature at its beginning.

---

MSX-Music is an MSX device based on the YM2413 chip. It has several variations and may be called FM-PAK, FM-Pac; it may have built-in SRAM or be without it.

### **Flash chip**

Flash chip is an electronic non-volatile computer storage medium that can be electrically erased and reprogrammed. GR8NET is having two flash chips in it: one is a parallel device, attached "front-end" and used for code run by MSX Z80 CPU (GR8NET ROM BIOS), and second is serial device, attached "back-end" and used for FPGA configuration and some other data.

### **GR8NET engine**

The term describes main functional core of the GR8NET – the FPGA chip and its configuration.

### **GR8cloud**

The GR8cloud provides virtual remote volume to the Nextor disk subsystem in mapper modes 8-14, and appears as local volume. Thus wherever you have GR8NET connected to the internet, you will have your GR8cloud volume available given GR8cloud server is accessible from the GR8NET connection location (e.g. ports are not blocked).



---

This page is intentionally left blank

---

## 0. Introduction

GR8NET internetworking adapter is designed to provide networking and multimedia capabilities to the MSX home personal computers.

GR8NET is relatively complex device made of contemporary high-technology components, and owning it is not only a fun, *but also a responsibility*. Please familiarize yourself with DO's and DON'Ts before you start operating it.

### DO:

- Install adapter only into MSX-compatible computers;
- Use adapter only with proven fault-free equipment – including network equipment, Micro-SD cards and audio equipment;
- Regularly examine edge connector of the cartridge, and slot connector cartridge to be installed into for dirt, and clean this dirt with ethanol / spirit or any other solution intended for connector cleaning;
- Install Micro-SD card with its pins facing front of the cartridge;
- Insert cartridge into the slot by slightly pressing onto its casing, and remove cartridge from the slot holding cartridge by its casing and gently pulling it out;
- Select different switch combinations for adapters installed in the system;
- In case of questions or problems seek advice from manufacturer/designer and from community.



Cotton bud after connector cleaning; dirty edge connector may cause instability in your PC operation using external devices

### DON'T:

- Do not disassemble cartridge and repair it yourself unless you are asked by authorized person;
- Do not store or operate cartridge in dusty places;
- Do not expose cartridge to the sun rays, excessive heat or excessive cold;
- Do not allow water, moisture or any other solutions getting into the box;
- Do not clean body of cartridge or labels with alcoholic solutions or solvents;
- Do not try disconnecting LAN cable by forcefully pulling it out using cord, press RJ-45 connector's latch first before removing the cable by its head;
- Do not forcefully insert Micro-SD card into its slot; it should be enough to apply slight vertical pressure to get card in.

There's a difference how adapter behaves on cold and warm boot. On cold boot (reset or power cycle) adapter will load its configuration from the flash chip; on warm boot GR8NET will not load configuration information making configuration set before reboot still being effective.

---

# 1. Quick user guide

This chapter covers main functionalities of GR8NET from user experience point of view; if you need more technical details please refer to the respective section of the manual. This manual is searchable, just press ^F and put keyword to search for in.

If you have issues please refer to **Troubleshooting** chapter. Then, if you do not find answers to your questions please contact [info@gr8bit.ru](mailto:info@gr8bit.ru) with your questions.

---

## Before installing adapter

If you install more than one GR8NET adapter into your system, please ensure that adapters are having different IDs set by their DIP switches at the bottom. ID set up in binary system, with both off being ID #0, and both on being ID #3.



---

## Getting help on the commands

It is as simple as typing `_HELP`, or just two underscores, `__` – and it will display list of all the commands available in default GR8NET. To view NET commands listing, type `_NETHELP`, to see specific command help type

`_NETHELP<command>`

If you find built-in help insufficient, please refer to the respective section of this manual.

```
callhelp
GR8NET (#2) Type CALL_HELP followed by commands listed below ( _ can be replaced
by spaces or removed completely). # character identifies family of the commands.
CALL command length should not exceed 15 characters (remove _ or spaces).
.NET# _IDSK# _FL#
[.NET] command family
.NET_BITOV _NET_PLAYBUF _NET_GET# _NET_SET#
.NET_VAR# _NET_IP _NET_MASK _NET_RAM
.NET_DNS _NET_DUMP _NET_STAT _NET_DHCP
.NET_FIX _NET_TELNET _NET_TERM _NET_DIAG
.NET_BLOAD _NET_LDBUF _NET_LDRAM _NET_RCHKS
.NET_SNDDTG _NET_BROWSE
Break
Ok
callnethelpsetmap
CALL NETSETMAP(A,M,MPD) Argument A defines type of the memory mapper GR8NET shou
ld switch to. Argument M is mapped RAM register read flag (0, 1 or 2 (autodetect
, default)). MPD is mapped RAM pending disable bit (0 or 1). To switch to mapper
mode 8-14 card should be installed in primary slot, otherwise illegal function
call error is given. After mapper type is set, machine reboots. If arguments are
omitted command tries to start ROM if it was loaded into GR8NET buffer RAM
Ok
color auto goto list run
```

---

## Getting firmware versions (firmware location)

`_NETVER` command shows you adapter ID (device), version of the code in flash chip, flash chip firmware build datecode and FPGA configuration version (engine).

```
callnetver
Device: 03
Flash: V.00.08 (2019/1/8)
Engine: V.00.08 (2018/12/29)
Image: 00 (factory), sector 00
Type: regular
HW_rev: 01 (stereo)
Ok
```

---

## Full configuration of the adapter

Configuration from scratch can be performed using `_NETCFG`, and exporting/importing of the configuration is done by `_NETEXPT`/`_NETIMPRT` commands.

---

## Setting up network

Most convenient mode is DHCP mode (`_NETDHCP` command). *Remote host* name should be resolved (otherwise there's an issue with DNS being used).

Note that NETSAVE command saves current mode (thus if card did not get DHCP configuration and booted in Fixed IP mode, after NETSAVE it will continue booting in Fixed IP mode).

```
DHCP successful
IP: 192.168.1.108
Mask: 255.255.255.0
DNS: 1.1.1.1
NTP: 192.168.1.155
DHCP: 1.1.1.1
MAC: 08:00:00:04:05:11
Mode: <#0>
Nov 05, 2018 20:55:14 (UTC)
Remote host:
www.gr8bit.ru (195.208.1.126)
```

---

---

## When SD-card can be used

- In mapper modes 8-14 with Nextor. Nextor will mount first and only one supported partition (FAT12/FAT16) of the card; to use other partitions you will have to use Nextor's `_MAPDRV` command. SD-card in DOS1 mode is not supported;
- With GR8NET commands like `_NETBROWSE`, `_NETPLAYWAV`, `_NETPLAYVID`, `_NETBLOAD`. These commands support first partition having FAT16/FAT32 formats and are read-only.

You can develop your own SD-card driver with full read-write functionality, GR8NET hardware allows it. It was deliberately chosen not to provide write functionality under BASIC without standard BIOS support (e.g. Nextor), thus there's no driver for DOS1 mode.

---

---

## Bootting from SD-card with Nextor (get system files)

- Only possible in mapper modes 8-14, e.g. to switch to mapper mode 8 use `_NETSETMAP(24)`. For these composite mapper modes do not forget to add 16 to the mapper number when using `_NETSETMAP`;
  - First partition of SD-card must be formatted as FAT12 or FAT16, it must contain files `NEXTOR.SYS` and `COMMAND2.COM`;
  - Card may be formatted using PC, or using Nextor's `_FDISK` utility (note there's a bug in Nextor and you may need to apply [this fix](#) to formatted card to have it working with GR8NET SD-card BASIC commands).
- 
- 

## Saving data onto SD-card

SD-card is available for writing in mapper modes 8-14 driven by Nextor, and by using `_DSKSVIMG` command in any mapper modes with GR8NET ROM (0 and 8-14). Useful application examples: [FTP](#) client for GR8NET written in BASIC and [Konamiman's networking tools](#).

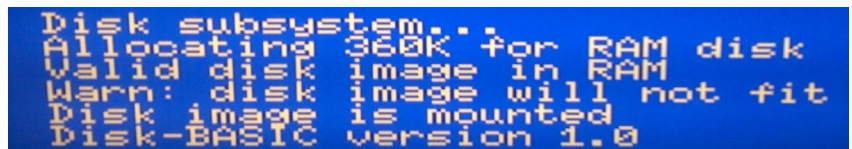
---

---

## Using GR8NET RAM-disk

During GR8NET initialization:

- Press and hold F4 during adapter initialization to enable built-in Disk-ROM;
- Press and hold F2 until key release notice to invoke web browser and select target image to load;
- Press and hold F1 to force image reload if there's valid image in RAM already;
- Press and hold F3 to disable built-in Disk-ROM;
- You can combine keys, e.g. pressing and holding F4 and F2 simultaneously.



```
Disk subsystem...
Allocating 360K for RAM disk
Valid disk image in RAM
Warn: disk image will not fit
Disk image is mounted
Disk-BASIC version 1.0
```

At any time you can **load new image** into RAM disk using `_DSKSETIMG` and then `_DSKLDIMG` commands; previous contents of RAM disk will be overwritten and lost.

At any time you can **save disk image** onto SD-card using `_DSKSVIMG` command

---

---

---

---

## Using built-in browser (ROMs, programs)

- Invoke browser by `_NETBROWSE` in BASIC;
- You can browse internet, and up to 4 first partitions of SD-card formatted with FAT16/FAT32.

When in the browser:

- Press alphanumeric key to get to next file starting with the respective character;
- Press SELECT key to reload the web page;
- Press CLS/HOME key to go to the bottom/top of the page;
- Press TAB key to play WAV file (when running regular FPGA image) or MP3 file (when running MP3 player FPGA image);
- Use ^V key combination to play video from SD-card;
- Press ENTER key to load and execute ROM, BASIC program or binary file;
- To have auto-run of the ROM with specific mapper type include target mapper type into the file name between curly braces, e.g. *metalgear2{3}.rom* will auto-start in Konami5 mapper mode.



Index of /roms/	Name
parent directory	
2,612	.htaccess
1,047	@license.txt
262,144	aleste{5}.rom
16,384	ant-adv{1}.rom
131,072	arkanoid2{2}.rom
32,768	arkanoid{1}.rom
16,536	aufmonty{4}.rom
131,072	blackonyx2{5}.rom
262,144	dragonslayer4{4}.rom
262,144	frogger{1}.rom
32,768	galaga{1}.rom
32,768	goonies{1}.rom
32,768	green-beret{1}.rom
48	header.html
16,384	hyper-olympic{1}.rom
16,384	hyper-sports{1}.rom
16,384	billiard{1}.rom
32,768	knightmare{1}.rom

Please refer to [Built-in web browser](#) chapter for supported web page formats, and `_NETBROWSE` execution flags

---

---

## Target mapper

GR8NET by default boots into mapper mode 0. It is possible to override it and request GR8NET to automatically reboot into other mapper mode 8-14 using `_NETTGTMAP` command. After changing the setting, do not forget to save it using `_NETSAVE`. Be aware that in composite mapper modes 8-14 GR8NET RAM buffers are limited to 512K only, and you will have to boot in mapper mode 0 to perform specific tasks (e.g. update firmware of the adapter). To skip target mapper re-initialization, press and hold arrow down key after machine reset and during adapter initialization, and it will forcefully boot in default mapper mode 0.

---

---

## Playing media files (audio, video)

- With regular FPGA image: you can play WAV files using web browser (TAB key) or `_NETPLAYWAV` command from network or SD-card. For network play please mind network throughput from the remote server, for far-away locations playback hiccups may occur;
  - With MP3 player FPGA image: you can play MP3 files using web browser (TAB key) or `_NETPLAYWAV` command from network or SD-card;
  - With external MP3 cartridge: you can play MP3 files using web browser (TAB key) or `_NETPLAYWAV` command from network or SD-card. External MP3 cartridge is detected as *MP3/ext* and also listed during GR8NET initialization as one of the identified devices;
  - You can listen to internet radio e.g. `_NETPLAYWAV("http://ic3.101.ru:8000/v13_1");`
  - You can play MSX video files (SC2, SC8 and SCC for respective screen mode) from SD-card (no network playback) using ^V in web browser or `_NETPLAYVID` command.
- 
-

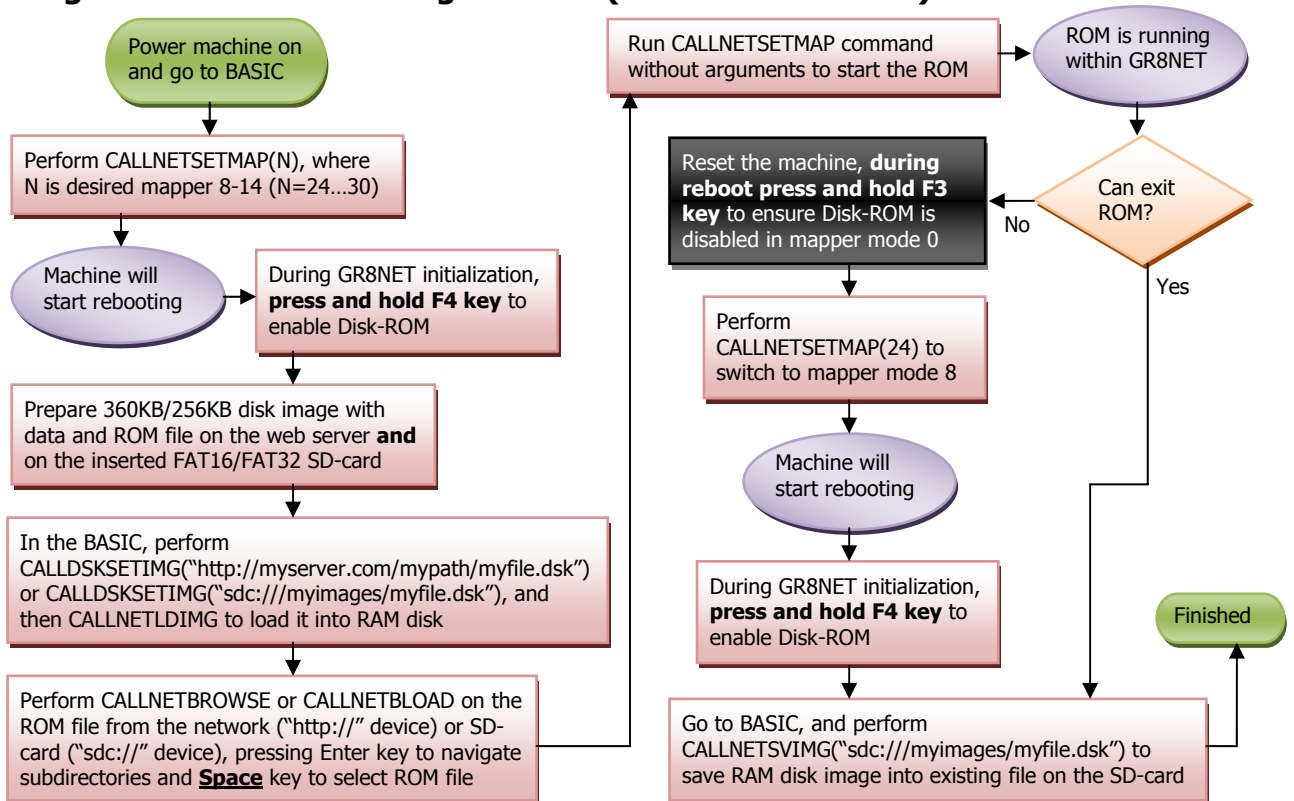


---

## Making ROM accessing data and saving results of its work

Some games and other ROMs may work with files – for example, loading and saving game states, or loading and saving audio samples and songs. Before deciding how you feed ROM with data, you must decide if it needs input data, and you need to save its output data. You will be able to use GR8NET RAM disk and GR8NET Nextor-based SD-card in game mapper modes 8-14, when GR8NET expands slot and having GR8NET ROM with RAM disk in subslot 0, 512K RAM in subslot 1 (can be disabled), Nextor in subslot 2 and game mapper in subslot 3. Remember that GR8NET RAM appearing in subslot 0 is the mirror of the game mapper ROM space, thus performing `_NETBLOAD` or `_NETBROWSE` will cause game mapper contents change/corruption.

### Using RAM disk as a working medium (for ROMs ≤ 128KB)



As you can see from the flowchart above, you can load data into RAM disk from network or from SD-card, but can save RAM disk image only to already existing disk image file on the SD-card. It is extremely important at step highlighted black to have Disk-ROM turned off during startup, because if enabled, GR8NET will start loading default image into GR8NET RAM location overwriting and corrupting previous RAM disk image.

It would be a good practice to have several diskette images on the SD-card, and rotate images when saving so that previous version of the disk image would be kept as a previous backup copy.

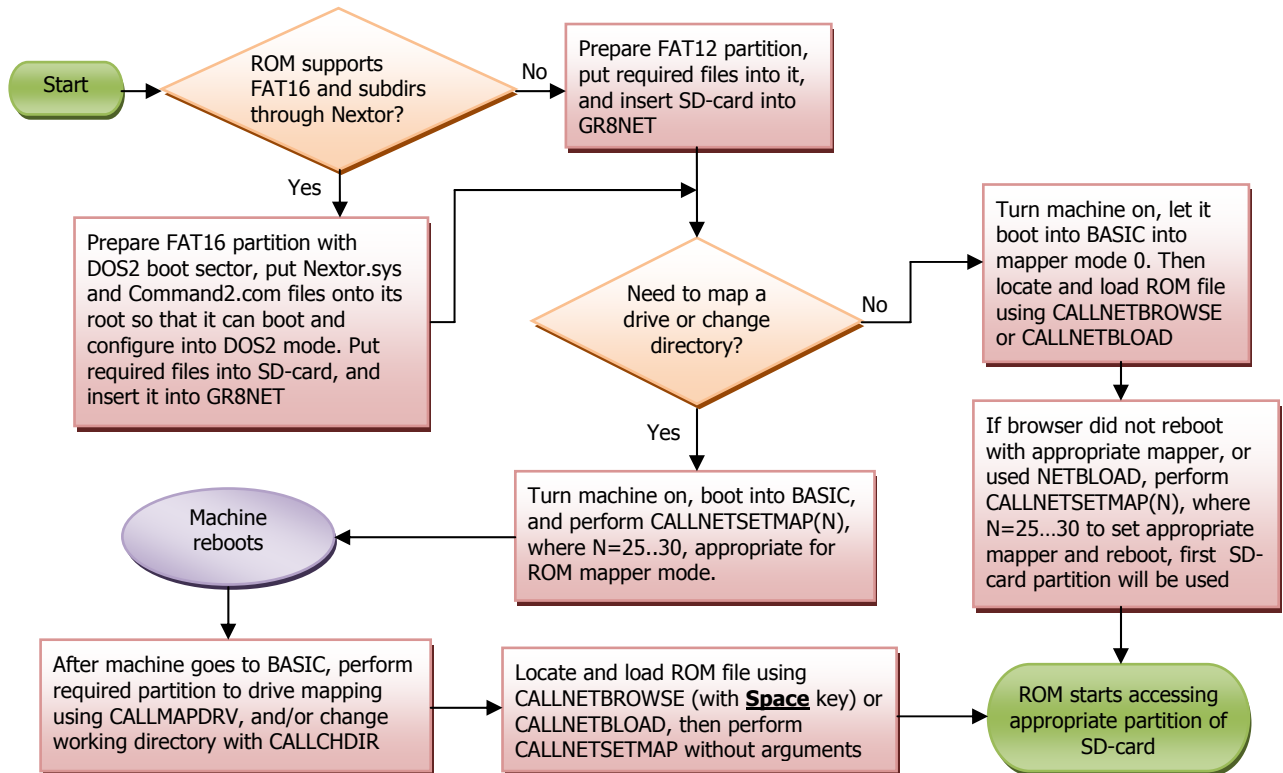
Some mappers may only have 256 KB space for RAM disk: these could be custom 256 KB volumes, or 360 KB volumes with space above 256 KB marked as bad blocks in the FAT.

The empty images of the diskettes can be found at <http://www.gr8bit.ru/software/booting/empty-images/>.

GR8NET RAM allocation within the flowchart described above is: 512KB of mapped RAM, 360K of reserved RAM disk space, and 152KB of user space (128KB for ROM). No Y8950 sample RAM possible in this configuration.

---

## Using SD-card with Nextor as data medium



Note about using FAT16: it may be required to boot into the DOS2 operating system before application can access FAT16 volume, that's why Nextor.sys and Command2.com files may be required on the FAT16 drive.

Note about using FAT12: to force DOS1 mode you may need to press and hold '1' key during GR8NET initialization up to its beeping forcing Nextor initializing in DOS1 mode.

---

### How to use SD-card with multiple partitions with \_NET commands

\_NET commands (e.g. \_NETBLOAD) support only FAT16 and FAT32 partitions; you address partition 0 by device SDC://, partition 1 by device SDD://, partition 2 by device SDE:// and partition 3 by device SDF://. To identify if SD-card is installed and supported partition exists, use NETRESST command. This command is useful to get fact of existence and properties (file attributes and size) of the resource on SD-card.

---

### How to connect GR8NET to my amplifier?

First of all, disconnect MSX with GR8NET installed and/or amplifier from the power mains to ensure there's no potential between systems when connecting, then insert standard 3.5 mm audio plug into GR8NET's 3.5 mm jack at its right side. GR8NET **output level** is within  $\pm 1.5 V_{PK}$ , and you may need to adjust receiving volume at your amplifier side.

---

---

## 1.1. How do I ...?

### ... Run game in ROM format up to 512KB ROMs

1. You must load ROM image into GR8NET buffer RAM using `_NETBLOAD` or `_NETBROWSE`. Source of data can be network or SD-card;
2. Prepare your system – in particular GR8NET sound subsystem with commands like `_NETSETMIX("bbblrl")`  
`_NETFKOPLLR`
3. Select appropriate mapper type with `_NETSETMAP`. See explanation of this command and mapper types.

### ... Run game in ROM format up to 1MB in size

1. Only ASCII-8 and ASCII-16 mappers support these mapper sizes;
2. You must ensure that, in composite mapper modes 9-12, you have RAM-disk disabled and MSX-Audio sample RAM set to 0 or MSX-Audio disabled completely.
3. Running the ROM of up to 1MB follows the same process as up to 512KB

### ... Run diskette image

#### System diskette images, demo images (Noisedisk, Unknown Reality disk 1...)

You can use GR8NET RAM disk to load disk image into – in mapper mode 0 size of image is up to 720 KB, in mapper modes 8-14 size are 360 KB and 256 KB (see memory allocation tables in [Memory manager](#) chapter).

1. Prepare your system – in particular GR8NET sound subsystem with commands like `_NETSETMIX("bbblrl")` « setting can be saved by `_NETSAVE`  
`_NETFKOPLLR`
2. Perform machine soft reset using  
`DEFUSR=0:A=USR(0)`  
Do NOT use reset button otherwise mixer settings will be reset;
3. When GR8NET starts initialization, press and hold F4 key to enable built-in Disk ROM. You can also press and hold F2 (simultaneously to F4) to invoke browser and select image with SPACE key. Alternatively, you can use `_DSKSETIMG` and `_DSKLDIMG` from BASIC, but for these command to work properly GR8NET Disk ROM must be already enabled, and space for RAM disk allocated at the GR8NET startup;
4. If diskette image is having DOS on it with `AUTOEXEC.BAT` file, or `AUTOEXEC.BAS` file, then they will start automatically.

### ... Run multi-diskette application

Use SofaRun application – copy it and required disk images into the directory on the SD-card, insert SD-card, prepare GR8NET audio subsystem, and switch GR8NET to required mapper mode if needed:

1. `_NETSETMIX("bbblrl")`
2. `_NETFKOPLLR`
3. `_NETSETMAP(28)`



---

The last command will soft reboot machine into GR8NET + 512K mapped RAM + Nextor + ASCII-8 game mapper in subslot 3 (GR8NET must be installed in primary slot).

**... Run application requiring SCC/SCC+**

The SCC/SCC+ device is now available in the mapper mode 8 in subslot 3 – together with MSX-Music ROM. Thus you can use mapper mode 8 if your game needs SCC device, but does not require K5 mapper as is – for example [project Melancholia's SD-snatcher](#) using SofaRun.

---

## 1.2. BASIC command reference

### Detection and identification

<b>_NETGETDA</b>	Get default adapter number and list of active adapters
<b>_NETSETDA</b>	Set default adapter number

### Command help, diagnostics and status

<b>_NETHELP</b>	Get help on the GR8NET commands and their arguments
<b>_NETCODE</b>	Returns communication status and HTTP response code
<b>_NETSYSINFO</b>	Get system information and system performance data
<b>_NETDIAG</b>	Turn built-in diagnostic output on or off
<b>_NETSTAT</b>	Display status information about the adapter
<b>_NETGETMMV</b>	Get memory manager values
<b>_NETSETMMV</b>	Set memory manager value

### Managing operating mode

<b>_NETSETMAP</b>	Set specified memory mapper type and reboot
<b>_NETGETMAP</b>	Get current memory mapper type and some other operating flags
<b>_NETTGTMAP</b>	Set target memory mapper configuration to switch to at the startup

### Managing configuration

<b>_NETSAVE</b>	Save current configuration into ROM configuration page
<b>_NETCFG</b>	Interactive GR8NET configuration
<b>_NETEXPRT</b>	Create BASIC program containing GR8NET configuration data
<b>_NETIMPRT</b>	Fill GR8NET system variables with data from BASIC program created by NETEXPRT

### Network management

<b>_NETIP</b>	Get current configured adapter's IP address
<b>_NETMASK</b>	Get current configured adapter's subnetwork mask
<b>_NETGW</b>	Get current configured gateway
<b>_NETDNS</b>	Get current configured domain name server (DNS) IP address
<b>_NETDHCP</b>	Perform DHCP discovery and dynamic configuration
<b>_NETFIX</b>	Configure fixed IP address information into network system
<b>_NETGETIP</b>	Get fixed IP address value
<b>_NETGETMASK</b>	Get fixed IP address mask
<b>_NETGETGW</b>	Get fixed IP address mode gateway IP address
<b>_NETGETDNS</b>	Get fixed IP address mode DNS IP address
<b>_NETSETIP</b>	Sets fixed configuration IP address value
<b>_NETSETMASK</b>	Set fixed configuration IP address mask
<b>_NETSETGW</b>	Set fixed configuration gateway IP address
<b>_NETSETDNS</b>	Set fixed configuration DNS IP address
<b>_NETCDTOF</b>	Copy DHCP configuration into fixed IP address configuration
<b>_NETVARRWTH</b>	Set networking RX window threshold
<b>_NETVARUDTO</b>	Set UDP packet timeout for DHCP and DNS operations

### System time management

<b>_NETNTP</b>	Get effective configuration of NTP server
----------------	---

<b>_NETGETNTP</b>	Get NTP server properties within fixed IP address configuration
<b>_NETSETNTP</b>	Set NTP server properties within fixed IP address configuration, and time setting flags
<b>_NETTSYNC</b>	Display and synchronize system time
<b>_NETSETTSHN</b>	Set time server host name
<b>_NETGETTSHN</b>	Get time server host name
Network or SD-card access	
<b>_NETBROWSE</b>	Invoke internet and SD-card browser in BASIC
<b>_NETBLOAD</b>	Load binary file from the remote web server using HTTP
<b>_NETRESST</b>	Get status of the resource
<b>_NETSETHOST</b>	Set remote host name, and perform simple DNS query if needed
<b>_NETSETPATH</b>	Set remote resource's path
<b>_NETSETNAME</b>	Set remote resource's file name
<b>_NETSETQSTR</b>	Set query string for remote resource processing
<b>_NETGETHOST</b>	Get remote host name and IP address
<b>_NETGETPATH</b>	Get remote resource's path
<b>_NETGETNAME</b>	Get remote resource's file name
<b>_NETGETQSTR</b>	Get query string set up for remote resource
<b>_NETSETPORT</b>	Set communication port numbers in the default URI structure
<b>_NETGETPORT</b>	Get communication port numbers
<b>_NETVARBSIZE</b>	Get size of bloaded data in bytes
<b>_NETVARBRSTR</b>	Get URI string of the location selected by user within the browser
Communication tools	
<b>_NETTERM</b>	Perform terminal session using TCP
<b>_NETTELNET</b>	Perform telnet session using TCP
Multimedia management commands	
<b>_NETPLAYWAV</b>	Play wave or MP3 file, or listen to internet radio
<b>_NETPLAYBUF</b>	Play wave from GR8NET RAM
<b>_NETPLAYVID</b>	Play video file from SD-card
<b>_NETSETMIX</b>	Set mixer configuration
<b>_NETGETMIX</b>	Set mixer configuration
<b>_NETSNDVOL</b>	Set or display GR8NET audio volume levels
<b>_NETGETCLK</b>	Get GR8NET or MSX bus clock frequency
<b>_NETSETCLK</b>	Set clock source for the GR8NET clock speed calculation
<b>_NETFKOPLL</b>	Fake OPLL ROM into mapped RAM
<b>_NETGETOPL</b>	Gets status of built-in OPLL/Y8950, and initial setting of sample RAM size
<b>_NETSETOPL</b>	Enables or disables built-in OPLL/Y8950, controls doubling of output amplitude, and sets sample RAM size
<b>_NETSETPSG</b>	Set built-in PSG properties
<b>_NETGETPSG</b>	Get built-in PSG properties
GR8NET RAM buffer / machine RAM access	
<b>_NETDUMP</b>	Dump data from adapter's buffer RAM
<b>_NETLDBUF</b>	Load data from main memory to adapter's buffer RAM

<b>_NETLDRAM</b>	Un-load data from adapter's buffer RAM to main memory
<b>_NETRCHKS</b>	Calculate simple 16-bit checksum on the buffer RAM contents
<b>_NETGETMEM</b>	Read 4 consecutive bytes (32-bit data) from the memory
<b>_NETSETMEM</b>	Write 4 consecutive bytes (32-bit data) into the memory
<b>_NETGETMD</b>	Get 32-bit double word from memory converted to double-precision and stored in variable
<b>_NETSETDM</b>	Convert double-precision value to 32-bit double word and store this dword into memory
SD-card low level access	
<b>_NETSDCRD</b>	Read sectors from the SD-card
Graphics-related routines	
<b>_NETBTOV</b>	Move binary data from the GR8NET buffer to VRAM
<b>_NETBITOV</b>	Move icon image from GR8NET buffer to VRAM
GR8NET RAM disk	
<b>_DSKGETIMG</b>	Get current disk image location
<b>_DSKSETIMG</b>	Set disk image location
<b>_DSKLDIMG</b>	Loads image into the RAM disk area in GR8NET buffer RAM
<b>_DSKSVIMG</b>	Saves RAM-disk image onto SD-card
<b>_DSKCFG</b>	Obtain or manage state of disk image
<b>_DSKFMT</b>	Initialize RAM-disk image
<b>_DSKSTAT</b>	Get/set state of the disk subsystem
GR8cloud management	
<b>_NETSETCLOUD</b>	Set up GR8cloud virtual volume access
<b>_NETGETCLOUD</b>	Prints GR8cloud virtual volume status onto the screen
GR8NET firmware update commands	
<b>_NETFWUPDATE</b>	Update the firmware – contents of the onboard flash ROM BIOS chip
<b>_NETFPGAUPD</b>	Update the FPGA (EPCS chip) firmware (discontinued)
BASIC network file support	
<b>OPEN</b>	Open network stream in HTTP, TCP, UDP or IP RAW modes
<b>CLOSE</b>	Close network stream and free the network socket
(other BASIC I/O commands)	PRINT#, PRINT# USING, LINEINPUT#, INPUT#, INPUT\$, MAXFILES=, LOC(f), LOF(f), EOF(f), LOAD, MERGE
<b>_NETSNDDTG</b>	Send datagram / pending data to the remote host
Flash chip management	
<b>_FLINFO</b>	Display information about flash chips installed
<b>_FLLIST</b>	List contents of the serial flash chip
<b>_FLUPDATE</b>	Update contents of the serial flash chip (replaced _NETFPGAUPD)

---

## 2. GR8NET adapter design

It is important that you understand how you can use adapter with your MSX system, and which configuration options you have – from hardware and from software/firmware perspectives.

### 2.1. Physical design

GR8NET internetworking adapter is packaged into standard Konami-size cartridge, having semi-transparent body. It has several indicating devices which are seen through this semi-transparent casing.



Figure 1. Adapter set

Product consists of the adapter, ribbon cable and adapter board to connect GR8blaster, USB-Blaster or ByteBlaster-II to the GR8NET adapter.

Please note that according to user feedback batch #2 (May 2016) and further Byteblaster-II device **is not supplied any more**.



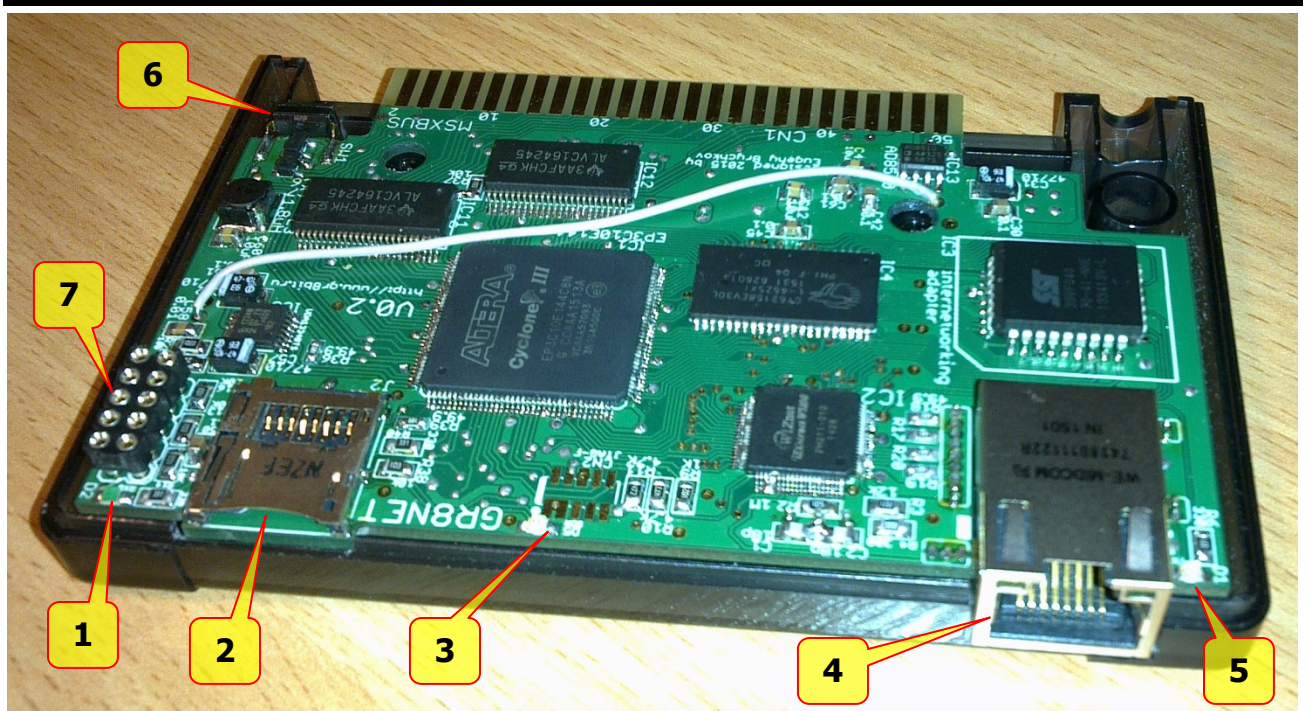


Figure 2. Components of the GR8NET adapter

Adapter consists of the following components (fig. 2):

1. Receive LED, green, seen through the semi-transparent cartridge casing;
2. Micro-SD card slot, card is inserted with its contacts to the front;
3. SD-card indication LED, red, seen through the semi-transparent cartridge casing. Off means no card or card is idle, regular flash of 3 times per seconds means SD-card initialization failure;
4. RJ-45 10/100 Mbit network connector;
5. Transmit LED, yellow, seen through the semi-transparent cartridge casing;
6. Configuration switch;
7. FPGA active serial configuration connector. This figure shows mono GR8NET cartridge; stereo one is having slightly different sized active serial configuration connector, but the orientation remains the same. Just follow guidelines matching pins and securely inserting the adapter board (more details – in [Offline update of FPGA firmware chapter](#)).



Figure 3. GR8NET view from the top

Top view (fig. 3):

- 8. Network activity LED, green;
- 9. Connection speed LED, on = 100 Mbit, off = 10 Mbit;
- 10. Slot in the cartridge to insert Micro-SD card – with card's pins to the front.



Figure 4. GR8NET view from the rear

Rear view (fig. 4):

- 11. Serial number/MAC address. Use light highlighting the text to see through the semi-transparent casing.

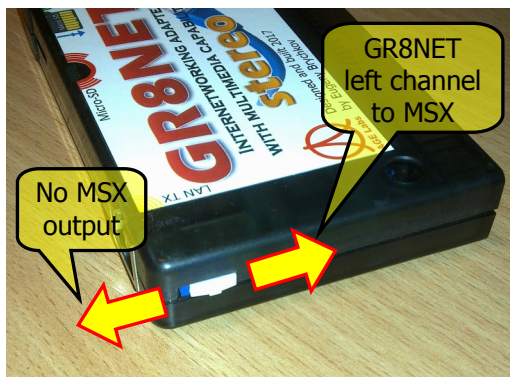




Figure 5. GR8NET view from the bottom

Micro-switches at the bottom of the cartridge (fig. 5) should be dealt with caution – when changing them please use thin, but not sharp object, and do not press towards the switch but rather in the direction of the switch thumb movement.

Stereo version of the GR8NET board is having two additional features



MSX mixer switch



3.5 mm jack line-out or headphones

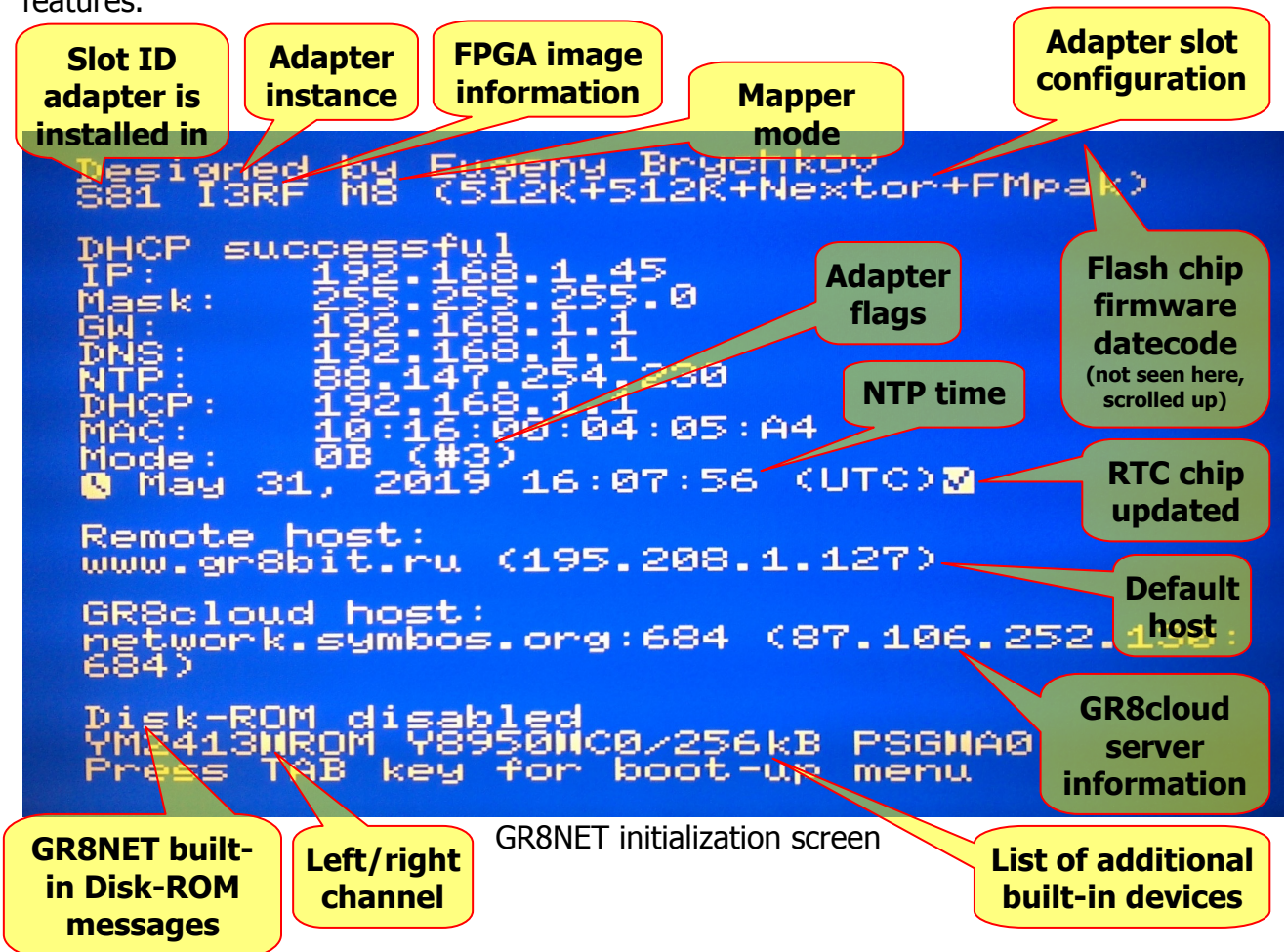


It is highly advisable that you disconnect MSX machine (or other device being connected) from the power mains when connecting that other device as line-in signal recipient. There could be a difference in grounding system of all devices involved, and thus AC potential, when connecting; as **tip of the connector** goes first – left channel wire – connecting to GR8NET ground when inserting into the connector, there's a risk of left channel experiencing excessive voltage at the GR8NET and/or signal recipient's side. This symptom is usually very well heard by the strong and loud low frequency (50 or 60 Hz) roaring sound from the output device. Damages caused by not following this rule will be considered as out-of-warranty cases. In case of other device damage or failure, neither AGE Labs nor Eugeny Brychkov might be held liable for such negative consequences.



## 2.2. Initialization messages and sequence

To better understand adapters' functionality let's look into how it initializes, and where you should look at to find vital information about adapter's configuration and features.



The following items are displayed on the GR8NET initialization screen:

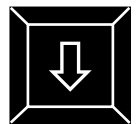
- Firmware datecode: the date flash chip firmware was built, for example 20190529 is 29 May 2019. This information is not seen on the screenshot as it was quickly scrolled up because of multiple messages GR8NET prints in composite mapper modes (in this case – mapper mode 8). In mapper mode 0 datecode will be seen for enough time for being pictured. With every release of firmware this manual is being updated, thus please look for updates regularly at <http://www.gr8bit.ru> site;
- Slot ID is the slot where adapter is installed in RDSLT format – with MSb being set if slot is expanded, bits 1:0 indicating primary slot number and bits 3:2 indicating secondary slot number (e.g. 86 means slot 2.1);
- Adapter instance # is selection of the micro-switch at the bottom of cartridge, please refer to [Operating in multi-adapter environment](#);
- FPGA image information shows the following identifiers: R=regular image, M=MP3 player image, F=factory image, A=application image;

- Mapper mode and adapter memory configuration shows how 1MB of RAM is allocated: it could be 1024K for mapper 0 (whole adapter RAM is used for GR8NET purposes), or 512K+512K in mapper 8-14 (512K is used for GR8NET purposes, and 512K is used as mapped RAM). Two last entries, if displayed, show the configuration of subslot 2 and subslot 3 respectively;
- [Adapter flags](#) identify status of the adapter, see [System registers](#). Disk ROM enable bit will be set to the state of disk subsystem status of the previous adapter initialization, and will be appropriately changed during further Disk ROM initialization. Since May 2018 there will be '+' character displayed after adapter flags value if GR8NET built-in SCC implementation is in SCC+ mode;
- NTP time is the time returned by the NTP server (if request was successful). RTC updated checkmark symbol means that RTC chip time is updated with NTP server time. See [Managing system time](#);
- Default host is the one set by \_NETSETHOST command, and is being resolved during startup;
- GR8cloud server information is only shown when Nextor subsystem is available (composite mapper modes 8-14), and identify the network location of your dedicated virtual volume being used;
- For details on Disk ROM initialization and operation please refer to [Using integrated MSX-DOS](#);
- And list of devices configured during adapter initialization – they may include YM2413 (OPLL/MSX-Music), Y8950 (MSX-Audio with configured access I/O port and sample RAM size), GR8NET built-in PSG. Sign within the device identifiers **PSG1A0** are left and right speakers – if channel is muted, then dash is displayed instead of the picture of the speaker.

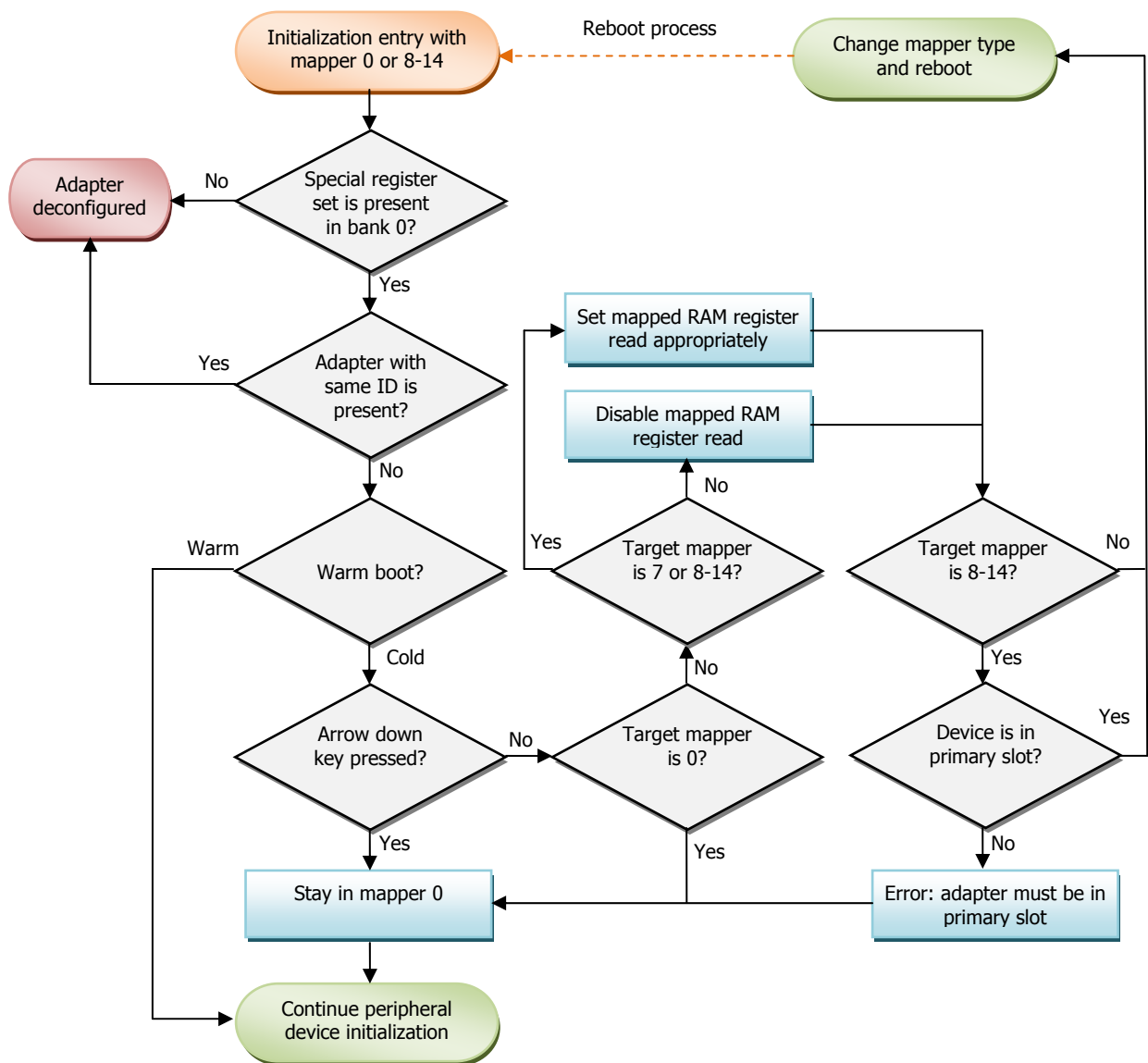
The picture below shows detailed GR8NET mode initialization sequence. During cold boot (after power cycle or hard reset) GR8NET firmware is initialized in mapper mode 0, and depending on the setting of *target mapper mode* (see [Setting memory mapper](#)), adapter will reboot machine in required mapper configuration. Most useful target mapper configuration is mapper mode 8, when half of GR8NET onboard RAM (512K) is dedicated for GR8NET, and another half (512K) is configured as mapped RAM for the machine.

Note that to operate in mapper modes 8-14 GR8NET adapter **must** be installed in primary slot, because GR8NET internally performs expansion of the primary slot and presents its functions in subslot 0 and mapped RAM in subslot 1 of the slot it is installed in.

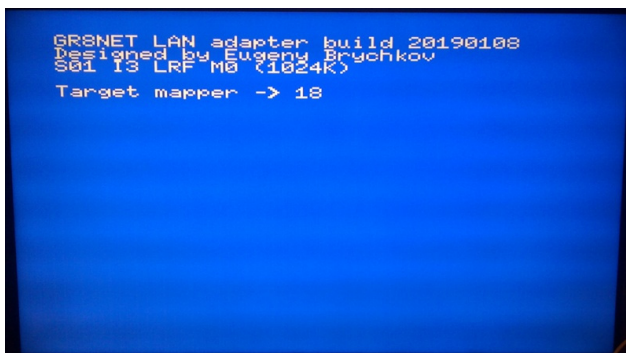
To cancel re-initialization of the adapter into target mapper configuration *during the cold boot*, press arrow down key on the keyboard before adapter starts initialization, and hold the key until adapter displays network configuration information. This action will force adapter to continue with mapper mode 0.



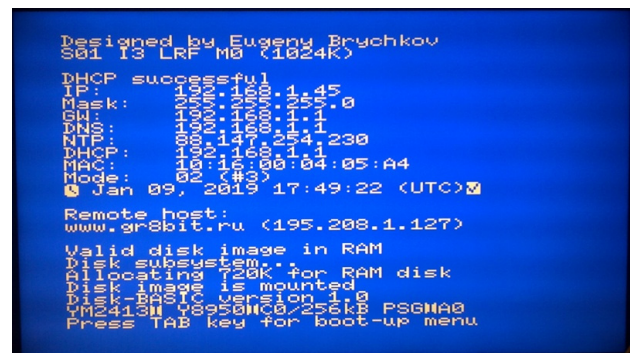
Such method was deliberately chosen to allow user cancelling adapter reconfiguration using MSX keyboard if installation is not supported, or adapter needs reconfiguration, or firmware update must be performed which can be done in mapper mode 0 only.



GR8NET mapper mode initialization sequence



Adapter automatic reconfiguration from mapper mode 0 to target mode 8 with special register set in GR8NET bank 0



Messages of adapter in mapper mode 0 (1024K), Disk ROM enabled and disk image present in GR8NET RAM



## 2.3. Boot-up menu

When GR8NET finishes its initialization in mapper modes 0 and 8-14, it displays message "Press TAB key for boot-up menu". If you press TAB key this time, or have TAB key held pressed, GR8NET will ask you to release TAB key and will enter boot-up menu.

```
Mask:      255.255.255.0
GW:        195.208.1.1
DNS:       195.208.1.1
NTP:       195.208.1.1
DHCP:      192.147.254.230
MAC:       10:16:00:04:05:A4
Mode:      0A (#3)
Dec 15, 2018 17:48:28 (UTC)

Remote host:
www.gr8bit.ru (195.208.1.127)

Disk-ROM disabled
YM2413M Y8950MC0/256kB PSGMA0
Release keys...

- Boot up menu -

1. Help screen
2. Clean up buffer RAM
3. Reset SCC mode
4. Change screen refresh rate
5. Finish
```

You can perform several very useful and important tasks using the boot-up menu before GR8NET attempts to start its game mapper (in subslot 3 for mapper modes 9-14) and before further cartridges are being initialized by the system.

```
>2
- Location to clean -
1. Y8950 sample RAM only
2. RAM disk space only
3. Whole GR8NET buffer RAM
5. Exit to main menu
```

Cleaning GR8NET RAM is essential if there's something in its game mapper location not allowing system to get to DOS or BASIC properly

```
>4
- Change refresh rate -
Current rate: 60 Hz
1. Set to 60 Hz
2. Set to 50 Hz
5. Exit to main menu
```

You can change VDP refresh rate. For MSX1.5 machines there will be no "current rate" displayed as it is not stored anywhere in the system.

```
>1

During GR8NET initialization you
may press and hold the following
hotkeys:

TAB  Invoke this boot-up menu
ESC  Skip start-up delay after
      adapter finishes initialization
F5   Forcefully disable GR8cloud
F4   Enable built-in Disk-ROM
F3   Disable built-in Disk-ROM
F2   Invoke browser to load RAM disk
      image, Disk-ROM must be enabled
      (e.g. using simultaneous press
      of F4 key)
F1   Forcefully reload RAM disk image
      even if valid image is detected
      in RAM, Disk-ROM must be enabled
      (e.g. using simultaneous press
      of F4 key)

Press space key to continue■
```

Help is very useful if you forgot GR8NET initialization hot keys. You will need to reset machine in order to use the keys as boot-up menu is a final step of GR8NET initialization and pressing keys at this stage will have no effect.

---

### 3. Using GR8NET in BASIC

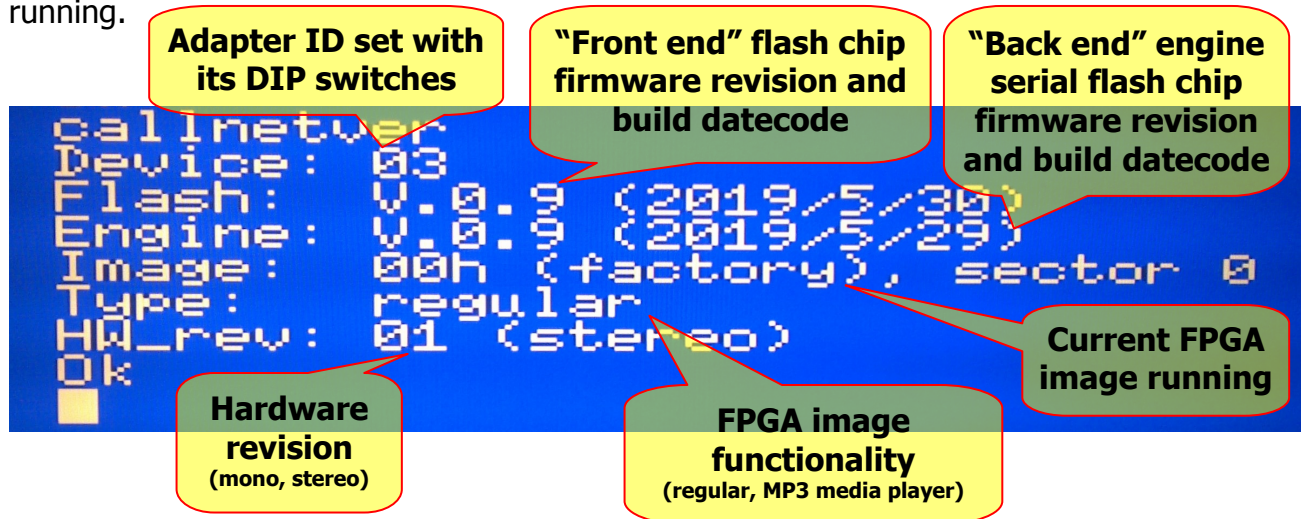
Setup and management of the adapter is performed using CALL statements, and standard BASIC file input/output operators.

Extended BASIC statements are used to configure and manage adapter's operation. When using listed CALL statements, you can leave space between keyword CALL and command name, but the command name should be typed without spaces.

While command names may look long, you will find that their names are convenient from their functionality point of view. There's a group of SET commands, and corresponding GET commands. CALL statements can not return value, and such division was decided as best fit for programming user interface with the adapter and internet.

Programmer has 3 sockets at disposal for BASIC file I/O operations, each socket can operate in TCP, UDP or RAW mode, and operate independently. These sockets are shared with TCP/IP UNAPI implementation.

First BASIC command worth knowing and using is CALLNETVER, giving the information about adapter, its hardware revisions, firmware revisions, and current image running.



#### 3.1. Operating in multi-adapter environment

MSX system may have up to 4 GR8NET adapters installed; they are differentiated by the switch setting of the adapter. Each installed adapter **must** have unique number set for it.

There's a notion of the **default** adapter – the GR8NET card which responds to the CALL and device I/O commands if these commands do not explicitly contain adapter number identification in their name. For example, `_NETGETIP0` will explicitly query adapter #0, while `_NETGETIP` will query *default* adapter. If system has only one GR8NET adapter, after reset it will be configured as the default adapter.

Addressing adapters using their numbers may not provide required flexibility, thus two statements were introduced in order to change default adapter number and operate needed adapter without adapter number following the CALL statement.

## NETGETDA

Get default adapter number and list of active adapters

### Format

CALL NETGETDA (A,B)

### Arguments

Argument A will receive current default adapter number (0-3), argument B will receive bitmap status of the adapters installed.

Bit	Adapter
0	Adapter #0 status
1	Adapter #1 status
2	Adapter #2 status
3	Adapter #3 status

If respective bit is set, adapter is active in the system and you can set it as default adapter and use its device I/O resources or manage it using CALL statements. If you believe adapter is physically present in machine, but respective bit in argument B is reset, this may mean that adapter under consideration is faulty or was deactivated during initialization process (after hardware reset) due to conflict with another adapter.

Any of arguments may be omitted. If statement is executed without arguments it displays information onto the screen.

### Example

\_NETGETDA

Default: 03

Present: 3x1x

## NETSETDA

Set default adapter number

### Format

CALL NETSETDA (A)

### Argument

Argument A is mandatory, it should have value 0-3 and identify active adapter in the system (see NETGETDA).

### Usage

After executing this command for e.g. adapter 2 "\_NETSETDA(2)" application can use e.g. "\_NETGETIP" (without adapter number identification) to manage adapter #2.

Example 1 – code is fixed to adapter numbers 2 and 3

```
10 _NETSETHOST2("www.google.com"):REM adapter 2
```

```
20 _NETSETHOST3("www.gr8bit.ru"):REM adapter 3
```

Example 2 – code is fixed to adapter numbers 2 and 3

```
10 _NETSETDA(2):_NETSETHOST("www.google.com"):REM adapter 2
```

```
20 _NETSETDA(3):_NETSETHOST("www.gr8bit.ru"):REM adapter 3
```

Example 3 – initialization of adapter is not hardwired to its number and defined by the variable

```
10 DATA "www.google.com","www.gr8bit.ru"
```

```
20 RESTORE 10
```

```
30 FOR I=2 TO 3:READ A$:_NETSETDA(I): _NETSETHOST(A$):NEXT I
```



## 3.2. Built-in helper

GR8NET contains at least some text help for all its CALL commands. To start with help subsystem, you can use the following commands:

`_HELP` or `__` (two underscores) or `CALL _`

All three commands display all the GR8NET commands in all the command trees. The list can be quite long, and you can press STOP to suspend listing, and CTRL-STOP to interrupts it.

If you need help on sub-tree "NET", type `_NET` or `_NETHELP`.

Only default adapter will respond to the CALL command, thus if you want to display another adapter's helper, add adapter number to the command: `_NETHELP2`.

Commands are organized as tree: for example, `_NETSETMAP` contains `NET`, `SET` and `MAP` – and it is well seen in `_HELP` command output – sub-trees are having \* identifier after them. The following commands are valid:

Command	Function
<code>_HELP</code>	Display top level help and all sub-trees
<code>_NETHELP</code>	Display NET level help and its sub-trees
<code>_NETSETHelp</code>	Display NET→SET help sub-tree
<code>_NETSETHelpMAP</code>	Display help for MAP command
<code>_NETSETHelp3MAP</code>	Display help for MAP command of adapter #3
<code>_NET3HelpSETMAP</code>	The same as above
<code>_NETSETMAP3</code>	Execute <code>_NETSETMAP</code> command for adapter #3

As you can see, the following rules apply:

Commands consist of the keywords listed in the `_HELP` command in the hierarchical format;

Keyword `HELP` is reserved word and causes executing helper instead of executing specific command;

Adapter identifier ("3" the examples above) and keyword `HELP` can appear at any place in the string– except `HELP` must NOT appear after "end leaf" command.



```
Ok
callnetsethosesthost
CALL NETSETHOST(N#1A,B,C,D) Set r
emote host property. If argument is a
string please use server part of URI
only, without protocol definition (e
.g. http://), and without trailing sl
ash, with maximal length of 31 chara
cters. Example: www.gr8bit.ru. Firmwa
re will try to resolve the host name
to IP address, and if this operation
is unsuccessful, further operations w
ith it may be refused with error. If
argument is IP address, hostname's st
ring is reset and you may not be able
to use HTTP-related functions which
require HOST: instruction in the requ
est field. If you input IP address, a
ny octet can be omitted, but at least
one should be present
Ok
■
color auto goto list run
```

Figure 6a. Help on `_NETSETHOST` command



```

GR8NET (#2) Type CALL_HELP followed by commands listed below (_ can be replaced
by spaces or removed completely). % character identifies family of the commands.
CALL command length should not exceed 15 characters (remove _ or spaces).
.NET%      _DSK%      _FL%
[.NET] command family
.NET_BITOV  _NET_PLAYBUF  _NET_GET%    _NET_SET%
.NET_VAR%   _NET_IP       _NET_MASK    _NET_GW
.NET_DNS    _NET_DUMP     _NET_STAT    _NET_DHCP
.NET_FIX    _NET_TELNET   _NET_TERM    _NET_DIAG
.NET_BLOAD  _NET_LDBUF    _NET_LDRAM   _NET_RCHK%
.NET_SND%DTG _NET_BROWSE   _NET_VER     _NET_CDTOF
.NET_SAVE   _NET_FWUPDATE  _NET_PLAYMAV _NET_SERVER
.NET_SDCRD  _NET_BTOV    _NET_CODE    _NET_SNDVOL
.NET_NTP     _NET_TSYNC    _NET_TGTHAP  _NET_CFG
.NET_SY%INFO _NET_FKOPLL%R  _NET_PLAYVID _NET_FPGAUPD
.NET_RE%ST   _NET_REC%G
[.NET_GET] command family
.NET_GET_HOST  _NET_GET_IP    _NET_GET_MASK  _NET_GET_GW
.NET_GET_DNS   _NET_GET_PORT  _NET_GET_PATH  _NET_GET_NAME
.NET_GET_MAP   _NET_GET_DA    _NET_GET_MEM   _NET_GET_MD
.NET_GET_MMV   _NET_GET_NTP   _NET_GET_TSHN  _NET_GET_QSTR
color          auto          goto          list          run

```

Figure 6b. Full command tree, part 1 (output is suspended by STOP key)

```

.NET_GET_HOST  _NET_GET_IP    _NET_GET_MASK  _NET_GET_GW
.NET_GET_DNS   _NET_GET_PORT  _NET_GET_PATH  _NET_GET_NAME
.NET_GET_MAP   _NET_GET_DA    _NET_GET_MEM   _NET_GET_MD
.NET_GET_MMV   _NET_GET_NTP   _NET_GET_TSHN  _NET_GET_QSTR
.NET_GET_CLK   _NET_GET_OPL   _NET_GET_MIX   _NET_GET_CLOUD
.NET_GET_PSG
[.NET_SET] command family
.NET_SET_HOST  _NET_SET_IP    _NET_SET_MASK  _NET_SET_GW
.NET_SET_DNS   _NET_SET_PORT  _NET_SET_PATH  _NET_SET_NAME
.NET_SET_MAP   _NET_SET_DA    _NET_SET_MEM   _NET_SET_DM
.NET_SET_MMV   _NET_SET_NTP   _NET_SET_TSHN  _NET_SET_QSTR
.NET_SET_CLK   _NET_SET_OPL   _NET_SET_MIX   _NET_SET_CLOUD
.NET_SET_PSG
[.NET_VAR] command family
.NET_VAR_BSIZE _NET_VAR_RWTH  _NET_VAR_BRSTR _NET_VAR_UDTO
.NET_VAR_IMPRT _NET_VAR_EXPRT
[.DSK] command family
_DSK_GETIMG   _DSK_SETIMG    _DSK_LDIMG    _DSK_CFG
_DSK_FMT      _DSK_STAT     _DSK_SVING
[.FL] command family
_FL_INFO      _FL_LIST       _FL_UPDATE
Ok
color          auto          goto          list          run

```

Figure 6c. Full command tree, part 2

### 3.3. Diagnostic commands and mode

GR8NET has several commands to display diagnostic information about itself, last command it was running, and your MSX computer, as well as turning the special networking diagnostic mode on when GR8NET prints messages regarding the stages of the communication with remote host – these messages will help figuring out what is wrong with the communication.

# NETDUMP

## Dump data from adapter's buffer RAM

### Format

CALL NETDUMP (P, A, C)

## Arguments

All arguments are variables or constant and are mandatory

## Usage

P is the logical page number to switch to in GR8NET bank 1 (6000-7FFF) for dumping, A is starting address, and C is byte count. You can dump any location of the memory visible to the CPU (see fig. 8), but when this command executes, CPU banks 1 and 2 (4000-BFFF) contain GR8NET, and therefore you will not be able to dump e.g. second part of the machine Main ROM (4000-7FFF), or dump RAM location occupied by BASIC program (in 8000-BFFF).

Dumping output will adjust to the width of the screen. Maximal dumping bytes per line is 16. Warning: be wary about dumping of special registers' prefetch data registers as their reading will cause prefetch pointer to change according to number of reads of the register. During command printing the dump, you can press STOP to suspend the listing, or CTRL-STOP to interrupt it.

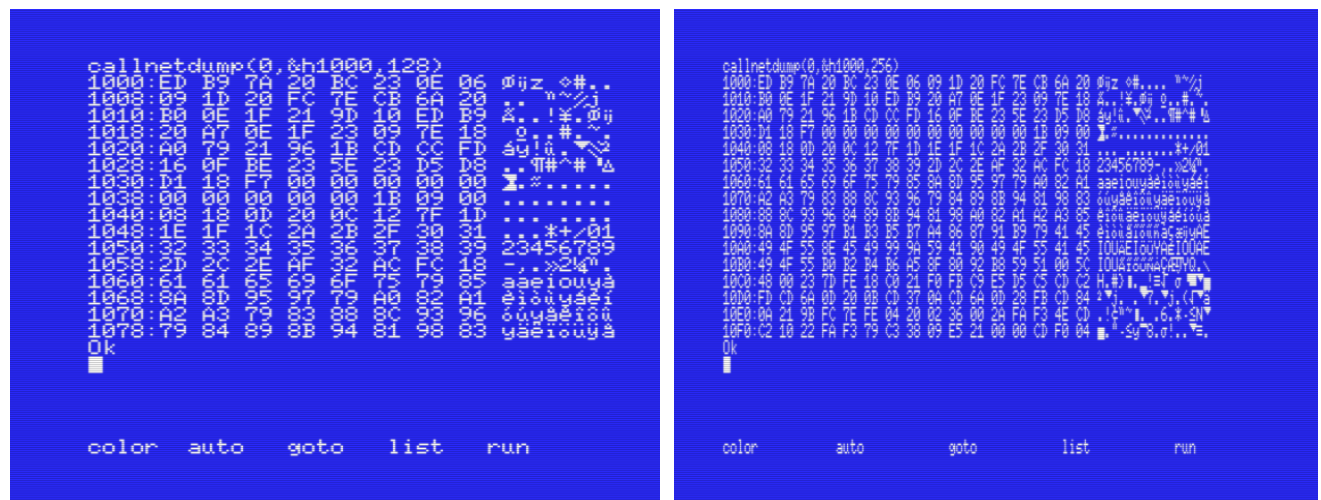


Figure 8. NETDUMP command output

## NETDIAG

Turn built-in diagnostic output on or off

### Format

CALL NETDIAG (V)

### Arguments

Argument can be numerical variable or constant

### Usage

If V is 0, diagnostics are turned off, if non-zero (for example 1), diagnostics are turned on. Normally diagnostics must be turned off, but If you turn them on, messages related to the stages of network communication will be displayed onto the screen in text modes. If communication problem occurs, analyzing the whole sequence and step when communication fails will help you understand what is wrong, and develop actions plan how to fix the issue.

## NETCODE

Return last operation status and HTTP response code

### Format

CALL NETCODE (E [,H])

### Arguments

E is variable receiving error code of the last operation, mandatory

H is variable receiving HTTP code of the last HTTP-related operation (e.g. \_NETBLOAD).

The value is only valid if there was no system error (E=0)

### GR8NET system error codes (hexadecimal)

00	No error	10	TCP open timeout	28	SD-card not ready
01	UDP open error	11	TCP state timeout	29	Invalid SD-card FS
02	String too long	12	TCP communication error	2A	Invalid URI structure
03	Domain name error	13	Connection close error	2B	SD-card I/O error
04	Transmit error	20	Memory overflow	2C	URI resource not found
05	Command timeout	21	Size mismatch	2D	Invalid format (e.g. WAV file)
06	UDP packet timeout	22	Invalid HTTP header	2E	Operation not supported
07	IPRAW open error	23	Unable to redirect	2F	Too many redirects
				30	Serial flash error

Operations setting the flags (E/H) include: NETBLOAD/E/H, NETDHCP/E, NETSETHOST/E, NETBROWSE/E/H, and many others accessing network or SD-card.

## NETSYSINFO

Get system information and system performance data

### Format

CALL NETSYSINFO (MV, CL, TP, VP, VM, MA, MR)

### Arguments

If arguments are omitted, system information is printed onto the screen

Any individual argument can be omitted

### Usage

This command is used to get system information, informing you about PC you are using, its performance, video system configuration and RAM configuration.

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft
Ok
callnetsysinfo
System: MSX2
MSXBUS: 3579560 Hz
T-perf: 60671*(51+8) per second
Wait: M1=279, Vrw=0, Vmw=0 (ns)
Video: V9938 @ 60 Hz (59.952), 128KB
Mapper: 8 (8) 16kB pages
RTCacc: +0.0000070 s (faster)
Ok
```

Wait state measurement (for diags and video player)

RTC accuracy measurement

Actual precise frequency measured by GR8NET

Here's the list of information available, with explanation

Arg	Output	Explanation
MV	System: MSX2	This is machine version. MV may be 0 (MSX1), 1 (MSX2), 2 (MSX2+) and 3 (MSX Turbo-R). If machine is Turbo-R, then MV bits 5 and 6 are meaningful: bit 5 set means Z80 mode, reset means R800 mode; bit 6 set means ROM mode, reset means DRAM mode
CL	MSXBUS: 3579560	MSX slot clock signal frequency, the same returned by _NETGETCLK command, in Hertz. It is NOT a clock frequency of the CPU, which may differ from the MSX BUS speed
TP	T-perf: 60670*(51+8)	This is T-cycle performance, and TP gets number of loops machine performed within one second with instructions of 51 cycle total duration (plus 8 for additional M1 wait state according to MSX specification). Thus for this output machine performed 1/60671/(51+8) T-cycles, and it is equivalent to 279,36 ns, exact value for calculated bus clock speed of 1/3579560 (see argument CL)
VP	Video: V9938	Video processor type: 0=TMS, 1=V9930, 2=V9958
VM	@ 60 Hz, 128KB	Combined 16-bit value: low byte (VM AND 255) is <i>current</i> vertical refresh rate (0=60Hz, 1=50Hz, 255=error identifying), high byte (VM\256) is size of VRAM in 4KB blocks (0=4KB, 1=8KB, 2=16KB, 4=32KB, 8=64KB, 16=128KB, 255=error identifying)

Arg	Output	Explanation
MA	8	Number of default mapper 16KB pages found by searching for RAM. Default mapper (slot) is the one which is selected for system data in bank 3. Note that for Turbo machines this value can be less than mapper size, and even not power of 2 (e.g. 12 in DRAM mode because Turbo-R protects highest 4 ROM-shadowed RAM pages). Maximal value is 256 (4096KB)
MR	8	Memory mapper size: number of mapper 16KB pages system tried identifying by accessing memory mapper registers. Maximal value is 256 (4096K), -1 if it can not identify mapper size. May be bigger than actual RAM – e.g. for Turbo-R A1ST with 256KB RAM installed mapper will show 32 pages (512KB mapper size), showing only 12 for argument MA

### 3.4. Overall subsystem status

If you need quick overview of the subsystem properties, use `_NETSTAT` command, which will display all effective IP addresses and status information.

#### NETSTAT

Display status information about the adapter

##### Format

CALL NETSTAT

##### Usage

DHCP host IP address will be displayed only if subsystem was initialized in DHCP mode.

Mode is a bitmap value:

7	6	5	4	3	2	1	0
Interface error	Diagnostic mode	N/A	N/A (reserved)	GR8cloud enabled	Disk-ROM enabled	DHCP (1) or Fixed IP (0)	Warm boot

Character '+' will be displayed after adapter mode flags if GR8NET's SCC is in SCC+ mode. Number displayed in the brackets is adapter number reporting the status.

##### Example

CALL NETSTAT

```

IP:      192.168.1.44
Mask:    255.255.255.0
GW:      192.168.1.1
DNS:     192.168.1.1
DHCP:    192.168.1.1
MAC:     10:16:00:04:05:06
Mode:    41+ (#3)
Ok

```

### 3.5. Memory manager

GR8NET is having relatively large – 1MB – onboard RAM, and there's a simple mechanism for its management. There're several variables related to RAM allocation, all three indicate logical page numbers.

- RAMMAX: maximal number of logical pages available for access within current GR8NET mapper;
- DSKLPG: RAM disk image starting page;
- RAMTOP: number of logical pages available for user data;
- UPRAMS: user protected RAM start logical page.









In any case, the following formula should be true:

$$\text{UPRAMS} \leq \text{RAMTOP} \leq \text{DSKLPG} \leq \text{RAMMAX}.$$








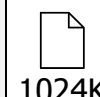













Logical page #	Usage
RAMMAX	Maximal page # available in current mapper configuration
...	RAM area reserved for the RAM disk image, maximal size of image for current mapper can be obtained by DSKCFG statement
DSKLPG	
...	Y8950 sample RAM space reserved by GR8NET firmware. If nothing was reserved, equals to DSKLPG
SAMLPG	
...	RAM area reserved by the GR8NET firmware. If nothing was reserved, RAMTOP equals to SAMLPG
RAMTOP	
...	RAM area reserved by user. Manageable by SETMMV statement
UPRAMS	
...	RAM available for user operations and NETBLOAD/NETBROWSE
0	

Only UPRAMS variable is user-manageable. Please refer to Using GR8NET disk subsystem chapter for detailed explanation of DSKLPG variable (RAM area allocated for RAM disk).














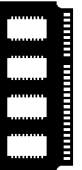


The following pages show the GR8NET RAM allocation in various mapper modes. Icons are having the following meaning:

	MSX-Audio sample RAM		Space for user data (NETBLOAD, NETBROWSE, NETPLAYWAV etc.)
	RAM disk space of 720K image size		Undefined space (theoretically can be used for user or game mapper data)
	RAM disk space of 360K image size		Game mapper of respective type
	RAM disk space of 256K available image size at the beginning (actually 360K disk image with last 104K marked as bad blocks)		
	Mapped RAM		



GR8NET mapper 0				GR8NET mapper 1 (plain ROM)		GR8NET mapper 2 (K4, Konami w/o SCC)	
RAMMAX=80h DSKLPG=26h RAMTOP=05h UPRAMS=05h	RAMMAX=80h DSKLPG=26h RAMTOP=13h UPRAMS=13h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h	No GR8NET variables are available		No GR8NET variables are available	
<div>RAM disk</div> <div>Sample RAM</div> <div>+</div> <div>+</div> <div>7F</div> <div>  <div>720K</div> </div> <div>26</div> <div>25</div> <div>  <div>256K</div> </div> <div>5</div> <div>4</div> <div>  <div>48K</div> </div> <div>0</div>	<div>RAM disk</div> <div>Sample RAM</div> <div>+</div> <div>-</div> <div>7F</div> <div>  <div>720K</div> </div> <div>26</div> <div>25</div> <div>  <div>304K</div> </div> <div>0</div>	<div>RAM disk</div> <div>Sample RAM</div> <div>-</div> <div>+</div> <div>7F</div> <div>  <div>256K</div> </div> <div>60</div> <div>5F</div> <div>  <div>768K</div> </div> <div>0</div>	<div>RAM disk</div> <div>Sample RAM</div> <div>-</div> <div>-</div> <div>7F</div> <div>  <div>1024K</div> </div> <div>0</div>	<div>Sample RAM</div> <div>+</div> <div>7F</div> <div>  <div>256K*</div> </div> <div>60</div> <div>5F</div> <div>  <div>992K</div> </div> <div>  <div>736K</div> </div> <div>4</div> <div>3</div> <div>  <div>32K</div> </div> <div>0</div>	<div>Sample RAM</div> <div>-</div> <div>7F</div> <div>  <div>992K</div> </div> <div>  <div>736K</div> </div> <div>4</div> <div>3</div> <div>  <div>32K</div> </div> <div>0</div>	<div>Sample RAM</div> <div>+</div> <div>7F</div> <div>  <div>256K*</div> </div> <div>60</div> <div>5F</div> <div>  <div>512K</div> </div> <div>8</div> <div>7</div> <div>  <div>256K</div> </div> <div>0</div>	<div>Sample RAM</div> <div>-</div> <div>7F</div> <div>  <div>768K</div> </div> <div>  <div>512K</div> </div> <div>8</div> <div>7</div> <div>  <div>256K</div> </div> <div>0</div>

\* MSX-Audio sample RAM will be allocated *before switching to respective mapper mode* if GR8NET is having MSX-Audio enabled.

GR8NET mapper 3 (K5, Konami with SCC)		GR8NET mappers 4/5 (ASCII-8/16)		GR8NET mapper 6 (Mirrored ROM)		GR8NET mapper 7 (Mapped RAM)	
No GR8NET variables are available		No GR8NET variables are available		No GR8NET variables are available		No GR8NET variables are available	
Sample RAM +	Sample RAM -	Sample RAM +	Sample RAM -	Sample RAM +	Sample RAM -	Mapped RAM - Sample RAM +	Mapped RAM + Sample RAM ?
7F  256K <sup>1</sup>	7F  512K	7F  256K <sup>1,2</sup>	7F	7F  256K <sup>1</sup>	7F	7F  256K <sup>1,4</sup>	7F
60 5F  256K		60 5F		60 5F  992K		60 5F  768K	
40 3F  512K	40 3F  512K	 1024K	 1024K	 736K			 1024K <sup>3</sup>
0	0	0	0	4 3  64K	4 3  64K	0	0

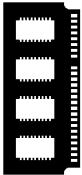
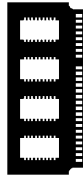


















<sup>1</sup> MSX-Audio sample RAM will be allocated *before switching to respective mapper mode* if GR8NET is having MSX-Audio enabled.

<sup>2</sup> Conflict between mapper contents and MSX-Audio sample RAM. Writing to MSX-Audio sample RAM will corrupt respective space of the game mapper.

<sup>3</sup> MSX-Audio sample RAM will always be de-configured in Mapped RAM mapper mode.

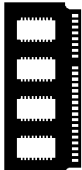

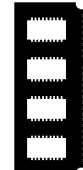




















<sup>4</sup> While mapped RAM is disabled and will not appear in the respective slot/subslot of the machine, MSX-Audio sample RAM will still be effective.

## GR8NET mapper 8

RAMMAX=40h DSKLPG=13h RAMTOP=09h UPRAMS=09h	RAMMAX=40h DSKLPG=13h RAMTOP=13h UPRAMS=13h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=53h RAMTOP=33h UPRAMS=33h	RAMMAX=80h DSKLPG=53h RAMTOP=53h UPRAMS=53h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
Mapped RAM + RAM disk + Sample RAM +	Mapped RAM + RAM disk + Sample RAM -	Mapped RAM + RAM disk - Sample RAM +	Mapped RAM + RAM disk - Sample RAM -	Mapped RAM - RAM disk + Sample RAM +	Mapped RAM - RAM disk + Sample RAM -	Mapped RAM - RAM disk - Sample RAM +	Mapped RAM - RAM disk - Sample RAM -
7F  512K	7F  512K	7F  512K	7F  512K	7F  360K	7F  360K	7F  256K	7F
40 3F	40 3F	40 3F	40 3F	53 52	53 52	60 5F	
 360K	 360K	 256K		 256K			
13 12	13 12	20 1F		33 32			
 32K*			 512K	 408K	 664K	 768K	 1024K
F E  120K	 152K	 256K					
0	0	0	0	0	0	0	0

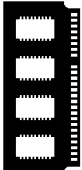
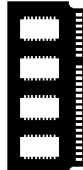
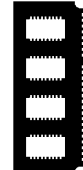




















\* Sample RAM will be downsized to 32 KB (4 x 8K pages), with 120K still available for the user.

## GR8NET mapper 9 (Plain ROM)

RAMMAX=40h DSKLPG=13h RAMTOP=04h UPRAMS=04h	RAMMAX=40h DSKLPG=13h RAMTOP=13h UPRAMS=13h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=53h RAMTOP=33h UPRAMS=33h	RAMMAX=80h DSKLPG=53h RAMTOP=53h UPRAMS=53h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
Mapped RAM + RAM disk + Sample RAM +	Mapped RAM + RAM disk + Sample RAM -	Mapped RAM + RAM disk - Sample RAM +	Mapped RAM + RAM disk - Sample RAM -	Mapped RAM - RAM disk + Sample RAM +	Mapped RAM - RAM disk + Sample RAM -	Mapped RAM - RAM disk - Sample RAM +	Mapped RAM - RAM disk - Sample RAM -
7F  512K	7F  512K	7F  512K	7F  512K	7F  360K	7F  360K	7F  256K	7F
40 3F  360K	40 3F  360K	40 3F  256K	40 3F  480K	53 52  256K	53 52  632K	60 5F  736K	 992K
13 12  120K*	13 12  120K	20 1F  224K	4 3  32K	33 32  376K	4 3  32K	4 3  32K	4 3  32K
0	0	0	0	0	0	0	0

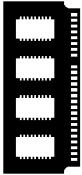





















\* Sample RAM will be downsized to 120 KB (15 x 8K pages).

# GR8NET mapper 10 (K4, Konami without SCC)

RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=53h RAMTOP=33h UPRAMS=33h	RAMMAX=80h DSKLPG=53h RAMTOP=53h UPRAMS=53h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
Mapped RAM + RAM disk + Sample RAM +	Mapped RAM + RAM disk + Sample RAM -	Mapped RAM + RAM disk - Sample RAM +	Mapped RAM + RAM disk - Sample RAM -	Mapped RAM - RAM disk + Sample RAM +	Mapped RAM - RAM disk + Sample RAM -	Mapped RAM - RAM disk - Sample RAM +	Mapped RAM - RAM disk - Sample RAM -
7F  512K	7F  512K	7F  512K	7F  512K	7F  360K	7F  360K	7F  256K	7F
40 3F  256K	40 3F  256K	40 3F  256K	40 3F  256K	53 52  256K	53 52  632K	60 5F  512K	 992K
20 1F  256K*	20 1F  256K	20 1F  256K	20 1F  256K	33 32  152K	20 1F  256K	20 1F  256K	20 1F  256K
0	0	0	0	0	0	0	0

\* MSX-Audio sample RAM will be de-configured (its size is set to 0), while FM will be available.

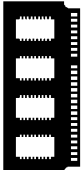
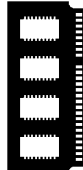









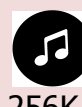







## GR8NET mapper 11 (K5, Konami with SCC)

RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=60h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=60h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
Mapped RAM + RAM disk + Sample RAM +	Mapped RAM + RAM disk + Sample RAM -	Mapped RAM + RAM disk - Sample RAM +	Mapped RAM + RAM disk - Sample RAM -	Mapped RAM - RAM disk + Sample RAM +	Mapped RAM - RAM disk + Sample RAM -	Mapped RAM - RAM disk - Sample RAM +	Mapped RAM - RAM disk - Sample RAM -
7F  512K	7F  512K	7F  512K	7F  512K	7F  256K	7F  256K	7F  256K	7F  512K
40 3F  256K *	40 3F  256K *	40 3F  256K *	40 3F	40 3F  256K	40 3F  256K	40 3F  256K	40 3F
20 1F  256K *	20 1F  256K	20 1F  256K	20 3F  512K	20 3F  512K	20 3F  512K	20 3F  512K	20 3F  512K
0	0	0	0	0	0	0	0

\* Conflict between game mapper of 512KB size with another device: you must use K5 ROM image up to the 256KB in size. In case of sample RAM, do not use sample RAM. SCC is still available for the game mapper at page game mapper's 3F at subslot 3.

\* MSX-Audio sample RAM will be de-configured (its size is set to 0).

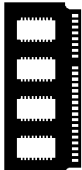

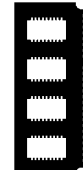






























GR8NET mapper 12/13 (ASCII-8/16)							
RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=20h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=60h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=60h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>+</div> <div>+</div> <div>+</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>+</div> <div>+</div> <div>-</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>+</div> <div>-</div> <div>+</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>+</div> <div>-</div> <div>-</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>-</div> <div>+</div> <div>+</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>-</div> <div>+</div> <div>-</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>-</div> <div>-</div> <div>+</div> </div>	<div> <div>Mapped RAM</div> <div>RAM disk</div> <div>Sample RAM</div> </div> <div> <div>-</div> <div>-</div> <div>-</div> </div>
<div>7F</div> <div>  <div>512K</div> </div>	<div>7F</div> <div>  <div>512K</div> </div>	<div>7F</div> <div>  <div>512K</div> </div>	<div>7F</div> <div>  <div>512K<sup>*</sup></div> </div>	<div>7F</div> <div> <div>  <div>256K<sup>*</sup></div> </div> <div>60</div> <div>5F</div> <div>  <div>256K<sup>*</sup></div> </div> </div>	<div>7F</div> <div> <div>  <div>256K<sup>*</sup></div> </div> <div>60</div> <div>5F</div> </div>	<div>7F</div> <div> <div>  <div>256K<sup>*</sup></div> </div> <div>60</div> <div>5F</div> </div>	<div>7F</div> <div> <div>  <div>1MB</div> </div> <div>768K</div> <div>768K</div> </div>
<div>40</div> <div>3F</div> <div>  <div>256K<sup>*</sup></div> </div>	<div>40</div> <div>3F</div> <div>  <div>256K<sup>*</sup></div> </div>	<div>40</div> <div>3F</div> <div>  <div>256K<sup>*</sup></div> </div>	<div>40</div> <div>3F</div> <div>  <div>512K</div> </div>	<div>40</div> <div>3F</div> <div>  <div>512K</div> </div>	<div>40</div> <div>3F</div> <div>  <div>768K</div> </div>	<div>40</div> <div>3F</div> <div>  <div>768K</div> </div>	
<div>20</div> <div>1F</div> <div>  <div>256K<sup>*</sup></div> </div>	<div>20</div> <div>1F</div> <div>  <div>256K</div> </div>	<div>20</div> <div>1F</div> <div>  <div>256K</div> </div>	<div>0</div>	<div>0</div>	<div>0</div>	<div>0</div>	<div>0</div>

<sup>\*</sup> Conflict between game mapper of 1MB size with another device: you must use ASCII-8/16 ROM image up to the size identified in the respective diagram (256K, 512K or 768K).

<sup>\*</sup> MSX-Audio sample RAM will be de-configured (its size is set to 0), while FM will be available.

## GR8NET mapper 14 (Mirrored ROM)

RAMMAX=40h DSKLPG=13h RAMTOP=08h UPRAMS=08h	RAMMAX=40h DSKLPG=13h RAMTOP=13h UPRAMS=13h	RAMMAX=40h DSKLPG=40h RAMTOP=20h UPRAMS=20h	RAMMAX=40h DSKLPG=40h RAMTOP=40h UPRAMS=40h	RAMMAX=80h DSKLPG=53h RAMTOP=33h UPRAMS=33h	RAMMAX=80h DSKLPG=53h RAMTOP=53h UPRAMS=53h	RAMMAX=80h DSKLPG=80h RAMTOP=60h UPRAMS=60h	RAMMAX=80h DSKLPG=80h RAMTOP=80h UPRAMS=80h
Mapped RAM + RAM disk + Sample RAM +	Mapped RAM + RAM disk + Sample RAM -	Mapped RAM + RAM disk - Sample RAM +	Mapped RAM + RAM disk - Sample RAM -	Mapped RAM - RAM disk + Sample RAM +	Mapped RAM - RAM disk + Sample RAM -	Mapped RAM - RAM disk - Sample RAM +	Mapped RAM - RAM disk - Sample RAM -
7F  512K	7F  512K	7F  512K	7F  512K	7F  360K	7F  360K	7F  256K	7F
40 3F  360K	40 3F  360K	40 3F  256K	40 3F  448K	53 52  256K	53 52  600K	60 5F  704K	 960K
13 12  88K*	13 12  88K	20 1F  192K	8 7  64K	33 32  344K	8 7  64K	8 7  64K	8 7  64K
8 7  64K	8 7  64K	8 7  64K	8 7  64K	8 7  64K	8 7  64K	8 7  64K	8 7  64K
0	0	0	0	0	0	0	0

\* Sample RAM will be downsized to 88 KB (11 x 8K pages).

---

Notes to the memory allocation diagrams above:

1. MSX-Audio sample RAM is being allocated with number of 8K pages set by `_NETOPL` command. Maximal number of pages is 32 (256K), minimal is 0 (no sample RAM). If number of pages to allocate is less than 32, then MSX-Audio sample RAM will be allocated with respective number of pages and the RAMTOP will adjust accordingly towards the RAMMAX;
2. When 256K of space is allocated for the RAM disk, all reads and writes above this space will return read or write error. You still can use 360K and 720K disk images error-free given that they have FAT entries of clusters above first 256K of disk image space filled with bad block identifier (0FF7h);
3. Disk size of 360K is a standard for MSX, thus there should be no problems applications using disk image formatted this way (for example, for game saves).

User protected RAM start and area (UPRAMS) is a logical page number, indicating top of the RAM for system utilities like `_NETBLOAD` and `_NETBROWSE`. For example, in mapper mode 8, with RAM disk enabled and sample RAM disabled, if you set UPRAMS variable to 10h, `_NETBLOAD` and `_NETBROWSE`, if trying to load data more than 131,072 bytes in size (16 logical pages, 0 to 15), will terminate with error and pages 10h-12h (3 pages of 24,576 bytes, between UPRAMS and RAMTOP) will be unmodified. At the same time, application may manage this 24KB *user protected* space using memory move statements like `_NETLDBUF` and `_NETLDRAM`.

## NETGETMMV

Get memory manager values

### Format

CALL NETGETMMV (U, T, D, M, S)

### Argument

All arguments are variables. Any argument may be omitted. If all arguments are omitted, command prints information onto the screen.

### Usage

Variable U will receive logical page number of user-protected area start (UPRAMS), T will get RAM top (RAMTOP), D will get RAM disk image start page (DSKLPG), M will get maximal number of pages in current mapper configuration (RAMMAX), and S will get starting page of the Y8950 sample RAM (SAMPLPG). Size of current sample RAM can be calculated as (D-S). See also [NETOPL command](#).

## NETSETMMV

Set memory manager value

### Format

CALL NETSETMMV (U)

### Argument

Argument is variable or constant

### Usage

It is possible to set only UPRAMS – user-protected RAM start page – with memory management command. Other variables (RAM top and RAM maximum) are managed by the GR8NET firmware, and are defined during GR8NET initialization.

### 3.6. Setting up the adapter

There're two modes adapter can operate in: DHCP mode and fixed IP address mode. DHCP mode is configured by the `_NETDHCP` command, or on PC startup if this mode was selected as default. If DHCP server is missing or did not reply, adapter configures itself in fixed IP address mode. It is advised to have single DHCP server on the local subnetwork, if there're multiple DHCP server, adapter will accept configuration from the first which replied to the DHCP discovery packet.

Default mode is preserved in adapter's flash memory, and you can update it using `_NETSAVE` command.

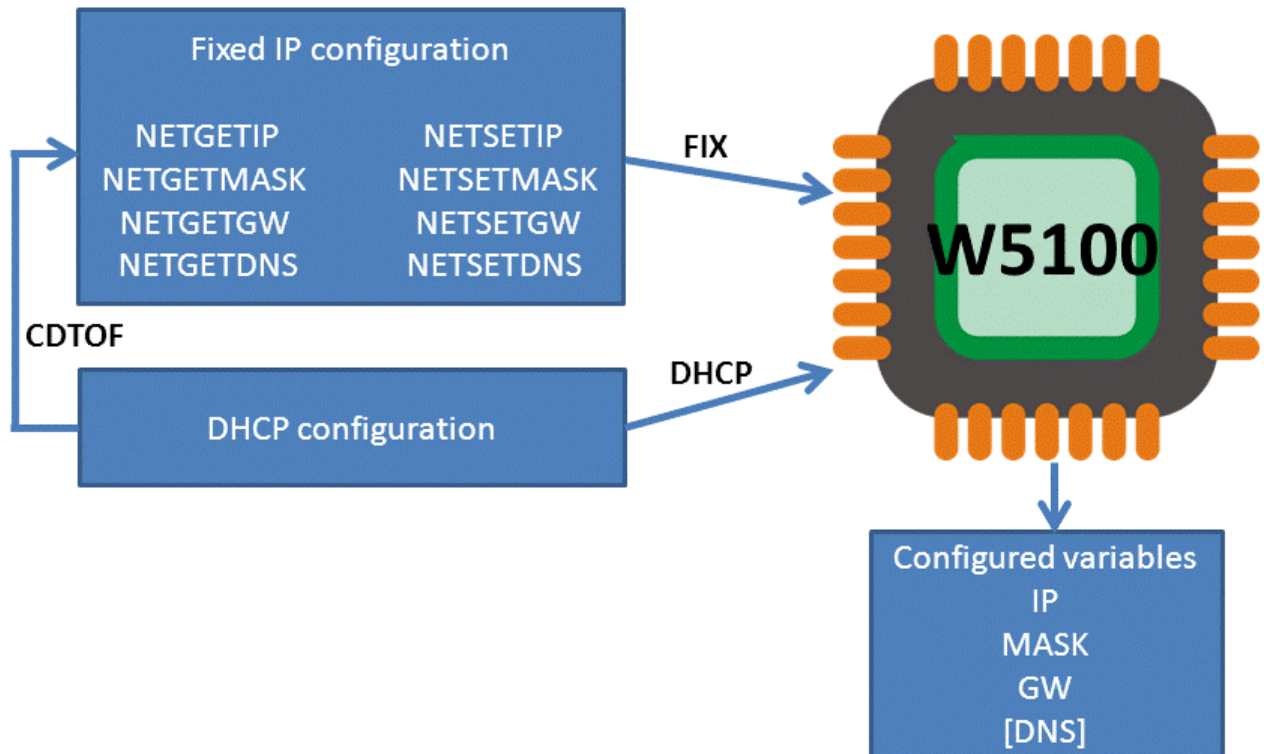


Figure 7. Configuration logic with MSX-BASIC statements

---

### 3.6.1. Getting effective configuration information

The following commands should be used to get effective configuration of the internetworking system. IP address, mask and gateway information come directly from the networking chip.

#### NETIP

Get current configured adapter's IP address

##### Format

CALL NETIP (A, B, C, D)

##### Arguments

Four arguments are variables; any can be omitted, but at least one present

##### Usage

Variables receive octets of the currently *configured* IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays IP address onto the screen in dot-decimal notation.

##### Example

```
CALL NETIP (A, B, C, D)
PRINT A;B;C;D
192 168 1 44
Ok
```

```
CALL NETIP (A, , , D)
PRINT A;D
192 44
Ok
```

```
CALL NETIP
192.168.1.44
Ok
```

#### NETMASK

Get current configured adapter's subnetwork mask

##### Format

CALL NETMASK (A, B, C, D)

##### Arguments

Four arguments are variables; any can be omitted, but at least one present

##### Usage

Variables receive octets of the currently *configured* subnet mask, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays mask onto the screen in dot-decimal notation.

##### Example

```
CALL NETMASK (A, B, C, D)
PRINT A;B;C;D
255 255 255 0
Ok
```

```
CALL NETMASK (A, , , D)
PRINT A;D
255 0
Ok
```

```
CALL NETMASK
255.255.255.0
Ok
```

## NETGW

Get current configured gateway

### Format

CALL NETGW (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Variables receive octets of the currently *configured* gateway IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays gateway's IP address onto the screen in dot-decimal notation.

### Example

CALL NETGW (A, B, C, D)	CALL NETGW (A, , , D)	CALL NETGW
PRINT A;B;C;D	PRINT A;D	192.168.1.1
192 168 1 1	192 1	Ok
Ok	Ok	

## NETDNS

Get current configured domain name server (DNS) IP address

### Format

CALL NETDNS (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Variables receive octets of the currently *configured* DNS IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays DNS IP address onto the screen in dot-decimal notation.

### Example

CALL NETDNS (A, B, C, D)	CALL NETDNS (A, , , D)	CALL NETDNS
PRINT A;B;C;D	PRINT A;D	192.168.1.1
192 168 1 1	192 1	Ok
Ok	Ok	



---

### 3.6.2. DHCP mode

DHCP initialization assumes adapter getting its IP address, subnet mask, gateway IP address and DNS server IP address from DHCP server configured and running on the subnetwork. This mode is the easiest for out-of-the-box running of the internetworking system.

All SET commands related to the fixed IP address configuration mode do not have effect unless you reconfigure adapter in fixed IP address mode.

#### **NETDHCP**

Perform DHCP discovery and dynamic configuration

##### Format

CALL NETDHCP

##### Usage

Adapter invalidates current configuration and tries to find local subnet DHCP server. If communication with DHCP server is successful, server supplied configuration takes effect, if communication is unsuccessful, then adapter loads fixed IP address configuration, and exits with "Device I/O error" error condition. This command resets internetworking system. Ensure closing all open files before issuing it.

---

### 3.6.3. Fixed IP address configuration mode

In fixed IP address configuration mode you can set up all the IP parameters manually. If you change any related value, it will take effect only after you reinitialize fixed IP address mode.

#### NETFIX

Configure fixed IP address information into network system

##### Format

CALL NETFIX

##### Usage

Adapter loads fixed IP configuration variables into its operating subsystem and networking chip. No checking on the information loaded is being performed, if fixed configuration does not work, please verify all the variables present in fixed configuration, and variables which are in effect (*configured* values). This command resets internetworking systems. Ensure closing all open files before issuing it.

#### NETGETIP

Get fixed IP address value

##### Format

CALL NETGETIP (A, B, C, D)

##### Arguments

Four arguments are variables; any can be omitted, but at least one present

##### Usage

Variables receive octets of the *fixed IP address configuration mode* IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays fixed IP address configuration mode IP address onto the screen in dot-decimal notation.

##### Example

CALL NETGETIP (A, B, C, D)	CALL NETGETIP (A, , , D)	CALL NETGETIP
PRINT A;B;C;D	PRINT A;D	5.20.1.11
5 20 1 11	5 11	Ok
Ok	Ok	

#### NETGETMASK

Get fixed IP address mask

##### Format

CALL NETGETMASK (A, B, C, D)

##### Arguments

Four arguments are variables; any can be omitted, but at least one present

##### Usage

Variables receive octets of the *fixed IP address configuration mode* subnetwork mask, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays fixed IP address configuration mode subnetwork mask onto the screen in dot-decimal notation.

---

### Example

CALL NETGETMASK (A, B, C, D)	CALL NETGETMASK (A, , , D)	CALL NETGETMASK
PRINT A;B;C;D	PRINT A;D	255.255.255.240
255 255 255 240	255 240	Ok
Ok	Ok	

## **NETGETGW**

Get fixed IP address mode gateway IP address

### Format

CALL NETGETGW (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Variables receive octets of the *fixed IP address configuration mode* gateway IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays fixed IP address configuration mode gateway's IP address onto the screen in dot-decimal notation.

### Example

CALL NETGETGW (A, B, C, D)	CALL NETGETGW (A, , , D)	CALL NETGETGW
PRINT A;B;C;D	PRINT A;D	5.20.1.1
5 20 1 1	5 1	Ok
Ok	Ok	

## **NETGETDNS**

Get fixed IP address mode DNS IP address

### Format

CALL NETGETDNS (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Variables receive octets of the *fixed IP address configuration mode* domain name server IP address, variable A being highest octet and variable D lowest octet. Variables supplied will keep their original type. If arguments are omitted, command displays fixed IP address configuration mode DNS IP address onto the screen in dot-decimal notation.

### Example

CALL NETGETDNS (A, B, C, D)	CALL NETGETDNS (A, , , D)	CALL NETGETDNS
PRINT A;B;C;D	PRINT A;D	5.20.1.2
5 20 1 2	5 2	Ok
Ok	Ok	

## NETSETIP

Sets fixed configuration IP address value

### Format

CALL NETSETIP (A, B, C, D)

### Arguments

Four arguments are numerical variables or constants; any can be omitted, but at least one present

### Usage

Sets card's IP address within fixed IP address configuration block. To make this IP address in effect, re-initialize with NETFIX command. If position in argument list is omitted, related octet of IP address is not changed.

### Example

CALL NETSETIP (192, 168, 1, 10)	CALL NETSETIP (, , , 15)	CALL NETSETIP
Ok	Ok	Illegal function call
		Ok

## NETSETMASK

Set fixed configuration IP address mask

### Format

CALL NETSETMASK (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Sets card's subnetwork mask within fixed IP address configuration block. To make this mask in effect, re-initialize with NETFIX command. If position in argument list is omitted, related octet of subnetwork mask is not changed.

### Example

A=128	CALL NETSETMASK (, , , 128)	CALL NETSETMASK
CALL NETSETMASK (255, 255, 255, A)	Ok	Illegal function call
Ok		Ok

## NETSETGW

Set fixed configuration gateway IP address

### Format

CALL NETSETGW (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Sets card's gateway IP address within fixed IP address configuration block. To make this address in effect, re-initialize with NETFIX command. If position in argument list is omitted, related octet of gateway IP address is not changed.

### Example

CALL NETSETGW (192, 168, 1, 1)	CALL NETSETGW (, , , 1)	CALL NETSETGW
Ok	Ok	Illegal function call
		Ok

## NETSETDNS

Set fixed configuration DNS IP address

### Format

CALL NETSETDNS (A, B, C, D)

### Arguments

Four arguments are variables; any can be omitted, but at least one present

### Usage

Sets card's DNS IP address within fixed IP address configuration block. To make this address in effect, re-initialize with NETFIX command. If position in argument list is omitted, related octet of DNS IP address is not changed.

### Example

CALL NETSETDNS (5, 20, 1, 2)	CALL NETSETDNS (, , , 2)	CALL NETSETDNS
Ok	Ok	Illegal function call
		Ok

## 3.6.4. Managing configurations

When GR8NET adapter initializes, it uses information stored in its ROM configuration page (logical page BF), mirroring these initial values into the configuration RAM (logical page FF). In addition to the actions described in 1.4.2 and 1.4.3, you can perform actions copying DHCP configuration into fixed IP address configuration and updating the configuration information in ROM so that next time adapter is initialized in fixed IP address mode the new saved configuration takes effect.

## NETCDTOF

Copy DHCP configuration into fixed IP address configuration

### Format

CALL NETCDTOF

### Usage

If you adapter configured using DHCP properly and you want to use clone of DHCP configuration, then you copy it into fixed IP address configuration variables and tune them. Variables copied are: IP address, mask, gateway and DNS server IP addresses.

### Example

```
CALL NETCDTOF
CALL NETSETIP(192,168,1,145)
CALL NETFIX
```

## NETSAVE

Save current configuration into ROM configuration page

### Format

CALL NETSAVE

### Usage

This command updates the following ROM variables: fixed IP address configuration, adapter operating mode ("Mode", "netmode", see NETSTAT) and default URI structure. It also writes MAC address back, but it is not advised to change it (through debugger or memory editor) because it may then cause device malfunction or conflict on the network.



### 3.6.5. Managing system time

When initializing, GR8NET tries to reach NTP (network time protocol) server to synchronize system time. If RTC chip is present, firmware updates RTC chip stored time if specific flag is set in GR8NET configuration. If RTC chip is not present in the system, firmware just displays current date and time onto the screen.

The following scheme is used to get NTP server's IP address:

- Fixed IP configuration: if default NTP server IP address set by NETSETNTP is starting with 0 (0.x.x.x), card tries obtaining IP address of host name set by NETSETTSHN using DNS address set by NETSETDNS; if default NTP server IP address starts with non-zero, then this default IP address is used for NTP server query;
- DHCP mode: if DHCP server responds to DHCP request with NTP server's IP address (DHCP option 42), this returned IP address will be used; if DHCP server does not give option with NTP server's IP address, then card tries obtaining IP address of host name set by NETSETTSHN using address of DNS returned by DHCP server.

#### NETNTP

Get effective configuration of NTP server

##### Format

CALL NETNTP (A, B, C, D, TZF)

##### Arguments

A-D are variables receiving effective NTP server IP address octets

TZF is current time zone and update flag

##### Usage

With this command application gets effective NTP server address, the address used for time configuration (even if unsuccessful). TZF has the following format:

7	6	5	4	3	2	1	0
TUF	Time zone definition						

- TUF is time update flag. If set, forces GR8NET to synchronize system's RTC (if present) with NTP server if NTP server query was successful
- Time zone definition is signed 7-bit number, identifying time zone in 15-minute quanta.
  - Value of 0 defines UTC time zone, thus zero time shift from time reported by NTP server;
  - Value of -64 (040h) means time shift of -16 hours (15 minutes \* 64);
  - Value of +63 (03fh) means time shift of +15:45.

#### NETGETNTP

Get NTP server properties within fixed IP address configuration

##### Format

CALL NETGETNTP (A, B, C, D)

##### Arguments

A-D are variables getting NTP server IP address octets of fixed IP address configuration

##### Usage

This IP address will be used in case DHCP server did not respond with NTP server address.

## **NETSETNTP**

Set NTP server properties within fixed IP address configuration, **and time setting flags**

### Format

CALL NETSETNTP (A, B, C, D, TZF)

### Arguments

A-D are variables or constants defining NTP server IP address within fixed IP address configuration

TZF is update flag and time zone (refer to \_NETNTP command)

## **NETTSYNC**

Display and synchronize system time

### Format

CALL NETTSYNC

### Usage

This command displays and synchronizes current system time (if system has RTC chip installed) using NTP server. It updates NTP server IP address displayed by NETSTAT command. Note if GR8NET obtains NTP server IP address using DHCP, GR8NET may use different IP addresses for synchronization.

## **NETSETTSHN**

Set time server host name

### Format

CALL NETSETTSHN (A\$)

### Arguments

A\$ is string variable or constant of length of 63 or less characters

### Usage

Sets time server host name, which will be used in DHCP mode to resolve to NTP server IP address, and in fixed mode configuration if IP address set with NETSETNTP starts with 0 (0.x.x.x).

## **NETGETTSHN**

Get time server host name

### Format

CALL NETGETTSHN [(A\$)]

### Arguments

A\$ is receives host name; if omitted, host name will be printed onto the screen

### Usage

This command is used to get time server host name (see NETSETTSHN).

---

### 3.6.6. Full configuration with NETCFG command

To assist you in configuration of the adapter and minimize number of commands you type and options you need to set, there's special interactive command `_NETCFG` which will guide you through almost all adapter settings. However you must clearly understand what you are setting; this subchapter will give you information how to run NETCFG properly from the first try.

#### NETCFG

##### Interactive GR8NET configuration

###### Format

CALL NETCFG

###### Arguments

No arguments are needed; all information is entered in interactive mode

```
GR8NET configuration utility
Sections are:
- Operating mode
- Network
- Time
- Multimedia
- Disk subsystem

At any time you can press CTRL-STOP
Empty input preserves current value

Press Enter to start,
any other key to abort█

color  auto  goto  list  run
```

Welcome screen

```
-- Operating mode --
Bitmap value,
- add 2 for using DHCP
- add 4 to enable Disk-ROM
- add 64 to turn diag mode on
(e.g. 2)

Current value:
1
New value:
█

color  auto  goto  list  run
```

First parameter to enter

#### Operating mode section

- **Operating mode:** should be a sum of values of power of 2. Bit 1 (+2) should be set if you want GR8NET to start in DHCP mode; bit 3 (+4) should be set if you want built-in DOS1 Disk-ROM to be initialized on startup; bit 6 (+64) is the same as performing `_NETDIAG(1)`, but diagnostics will be turned on just after GR8NET starts, and you will be able to see how it works;
- **Target mapper mode** (NETTGTMAP): here you put target mapper number in decimal format. Value of 16 deactivates target mapper mechanism. Do not forget to add 16 to mapper mode value to have special register set turned on in GR8NET bank 0;
- **Multimedia configuration** (similar to NETSETOPL): bit 1, if set (+2), causes OPL/Y8950/PSG being fed with internal clock rather than external from MSX-BUS; bit 2, if set (+4), turns OPLL off; bit 3, if set (+8), causes GR8NET its OPL\* output waveform amplitude to double; bit 4, if set (+16), turns Y8950 off; bit 6, if set (+64) causes built-in PSG to be enabled on (re) configuration; and bit 7, if set (+128), selects built-in PSG location to port 010h instead of port 0A0h (internal machine PSG mirror).

- 
- **Receive window threshold:** this is service setting, normally should be set to 0, but for troubleshooting purposes you can set it to 1024 to decrease operating size of RX buffer from 2048 bytes to 1024 bytes;

### Fixed IP config section

- **My IP address** (NETSETIP): IP address GR8NET will come up if it starts in, or is being reconfigured to the fixed IP address mode using NETFIX;
- **Network mask** (NETSETMASK): IP address mask GR8NET will come up if it starts in, or is being reconfigured to the fixed IP address mode using NETFIX;
- **Gateway** (NETSETGW): IP address of the gateway to use if GR8NET will come up if it starts in, or is being reconfigured to the fixed IP address mode using NETFIX;
- **DNS** (NETSETDNS): IP address of the domain name server to use if GR8NET will come up if it starts in, or is being reconfigured to the fixed IP address mode using NETFIX;

### Time

- **NTP server name** (NETSETTSHN): host name of the NTP server to be used in fixed IP or DHCP configuration mode;
- **NTP server IP address** (NETSETNTP): sets NTP server IP address to fall back to in case NTP server host name was not resolved by the DNS. Note that in DHCP mode, if DHCP server provides its own NTP server IP address, then provided IP address will be used as a fall back IP address;
- **Time zone** (NETSETNTP): sets flags for the GR8NET time configuration; bit 7 (+128) should be set if you want GR8NET to update machine's RTC on command run (including machine startup), and bits 6:0 are representing time zone in 15 minute increment as 7-bit signed integer. For example, GMT+1 (CET) will be coded as +4, GMT+9 (JST) will be coded as +36, GMT-5 (EST) will be coded as +108 ( $128-5*4$ ), GMT-10 (HAST) will be coded as +88 ( $128-10*4$ ), and GMT+5.30 (IST) will be coded as +22 ( $5*4+2$ ).

### Default URI structure

- **Remote host name** (NETSETHOST): sets default host name for GR8NET all other locations will be derived from. Size of the field is 63 characters for firmware 20171111 and later and 31 characters for older releases;
- **Remote host port** (NETSETPORT): default remote port for TCP and UDP communication, 16-bit word;
- **Source port number** (NETSETPORT): default value is 0 instructing GR8NET select dynamic port number; it is very advisable to leave this value 0, otherwise port number reuse problem may happen;
- **Remote host path** (NETSETPATH): string setting default path on the remote resource, maximal length is 239 characters;
- **Remote host file name** (NETSETNAME): string setting default file name on the remote resource, maximal length is 63 characters;
- **Remote host query string** (NETSETQSTR): string setting default query string on the remote resource, maximal length is 63 characters. Logically that it must start with '?' character (GR8NET does not check for such compliance of the string);

---

## Multimedia

- **Master DAC volume** (NETSNDVOL): main multimedia subsystem volume, officially its maximum is 128, but can be set to 255 to digitally amplify output waveform by almost 2;
- **Built-in SCC volume** (NETSNDVOL): maximum is 128, not used in MP3 media player mode;
- **Digital waveform volume** (NETSNDVOL): maximum is 128;
- **PCM volume** (NETSNDVOL): maximum is 128, in MP3 media player mode is used for MP3 output;
- **YM2413 volume** (NETSNDVOL): maximum is 128, not used in MP3 media player mode;
- **YM8950 volume** (NETSNDVOL): maximum is 128, not used in MP3 media player mode;
- **PSG volume** (NETSNDVOL): maximum is 128, not used in MP3 media player mode;
- **Mixer setting** (NETSETMIX): is a string coding mixing of respective devices into left and right channels. There should be maximum 6 character string consisting of letters B (both), L (left), R (right) and M (mute), with first letter for PCM (or MP3 channels in MP3 media player mode), second letter for SCC, third letter for input waveform, fourth letter for YM2413, fifth letter for Y8950 and last, sixth letter, for built-in PSG. Example: BBBLRL.

## Disk subsystem

- **Default RAM-disk URI** (NETDKSETIMG): sets location of the image to load on start up when Disk-ROM is enabled (and there's no valid image in RAM and F2 key is not being held), and DSKLDIMG command is performed.

## GR8cloud configuration

- **GR8cloud server** (NETSETCLOUD): sets GR8cloud server location and port number separated by semicolon `:`;
- **GR8cloud password** (NETSETCLOUD): sets password for GR8cloud volume, maximum 16 characters.

---

### 3.6.7. Exporting and importing GR8NET configuration

While NETCFG command allows you to configure your GR8NET from the scratch, there's a way for exporting existing configuration. The process is very simple, but you must watch it to ensure there're no warnings or errors.

First, you must export configuration using \_NETVAREXPRT command. It will create BASIC file in specific format with current GR8NET configuration data. Existing BASIC program will be erased, however command will ask if you want to proceed if it detects you have some BASIC program in the RAM.

Second, you save this created BASIC program to any storage media you have. You can use SAVE command, and save in ASCII (",A" option) or tokenized format. You will use this saved file after you perform operations with GR8NET which will modify or invalidate its custom configuration (e.g. you perform firmware update). Do **NOT** edit the program.

Third, when you need to restore configuration of your GR8NET, you load previously saved BASIC program, and perform \_NETVARIMPRT command. It will parse the BASIC file, and will be printing messages onto the screen. You can press STOP key to stop printout or CTRL-STOP to break the process. Watch for warnings and errors during parsing.

Fourth, and last, after successful parsing without errors you perform CALL NETSAVE command to save configuration to your GR8NET.

**Note:** configuration becomes effective immediately after you perform \_NETIMPRT command. It is not recommended to run adapter in this state, thus perform \_NETSAVE and then reset machine so that GR8NET re-initializes properly.

**Important:** Keep in mind that configuration data contains private information (your GR8cloud password). While it is salted and ciphered, please keep configuration file in safe place and avoid sharing it with unauthorized entities.



#### NETVAREXPRT

Create BASIC program containing GR8NET configuration data

##### Format

CALL NETVAREXPRT

##### Arguments

No arguments are needed

##### Usage

This command completes quietly, but it will ask for active approval to erase current existing BASIC program if it is detected. After completion you can perform LIST to see the program, and even perform RUN on it to see its purpose and usage message. Do NOT edit the program.



---

## NETVARIMPRT

Fill GR8NET system variables with data from BASIC program created by NETEXPRT

### Format

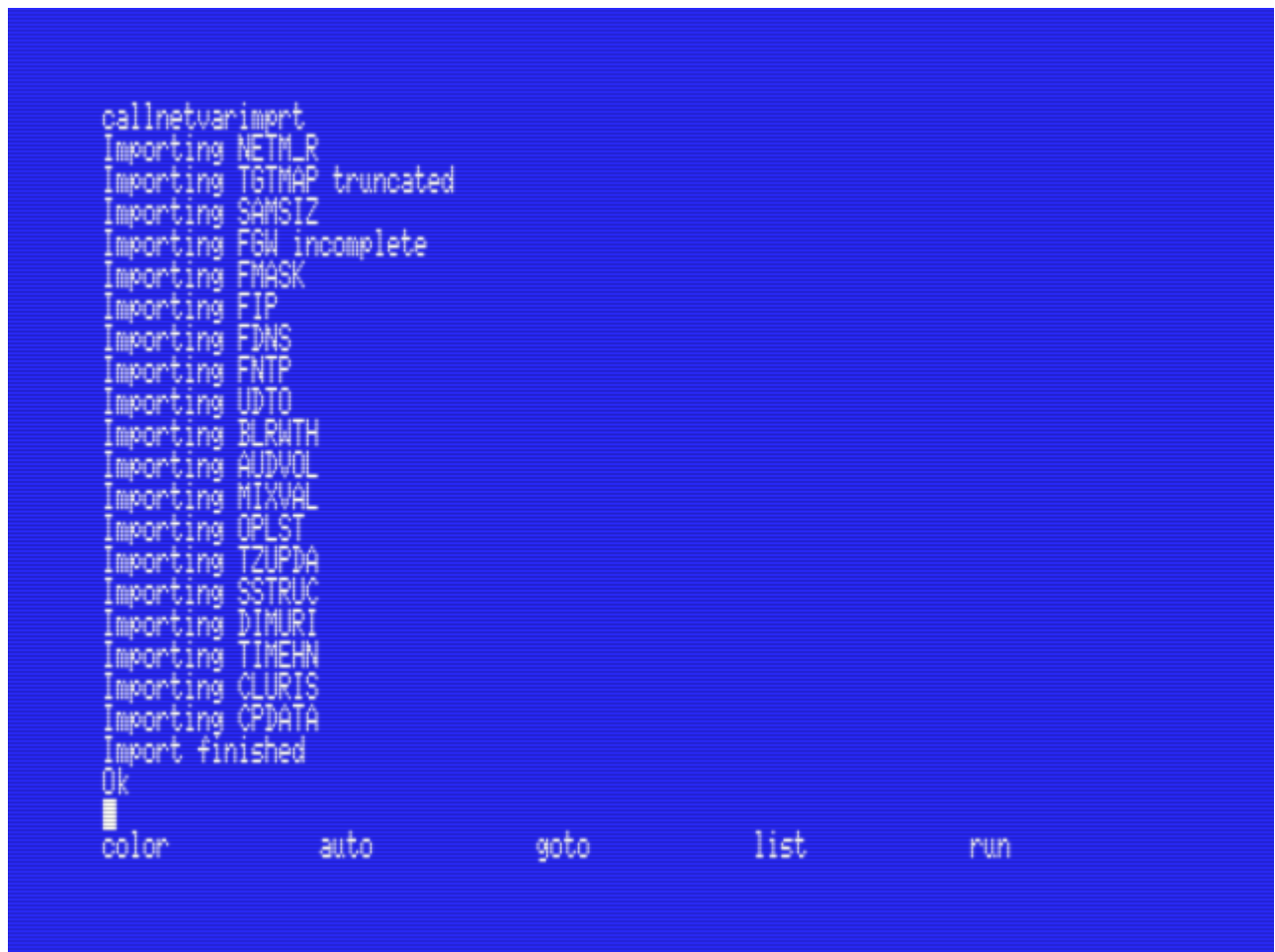
CALL NETVARIMPRT

### Arguments

No arguments are needed

### Usage

This command will print diagnostic information onto the screen. Normally it will display the list of variables, but there're could be a number of warnings or error displayed. Command will inform if it completes successfully or with error. Below is sample output of the command.



```
callnetvarimpert
Importing NETMLR
Importing TGTMAP truncated
Importing SAMSIZ
Importing FGW incomplete
Importing FMASK
Importing FIP
Importing FDNS
Importing FNTF
Importing UDT0
Importing BLRATH
Importing AUDVOL
Importing MIXVAL
Importing OPLST
Importing TZUPDA
Importing SSTRUC
Importing DIMURI
Importing TIMEHN
Importing CLURIS
Importing CPDATA
Import finished
Ok
color      auto      goto      list      run
```

Here's the list of possible messages:

- **Importing *variable\_name*:** process of parsing variable has started, if it completes successfully no messages will be displayed after this text;
- **Variable *variable\_name* not recognized:** means that current firmware does not have variable listed in the configuration file;

- 
- **Truncated:** configuration data contains *more data* than current firmware supports, and means that some data from configuration file was not put into GR8NET adapter configuration;
  - **Incomplete:** configuration data contains *less data* than current firmware supports, and means that some data in GR8NET adapter remains as default;
  - **Incorrect format:** this error message happens when unexpected format is encountered. It halts import process;
  - **No GR8NET configuration BASIC program loaded:** means that you did not load proper BASIC program containing GR8NET configuration data;
  - **Import finished:** import process has finished without errors, but may have warnings.

---

## 3.7. Using built-in HTTP-related commands

There're a number of commands and functions (including BASIC I/O), they include accessing remote resources using HTTP protocol –downloading information from web servers and uploading information to them. One of such commands is \_NETBLOAD.

### NETSETHOST

Set remote host name, and perform simple DNS query if needed

#### Format

CALL NETSETHOST (N\$ | A, B, C, D)

#### Arguments

Either string variable or string constant; or four numerical variables or constants (any can be omitted, but at least one present)

#### Usage

Sets remote host name, or converts network URI structure to SD-card URI structure.

- If string is used as an input *and string is not empty*, adapter uses DNS server to perform name resolution into IP address. String should be without protocol definition (e.g. http://) and without trailing slash. Maximal string length is 63 characters. If resolution is unsuccessful, "Illegal function call" error is given;
- If string is empty, then URI structure is converted to SD-card URI structure, and when calling NETBLOAD it will access SD-card path rather than network path. Be sure to set appropriate SD-card path and file name before proceeding to \_NETBLOAD.

In case IP address octets are entered, host name string is emptied, and in some cases web access may be unsuccessful if remote web server requires valid "Host:" HTTP header.

#### Example

CALL NETSETHOST ("www.gr8bit.ru") Ok	CALL NETSETHOST (195, 208, 1, 126) Ok <i>(NETBLOAD will not work, web server requires header "Host: www.gr8bit.ru")</i>	CALL NETSETHOST Illegal function call
CALL NETSETHOST ("") Ok		Ok
CALL NETSETHOST ("192.208.1.126") Ok		

### NETSETPATH

Set remote resource's path

#### Format

CALL NETSETPATH (N\$)

#### Arguments

String variable or constant

#### Usage

Sets resource's path. Path is absolute, without trailing slash. Maximal length of the string is 239 characters. The path can address network resource (if URI structure is network one), or resource on SD-card (if URI structure if SD-card one). Use \_NETGETHOST to identify current URI structure type.

---

### Example

A\$="/software/roms"	CALL NETSETPATH ("/software/roms")	CALL NETSETPATH
CALL NETSETPATH (A\$)	Ok	Illegal function call
Ok		Ok

## **NETSETNAME**

Set remote resource's file name

### Format

CALL NETSETNAME (N\$)

### Arguments

String variable or constant

### Usage

Sets remote resource's file name. Maximal length of the string is 63 characters. The name can address network resource (if URI structure is network one), or resource on SD-card (if URI structure is SD-card one). Use \_NETGETHOST to identify current URI structure type.

### Example

A\$="vkiller.rom"	CALL NETSETNAME ("vkiller.rom")	CALL NETSETNAME
CALL NETSETNAME (A\$)	Ok	Illegal function call
Ok		Ok

## **NETSETQSTR**

Set query string for remote resource processing

### Format

CALL NETSETQSTR (N\$)

### Arguments

String variable or constant

### Usage

Query string must start with question mark. Maximal length of the string is 63 characters. Query string has effect only for network URI structure type

### Example

A\$="?forumtopic=13"	CALL NETSETQSTR ("?forumtopic=13")	CALL NETSETQSTR
CALL NETSETQSTR (A\$)	Ok	Illegal function call
Ok		Ok



During HTTP request construction query string is just added after the file name, thus in total both strings may make up to 126 characters. If you have very long file name, you can use query string splitting file name into two parts which fit into file name and query string limitations.

## **NETGETHOST**

Get remote host name and IP address

### Format

CALL NETGETHOST (F, N\$ | A, B, C, D)

### Arguments

F is numerical variable, N\$ is string variable, A, B, C, D are numerical variables. Any argument can be omitted

---

### Usage

Gets remote host name or IP address. F is the byte flag having the following meaningful bits:

- bit 0 reset for network URI structure, set for SD-card URI structure;
- bit 1 reset if hostname/file properties are not valid, set if they are valid;
- bits 3 and 2: for SD-card type of URI structure mean SD-card partition number;
- bit 6 is set if default URI structure is version 1.

For network URI structure, N\$ will receive host name; for SD-card URI structure N\$ will be empty string.

If numerical variables are supplied, they get octets of the IP address of remote host in case URI structure is network one, otherwise displays *Illegal function call*.

If all arguments are omitted, displays information about remote host onto the screen.

### Example

CALL NETGETHOST (F,A\$)	CALL NETGETHOST (, A, B, C, D)	CALL NETGETHOST
PRINT F;A\$	PRINT A;B;C;D	<a href="http://www.gr8bit.ru">www.gr8bit.ru</a> (195. 208.1.126)
2 www.gr8bit.ru	195 208 1 126	Ok
Ok	Ok	

## **NETGETPATH**

Get remote resource's path

### Format

CALL NETGETPATH (N\$)

### Arguments

String variable

### Usage

Gets remote resource's path. If argument is omitted, prints path onto the screen

### Example

CALL NETGETPATH (A\$)	CALL NETGETPATH
PRINT A\$	/software/roms
/software/roms	Ok
Ok	

## **NETGETNAME**

Get remote resource's file name

### Format

CALL NETGETNAME (N\$)

### Arguments

String variable

### Usage

Gets remote resource's file name. If argument is omitted, prints file name onto the screen

### Example

CALL NETGETNAME (A\$)	CALL NETGETNAME
PRINT A\$	vkiller.rom
vkiller.rom	Ok
Ok	

## NETGETQSTR

Get query string set up for remote resource

### Format

CALL NETGETQSTR (N\$)

### Arguments

String variable

### Usage

If argument is omitted, prints query string path onto the screen

### Example

CALL NETGETQSTR (A\$)	CALL NETGETQSTR
PRINT A\$	?forumtopic=13
?forumtopic=13	Ok
Ok	

## NETSETPORT

Set communication port numbers in the default URI structure

### Format

CALL NETSETPORT (A, B)

### Arguments

A and B are variables of constants

If B is 0 (default setting), then dynamic port number is used (recommended)

### Usage

A is remote port number for communication (e.g. 80 for web server), B is local source port. Command does not check validity of the ports. Please ensure that for generic HTTP communication you use port number above number 49152.

**Note:** remote port number's lowest 8 bits are used to identify IP RAW protocol ID in case network file is being open with OPEN statement.

### Example

A=80	CALL NETSETPORT (80)	CALL NETSETPORT
CALL NETSETPORT (A,50000)	Ok	Illegal function call
Ok		Ok

## NETGETPORT

Get communication port numbers

### Format

CALL NETGETPORT (A, B)

### Arguments

A and B are variables

### Usage

Variable A gets remote port number for communication (e.g. 80 for web server), B gets local source port number. If arguments are omitted, prints port numbers onto the screen.

### Example

CALL NETGETPORT (,B)	CALL NETGETPORT	CALL NETGETPORT
PRINT B	Dest: 80	Dest: 80
-15536	Src: 50000	Src: 0 (dynamic)
Ok	Ok	Ok



---

## 3.8. Using NETBLOAD command

\_NETBLOAD command loads binary data from the remote server using HTTP protocol, or from installed SD-card. You have to supply the following information for \_NETBLOAD to succeed:

- Loading from the network:
  - NETSETHOST » valid host name or IP address; if hosting server has multiple web server names configured for it, you will have to set up host name and ensure successful IP address resolution;
  - NETSETPATH » valid path to the remote resource;
  - NETSETNAME » valid remote file name; if name is empty network operation will try to return folder contents as returned by web server;
  - NETSETQSTR » valid query string;
  - NETSETPORT » valid port numbers: remote and local;
  - Properly configured GR8NET adapter with its IP address, subnet mask, gateway, DNS server IP address.
- Loading from SD-card:
  - Have SD-card installed with FAT32 or FAT16 file system on it;
  - NETSETHOST » to set up SD-card URI structure type;
  - NETSETPATH » set path on the SD-card;
  - NETSETNAME » set file name; if file name is empty, directory image will be loaded.

File size to download should not exceed size of the buffer available under user protected RAM (UPRAMS), otherwise it will be truncated and \_NETBLOAD will finish with error. If download fails, use \_NETDIAG command to turn on diagnostic output and track \_NETBLOAD command execution, and \_NETCODE command to find out error code and HTTP return code.

For information about URI structure and URI string, refer to [URI structure](#) chapter.

### NETBLOAD

Load binary file from the remote web server using HTTP or from SD-card

#### Format

CALL NETBLOAD (P\$, A, PG, ADDR)

#### Arguments

P\$ is URI string to the remote resource (use SDC:// device for SD-card's first partition)

A is byte variable or constant

PG is starting logical page number

ADDR is starting page address in GR8NET bank 1 area (6000-7FFF)

#### Usage

By default NETBLOAD takes default URI structure managed by the \_NETSETHOST, \_NETSETPATH, \_NETSETNAME and \_NETSETQSTR commands, however if P\$ is supplied, it overrides specific parts of the URI, or even changes its type to SD-card structure.

Value of A makes sense for binary executables only (files which are created by BSAVE having 7-byte header in them). If A is 2, executable data loaded will be moved to location indicated by the header and will be executed; if A is 1, then executable data will be loaded into respective location but will not be executed. If A is 0 then data will not be loaded into respective location and will not be executed. If A is omitted, value 0 is assumed.

PG:ADDR value set identify offset to start loading data to. PG is logical page number in range 0...07Fh, and ADDR is CPU RAM space within GR8NET bank 1 (6000h-7FFFh) where page number PG will be presented during data load start. If file will not fit into the RAM (it will overflow to logical page above indicated by UPRAMS) then *Device I/O error* will be generated.

#### Examples

CALLNETBLOAD	will load data, and if it will detect that it is binary no action will be taken
CALLNETBLOAD(,2)	will load data and if it is binary, move it and execute
CALLNETBLOAD ("http://www.gr8bit.ru")	will use host <a href="http://www.gr8bit.ru">www.gr8bit.ru</a> , and empty path and file name (will read root of the web server)
CALLNETBLOAD ("/software/binary/file.bin")	will load indicated file from indicated path from the web server defined by NETSETHOST command
CALLNETBLOAD("file.bin",2,&h7000)	will load indicated file from path set by NETSETPATH command and web server set by NETSETHOST command, starting logical page 2 and GR8NET bank 1 address 7000h (thus data will start at $2*2000h+1000h=9000h$ in the GR8NET physical RAM).
CALLNETBLOAD("sdc:///kvalley.rom")	Load ROM file from the root (/)of the first partition of the SD-card
CALLNETBLOAD("sdf:///dir/file.dat")	Load file with name <i>file.dat</i> from <i>/dir/</i> directory located in the 4 <sup>th</sup> partition of the SD-card

### **NETVARBSIZE**

Get size of bloaded data in bytes

#### Format

CALL NETVARBSIZE (S)

#### Arguments

Numeric variable

#### Usage

After bloading data from remote server or SD-card and checking for successful completion using \_NETCODE, application can obtain size of data bloaded.

---

## 3.9. Built-in communication tools

GE8NET firmware allows you connecting to other network devices and controlling them in real time with the applications described below. The difference is that telnet application supports telnet protocol (RFC 854) and selected ESC sequences, while terminal application does not support any application layer data communication protocol, and will display all the characters incoming through network according to flags provided.

**Important notice:** If you use `_NETHOST` command defining IP address of the remote host using IP address octet numbers instead of host name string, e.g.

```
_NETSETHOST(192,168,1,10)
```

then host name string will be cleared, and terminal will go into listening state waiting for incoming connection rather than performing active connection. To get terminal actively connecting to the remote device when you need to identify this remote device using IP address, use IP address as a string:

```
_NETSETHOST("192.168.1.10")
```

### 3.9.1. Terminal application

Terminal is a bare communication program allowing you seeing what another network node is sending. It has rich set of options so that you can control your input style and display incoming control codes as characters.

The main purpose of terminal would be debugging the connection, or performing very simple remote tasks, not involving any application layer communication protocol. If you need to control the router, or access any other service using application layer protocols, please use telnet application.

#### NETTERM

Perform terminal session using TCP

##### Format

```
CALL NETTERM ([HN$,] F)
```

##### Arguments

HN\$ is a variable or constant, it is host name, with port separated with colon

F is variable or constant, flags, if omitted – input is assumed to be 0

##### Usage

The terminal application does the simplest thing of receiving characters from the remote host and displaying it on your screen, and sending characters you type to the remote host (and optionally display it on your screen). It does not perform *special* ESC code translation, thus if ESC code (&h1B) is displayed as control character, following sequence is treated according to available MSX escape sequences (see appendix 10 of the MSX Technical Handbook [here](#)).

If first argument HN\$ is present, it designates host name and remote port, for example "myhost.com:23". Port number may be omitted (together with colon), then port number set by `_NETSETPORT` is used. If host name is empty, then terminal waits for the

connection rather than actively connects to remote host. If host name is omitted at all, host name set by \_NETSETHOST will be used.

Second argument F is flags.

- Bit 0 set means echo characters you type onto to your screen. Should be kept 0 (default) if remote network device perform echoing of what it receives back to GR8NET;
- Bit 1 set means terminal application not to add LF to the CR character. If bit is 0, when you press ENTER key there will be LF (new line character) added and sent to the remote host (and displayed onto your screen if echoing is on);
- Bit 2 set instructs terminal application displaying all characters, except CR and LF, as graphical characters, thus, for example, if remote client sends decimal character code 12 (CLS), there will be character with code 12 displayed instead of your screen being cleared;
- Bit 3 set instructs to display CR as well as LF characters as raw characters (see bit 2). Thus if remote host sends CR (presses its ENTER key) it will be displayed as graphical character rather than cursor moving to the beginning of the line;
- Bit 4 set instructs to enter network listening state waiting for the connection from another host. It has the same effect as having host name string empty.

### Examples

CALLNETTERM("www.gr8bit.ru:80")	` actively connect to www.gr8bit.ru on port 80
CALLNETTERM("192.168.1.1:23")	` actively connect to 192.168.1.1 on port 23
CALLNETTERM("www.google.com",15)	` will use port # defined by NETSETPORT
CALLNETTERM("":23")	` wait for remote connection on port 23
CALLNETTERM(16)	` wait for remote connection on port set by NETSETPORT

Here're the screen shots for the CALL NETTERM command.

**Says "Connected" when successful**



```
Copyright 1986 by Microsoft
Ok
callnetterm
Connected at 195.208.1.127:80
HTTP/1.1 400 Bad Request
Server: nginx/1.10.1
Date: Thu, 05 Apr 2018 20:18:49 GMT
Content-Type: text/html
Content-Length: 173
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.10.1</center>
</body>
</html>

Connection was closed
Ok
```

Picture above shows terminal connecting to the host defined by the \_NETSETHOST command to the port defined by the \_NETSETPORT command.

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft
Ok
callnetterm(":23")
Waiting at 192.168.1.46:23
```

Waiting for  
incoming  
connection  
request

Picture above shows terminal waiting for the connection. There will be no action until remote device will request connection onto the port 23 defined by the command on the picture.

### 3.9.2. TELNET application

TELNET application supports TELNET protocol and is able to display selected ESC sequences onto the screen, providing convenient and useful way communicating with network devices using formatted output and TELNET control protocols.

TELNET application requires MSX2 machine because it works in SCREEN 7 graphical mode, and makes use of ANSI colors.

#### NETTELNET

Perform telnet session using TCP

##### Format

CALL NETTELNET ([HN\$,] F)

##### Arguments

HN\$ is a variable or constant, it is host name, with port separated with colon

F is variable or constant, flags, if omitted input is assumed to be 0

##### Usage

Using and handling arguments is similar to the terminal application, but TELNET is having only one meaningful bit within its flags:

- Bit 1 set means TELNET application not to add LF to the CR character. If bit is 0, when you press ENTER key there will be LF (new line character) added and sent to the remote host (and displayed onto your screen if echoing is on).

In addition, TELNET application can only work in client mode.

##### Examples

CALLNETTELNET("192.168.1.1:23")

` connect to 192.168.1.1 (e.g. router) on port 23

CALLNETTELNET("bbs.hispamsx.org:23")

` connect and use HispaMSX BBS



Here're the screen shots for the CALL NETTLNET command.

```
Connecting to 192.168.1.1:23 - OK
Login: admin
Password: *****
(config)> _
```

Connection to the router

```
Synchronet BBS for Linux Version 3.17 Copyright 2016 Rob Swindell

CLIENT CONN: Telnet
ADDR: broadband-46-242-12-136.moscow.rt.ru [46.242.12.136]
SERVER NAME: HispaMSX BBS
ADDR: bbs.hispamsx.org
NODE: 1 (of 8)
TIME: Thu Apr 12 2018 12:07:08 UTC+1:00
ADMN: Karloch

If you are a new user to the system, type "New" now.
Otherwise, enter your user name or number now.

Enter User Name or Number or 'New' or 'Guest'
Login: _
```

HispaMSX BBS logon screen

```

HISPAMSX Local Time: 12:08:07 HispaMSX BBS
Press ? anywhere to get online help

  Read/Post Messages      Message Area Selection    Electronic Mail
  NRZBPQA                J Jump to new msg area    E Read/Send E-mail
  New message scan        * List sub-boards
  Read message prompt     * List groups
  Continuous new scan     { } # Select sub-board
  Browse new scan         [ ] # Select group
  QWK packet transfer
  Post a message
  Post auto-message

  Message Search          Go to                      Other Commands
  F Find text in messages T File Transfer section
  S Scan for msgs to you  G Text file section
                          C Chat section
                          X External programs
                          D Default user config
                          & Message scan config
                          U User lists
                          I Information
                          M Minute Bank
                          ^L Node activity
                          ^K Ctrl-key Menu
                          O Logoff BBS (or ^O)

Anytime | Ctrl-U Who's online Ctrl-P Send private msg Ctrl-C Abort cmd/text

Main | 3:59:34 [1] Local [7] Main: _
```

HispaMSX BBS menu



The following characters and control sequences are supported from the remote device (e.g. telnet server):

Char code or sequence	Meaning	Notes
0	Null character	Discarded
7	Bell	Standard MSX beep sound
8	Back space	
9	Tabulation	Cursor is moved into next column position of multiple of 8
10	Line feed	Cursor goes one line down with screen scrolling
13	Carriage return	Cursor goes into column 1
<b>VT52 emulation</b>		
ESC A	Cursor up one line	
ESC B	Cursor down one line	
ESC C	Cursor right one column	
ESC D	Cursor left one column	
ESC H	Cursor home	
<b>VT100 emulation</b>		
ESC [ {r};{c} H	Cursor positioning	
ESC [ {r};{c} f	Force cursor position	same as ESC [ H
ESC [ * A	Cursor up * lines	Cursor moves till the respective border of the screen without any further scrolling
ESC [ * B	Cursor down * lines	
ESC [ * C	Cursor right * columns	
ESC [ * D	Cursor left * columns	
ESC [ s	Save current cursor position	
ESC [ u	Unsave (restore) cursor position	
ESC [ * J	Erase screen portion	0 or no * argument = from, 1 = to, 2 = all and move home
ESC [ * K	Erase line portion	0 or no * argument = from, 1 = to, 2 = all and move to the line start
ESC [ * m	Assign attributes	0 = reset all, 1 = bright color, 30-37 = foreground color, 40-47 background color
ESC [ 6 n	Report cursor position	Telnet client sends current cursor position to the requesting device

---

The following characters and key presses are supported at the telnet client side:

Key, code or combination	Explanation
^A to ^H	Control character of respective code (1 to 8) is sent to the server
^I or TAB key	Control character 09h (tabulation) is sent to server, server may respond with respective number of space characters
^J to ^_	Control character of respective code (10 to 31) is sent to the server
32 to 126	Printable characters sent to server as is
127	DEL character is sent as is, server must issue required ESC sequences to erase characters within the telnet client's screen
128 to 254	Printable characters sent to server as is
Arrow keys	Respective ESC [ A/B/C/D code is sent to server
^ arrow keys	CTRL key with up/down function as page up (ESC [ 5~) and page down (ESC [ 6 ~), with right and left function as end (ESC [ 4 ~) and home (ESC [ 1 ~) respectively
INS key	Sends ESC [ 2 ~ to the server
ESC key	Sends character with code 27 to the server
BS key	Sends character with code 8 to the server

Function keys, and keys SELECT, CLS/HOME have no effect during telnet session.

---

## 3.10. Built-in media player

Three formats are supported by the card's software and firmware:

- Wave files (.WAV) – mono and stereo. For stereo files, mono GR8NET will only output one of the channels. When creating WAV files you may balance between number of channels (mono/stereo) and sampling rate;
- MPEG-1 layer 3 files (MP3); to play this format you will need to have GR8NET reconfigured into MP3 player FPGA image, or have additional [MP3 cartridge](#) installed in your system as an output device; MP3 file playback from RAM buffer is not supported;
- MSX video files (SC2, SC8 and SCC for respective video modes) will only play from the SD-card because network bandwidth is not enough to provide appropriate streaming speed for smooth video playback.

### 3.10.1. Playing WAV or MP3 file from network source

You can play WAV file in two ways: from built-in browser by pressing TAB key on the directory entry, or by calling `_NETPLAYWAV` from BASIC. It will contact remote server, identify type of the file, and if it will appear to be valid file, will proceed to playback.

To be identified as WAV file it should have valid RIFF header; to be identified as MP3 file the content should start with 'I' (ID3 tag) or byte 0ffh (first 8 bits of MP3 frame).

To play WAV data the utility is using hardware acceleration – PCM function and prefetch directly from W5100 buffers, which allow proper play of the 8-bit WAV samples at 22050 kHz frequency, 16-bit samples at 11025 kHz from the network, and up to stereo 16-bit at 22050 kHz from SD-card.

It is advised to select 22050 kHz with 8-bit WAV samples because quality of the output sound is defined by the sampling rate rather than difference between 8 and 16 bits in the sample.

Important: if network data stream is delayed, playback will hiccup, causing unpleasant experience. Ensure minimal delays on the network by having server as close to GR8NET as possible from timing perspective and remove unneeded traffic from the network during playback.

Warning: player allows user pressing ESC key to terminate playback. While this is very useful feature in terms of user interface, TCP connection being used for data exchange does not assume graceful unilateral exit in the middle of the TCP data exchange process. While GR8NET sends disconnect request to the server, it then sends rude connection close request to the server. Server's web server sending process may still need to send all data through its socket, thus, while GR8NET will not be accepting data at its side, server will have port open, and thus GR8NET application will experience problems when *reusing same source port number* after such ungraceful termination of the connection. The solution is to turn dynamic port allocation on by `_NETSETPORT` statement.



## NETPLAYWAV

### Play wave file

#### Format

CALL NETPLAYWAV (A\$)

#### Arguments

A\$ is a string variable or constant identifying URI to the remote file. Argument is mandatory. If URI provided is missing required parts (e.g. host name, remote port #), then this information will be obtained from values in default URI structure managed by \_NETSETHOST, \_NETSETPATH, \_NETSETPORT statements.

#### Example

```
CALL NETPLAYWAV("http://www.somehost.com/somepath/somefile.wav")
```

```
CALL NETPLAYWAV("sdc:///audio/mywavfile.wav")
```

```
CALL NETPLAYWAV("somefile.wav")
```

### 3.10.2. Listening to internet radio

Internet radio providers do as simple as send stream of MP3 frames to the connected client, and technically there's absolutely no difference between playing MP3 file over network and playing internet radio stream. You just need to point URI to the right host, destination port and path to start receiving MP3 frames.

There're two ways to play internet audio stream:

1. Using NETBROWSE in the MP3 medial player mode, there will be additional entry available for browsing *Online radio*. Pressing ENTER key on this entry will redirect browser to [this location](#). If you want to listen to specific list, then you press TAB key on the entry;
2. Using NETPLAYWAV command and pointing to the resource; e.g.

```
CALL NETPLAYWAV("http://ic3.101.ru:8000/v13_1")
```

will play Russian internet radio called "Relax FM". List of all stations served by this media server can be found [here](#).

You can still click ESC key to interrupt playback. It may continue for some moments after cancellation, until MP3 decoder frame buffer depletes.

Note about multicast: multicast does not work properly over the internet, at least in relation to internet radio transmissions.

### 3.10.3. Playing WAV or MP3 files from SD-card

Playing media files from the SD-card is very similar to playing them from network, the only difference is that you use local SDC:// to SDF:// device (depending on the SD-card partition being used), which has minimal data transmission delays, for example

```
CALL NETPLAYWAV("sdc:///mypath/myfile.mp3")
```

---

thus playing from SD-card can be considered as more reliable from *quality predictability* point of view. If you are going to design application which will use wave or MP3 playback, the best way would be have media file on the SD-card rather than on the network.

### 3.10.4. Playing wave from GR8NET RAM

This feature provides flexible way to play WAV samples in GR8NET RAM, which can be of various size, various sample rate, and which can be dynamically synthesized in the RAM. This functionality is used in [Heroes of Might and Magic III demo](#) for MSX.

Note: MP3 playback from GR8NET RAM is not supported.

The set of commands described below use GR8NET hardware acceleration – prefetch from RAM, PCM playback function and controlled interrupt generator, thus you should not make changes to these hardware functions while wave play functionality is in use. In addition, commands may use Math-Pack's BDC-to-integer conversion, changing its input and thus output values.



**Important:** command name is NETPLAYBUF, and format of the respective commands described below deviate from standard. NETPLAYBUF command must be followed by adapter number (0...3), if not default adapter should be used, and then by **modifier letter** which defines functionality.

Thus to address adapter #2 setting addresses application should issue \_NETPLAYBUF2A (and **not** NETPLAYBUFA2). This mechanism of modifier letter, rather than additional parameter for the command, was chosen in order to improve speed of processing. **[#]** in command descriptions below designate possible one digit number of adapter being addressed. If this number is omitted, current adapter is assumed.

#### **NETPLAYBUF[#]A**

Set up starting buffer address and data size and perform initial buffering

##### Format

CALL NETPLAYBUFA (PG, ADDR, SIZ)

##### Arguments

All arguments are mandatory

PG is logical page number and ADDR is pointer within this logical page for wave data start; SIZ is data size (ensure position does not slip outside of GR8NET 1MB buffer RAM)

##### Usage

Command calculates absolute starting address and ending address of the wave data, and sets prefetch function up. Then it fills PCM memory with data (performs initial buffering). PG should be within 0...&h7F range, ADDR within 6000-7FFF (GR8NET bank 1).

##### Example

CALL NETPLAYBUF3A (2, &h675F, 20000)

## NETPLAYBUF[#]P

Initiates playback of pre-buffered data

### Format

CALL NETPLAYBUFP (B, FREQ)

### Arguments

B is size of the data: 8 or 16 (bits)

FREQ is frequency in Hertz: 1...65535 Hz

### Usage

This command starts controlled interrupt generator and PCM function to play pre-buffered PCM data. Playback stops when PCM buffer depletes, thus it is vital to execute \_NETPLAYBUFC command before it happens.

### Example

CALL NETPLAYBUFP (8, 13200)

## NETPLAYBUF[#]C

Continues playback, replenishing PCM buffer from prefetch

### Format

CALL NETPLAYBUFC

### Usage

This command replenishes data from prefetch into PCM buffer RAM. If input data has depleted, or PCM buffer is full, command does nothing.

Use this command in interrupt handler to regularly replenish data so that wave playback continues without interruptions.

### Example

```
10 ON INTERVAL=40 GOSUB 100
20 CALLNETBLOAD("sdc:///mywavedata.raw")
30 CALLNETPLAYBUFA(0,&h6000,573168)
40 CALLNETPLAYBUFP(8,13200):INTERVAL ON
50 CALLNETPLAYBUFS(A):IF A=0 THEN 50 ELSE STOP
100 CALLNETPLAYBUFC:RETURN
```

## NETPLAYBUF[#]S

Get status of the playback

### Format

CALL NETPLAYBUFS (S)

### Arguments

S must be a variable, returning -1 when PCM playback has ended, and 0 when it is still running. End of playback is the state when prefetch has no more data, and PCM has finished playing all its buffer data.



## NETPLAYBUF[#]R

Reset playback engine

### Format

CALL NETPLAYBUFR

### Usage

This command resets PCM function and controlled interrupt generator.

The following algorithm shows flexibility and usefulness of the RAM buffer playback:

1. Application loads data into the GR8NET RAM (e.g. using \_NETBLOAD command), for example data is 8-bit, 16 kHz and size is 500,000 bytes;
2. Runs \_NETPLAYBUFR to ensure that subsystem is reset;
3. Runs \_NETPLAYBUFA (2, &h675F, 500000!) command with arguments stating the logical page 2 and address 675Fh within GR8NET RAM, and setting full size of data to play. Command buffers about 32 KB of initial data, but does not start its playback;
4. At required time application runs \_NETPLAYBUFP (8, 16000) to start playback of the wave data. Command exits immediately.
5. Playback of 8-bit data at 16 kHz will take 0.75 seconds, thus sound is expected to stop within these 0.75 seconds. BASIC program must, within this time, call \_NETPLAYBUFC command to replenish data in PCM buffer; the best way to do it is using ON INTERVAL GOSUB mechanism, ensuring that this command is called some time before data in PCM buffer is expected to deplete.
6. To find out if wave data finished playing, application uses \_NETPLAYBUFS(S) command.
7. Finally, program resets hardware acceleration functions use \_NETPLAYBUFR command.

### 3.10.5. Playing video file from SD-card

Since end of March 2017 utility GR8VIDEO.COM is replaced with GR8NET firmware built-in BASIC command \_NETPLAYVID. This command supports video files created using method, described in chapter [Making videos for MSX](#) for screen modes 2, 8 and 12. Since September 2018 capability to play videos on TMS VDPs (MSX1 machines) was added, but please note that in order to look good on TMS VDPs video must be prepared converting frames for MSX1 with default MSX1 palette.

Mode	VDPs	Video configuration	Audio configuration
SCREEN 2 (.SC2)	T, 3, 5	256 x 192 pixels	22050 Hz, 8-bit, mono
SCREEN 8 (.SC8)	3, 5	X*Y to be not greater than 13,872, but not less than 11,098 (for example 136*102)	
SCREEN 12 (.SCC)	5		

T: TMS99xx, 3: V9938, 5: V9958

Below is the table for supported machine types and file versions and formats. SCREEN2 video files are having two types of formats, old (version 0), and new (version 1), and you can play version 0 files only on MSX2 and above. TMS VDPs with less than 16K of VRAM are not supported.

Machine	VDP	VRAM size (min)	File version support
MSX1	TMS 9918/28 or TMS 9929	16K	.SC2 V1 only
MSX1.5	V9938	16K	.SC2 V1 only
MSX2		32K	.SC2 V0 and V1, .SC8
MSX2+	V9958	32K	.SC2 V0 and V1, .SC8, .SCC
Turbo-R			.SC2 V0 and V1, .SC8, .SCC

SCREEN 2 versions differ in the way they have video data laid out: in version 0 (non-interlaced) full frame color data follows full frame of patterns, but in version 1 (interlaced) pattern data and color data of 256 byte chunks each are being interlaced. This format is specifically designed for MSX1 and MSX1.5 machine when quick update to same regions of both pattern and color data is required for display with minimal visible artifacts.

You can see that .SC2 V1 videos are supported in each mode, and you have an easy way to migrate your existing videos from V0 format to V1 format. Please use **convert-sc2-video.xls** Microsoft Excel file available in the location <http://www.gr8bit.ru/software/video/>.

SCREEN 2 quality of the video assessment information is provided in the table below. Note that MSX1 and MSX1.5 can only play interlaced files (version 1).

Machine	Display properties	Quality	Comments
MSX1 (TMS)	Built-in palette, single video page, interlaced output	Good	Pattern artifacts on the screen, but no palette "flashing" effects
MSX1.5 (V9938)	Modifiable palette, single video page, interlaced output	Fair	.SC2 files with modifiable palette: less pattern artifacts on the screen, but "flashing" due to palette changes at the same video page
		Good	.SC2 files with MSX1 built-in palette: quality is the same as on MSX1
MSX2 (V9938)	Modifiable palette, two video pages, interlaced and non-interlaced output	Very good	.SC2 files with modifiable palette: no artifacts on the screen, but dot approximation due to SCREEN 2 mode limitations
MSX2+ (V9958)			
Turbo-R (V9958)			

To prepare video for playback, use your PC to decompress video and copy it to SD-card.

The NETPLAYVID command will run supported videos on any machine, but you should mind the limitations:

- If machine is having only 16kB of VRAM, command can play only SCREEN 2 videos, and will use only single VRAM page thus there will be artifacts visible on the screen;
- On Turbo and overlocked machines above 5.37 MHz video player makes its best to return to standard Z80 operating conditions (to timing coded into the video file – e.g. switching Turbo mode or boost mode off during playback), but videos in SCREEN 8 and

---

12 modes may run faster or slower (!) than expected. Application or user should set nominal machine clock mode before playing the video;

- Playback from the network is not supported;
- During playback command switches Turbo-R's R800 CPU into Z80 mode, and Panasonic MSX2+ turbo mode off, and restores original operating modes after playback is finished;
- After playing SCREEN 2 mode video files, default "blue" MSX palette will be reloaded, but screen's foreground, background and border color assignments will be preserved.

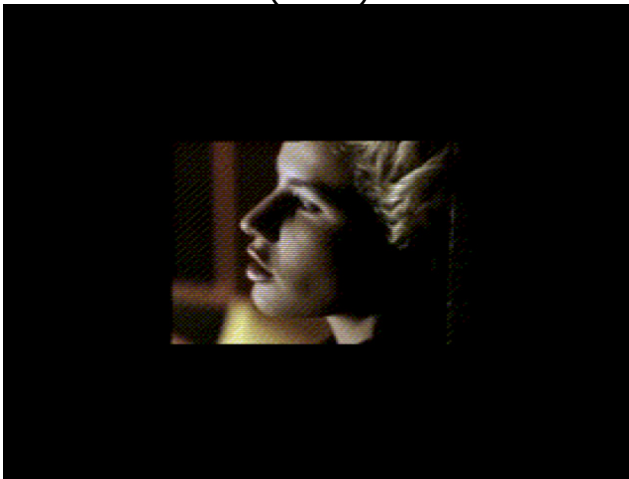
Below are the screen shots for the various machine and screen modes. You can find sample files at the same location <http://www.gr8bit.ru/software/video/>.



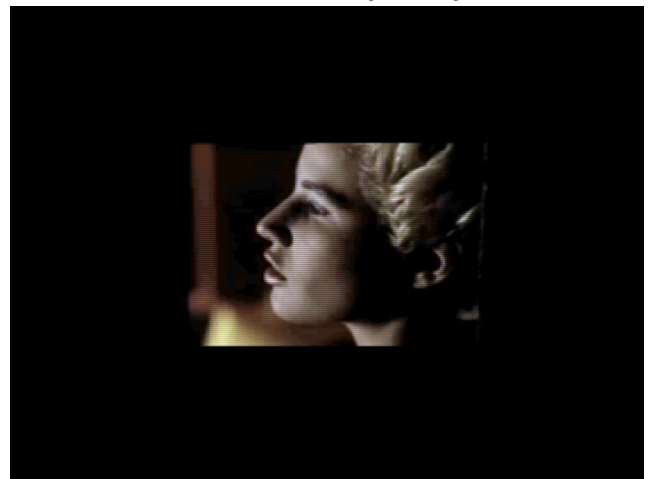
SCREEN 2 video (brfestlt.sc2) on TMS9929 (MSX1)



SCREEN 2 video (brfestlt.sc2) on V9938 with 128K VRAM (MSX2)



SCREEN 8 video (brfestlt.sc8) on V9938 (MSX2)



SCREEN 12 video (brfestlt.scc) on V9958 (MSX2+)

## NETPLAYVID

Play video file from SD-card

### Format

CALL NETPLAYVID (SDU\$ [, SI])

CALL NETPLAYVID (SM)

### Argument

The command works in two modes, depending on the type of first argument

Mode 1: first argument is string

SDU\$ is a string constant or variable pointing to the video file on the SD-card, for example "sdc:///video.sc8" or "sdc:///videos/msxvideos/myvideo.sc2";

SI bit [0] is screen initialization flag, if set to 1 then screen will not be initialized. If omitted, value of 0 is assumed;

SM bit [6] is set to force black background and border color in modes 8/12 (effective for background color only if bit 0 is not set);

Mode 2: first argument is integer

SM bits [3:0] is screen mode to initialize screen with, valid values are 2, 8 and 12;

SM bit [7] is disable screen flag, if set to 1 screen will be disabled for display;

SM bit [6] is set to force black background and border color in modes 8/12.

### Usage

The command usage is split into modes for specific purpose so that application can have screen initialized, then put some its own art into the screen around video display area, and then call actual video playback. [Heroes of Might and Magic III demo](#) uses this algorithm.

Application can use BASIC's SCREEN operator to initialize desired screen mode, but \_CALLNETPLAYD(SM) is having several important features: (a) it initializes screen mode as SCREEN does; (b) it clears both video pages to be used by video player, and (c) it can disable screen (using BL bit of VDP register 1) so that further VRAM/drawing operations are not visible to the user. Screen is got enables by the following \_NETPLAYVID (SDU\$, 1) command which starts actual video playback.

In video modes 8 and 12, \_NETPLAYVID(SM) initializes video pages with the background color set by *COLOR* *x*, *y*, *z*, thus to have black background perform e.g. *COLOR 15,0,0* before initializing screen mode. In video mode 2 background and border colors are always forced to be 0 (black).

Video file name does not indicate format, screen mode or geometry; all these properties are coded into the video file.

---

### Typical video playback BASIC program

10 gc=peek(&hf3ea):bc=peek(&hf3eb):color 15,0,0	Get current background and border colors, set current to 0 (black)
20 callnetplayvid("sdc:///3do.sc2")	Play video (screen 2 mode)
30 callnetplayvid(136)	Initialize screen 8, disable screen output (136=128+8)
40 callnetbload("sdc:///hmm3-intro-frame.sc8")	Blod image into GR8NET RAM
50 callnetbtov(0,0):callnetbtov(1,0)	Move this image to VRAM's page 0 and page 1 so that when pages are alternated this image appears on both even and odd video frames
60 vdp(10)=vdp(10) and 127	Set 192 lines per screen, display is still disabled
70 callnetplayvid("sdc:///hmm3-intro-video.sc8",1)	Start playing screen 8 video, do not re-initialize the screen (1) and enable screen output (automatically)
80 color 15,gc,bc	Revert back to initial background and border colors

Note the following:

- `_NETPLAYVID(SM)` uses first RAM buffer page, thus image load must be performed after screen 8 initialization (otherwise previously loaded image may be corrupt). Alternatively, image may be loaded into RAM page 1 or further with `_NETBLOAD("sdc:///mygame/image.sc8",,1);`
- Video player alternates two video pages when displaying video, thus take care to load background image into both video pages, as shown in line 50 above. Screen 2 videos are displayed full screen, thus there can be no images around the video display area;
- When playing SCREEN 2 videos border and background colors will be automatically set to 0 (not depending on COLOR operator setting) because color 0 may be reserved for black, while others may change their palette.

### Errors

If there's error, command terminates with *Device I/O error* for first mode and *Illegal function call* for second mode with:

- NETCODE 2B if there was SD-card read;
- NETCODE 2E if screen mode coded into the file is not supported on the machine;
- NETCODE 20 if there's no enough VRAM/GR8NET RAM to display the image.

It is the task of application to ensure file will play on the target machine (e.g. using NETSYSINFO).

Playback can be interrupted by pressing ESC key.

### 3.11. Setting memory mapper

Mapper mode is switched by the `_NETSETMAP` command, which sets specified memory mapper type and appearance of the special register set, and reboots machine. Important: 1 MB buffer RAM is not used during GR8NET adapter initialization, and after soft or hard reset it is kept intact if no other networking operation is explicitly performed (e.g. BLOAD, RAM disk image load). Thus you still can issue `_NETSETMAP` command to start the image. Power cycle erases the data in buffer RAM.

GR8NET adapter can function in the following memory mapper modes (see [Setting operating mode and mapper type](#) chapter for more information):

Mode	Purpose	NETSETMAP Argument A	Game mapper size
Modes 0 – 7: cartridge may be in primary or expanded slot			
0	GR8NET	<b>16 *</b>	-
1	Plain write-protected 32K	<b>1</b>	<b>32K</b>
2	Konami without SCC	<b>2</b>	<b>256K</b>
3	Konami with SCC	<b>3</b>	<b>512K</b>
4	ASCII 8	<b>4</b>	<b>1024K</b>
5	ASCII 16	<b>5</b>	<b>1024K</b>
6	Mirrored ROM	<b>6</b>	<b>64K</b>
7	1 Megabyte mapped RAM	<b>7</b>	-
Mode 8 – 14: cartridge must be in primary slot; GR8NET with 512K is in subslot 0, 512K mapped RAM is in subslot 1 (can be disabled), Nextor is in subslot 2			
8	FM-Pak and SCC/SCC+ is in subslot 3	<b>24 *</b>	-
9	Plain write-protected 32K in subslot 3	<b>25 *</b>	<b>32K</b>
10	Konami without SCC in subslot 3	<b>26 *</b>	<b>256K</b>
11	Konami with SCC in subslot 3	<b>27 *</b>	<b>512K **</b>
12	ASCII 8 in subslot 3	<b>28 *</b>	<b>1024K **</b>
13	ASCII 16 in subslot 3	<b>29 *</b>	<b>1024K **</b>
14	Mirrored ROM in subslot 3	<b>30 *</b>	<b>64K</b>
15	Not allowed	-	-

\* For mapper modes 0 and 8-14 **it is required to add 16** to the mapper mode number so that special register set appears within the GR8NET workspace. If you forget to add 16, you will get "Config error: 00" message when GR8NET initializes, and this message means that GR8NET firmware did not detect special register set in its required location of 5FC0h-5FFFh.

\*\* These composite mapper modes may cause GR8NET RAM space allocation conflicts as there're competing services like mapped RAM and MSX-Audio sample RAM. Please read more details in chapters [Memory Manager](#) and [RAM allocation conflicts in composite mappers](#), and how to prevent these conflicts.



## NETSETMAP

Set specified memory mapper type and reboot

### Format

CALL NETSETMAP [(A, M, MRPD)]

### Arguments

A is variable or constant *identifying mapper type and location of special register set*

M is variable or constant, identifying if GR8NET will respond to mapped RAM register ports' read. Read more about setting this argument in [Mapper read flag of the chapter System registers](#). Valid values are 0 (read disabled), 1 (read enabled), and 2 (auto-detect, default).

MRPD is mapped RAM pending disable bit, value must be 0 (enable) or 1 (disable). When this value is 1, mapped RAM will be disabled *after mapper type is changed*.

If argument A is omitted, M is set for current mapper mode.

If GR8NET *is already in the respective game mapper mode (9-14)*, then you can run command without arguments: command checks the GR8NET RAM buffer for the valid ROM image, and tries starting it without reset (first calling ROM's initialization code, and then calling HSTKE hook if ROM initialization code returns).

### Important note

Properly designed software will never read mapped RAM mapper ports located at 0fch-0ffh, keeping track of its changes in its memory or through system. However, some applications are designed the way they identify size of memory mapper by reading these mapper ports, and will misbehave in case mapper size does not match information read from the port. Example: system is having built-in 128K mapper, which has 3 significant bits on data bus. If you will run GR8NET with mapper port read disabled, application, after writing 0 to port 0fch will read 0e0h, thinking there're only 128K mapper, and proceed setting 0e0h as first mapped RAM page. However, if GR8NET mapped RAM is chosen as main RAM, in mode 8 (5 significant bits) it will have page 20h set instead of 0, and machine will crash. One of the applications affected is MSX debugger DBG.COM (DBGE.COM), which will not run if mapper read is disabled.

### Example

(netblood knightmare.rom)  
CALL NETSETMAP (1)  
(reboot)

CALL NETSETMAP  
This mapper mode does  
not have the ROM mapper  
Ok

CALL NETSETMAP (24, 1)  
(switch to mapper 8 with special  
register set and respond to mapper  
regs reads)

(netblood metal gear 2)  
CALLNETSETMAP(27,,1)

(will run Metal Gear 2 with mapper 3 in subslot 3 with GR8NET mapped RAM disabled in subslot 1, and Nextor available in the system in subslot 2)



## NETGETMAP

Get current memory mapper type and some other operating flags

### Format

CALL NETGETMAP (I)

### Arguments

I is 16-bit variable getting mapper type and flags; its bits 7 to 0 are the value of current [mapper register](#), and bits 8, 13 and 4 are bits from [system mode register](#).

### Example

CALLNETGETMAP

411B M3 (512K+none+Nextor+K5)

Ok

The above output means that GR8NET is having slot expansion internally (mapper mode is 1B = 16+8+3), operating in mapper mode 3 in subslot 3 (Konami 5), has Nextor in subslot 2, has mapped RAM disabled ("none", bit 6 is set 411B).

## NETTGTMAP

Set target memory mapper configuration to switch to at the startup

### Format

CALL NETTGTMAP (A [, M])

### Arguments

See NETSETMAP command

### Usage

The difference between \_NETTGTMAP and \_NETSETMAP is that former will cause adapter to restart machine in target mapper mode when machine cold-starts. The value set by \_NETTGTMAP is preserved by \_NETSAVE command to become in effect on next cold boot.

---

## 3.12. Managing audio mixer

There're two versions of GR8NET – mono and stereo, and they handle mixing and output to the MSX machine audio mixer differently.

- Mono GR8NET: single DAC channel – **right channel** – is connected to the MSX machine mixer, thus to hear the respective device through MSX you must set it to output to right channel, output to the left channel will go nowhere unless you perform the relatively complex modification described in [DN0006: GR8NET mono to stereo modification](#) article;
- Stereo GR8NET: you can immediately distinguish stereo GR8NET by the slider switch at the left side of it near the RJ-45 networking connector, and stereo 3.5 mm audio jack at the cartridge's right side. Only **left channel** is connected to the internal MSX mixer through the slider switch; if slider switch is in upper position, you will not hear GR8NET's left channel through MSX machine audio output.

Therefore, the mixer setting will be different for mono and stereo GR8NETs, and will also very depend on how application software uses GR8NET's devices – for example complementary output of MSX-Audio and MSX-Music, or pseudo stereo output with MSX-Music in one channel and MSX-Audio is another. Also mind the channel PSG sound will appear in – and it will also very depend on the application software.

For example, for demo Unknown Reality using pseudo-stereo, MSX-Music and PSG should be output together through one channel, and MSX-Audio should be output through another. Another example: Aleste 2 game which uses MSX-Music and PSG simultaneously, and you can output both MSX-Music and PSG together into both channels.

**Important:** if you hear incorrect. Bad or corrupt MSX-Audio and/or MSX-Music sound, ensure that you did not mistakenly configure these devices to sound into same channel(s) when application expects them to sound into different channels. Superposition of MSX-Audio and MSX-Music may really sound bad if software is not specifically designed for it.

## NETSETMIX

### Set mixer configuration

#### Format

CALL NETSETMIX (M)

CALL NETSETMIX (M\$)

#### Arguments

You can supply numerical value or string value through variable or constant

#### Usage

If numerical value is supplied, it represents bitmap for the channel output. For more information refer to [Mixer and DAC](#) chapter.

If string value is supplied, it has the following format: "D<sub>1</sub>D<sub>2</sub>D<sub>3</sub>D<sub>4</sub>D<sub>5</sub>D<sub>6</sub>", where:

Position	Device
D <sub>1</sub>	PCM
D <sub>2</sub>	SCC
D <sub>3</sub>	Wave
D <sub>4</sub>	OPLL
D <sub>5</sub>	Y8950
D <sub>6</sub>	PSG

Value in position	Behavior
'M'	Mute, device does not appear in any channel
'L'	Left, device appears in left channel only (stereo GR8NET has left channel connected to MSX mixer)
'R'	Right, device appears in right channel only (mono GR8NET has right channel connected to MSX mixer)
'B'	Both, device appears in both left and right channels
Any other	Current channel assignment is preserved

#### Examples

CALL NETSETMIX(&HACD) ` PCM=left, SCC=both, Wave=mute, OPLL=both, Y8950=right, PSG=both

CALL NETSETMIX("LBMBRB") ` same as above

CALL NETSETMIX("BxxLRL") ` PCM=both, SCC/Wave=keep current setting, OPLL=left, Y8950=right, PSG=left

## NETGETMIX

### Set mixer configuration

#### Format

CALL NETGETMIX (M)

#### Arguments

M is numerical variable, if omitted status is printed onto the screen

#### Usage

This command returns current status of the mixer in the format explained in [Mixer and DAC](#) chapter. In addition, it returns bit 15 of its bitmap value reset if GR8NET is monophonic, and set if stereo.

#### Example

CALL NETGETMIX

Type: Stereo

PCM: Left + Right

SCC: Left + Right

Wave: Left + Right

OPLL: Left

Y8950: Right

---

PSG:       Left  
Ok

The following command allows setting linear volume levels for all the GR8NET internal devices. While all volume levels are limited by 128 (maximal volume), there's a hack setting master volume to 255, this setting will cause almost 2\* *digital amplification* of the signal.

## **NETSNDVOL**

Set or display GR8NET audio volume levels

### Format

CALL NETSNDVOL (M, S, W, P, O, Y, A)

### Arguments

Arguments should be in range 0 (mute) to 128 (full volume).

Any of arguments can be omitted.

### Usage

If all the arguments are omitted, command displays values of all seven volume levels onto the screen. If specific argument is omitted, volume of specific channel is not changed. M is master mixer volume level, S is SCC volume, W is waveform input volume, P is PCM volume, O is built-in OPLL volume, Y is built-in Y8950 volume and A is built-in PSG volume.

In MP3 player mode the only effective values are M (master), W (waveform) and P (MP3 sound volume, not PCM as PCM function is not available in MP3 player mode).

### Examples

CALLNETSNDVOL(,&h60)	` lower SCC's volume by 25%
CALLNETSNDVOL(0)	` mute GR8NET output completely

---

### 3.13. Getting state of the resource (SD-card or network)

There's a way to get SD-card file or network resource properties without actually loading the file. The command explained below works with SD-card partitions formatted with FAT16 or FAT32.

#### NETRESST

Get status of the resource

##### Format

CALL NETRESST (F)

CALL NETRESST (I\$, O\$, SF, SI)

##### Arguments

F is variable, bitmap of the SD-card and its partition status and network file open status

I\$ is input URI string (SD-card or network one)

O\$ is output URI string variable (may differ with input in case of redirects)

SF is resource flags variable

SI is size of the resource variable

##### Usage

If arguments are omitted, command prints flags F onto the screen.

F bitmap variable returned the following information:

7	6	5	4	3	2	1	0
SDINS	FILS2	FILS1	FILS0	PART3	PART2	PART1	PART0

Bit	Description
SDINS	This bit is set if SD-card is inserted and initialized to be used
PART0 – PART3	If respective bit is set, then SD-card's specific partition can be accessed using NET commands. SD-card's partition 0 is addressed using SDC:// device, partition 1 – by SDD:// device, partition 2 by SDE:// device and partition 3 by SDF:// device
FILS0 – FILS2	If set, means that user socket with respective number (0, 1 or 2) is open for BASIC file I/O access (using OPEN operator)

In case first argument supplied to the NETRESST command is string variable or string constant, command will perform search of the resource pointed by I\$ on the device, defined in I\$. If you perform network resource query, please include port # (e.g. ":80" for HTTP).

In case SD-card resource is being searched for, then O\$ will return empty string "" if resource is not found and both SF and SI will be 0. If resource is found, O\$ will return URI string pointing to that resource (the same as I\$, but reconstructed), SF will be file attributes (e.g. 0x10 if entry searched for is a subdirectory), and SI will contain file size (note: directory sizes will be calculated and returned).

In case network resource is being searched for, then command will perform HEAD HTTP request first, and GET next if HEAD is not properly supported by the remote server, and check if there's 3xx redirect return code, then it will take new URI from *Location* header

---

and proceed searching in that location. Command will support only 4 redirects, returning 2F error (shown by \_NETCODE) if there're more redirects occur.

As soon as code 2xx (successful) is returned, command puts queried resource URI into O\$, puts HTTP code into SF, and resource size into SI, if this resource size was supplied in *Content-Length* header (otherwise it returns 0).

If command has communication problems, or remote server returns malformed HTTP or ICY response, then command completes with BASIC *Device I/O error*, and application can use \_NETCODE command to get diagnostic information.

If command can not reach resource (HTTP return code is other than 3xx or 2xx), it will return last queried URI in O\$, and last HTTP code in SF. Information in SI will be invalid.

#### Examples

CALL NETRESST

SD-card: Ready  
Partition 0: Yes  
Partition 1: Yes  
Partition 2: No  
Partition 3: No  
Socket 0 file: Closed  
Socket 1 file: Closed  
Ok

The following command gets properties of the AUTOEXEC.BAT file from the partition 1 of the SD-card. Note that 8:3 file names must have the same case as in the directory of the SD-card, thus querying "sdd:///autoexec.bat" will return no match (O\$ as empty string).

CALLNETRESST("sdd:///AUTOEXEC.BAT",a\$,b,c)

Ok  
?a\$;b;c  
SDD:///AUTOEXEC.BAT 32 194  
Ok

The following command gets properties of the network resource. Note it has port number, and as the result it was redirected to the new URI. If HEAD request will not return Content-Length header information, variable C will return 0.

CALLNETRESST("http://www.somehost.com:80/somepath/somefile",a\$,b,c)

Ok  
?a\$;b;c  
HTTP://www.somehost.com/newpath/newfile 200 1024  
Ok

---

### 3.14. Serial flash chip and configuration image management

GR8NET is having two flash chips where its firmware is stored: 512K chip for “front-end” ROM BIOS run by the main MSX Z80 CPU, and “back-end” serial flash used by the FPGA (GR8NET engine) for the its configuration.

Serial flash chip is located at the backend of the FPGA interface, and its contents are not mapped to any memory or I/O location for Z80, and it is one of the reasons why its space was not available to the Z80 CPU from the beginning.

All GR8NETs are originally having Altera EPCS16 serial flash chips of 2 MB in size. It appeared that this amount of flash space it too much to hold single image for FPGA, and in later releases GR8NET may have had up to 4 “locations” with various FPGA configuration images. These images were 512 KB in size each, and this size was selected for simplicity of image creation and image location management within the chip.

However starting firmware version 0.8 this has changed: now firmware operates at the serial flash chip 64 KB sector level and provides mechanism to list the flash contents to see the useful information it has.

As \_NETFPGAUPD command was accepting location number as an argument (0 to 3 for original 2 MB chip), it is discontinued, and new \_FLUPDATE command was developed accepting sector number as an argument.

Starting version 0.8 also allows having user data in the serial flash chip in the form of Catalog. The Catalog is a data in special format containing compressed ROMs – very like as you would have these ROMs on the SD-cards, but in this case you do not need SD-card and may have your favorite ROMs always with you inside the GR8NET.

#### FLINFO

Display flash chip information

Format

CALL FLINFO

Arguments

No arguments

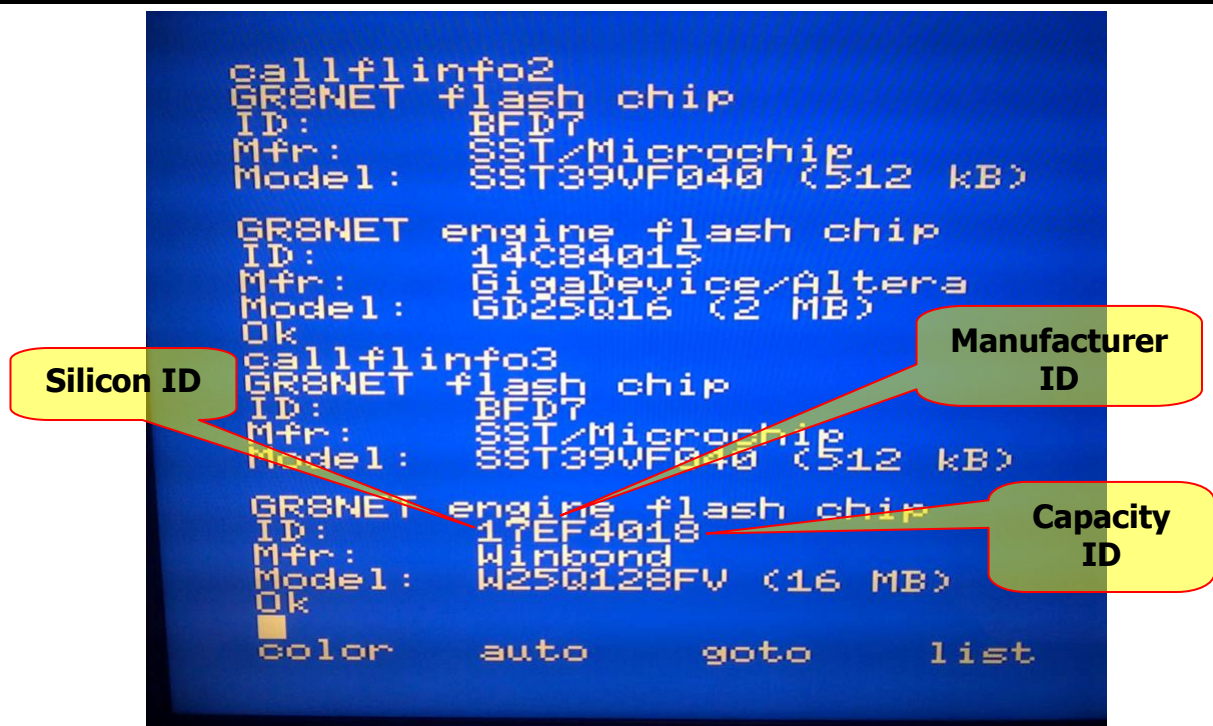
Usage

This is diagnostic command showing the flash devices installed in your GR8NET. Surely both will be listed (as they both are vital for GR8NET operation). Firmware can identify a number of the flash chip types; even if it can not identify the chip, it will identify and display the size properly. Photo below shows machine with two GR8NETs installed: #2 with original EPCS16 serial flash chip, and #3 with this chip replaced by the W25Q128FVSIQ chip.



Important: if you are replacing the serial flash chip, ensure it **does not** have Q letter in its marking, and it is **not** configured to run in quad mode. GR8NET will not support serial flash in quad mode, even more, due to the fact that it was designed for EPCS16 serial flash – other flash chips may be damaged if in quad mode. Please refer to the serial flash chip replacement guidelines.





## FLLIST

Lists contents of the serial flash chip

Format

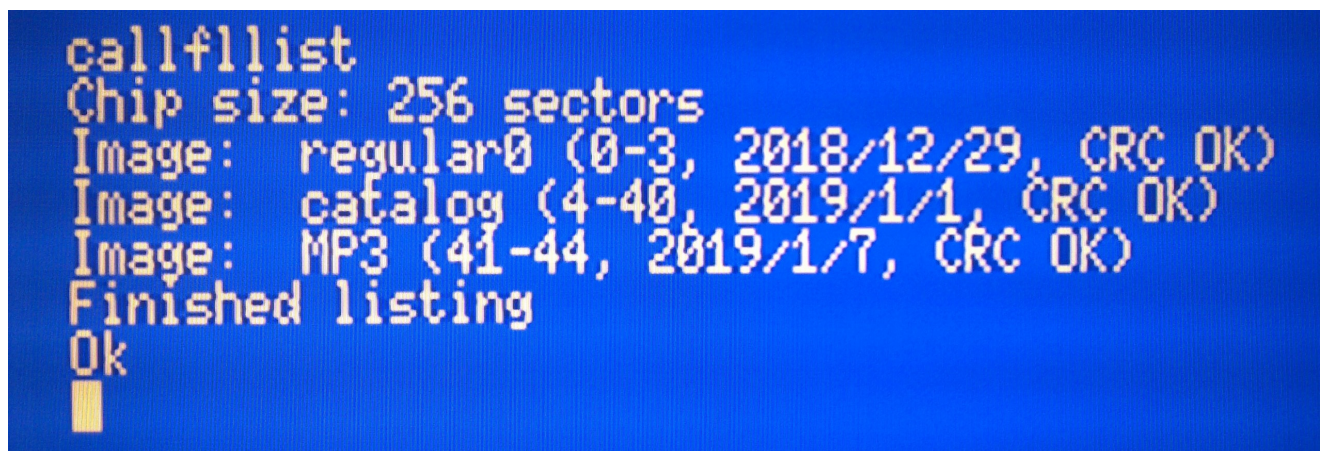
CALL FLLIST

Arguments

No arguments

Usage

This command displays identifiable contents it finds in the serial flash chip. It may run for quite a time because it not only checks signatures, but also compares checksums to ensure that images are not corrupt.



Chip size displays decimal number of 64KB sectors in the flash chip, here it is W25Q128 chip, having 256 sectors (16 MB);

Then list of images follows:

- Regular images are runnable by the GR8NET having various functionalities; you can perform \_NETRECFG to these images (if current running image supports reconfiguration);
- MP3 image is a subset of regular image having MP3 player capability;
- Catalog is a set or compressed ROMs, which can be accessed and run from the browser.

Each image has range of sectors displayed for it in decimal format, and date of creation of the update file (not necessarily equal to catalog or regular/MP3 image creation date), and then state of the CRC check.

Note that images must not intersect in their sectors, and in this screenshot three images are located consecutively. The only requirement for GR8NET to operate is having valid regular image at the beginning of the serial flash chip (starting sector 0 – also called “factory image”), all other contents’ location does not matter as long as those contents integrity is preserved (OK for CRC check).

## FLUPDATE

Update serial flash chip starting specific sector

### Format

CALL FLUPDATE (S [,F])

### Arguments

S is sector number – variable or constant

F if set forcing no prompt confirming intention to perform update

### Usage

This command is a replacement for \_NETFPGAUPD command, and it accepts sector number (instead of location number for FPGAUPD).

You must use this command with extreme caution not to overwrite factory image at sectors 0-3. If any of these sectors selected, command will prompt to confirm operation even if flag F is set.

If, for some reason, factory image is corrupt, you will have to use USB blaster, ByteBlaster-II or GR8blaster device to reload factory image into the GR8NET because without this image GR8NET will not operate.

```
ROM file loaded
Ok
callflupdate(48)
Buffer: MP3, OK
Chip size 256, range 48-51
CRC: DB05, OK
Verifying sector 30 - Mismatch
Type Y to update: ■
```

In this example we are about to write MP3 FPGA configuration image starting sector 48 (range 48-51). This operation is correct as this space is not occupied as FLLIST says. Pressing capital Y (with SHIFT key pressed) will start the update process.

“Mismatch” is not an indication of an error before update begins, it means that

contents of flash chip differs with contents of the buffer.

---

## NETRECFG

### Reconfigure GR8NET's FPGA

#### Format

CALL NETRECFG (S)

#### Arguments

S is a variable or constant of the starting sector number for the configurable image.

#### Usage

Note that argument is no more "image number" or "location identifier" as it was in firmware version 0.7. You can find list of images to reconfigure to with their starting sectors using \_FLLIST command. You can always safely reconfigure to image at the "factory location" sector 0 – if image you are currently running supports reconfiguration. If currently running FPGA image does not support reconfiguration, then power cycle the machine (power it off and then on – it will force GR8NET reloading factory image from sector 0).

Machine will reboot after completion of reconfiguration. It is required operation because resources GR8NET was providing to the machine may be gone after reconfiguration, and machine will require re-initializing itself and GR8NET.

---

### 3.15. Other utilities

GR8NET has some other useful utilities for developer and programmer to run and use within development and debugging process. They include calculating checksum and transferring data between conventional RAM and adapter's buffer RAM, putting pictures and icons into the VRAM.

#### **NETLDBUF**

Copy data from main memory to adapter's buffer RAM

##### Format

CALL NETLDBUF (P, BA, C, RA [, M])

##### Arguments

All arguments are variables or constant, start of chunk (BA) and end of chunk (BA+C) should reside in the GR8NET bank 1 (6000-7FFF)

##### Usage

Copies data from main RAM pointed by RA into the adapter's buffer RAM page P starting address BA. Whole transfer should remain within buffer RAM ( $BA+C \leq 7FFF$ ). You can transfer only to single buffer RAM page with single LDBUF command. Argument M, if set, causes change of the mapper type to M and machine reboot (if change is performed to mappers 7 or 8-14, resulting mapped RAM register read bit state will be detected automatically). Value of 255 for M is reserved.

Example: BASIC program called MAPLOAD.BAS to load 128kBytes ROM image divided into 8\*16kBytes chunks into buffer RAM

```
10 definta-z:INPUT"FILE W/O EXTENSION";A$
20 for i=0 to 7: f$=a$+"." + chr$(i+48): print "Loading #";str$(i);": ";: bload f$
30 callnetldbuf(i*2,24576,8192,&h9001):callnetrchks(i*2,24576,8192,a):PRINTHEX$(A);" ";
31 callnetldbuf(i*2+1,24576,8192,&hb001):callnetrchks(i*2+1,24576,8192,a):PRINTHEX$(A)
40 next i: print "Load complete"
```

After this BASIC program completes, issue CALLNETMAP(2) to restart machine and start execution of the loaded ROM image.

#### **NETLDRAM**

Copy data from adapter's buffer RAM to main memory

##### Format

CALL NETLDRAM (P, BA, C, RA)

##### Arguments

All arguments are variables or constant and are mandatory, BA=6000-7FFF

##### Usage

Copies data from the adapter's buffer RAM page P (0...RAMMAX-1) starting address BA into main RAM pointed by RA. Whole transfer source data should remain within buffer RAM ( $BA+C \leq 7FFF$ ). You can transfer only from single buffer RAM page with single LDBUF command.

## NETRCHKS

Calculate simple 16-bit checksum on the buffer RAM contents

### Format

CALL NETRCHKS (P, A, C[, R])

### Arguments

P, A and C are variables or constants, R is variable

### Usage

Calculates simple 16-bit checksum for the data block setting page P in GR8NET bank 1 beforehand, from address A with byte count C. If variable R is supplied, puts this checksum into this variable. If R is not supplied, prints checksum onto the screen.

Example: see NETLDBUF's MAPLOAD.BAS for implementation of this command

## NETGETMEM

Read 4 consecutive bytes (32-bit data) from the memory

### Format

CALL NETGETMEM (P, ADDR, [A,][B,][C,][D])

### Arguments

P is logical page number to be presented in GR8NET bank 1 (6000-7FFF) for access

ADDR is address of Z80 visible memory to access

A, B, C and D are variables receiving contents of the memory

### Usage

This statement loads variables A to D with consecutive memory values: A=(ADDR), B=(ADDR+1), C=(ADDR+2), D=(ADDR+3). Variables A to D will keep their original type, having contents in 0-255 range. If variable location is omitted, the memory location is skipped however *is being read*. The benefits of using this statement are (1) it replaces up to 4 PEEKs from general memory location, and (2) it gives access to GR8NET RAM buffer memory (as you can not access it with PEEK). Note that when command is running, CPU banks 1 and 2 (4000-BFFF) have GR8NET pages in them, thus you can not read RAM location occupied by BASIC program (in 8000-BFFF) – use PEEK operator instead.

## NETSETMEM

Write 4 consecutive bytes (32-bit data) into the memory

### Format

CALL NETSETMEM (P, ADDR, [A,][B,][C,][D])

### Arguments

P is logical page number to be presented in GR8NET bank 1 (6000-7FFF) for access

ADDR is address of Z80 visible memory to access

A, B, C and D are variables or constants to write to the memory

### Usage

This statement writes variables A to D with consecutive memory values: (ADDR)=A, (ADDR+1)=B, (ADDR+2)=C, (ADDR+3)=D. Variables A to D should be in 0-255 range of any type, otherwise *Overflow* error occurs. If variable location is omitted, the memory location is skipped and not written to. Note that when command is running, CPU banks 1 and 2 (4000-BFFF) have GR8NET pages in them, thus you can not write to RAM location occupied by BASIC program (in 8000-BFFF) – use POKE operator instead.

## NETGETMD

Get 32-bit double word from memory, convert and store the value into BASIC variable

### Format

CALL NETGETMD (P, ADDR, A)

### Arguments

P is logical page number to be presented in GR8NET bank 1 (6000-7FFF) for access

ADDR is address of Z80 visible memory to access

A is a variable of any numerical BASIC type able to accommodate the value

### Usage

This statement takes 4 bytes from the location pointed by P/ADDR, converts it to appropriate BASIC format and stores in variable A. Conversion is performed in software, MathPack is not used anymore because this functionality was removed. This command is very useful in couple with NETSDCRD to get sector of cluster number (which are 32-bit) into double-precision BASIC variable, which can hold such value. If 32-bit value is too big for the variable (e.g. storing 32768 into integer variable, identified by % sign or set by DEFINT) command will terminate with Overflow error.

### Example

` Imagine first FAT sector is loaded into location 1:6000, command takes cluster 2 number from FAT sector and puts it into V

```
DEFDBL V:CALL NETGETMD (1, &H6004, V)
```

` and then uses this variable to read first sector of cluster 2 into location 0:6000

```
CALL NETSDCRD (0, &H6000, V, 1, N)
```

` if command returns mismatching number of sectors it has read

```
IF N<>1 THEN PRINT "Read error"
```

## NETSETDM

Get value from BASIC variable, convert it to 32-bit double word and store it into memory

### Format

CALL NETSETDM (P, ADDR, A)

### Arguments

P is logical page number to be presented in GR8NET bank 1 (6000-7FFF) for access

ADDR is address of Z80 visible memory to access

A is a variable of any numerical BASIC type

### Usage

This statement converts A into 32-bit dword and stores it into location pointed by P/ADDR. If value is single or double precision floating point, fractional part is discarded. If overflow condition is detected ( $A > 2^{32}-1$ ) then *Overflow* error is given. Conversion is performed in software, MathPack is not used anymore because this functionality was removed. This command may be useful when working with SD-card's FAT file system to convert double-precision variable into 32-bit value to update FAT entry. See \_NETGETMD command.

### Examples

```
CALLNETSETDM (255,&H6800,120*8)
```

```
DEFDBLX:X=2^24-1:CALLNETSETDM (255,&HD000,X)
```

## NETSDCRD

Read sectors from the SD-card

### Format

CALL NETSDCRD (P, ADDR, S, N, O)

### Arguments

P is *target* logical page number to be presented in GR8NET bank 1 (6000-7FFF) for access

ADDR is *target* address of Z80 visible memory to copy read data to

S is BASIC variable of any numerical type, absolute sector number to read

N is number of sectors to read, maximum 8

O is number of sectors actually read. If O is not equal to N, check SD-card status registers

### Usage

This statement reads N number of sectors starting sector number S into SD-card buffers, and then moves data into the location pointed by ADDR switching page P in GR8NET bank 1 before the move. Important: target address space (pointed by ADDR as start and ADDR+N\*512-1 as an end) can only reside in areas 6000-7FFF and C000-FFFF. All other addressing space is not available.

### Examples

` The following command switches to logical page 0, but copies data into address &hC000 which is main memory

CALLNETSDCRD (0,&HC000,0,1,O):IF O<>1 THEN PRINT"Error"

` The following command switches to logical page 0, and copies data into its beginning (as address points into area of 6000-7FFF)

S=10\*80:CALLNETSDCRD (0,&H6000,S,8,O):IF O<>8 THEN PRINT"Error"

## NETBTOV

Move binary data from the GR8NET buffer to VRAM

### Format

CALL NETBTOV (P, ADDR)

### Arguments

P is an argument consisting of two fields: bit 0 is part of the VRAM to write to: 0=0000-FFFF, 1=10000-1FFFF, bits 8:1 identify GR8NET logical page data starts in

ADDR is VRAM address offset from the position indicated in binary header

### Usage

Binary image loaded into the buffer (e.g. by \_NETBLOAD) should have valid binary header per BASIC's BSAVE/BLOAD specification, otherwise *Illegal function call* error is generated.

### Examples

CALLNETBTOV                   ` image is located in logical page 0, load at VRAM address indicated in file's header

CALLNETBTOV(&h100)       ` load image with VRAM offset of 0100h

CALLNETBTOV(184\*2+1)   ` VRAM image is located in logical page 0B8h (in GR8NET ROM), and load this image into VRAM area starting 010000h



## NETBITOV

Move icon image from GR8NET buffer to VRAM

### Format

CALL NETBITOV (P, ADDR, VP, VADDR)

### Arguments

P is a logical page number of the start of the icon data;

ADDR is an address within 6000-7FFF window of the start of icon data;

VP is the video bank to write to (0=0000-FFFF, 1=10000-1FFFF)

VADDR is the address in the VRAM within the video bank.

### Usage

Icon must have special format: first byte is the width of the icon in bytes, second byte is height of the icon, followed by the data. When source address (ADDR) overflows to the next RAM logical page, logical page number is increased. When VRAM destination address (VADDR) overflows the video bank, pointer is reset to the beginning of the same bank.

### Examples

SCREEN 2:CALLBITOV(2,&H7F39,0,&H390)

If first byte of the icon is width 40 (multiply of 8), and second byte is height 6, then there will be a rectangular icon on the screen of 40x48 dots in size located with its left-top corner located in (24,144). To display rectangle properly in SCREEN 2 mode VADDR must also be a multiply of 8.

SCREEN 8:CALLBITOV(&H3E,&h6458,1,0)

If first byte of the icon is width 13, and second byte is height 11, then there will be a rectangular icon on the screen of 13x11 size located in the left top corner of the displayable area.

## NETGETCLK

Get clock speed of the specific source within GR8NET

### Format

CALL NETGETCLK (SF, BC)

### Arguments

SF is flag identifying the source being measured. If SF is returned zero, then MSX system bus clock is being measured; if non-zero, then GR8NET internal oscillator (typically 3.579545 MHz) is being measured;

BC receives clock frequency of the respective source – MSXBUS clock or internal GR8NET oscillator

### Usage

This command is used to identify the clock speeds within GR8NET and MSX system. Since version 0.9 it is not possible to change the source of audio clock – internal audio devices will always be clocked by the GR8NET internal oscillator.

### Examples

```
CALLNETGETCLK          100 CALLNETSETCLK(1):CALLNETGETCLK(,B)
Source: MSXBUS          110 IF B>3580000 THEN PRINT"Machine is overclocked"
Freq: 3579561 Hz
Ok
```

## NETSETCLK

Set clock source to measure speed of

### Format

CALL NETSETCLK (SF)

### Arguments

If SF is zero, GR8NET will measure clock speed of MSX system bus; if non-zero then GR8NET internal oscillator clock speed (typically 3.579545 MHz) will be measured.

Argument is mandatory

### Usage

Command will set the source for the measurement of the clock speed. Actual clock speed can be obtained by \_NETGETCLK command. Since version 0.9 it is not possible to change the source of audio clock – internal audio devices will always be clocked by the GR8NET internal oscillator.

### Examples

CALLNETSETCLK(1)

Ok

## NETVARRWTH

Set networking RX window threshold

### Format

CALL NETVARRWTH (G, N)

### Arguments

G is a variable receiving current value of threshold, default is 0

N is variable or constant setting threshold (0 to 2047)

### Usage

W5100 chip, unfortunately, is not designed to work in heavy loaded and switched environment, especially where packets may get massively lost, out-of order packets are being received, or spurious retransmissions occur which hinder normal and sequenced TCP/IP data packet flow. For technical information about issue please refer to the WIZnet support forum website at <http://wizwiki.net/forum/viewtopic.php?f=4&t=3670>.

The issue is that remote hosts/proxies may expect W5100 to behave in specific way, but W5100 does not satisfy expectations. To mitigate this issue there's threshold value used which makes \_NETBLOAD and \_NETPLAYWAV statements keeping threshold number of bytes in the W5100's RX buffer so that remote host/proxy did not have an option sending more than one data packet a time without W5100's acknowledgement, and thus performing retransmissions immediately as requested by W5100 without filling its buffer (which is not accepted by W5100) prior performing retransmission.

Default value of RX buffer threshold is 0, assuming whole RX window is used and as soon as at least one byte appears in RX buffer it is withdrawn by the firmware.

*By default* W5100 is having 2048 bytes buffer per socket, thus remote host/proxy, seeing W5100 reporting 2048 bytes window, *may* divide it to two 1024 byte areas, and send packets of 1024 data bytes long. This means that 2 packets fill W5100's RX socket buffer completely, thus, in such situations, to keep space for only one packet application, you may set the threshold to 1024.

Please note that limiting receiving buffer using RX buffer threshold may slow down transfer speed noticeably, but it increases reliability of the transfer in highly loaded and erroneous networking environments.

## NETVARUDTO

Set UDP packet timeout for DHCP and DNS operations

### Format

CALL NETVARUDTO (G, N)

### Arguments

G is a variable receiving current value of timeout and DHCP request retry count

N is variable or constant setting new timeout value and DHCP request retry count

### Usage

Both arguments are having same format:

- Bits [7:0] identify UDP timeout value (0-255), in 100 ms periods, default is 20 (2 sec)
- Bits [11:8] identify number of DHCP request retries attempted when GR8NET initializes, 0 means try once, 2 means try twice (thus maximum retries is 16), default is 0.

The UDP timeout controls the time system waits for incoming UDP packet to arrive for DHCP and DNS requests. Value indicates number of 100 ms periods, thus value of 20 means 2 seconds timeout; value of 0 means 256 periods (25.6 seconds). This command is instrumental if network is heavily loaded, DHCP server is relatively away from GR8NET or is just being slow in responding to the requests.

Increased number of retries could be instrumental if the router is too slow to start its port GR8NET is connected to, and there's no option to set the port into fast-start mode (e.g. using PortFast command for Cisco switches/routers). Using this setting you may have up to 16 DHCP query retries, with maximum of 25.6 seconds of timeout.

The values explained above are being preserved by \_NETSAVE command. If both arguments are present, G gets *previous* value, and then new value of N is set.

### Examples

CALLNETVARUDTO(G,256*2+50)	CALLNETVARUDTO(,50)	CALLNETVARUDTO(G)
?G	Ok	?G
532		20
Ok		Ok

## NETFKOPLL

Fake OPLL (MSX-Music) ROM into mapped RAM

### Format

CALL NETFKOPLL

### Usage

This command is targeted for the software which runs in GR8NET mapper modes 1-6 (game mapper), and can not run in mapper mode 8 when MSX-Music ROM is available in GR8NET subslot 3.

The ROM image, to be exact – specific signature in this ROM, is required for software to detect presence of MSX-Music chip (YM2413). \_NETFKOPLL checks if there's mapped RAM available and it has at least 48KB, and copies MSX-Music ROM from its flash chip into

---

mapped RAM page 2 (otherwise *Out of memory* error is given). Then, after machine reboot, MSX BIOS detects ROM image in RAM; applications will also detect presence of MSX-Music and find required signature.

However, if RAM mapper page 2 contents will get modified (for example, by loading application in MSX-DOS mode), machine may malfunction as BASIC "CALL" commands and file I/O may be calling corrupt locations.

In addition, this command overwrites mapped RAM page 2, and its original contents will be gone.

The primary purpose of this command is to allow games, running in GR8NET mapper modes 1-6 (e.g. Aleste) to produce FM-sound using GR8NET built-in YM2413 implementation. All other uses should be thoroughly considered for side-effects.

---

## 4. Using GR8NET as BASIC I/O device

You can use BASIC I/O commands to access remote resources over network or send datagrams using GR8NET. Important to know that if your I/O program is stuck in GR8NET waiting for input from remote host, you can press CTRL-STOP to abort the I/O operation, and BASIC will return with "Device I/O error".

Devices available for use are:

Device	Purpose	Arguments
TCP:	Open the socket to use TCP (transmission control protocol), in <i>client</i> mode. This means the socket will attempt actively connecting to remote resource to establish connection	Default URI structure values set by NETSETHOST (host name) and NETSETPORT (source port number)
TCS:	Open the socket to use TCP (transmission control protocol), in <i>server</i> mode. This means the socket will be set in listening mode waiting for incoming connection. To identify if connection is established use EOF function	Source port number to be listened to
UDP:	Open the socket to use UDP (user datagram protocol)	Source port number
IPRAW:	Open the socket to use IP protocol raw capabilities	Source port value's lowest 8 bits serve as <a href="#">IP raw protocol ID</a>
HTTP:	Open the socket to use TCP (transmission control protocol), in <i>client</i> mode, and send GET request to the remote resource. Header is being checked for HTTP response status code. If code is not 200 (OK) <i>Verify error</i> is generated	Full set of default URI structure values set by NETSETHOST, NETSETPORT, NETSETPATH, NETSETNAME and NETSETQSTR
HTTR:	<i>Raw</i> HTTP mode – same as previous but server response header should be read and processed by the application	

All devices listed above are **binary access devices**, capable of receiving any character through BASIC I/O statements and functions (according to those function specifications). These devices were ASCII character devices before Sep 2017, and it caused issues handling EOF character (code &H1A). There's no more need to catch EOF character through ON ERROR GOTO because *Input past end* error will no more happen when receiving EOF character.

There're three networking sockets, thus 3 (three) network BASIC files can be opened simultaneously. Trying opening more files than sockets will cause *Bad file number* error for fourth network file open request.

---

## 4.1. Using network device names in BASIC I/O

Device names in the table above must be used with BASIC file OPEN operator to request specific access to the networking resources, for example

OPEN"TCP0A:S\$\*"AS#1

Let's consider this example in detail

TCP	0	A	:	S\$	*
Device you are going to open, from the list of table above	GR8NET adapter identification (0-3). If omitted, default adapter is used (set by _NETSETDA)	(discontinued) Socket # to be used, kept for compatibility purposes. <b>See note 1 below</b>	Device name and file name separator	BASIC file name, a string, must NOT start with space. <b>See note 2 below</b>	Delayed data sending, no effect in UDP and IPRAW modes. <b>See note 3 below</b>

Notes:

1. Socket number.
  - There are three (3) sockets available for user, and they are shared between BASIC file I/O and TCP/IP UNAPI implementation. Now applications are not required to use explicit socket identification number (A or B) in device name, this socket number will be discarded, and actual socket location will be selected by the GR8NET firmware automatically. Thus, this change is backward compatible.
  - For firmware before Sep 2017 there were two sockets available for user, 0 ("A") and 1 ("B"). Each following device name had to be appended with the socket identification letter to identify the socket.
2. BASIC file name. Must be maximum 2 letters ending with \$ sign (as per MSX BASIC string variable specification). As in the example "S\$" itself is a string, but GR8NET firmware considers it as identifier of the BASIC variable holding URI to the remote resource to access. This string has no meaning if opening in TSC (listening – TCP server) and RAW modes.
3. Asterisk modifier. If it is not present, all TCP-based sockets (TCP, TCS, HTTP, HTTPR) do send characters to the network immediately as they are placed into the buffer (e.g. using PRINT#1); however if this sign is present, these TCP-based sockets will not immediately send packets with single characters, but accumulate data in the network buffer instead. When application is finished filling buffer, it calls \_NETSNDDGT, sending all accumulated data in one (or several) large packets. Using this mode application will have higher performance in its software I/O as well as from networking perspective. Note that normally TX buffer size of the socket is 2048 bytes, thus you should not place more than this amount of unsent data into the TX buffers without performing \_NETSNDDTG command.

Examples of the supplied string variable

S\$="http://www.gr8bit.ru:80" will cause DNS query to resolve www.gr8bit.ru

S\$="http://192.168.1.37:23" will use direct IP address without DNS query

S\$="http://www.myserver.com" will reuse remote port setting set by NETSETPORT

---

## 4.2. Identification of the network resource

When opening or accessing network resources *default URI structure* (or its parts) is reused as an input to the resource identification. The commands associated with setting (and respective getting commands) are:

\_NETSETHOST  
\_NETSETPORT  
\_NETSETPATH  
\_NETSETNAME  
\_NETSETQSTR

They set respective target URI parts, which then resemble full URI to the resource.

If you use string variable with OPEN operator as explained in previous section, defined parts of this variable will override values set in the default URI structure, in other words:

A\$="http://www.myserver.com:8080/mypath/myfile?myquery"

Will override all the values in default URI structure, *except source port*

A\$="http://:8080"

Will only override destination port

A\$="myfile"

Will only override file name

## 4.3. BASIC operators and functions to use for network access

You can use most of the BASIC I/O operators on the network resources. You must keep in mind that **network communication is sequential** and if one peer sends some data to another peer, latter peer will receive sent data in its buffer, and will have to read it sequentially to get access to further data.

All network connections share the same pool of sockets, thus ensure you close unneeded network files as soon as possible.

Command	Application
OPEN	Opens network socket, and connects to remote server (TCP, HTTP, HTTPR), sets to listen for incoming connections (TCS), or sets to receive or send datagrams (UDP and IPRAW). When device is being open, destination IP address, source port and IP protocol ID (for IP RAW) are automatically populated from the default URI structure modified by provided string variable in the device file name field. Open the network resource in binary mode, if you will try random access on it, BASIC will return "Sequential I/O only" error. If there will be any problem (e.g. hardware error, remote resource not found) execution will interrupt with Device I/O error and you can use _NETCODE statement to get cause of the error



Command	Application
PRINT#, PRINT #USING	Use these commands like you do on any other file, but be cautious: (a) at the end of each print without ";" BASIC prints CRLF. If you perform binary output, CRLF will corrupt your packet; (b) if you print numeric values, they may print with heading or trailing spaces.
LINEINPUT# INPUT#	Use these input commands with caution: they expect line to terminate with CRLF, however many internet resources use only LF as next line character, and command will terminate with "String too long" error, or being stuck at the end of the file waiting for CRLF.
INPUT\$	The same use as usual, but before requesting specific number of characters, ensure this number of characters is present in the RX buffer using LOC function, otherwise system may get stuck waiting for incoming character
CLOSE	Closes the network file, closes socket and removes file assignment with the socket
MAXFILES=	Important to know that invoking this BASIC feature is not only reallocates memory for I/O buffers, but it forcefully closes all files open before running it. Thus doing this command may interrupt your communication process. Please use it before you start communication ensuring you have needed file handles allocated for your software
LOC(f)	Function returns number of bytes waiting in the receive buffer
LOF(f)	Function returns number of bytes free in TX buffer
EOF(f)	In TCP modes, returns 0 (not EOF) if there's data in receive buffers (see LOC function), or there's no data but TCP connection is not closed. In UDP mode returns 0 (not EOF) if there's data received for the socket

You can load or merge programs from the internet onto your MSX PC:

Command	Application
LOAD MERGE	These operators will properly work <i>only</i> with HTTP: device, which parses header and feeds contents of the remote data to the BASIC. Remote BASIC file should be in ASCII format. Example: A\$="http://www.gr8bit.ru/software/basic/kove.asc" LOAD"HTTPA:A\$" After execution of the LOAD (successful or unsuccessful) variable A\$ is cleared.

#### 4.4. Sending datagrams or delayed TCP data

There're two modes network access can operate in:

- Immediate data send mode: this mode is used for all TCP-related connections by default (TCP, TCS, HTTP, HTTR). When new character is being output onto the network file, this single character is being sent in separate packet. It is clear that such mechanism, while provides real-time data communication, causes big overhead on the network, thus for the cases when application needs to accumulate characters

---

first and then send them at once, it uses \* asterisk modifier as explained in [Using network device names](#) chapter;

- Delayed data send mode: this mode is used for UDP and RAW connections, and also for TCP connections which are open with \* asterisk modifier. In this mode characters accumulate in the transmit buffer, and to send all characters accumulated application uses \_NETSNDDTG statement.

## **NETSNDDTG**

Send datagram / pending data to the remote host

### Format

CALL NETSNDDTG (F, A, B, C, D, RP)

### Arguments

F is BASIC file number, mandatory

A, B, C and D are remote device IP address octets, may be omitted

RP is remote device port for TCP/UDP, or IP protocol ID for IPRAW, may be omitted

### Usage

When network file is open (in any mode), its respective socket is associated with default URI structure's remote IP address, remote port, and IP RAW protocol ID (if not overridden by string variable as file name). Specifying A-D and/or RP parameter application can override destination datagram to be sent to, and IPRAW protocol being used.

After you put required data into the networking chip TX buffer using BASIC file I/O statements, you use \_NETSNDDTG command to send the datagram to the remote host.

### Usage

CALLNETSNDDTG(1)

` Send file #1 data to the preset IP/remote port

CALLNETSNDDTG(1, , , , , 23)

` In case file is open in UDP mode, sends file #1 data to preset IP address, but change port to 23 (telnet)

CALLNETSNDDTG(1,192,168,1,47,1)

` In case file is open in IPRAW mode, sends data written to the buffer using IP RAW protocol ID 1 (ICMP)

## **4.5. BASIC I/O using TCP devices**

TCP is based on persistent connection, and each byte you put into the output file is sent immediately, and as soon as there's free space in adapter's networking chip buffer, next character is being received from remote host. But you should be aware that remote hosts may have timeout settings, thus for reliable TCP communication you should use EOF function to check if remote host is still on the line.

Settings required for proper opening and operation:

- TCP: device – host name, remote port number, source port number
- TCS: device – source port number only, set by \_NETSETPORT. Note that source port number 0 (identifying automatic selection of the source port number) is not allowed, and application must change source port # from 0 to valid port number to be listened to using \_NETSETPORT.

Let's look at how communication takes place on the example. This simple program reads file from the remote server using HTTP protocol and displays it onto the screen:

```

10 CALLNETSETHOST("www.gr8bit.ru")
20 OPEN"TCP0:"AS#1
30 PRINT#1,"GET /software/roms/@license.txt HTTP/1.0"
40 PRINT#1,"HOST: www.gr8bit.ru":PRINT#1,""
50 IF LOC(1)<>0 THEN LINEINPUT#1,A$:PRINTA$
60 IF EOF(1) THEN PRINT:PRINT"RECEIVED":END
70 GOTO 50

```

Line 10 sets up the remote host, with firmware resolving the host name into its IP address. Line 20 opens stream in TCP mode, using adapter #0, in binary I/O mode. Line 30 and 40 print HTTP headers into the output stream, line 50 checks if there's anything in the input buffer, and performs line input from it. Line 60 checks if data exhausted, and ends if it is. Receive is performed in loop until there's no received data and remote host disconnects.

Now let's look to slightly performance-optimized TCP program:

```

10 A$="http://www.gr8bit.ru:80":OPEN"TCP0:A$*"AS#1
20 PRINT#1,"GET /software/roms/@license.txt HTTP/1.0"
30 PRINT#1,"HOST: www.gr8bit.ru":PRINT#1,""
40 CALLNETSNDTG(1)
50 IF LOC(1)<>0 THEN LINEINPUT#1,A$:PRINTA$
60 IF EOF(1) THEN PRINT:PRINT"RECEIVED":END
70 GOTO 50

```

Line 10 sets string variable, which will be parsed by OPEN statement into the remote IP address (using DNS query on the name) and destination port. Network file will be open in delayed-send mode (\* asterisk sign in the file name).

Lines 20 and 30 will put a number of characters into the transmit buffer, and line 40 will send all these characters altogether in single packet.

Please note that when using delayed-send mode, you must check space remaining in the transmit buffer using LOF function. Writing more data than transmit buffer will cause *Out of memory* BASIC error.

Function EOF returns true when there's no pending data to retrieve (by INPUT\$ for example) and the connection is closed. Thus it will return false when there's no data, but TCP connection is still established, in this situation input operators will wait for next byte from the remote host.

Function LOC returns number of bytes currently available to read from the receive buffer, and you can freely use INPUT\$(N,F) where N is a number of bytes, and F is file number; however you must keep in mind maximal string length within your environment not to cause *String too long* error. For example:

```

L=LOC(1):IF L>128 then L=128
A$=INPUT$(L,1)

```

---

## 4.6. BASIC I/O using UDP device

UDP is connectionless protocol, which assumes accumulation of set of data, and sending this data to predefined recipient in a single packet (or several packets with segmented data). UDP mode will always be delayed-send mode, thus to send data application will have to use NETSNDDTG statement explicitly.

Example: application wants to perform DNS query to the Google server 8.8.8.8. First, it opens connection in UDP mode, for example:

- A\$="http:// google-public-dns-a.google.com:53":OPEN"UDP:A\$" AS#1
- OPEN"UDP:" AS#1

First example will resolve host name to 8.8.8.8, and set remote port to 53, second example will use current setting on \_NETSETHOST/PORT, which has to be overridden in the code preceding OPEN command.

Next, application will PRINT#1 packet contents into the transmit buffer, minding free space in this buffer using LOF function.

And then application sends the data written to buffer using

```
CALLNETSNDDTG(1)
CALLNETSNDDTG(1,8,8,8,53)
```

Both lines will work properly for both OPEN commands mentioned above, but if application used second open command without A\$ as location modifier – and thus used default URI structure's host name/IP address and remote port number, first \_NETSNDDTG command will most probably send packet to wrong location, thus only second must be used.

Function EOF returns true if there's no received data, and if it is false, then function LOC is having non-zero value of the number of bytes received.

Datagrams received in the UDP mode are having special format, and you must ensure you read and check the header first before you get access to the actual data sent by the remote device. Header is having the following format:

0	1	2	3	4	5	6	7	Data
				HI	LO	HI	LO	
Sender IP address				Remote port		Data size		

First 4 bytes (0-3) of the header you read is sender's IP address. Bytes 4 and 5 represent remote port number datagram was sent from, and bytes 6 and 7 represent size of data in this datagram. After you read the data of *data size*, you must expect another header for next datagram.

Note that remote port and data size are **big-endian**, meaning that most significant byte goes first.

---

## 4.7. BASIC I/O using IPRAW device

IPRAW is the mechanism providing communication at the OSI layer 3 (transport level), one level below TCP and UDP. Transport level operates IP addresses, but does not have source/destination port abstraction of TCP and UDP.

IP packets define [types of protocol](#) their datagrams contain, for example – ICMP (Internet Control Message protocol) has ID 1, TCP (Transmission Control protocol) has 6 and UDP (User Datagram protocol) has 17. You see that ping, which operates using ICMP protocol, can be sent using IPRAW mode. You can not send ping request and receive response to it using TCP or UDP. Thus implementation of IPRAW provides greater networking opportunities than TCP and UDP provide at their higher, 4<sup>th</sup> OSI layer.

Opening the IPRAW network file is very similar to the opening UDP network file, however remote port number's least significant byte will be treated as IP protocol ID, for example:

```
A$="http://www.gr8bit.ru:1":OPEN"IPRAW:A$"AS#1
```

will set target IP address to resolved of www.gr8bit.ru, and set IP protocol ID to 1. (It will also set TCP/UDP remote port number to 1 within networking chip, but it does not make sense as soon as socket is being open in IPRAW mode).

In case of IPRAW, \_NETSNDDTG's last argument is treated as protocol ID rather than communication port number:

```
_NETSNDDTG(1,192,168,1,37,1)
```

where last argument 1 is IP protocol ID.

There's also a small difference from the UDP when receiving data. In IPRAW mode remote port number is missing, making up the header of only 6 bytes:

0	1	2	3	4	5	Data
				HI	LO	
Sender IP address				Data size		

Also note that data size is stored in **big-endian** format.

---

## 4.8. BASIC I/O using HTTP/HTTR devices

These device modes are very similar to the TCP device, and the difference is that HTTP (HyperText Transfer protocol) is used above the TCP protocol. It means that in addition to the connection to remote network device, HTTP/HTTR device will perform remote resource request operations.

After successful connection using transmission control protocol (TCP), device sends HTTP GET request to remote device using default values set by `_NETSETHOST`, `_NETSETPATH`, `_NETSETNAME` and `_NETSETQSTR`. `_NETSETHOST` is used in "Host:" HTTP header, and `_NETSETPATH/_NETSETNAME/_NETSETQSTR` are used to make up URI of the GET request.

These default values can be overridden with supplying string variable as a file name when opening the device, e.g.

```
10 A$="http://www.gr8bit.ru:80/software/roms/@license.txt"  
20 OPEN "HTTP:A$" AS #1
```

will cause the following HTTP request sent to the remote server

```
GET /software/roms/@license.txt HTTP/1.0  
Host: www.gr8bit.ru  
...
```

Remote server will reply to this request with the HTTP response, and data if appropriate. And here's the difference between HTTP: and HTTR: devices:

- HTTP: OPEN statement, as part of its execution, will read response header, parse it for the HTTP response code and open BASIC file if code returned is 200. Therefore BASIC program will start reading remote file data immediately;
- HTTR: OPEN statement will just send request, and successfully finish; response header reading and parsing is left to the application.

HTTP device is convenient for gaining quick access to the data, HTTR device provides more flexibility in application parsing response headers and thus acting in some custom way.

---

## 4.9. The URL parser

GR8NET CALL and file I/O commands work using URI structures, and as explained in the subchapters above it is possible to supersede existing fields of the default URI structure by providing URL string. This string will be parsed into the fields, and respective transient URI structure's fields will be updated. All fields which are not present in the URL string will be kept intact in accordance with default URI structure.

The following URL formats are valid and meaningful:

Value	Meaning
file.dat	New file name only
/file.dat	New file name located in the root
/path/	New path from the root
path/	Path to append to current path
http://host	New host name
http://host/	New host with new root path
http://host/file	New host name with file in the root
http://host/path/	New host with new path
http://host:port	New host with new destination port
http://host:port/path/file.dat	New host with new port, new path and new file

The list above is not exhaustive, it just shows you some important examples of network URLs you can use, and what they will mean and change for OPEN, LOAD and MERGE statement. Parser will not update *source* port number, please use \_NETSETPORT command to set it.



## 5. Built-in web browser

There's simple web browser built into the GR8NET firmware, its primary purpose at the time or initial release is to let user conveniently browsing directory listings provided by the web servers and SD-card. Browser is capable of displaying relatively small text files. To invoke browser use \_NETBROWSE command in BASIC.

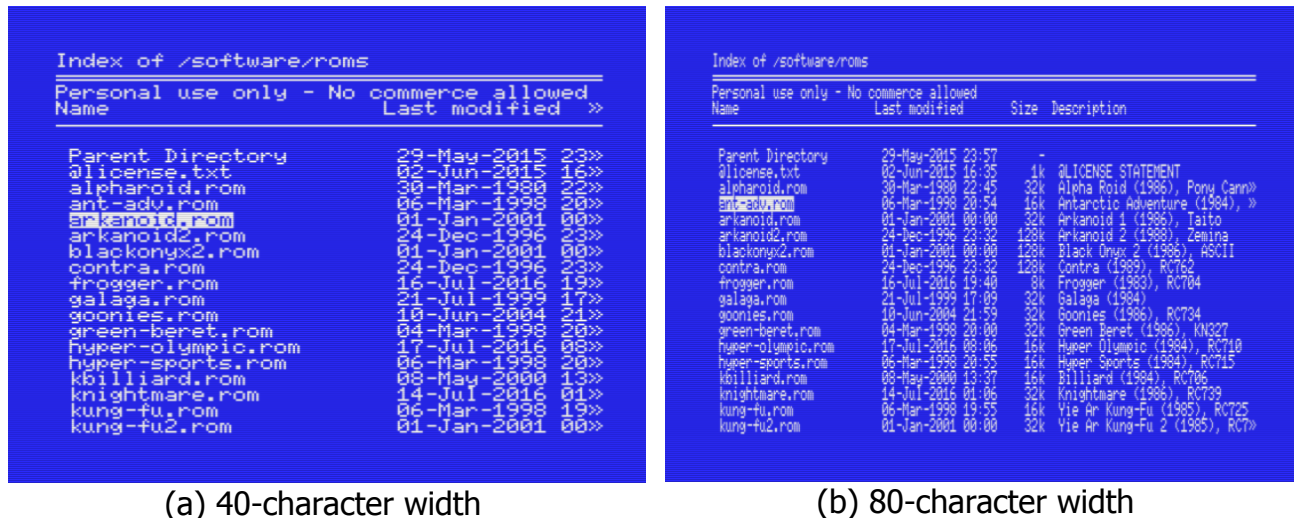


Figure 9. Browser screens

Figure 9 shows 40- and 80-character display of the browser. If your machine has V9938 and above we recommend using 80-character display mode, however if you have MSX1 machine you will be anyway able using the browser in 40-character screen mode. To choose the mode, set SCREEN0's screen width using WIDTH command before launching the browser.

When starting browser, it checks for SD-card being installed, and if SD-card is in the slot, it prompts for the source to browse (fig. 9c). Please do not make files with names longer than 28 characters. Supported SD-card file systems are FAT32 and FAT16.

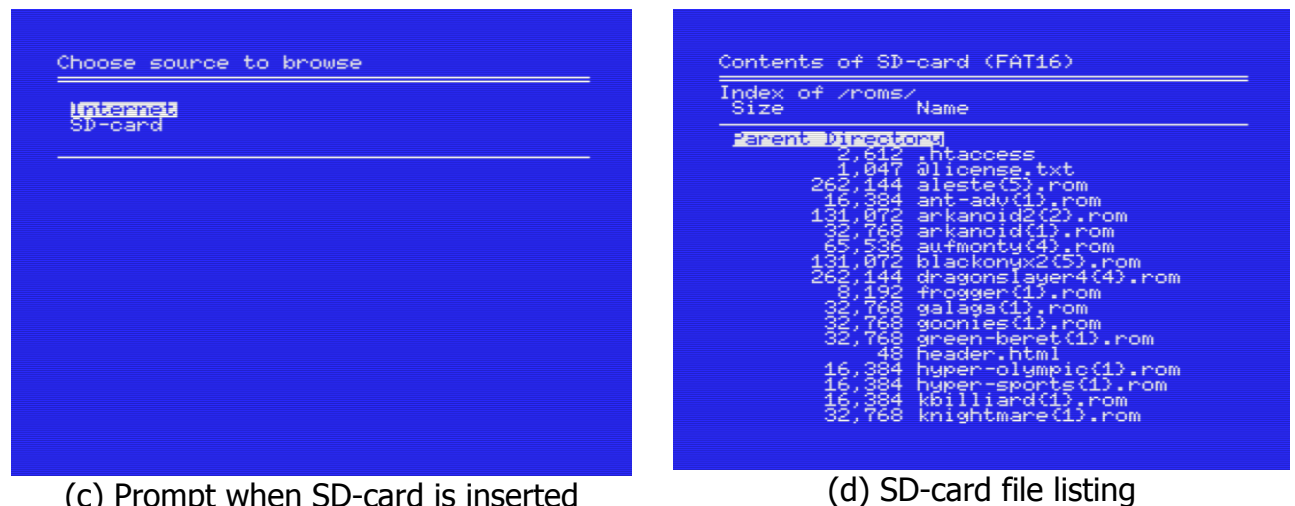

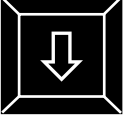
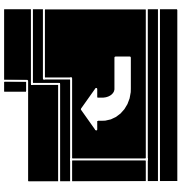
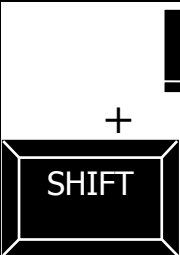
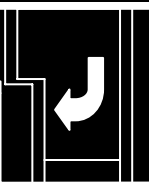





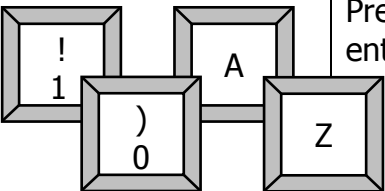
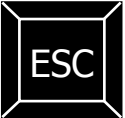
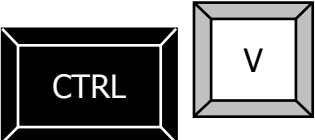


Figure 9 (continued). Browser screens

Navigation is performed by the following keyboard keys:

Key	Function
 Arrow up  Arrow down	Using these keys you navigate the web page. If next or previous link is visible, focus will change to that link, if not, page will be scrolled in required direction – up or down
 ENTER key	This key selects the link for activation. If you select text file or subdirectory their content will be loaded into the browser. If you select binary file, it will be loaded and executed. If execution will return back to browser, browser will reload directory listing file was selected in. If you select ROM image, it will be loaded, and if ROM image's file name contains special digits in {} braces, then ROM will be launched using specified mapper type, otherwise GR8NET will try to identify ROM mapper type and launch it, prompting to confirm correct identification
 + 	This combination of keys will override {} setting of the file name of the ROM image (see Enter key section above) and GR8NET will try to identify ROM mapper type.
 Space key	Space key acts the same way as Enter key, except images will not be executed, and after load browser terminates. ROM images are loaded into GR8NET buffer RAM, binary files are being transferred into their required location, and message will be given about binary image start and execution addresses. <i>This behavior can be modified by special flags within second argument to the command</i>
 Backspace	This key allows going back <i>one</i> web page. If you opened text file which does not have links at all and you want to return to directory listing without re-launching the browser, click Backspace key.
 TAB	Pressing TAB key on the entry in the list which presents wave file will cause browser to invoke built-in WAV player (for regular FPGA image) or MP3 player (for MP3 player FPGA image) and play this file. After playback completes, or user interrupts the playback, system will return to the browser
 CLS HOME	Move to the top or bottom of the page. If top of page is displayed and first link is on focus, moves view to the bottom of the page. If bottom or other part of the page is displayed, moves view to the top of the page.

Key	Function
	Refresh the page (reload and display)
	Pressing alphanumeric keys will move focus to the next linked entry with its description (link text) starting with the respective character pressed. Case of alphabetical character does not matter.
	Exit web browser
	Play video file from the SD-card. After playback is finished, execution returns to the browser.

As mentioned in Enter key section, browser can run ROM automatically if file name of the resource is having special section – braces {} with maximum 3 characters in them:

- First character must be present, it selects mapper type from 1 to 6, e.g. {3} to start ROM in Konami SCC mode;
- Second character may be omitted if third character is not present, and identifies if GR8NET will restart in composite mapper mode having GR8NET, mapped RAM and Nextor in expanded slot. Value can be 0 or 1. Example: {31} will cause GR8NET to restart in mapper mode  $1 \cdot 8 + 3$ , thus in mapper mode 11;
- Third character may be omitted, and identifies if composite mapper mode will have mapped RAM disabled (nothing will be present in subslot 1); values can be only 0 or 1. Example: {311} will cause restart with mapper mode 11, and no mapped RAM in subslot 1.

Web browser has several limitations; please keep them in mind when creating web pages/directory listings to be accessible by GR8NET:

- While file names allow 63 characters, keep file names limited to 23 characters, otherwise they may be truncated by the display;
- Do not use double quotes " within file names. This character is reserved for identification of hypertext reference boundaries during HTML document parsing;
- Create comprehensive descriptions for files and directories in the web server index pages – these descriptions will help understand the purpose of the file and will be very useful in 80-character mode view;
- Ensure number of links per page is not more than 250 – including header's *Name*, *Last modified*, *Size*, *Description* and *Parent directory* means limit number of entries (files and subdirectories) up to 245 maximum;
- Images are not displayed and discarded;

- Please use HTML checking tool in case you create custom web pages to ensure tag consistency;
- Some HTML constructions are not supported at all, some characters are considered as bad characters. In this case browser will display *stream error* and terminate.

Browser supports simple directory listing format provided by web servers, and supports so called *fancyindexing* view. Below is an example of the .htaccess configuration file for Apache:

```
HeaderName header.html
IndexIgnore *.html
IndexOptions FancyIndexing SuppressIcon FoldersFirst
Options +Indexes
IndexOptions DescriptionWidth=*
Adddescription "@LICENSE STATEMENT" @license.txt
Adddescription "Sample WAV files for GR8NET testing" audio
Adddescription "Sample BASIC programs" basic
Adddescription "Bloable executables" binaries
Adddescription "Firmware update and troubleshooting tools" firmware
Adddescription "ROM images for GR8NET testing" roms
Adddescription "Diskette images" bootimg
Adddescription "MSX images to load to VRAM" images
Adddescription "MSX videos, software to create MSX videos" video
```

Do not forget that settings in .htaccess are automatically inherited by the subdirectories if not overridden in the hierarchy, thus if you add description to specific file on one level then if there's file with same name down the subdirectory tree the defined description will be displayed if not overridden by *Adddescription* in the directory containing the file under consideration.

## NETBROWSE

Invoke internet and SD-card browser in BASIC

### Format

CALL NETBROWSE [(I\$, F)]

### Arguments

I\$ is string variable or constant used as source URL to start browsing with. Command re-uses default structure set by \_NETSETHOST, \_NETSETPATH, \_NETSETNAME and \_NETSETPORT, thus take special care to put as much information into I\$ as possible: host name, destination port, path and name (or make sure appropriate values are set by abovementioned commands before you run browser). Please refer to [The URL parser](#) chapter;



If you invoke browser with intent to browse internet, but are brought to browsing SD-card, ensure that URI structure and URL string you use as input is network-type (contains *http://* in its definition).

F is an integer representing a set of bitmap flags. F can be a constant, expression, formula or variable. In case F is an integer *variable*, several status bits will be returned in it. If this argument is omitted, it is assumed of value 0.

7	6	5	4	3	2	1	0
ESCF (output)	DIR (output)			NOSEL (input)	DIRENA (input)	NOLOAD (input)	SPCMV (input)

- SPCMV: if this bit is set, when user presses SPACE key on the selection, target will be loaded into the GR8NET RAM, but no action will be taken on it (e.g. binary file with header will not be moved into its designated location), and browser exits;
- NOLOAD: if this bit is set, browser will not load selected target into GR8NET RAM. This bit makes sense only when SPCMV bit is set. Application can use \_NETVARBRSTR command to get and process selection and load it afterwards;
- DIRENA: if this bit is set, pressing SPACE key on directory will select directory and exit; if this bit is not set pressing SPACE key on directory will cause content load and continuation of browsing. This option has effect only when SPCMV is set;
- NOSEL: if set, forces no source device selection page (Internet/SD-card), browsing will proceed directly to device identified by URI structure/URL string (http:// or sdc://);
- ESCF: this bit is set when browser exits having user pressed ESC key. In this case output of NETVARBRSTR will point to last directory user have been viewing;
- DIR: on completion of browser process, this bit is set if contents loaded or pointed to is not a file entry, but directory entry. For network operations, contents should be a web page with directory list generated by web server, or any other HTML text returned by web server for the directory; for SD-card operation, loaded content will be operating system image of the directory.

#### Examples

CALLNETBROWSE	Start browsing using default URI structure set by _NETSETHOST, _NETSETPATH and _NETSETPORT
CALLNETBROWSE ("http://myserver")	Start browsing server <i>myserver</i> with path set by _NETSETPATH and port set by _NETSETPORT
CALLNETBROWSE ("http://myserver:80/mypath/")	Start browsing server <i>myserver</i> with path <i>/mypath/</i> (trailing slash is mandatory to distinguish path and name)
CALLNETBROWSE ("sdc:///roms/konami/", 8)	Start browsing SD-card in <i>/roms/konami/subdirectory</i> without having source selection web page displayed
CALLNETBROWSE ("sdc:///roms/konami/")	This command will first bring Internet/SD-card selection screen, and whatever selection you will make, proceed to browsing SD-card
F=8+4+2+1 CALLNETBROWSE(A\$, F) IF F AND 128=128 THEN PRINT "ESC pressed" ELSE CALLNETVARBRSTR(O\$):PRINT O\$	Starts browsing location pointed by A\$, without source selection screen, with SPACE key selecting file or subdirectory entry and not loading its contents.

---

## NETVARBRSTR

Get URL string of the location selected by user within the browser

### Format

CALL NETVARBRSTR (O\$)

### Arguments

O\$ is string variable receiving UR string corresponding to user selection by Enter or Space key. If browsing was terminated by ESC key (ESCF flag is set), O\$ will identify directory user has chosen to press ESC key in to terminate browsing.

Important: if resulting full URL string will not fit into string variable allowed by BASIC (due to its size being longer than 254 characters or string variable area depletes), *String too long* error will be generated. To ensure smooth execution of BASIC program, ensure you handle this possible error through ON ERROR statement.

### Usage

Use this command immediately following \_NETBROWSE command completion. Other commands may use same memory location for storing URL/URI information thus may modify content and returned URL string may be invalid.

---

## 5.1. Opening SD-card located file in the browser

Browser was designed for the purpose displaying contents provided by the web server or browsing contents of the SD-card as directory tree. When entering browser, input URI location is parsed and file name is removed in order for URI to point to directory location. For loading and executing files from the BASIC programs and BASIC command prompt \_NETBLOAD command should be used.

However there're could be cases when it is required to open file in the browser – as text or HTML – for example, showing readme file contents or provide custom listing of the SD-card resources using HTML index files. Using command

```
CALLNETBROWSE("sdc:///mypath/myfile.html",8)
```

will open contents of the subdirectory *mypath* (as file name is removed). There's a hack available for SD-card browsing presenting file as directory, e.g.

```
CALLNETBROWSE("sdc:///mypath/myfile.html/",8)
```

when after URI parsing directory path is */mypath/myfile.html/*. Adding trailing slash to the file name will cause parser to treat file name as part of the path, and will seek to the start of the file (not knowing that it is file and not subdirectory). Then, when opening the contents of the *myfile.html*, browser identifies that it is not a subdirectory, but a HTML file, and displays it appropriately.

The following rules should be followed:

- File must exist of the SD-card;
- HTML file must be properly formatted – it should start with *!DOCTYPE* or *HTML* directive, otherwise it will be displayed as text file. For the example see source of any directory listing, e.g. <http://www.gr8bit.ru/software/firmware/GR8NET/> (right click into the page display area and select "View page source");
- All anchors in HTML file should be absolute, and all file URIs to be displayed in the browser must have trailing slash in their URI strings;
- Second parameter 8 (NOSEL bit set) is required to suppress display of browsing source selection and browsing from the SD-card root.



---

## 6. Using integrated MSX-DOS

**Disclaimer:** you agree that AGE Labs/Eugeny Brychkov (GR8NET device), and Nestor Soriano (Nextor software) are not liable, and can not be held liable for the adverse effects of managing data contents using GR8NET Disk subsystem and Nextor built into GR8NET device, including, but not limited to data loss, data corruption, data unavailability, storage device format consistency. You should perform regular backups of the storage media contents, and, in case of proven storage system malfunction, report it, along with the plan to reproduce malfunction, to abovementioned developers. Abovementioned developers do not perform restoration of the damaged volumes and related data, and do not guarantee that there will be updates or fixes available for reported, known or unknown issues with the GR8NET built-in storage systems.

Disk subsystems described below are independent, can run separately, or, if Nextor is started, Nextor will take over control of GR8NET disk subsystem (if this subsystem is activated).

### 6.1. GR8NET Disk subsystem (DOS1)

Since February 2016 GR8NET is having integrated Disk-BASIC in its ROM logical pages 88h-89h (16 Kbytes). *This disk subsystem* is available in mapper 0 (GR8NET) and composite mappers 8-14. Use this disk subsystem in mapper modes 9-14 with caution because GR8NET RAM is being shared between game mappers, and loading disk image may corrupt ROM image in game mapper.

This implementation of DOS version 1 supports all the disk I/O functionality, its ROM is switching in the CPU bank 1 (4000-7FFF), thus there should be no functional difference between regular DOS1 implementation and GR8NET's one.



**WARNING!** It is not allowed for applications to call Disk-ROM directly. Applications should use standard interfaces like BDOS or MSX BIOS entries and hooks related to management and control of the standard storage devices.

Because of control transfer between GR8NET pages and CPU page switching, integrated Disk-ROM has slightly lower performance than original Disk-ROM. If original performance is needed, please do one of the following:

- Install MSX-DOS version 2 cartridge into the system, this cartridge will take control over integrated MSX-DOS version 1 BIOS and use GR8NET as an end storage device;
- Install another storage device in slot ID lower than that GR8NET installed into. This will make this another device as a master, and it will use GR8NET as an end storage device.

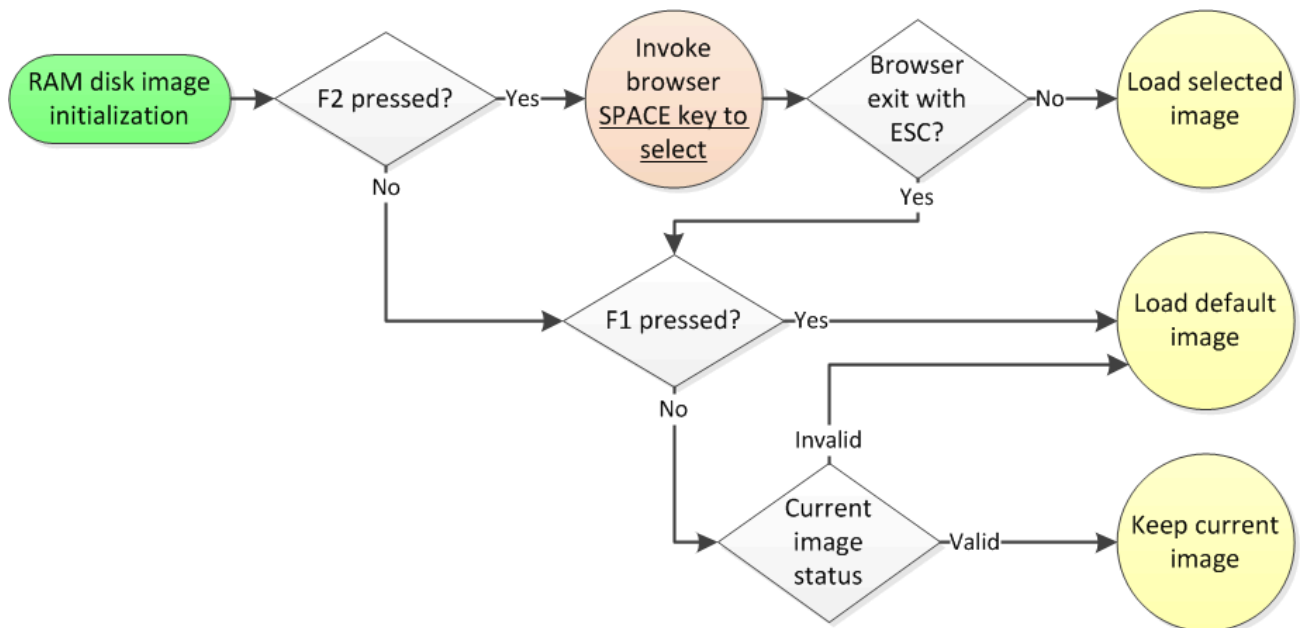
GR8NET disk subsystem initializes after GR8NET networking initialization finishes. Disk and networking work independently, thus it is possible to have Disk-ROM in the GR8NET RAM as a RAM-disk with no valid networking connection. Source for RAM disk image could be an SD-card. If RAM disk is uninitialized, then you will need to format it before using.

### 6.1.1. Initialization of the DOS1 disk subsystem

First step of the disk subsystem initialization is memory allocation for the RAM disk; please refer to [Memory manager](#) chapter. Then GR8NET is checking the GR8NET RAM for valid diskette image. This operation is performed even if disk-ROM is disabled, and gives you opportunity to enable the disk subsystem using F4 key to use available diskette image. If valid disk image is detected, *Valid disk image in RAM* message is displayed.



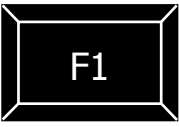

If you pressed and held F4 key during GR8NET adapter initialization, or disk subsystem was already enabled before GR8NET initialization (e.g. after warm boot), then the following workflow applies:

1. Space is allocated for the disk image: 720KB (90 logical pages) in GR8NET mapper mode 0, and 360K (45 logical pages) or 256KB (32 logical pages) in GR8NET mapper modes 8-14 (depending on the mapped RAM configuration and mapper type);
2. Firmware displays *Disk subsystem* message and waits for 3 seconds, displaying dots on the screen. You have an opportunity to press and hold modifier keys (F3 to disable disk ROM, F1 to refresh image, F2 to invoke browser) during this period;
3. If Disk-ROM is still enabled, then, depending on the keys being pressed, actions are taken according to the image below.



4. If load of remote image was successful or there was valid image identified before, firmware checks if image fits into the RAM allocated for the image. If it does not, displays warning message *Warn: disk image will not fit*, and then *Disk image is mounted* message;
5. GR8NET firmware starts initialization of the disk subsystem.

You have an opportunity to change behavior of the disk subsystem initialization procedure by pressing and holding the following keys during GR8NET initialization:

Key	Behavior
	<i>Disable disk-ROM.</i> By pressing this key you disable disk-ROM and prevent new image load. This key has immediate effect – as soon as you press it <b>during dot display process</b> , further disk-related operations are aborted
	<i>Force disk-ROM enable.</i> In case disk-ROM is disabled in the adapter's mode register, by <b>pressing and holding this key when GR8NET finishes initialization of its networking subsystem</b> will cause override of disable bit, and forces GR8NET to perform disk-ROM initialization
	<i>Force image reload.</i> By pressing this key you instruct to reload default image even if valid image is present in the RAM. If F2 is pressed at the same time, you go to browser, and default image is reloaded only if browser is exited with ESC key
	<i>Invoke browser.</i> If you do not want default disk image to be loaded you can press this key and invoke browser to browse the directory pointed by the default image URI structure. Example: default URI is <a href="http://www.gr8bit.ru/software/bootimg/bootimg0.dsk">http://www.gr8bit.ru/software/bootimg/bootimg0.dsk</a> , by pressing F2 you will be brought to browsing of <a href="http://www.gr8bit.ru/software/bootimg/">http://www.gr8bit.ru/software/bootimg/</a> directory. Please note that <u>to have disk image loaded properly by the browser you should press Space key in the browser, not Enter key.</u>

After initialization is finished, GR8NET in its Disk-ROM part functions as any other standard storage controller:

- It tries to load MSXDOS.SYS and then COMMAND.COM to boot into the disk operating system mode;
- If DOS mode is unsuccessful, it tries loading AUTOEXEC.BAS and executing it;
- If both previous steps are unsuccessful, machine goes to MSX BASIC. You can easily identify if GR8NET is mastering the disk operations by the string it displays when BASIC is entered.

---

### 6.1.2. Using GR8NET DOS1 disk subsystem

Standard Disk-BASIC commands and standard BDOS function are supported by the disk subsystem.

By default disk-ROM initializes with single logical drive (e.g. A:) for RAM disk. It is possible to force two-drive configuration by putting number of drives in brackets into *file name* of the image being loaded. For example, mydiskimg{2}.dsk.

You have the following commands to manage GR8NET-specific implementation of the disk subsystem and its driver:

#### DSKGETIMG

Get current disk image location

##### Format

CALL DSKGETIMG [(P\$)]

##### Arguments

P\$ is a string variable. If P\$ is empty, image was not loaded or invalidated

##### Example

CALL DSKGETIMG

http://www.gr8bit.ru:80/software/bootimg/diskimg0.dsk

Ok

#### DSKSETIMG

Set disk image location

##### Format

CALL DSKSETIMG [(P\$)]

##### Arguments

P\$ is a string variable or constant

##### Usage

If argument is supplied, statement sets new image location to be loaded from, but does not start image loading. Logically, this command is followed by \_DSKLDIMG command to load the image from new location.

If argument is omitted, raises disk change flag forcing Disk subsystem re-read RAM disk's control structures.

{2} in the file name of the image will force two drive configuration, (e.g. A: and B:), where drive A: will be a RAM disk, but drive B: will always return *Not ready* error.

**Note:** during URI parsing default URI structure is used, thus to avoid dependence of default URI setting define all fields in P\$: server name, remote port number, path, and file name.

Note: \_NETSAVE command preserves current remote disk URI information, and preserved URI will be used as a boot location on next GR8NET initialization.

##### Examples

CALL DSKSETIMG("http://www.gr8bit.ru:80/software/bootimg/diskimg0.dsk")

CALL DSKSETIMG("sdc:///disks/laydock.dsk")

CALL DSKSETIMG("sdc:///disks/dsbest{2}.dsk")

## DSKLDIMG

Loads image into the RAM disk area in GR8NET buffer RAM

### Format

CALL DSKLDIMG

### **Important notice**

(Re-) loading disk image will overwrite current RAM-disk's image present in the RAM, and all current information will be lost. Before loading new image ensure you backed required data up from current image of RAM disk onto another device (e.g. using \_DSKSVIMG, using floppy disk, hard disk, SD-card).

### Usage

Statement uses previously set disk image location loading it into the RAM. Location to load data into is defined by DSKLPG variable (see [memory management](#) chapter). If size of image being loaded is bigger than space allocated for RAM disk, *Device I/O error* is given. If disk configuration (i.e. number of sectors in its maximal size) does not fit into space allocated for RAM disk, image will be loaded successfully, but for all sectors above space available for RAM disk subsystem will return *Not ready* error. If HTTP return code is not successful (not 200), *Verify error* is given.

## DSKSVIMG

Saves RAM-disk image onto SD-card

### Format

CALL DSKSVIMG(P\$)

### Argument

P\$ is string constant or variable, if omitted location set by \_DSKSETIMG is used

### Usage

This command is not designed to be used programmatically, please use it from command prompt.

Target file on SD-card must be already present, with matching size of the image in the GR8NET RAM. For example, if you have 360K disk image loaded into GR8NET RAM, it can only be saved into already existing 360K file located on SD-card.

In case of errors, command will return *Illegal function call* error in BASIC, and further information can be obtained by \_NETCODE command. Possible causes: SD-card not ready (install operational SD-card), operation not supported (supplied URI must be SD-card SDC:// device, target file size should match image size), URI resource not found (point to existing file), SD-card I/O error (there's physical read/write card error), Invalid SD-card FS (in case there're problems with file system).

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft

GR8NET Disk BASIC version 1.0
Ok
call dsksvimg("sdc:///laydock.dsk")
DSK image saver
Target: [laydock.dsk]
Warning: RAM-disk image
Image sectors: 65535
RAM-disk space: 1440
Required size: 737280/737280

Ready to save image to SD-card
Type two letters SY to continue SY
Saving...
Ok
■

color auto goto list run
```

Example to the left shows command execution flow, please notice that:

- There's warning displayed, indicating that there's some issue with image of RAM-disk;
- *Image sectors* does not match *RAM-disk space*, another indication that there's something wrong with image in RAM.

In order to proceed with saving the data, you must press keys S and Y sequentially, otherwise command will abort.

### **Important notice**

Regularly back your SD-cards up. This command \_DSKSVIMG writes to the SD-card, and AGE Labs/Eugeny Brychkov will not be held responsible for SD-card data integrity, usability and validity.

### **Preparing SD-card for disk image saving**

This task is as easy as:

- Choose correct size of the target file – 720K, 360K or 256K;
- Locate existing, or create new .DSK image using diskette image creator or emulator. Empty images can be found here <http://www.gr8bit.ru/software/bootimg/empty-images/>;
- Copy this file to SD-card using Nextor or another platform (Linux/Windows/Mobile OS).

## **DSKCFG**

Obtain or manage state of disk image

### Format

CALL DSKCFG (MS, SETS)

### Arguments

MS is a variable receiving maximal number of logical pages RAM disk can be set to;

SETS is variable or constant setting size of RAM disk, should be  $0 \leq \text{SETS} \leq \text{value of MS}$

### Usage

Each GR8NET mapper mode has maximal size of RAM disk preset, you can not create RAM disk of more than MS pages in size. Thus SETS should be  $\leq \text{MS}$ . Please refer to [Memory manager](#) chapter.

When SETS argument is present, disk subsystem re-allocates RAM disk structures so that there's required number of pages between RAMMAX and DSKLPG memory manager's variables. This action will invalidate image location (managed by \_DSKSETIMG), and will set disk change flag for RAM disk. It is advised to format RAM disk using \_DSKFMT command after resizing it because if disk image start changes RAM disk's control structures will appear invalid, and image will not be handled properly. If SETS is 0, then RAM disk's size is 0, and for any read or write operation system will respond with *Not ready* error.

---

When re-allocating RAM disk space:



- If RAMTOP value before re-allocation is equal to DSKLPG value, top of RAM will be aligned to the new start of disk image start page. In other words, if there's nothing else being allocated by the GR8NET system below disk image, RAMTOP frees space according to DSKLPG change. If RAMTOP value before re-allocation is not equal to DSKLPG, it remains unchanged and no memory size below this page changes.
- Same holds true to UPRAMS variable (user protected RAM area start), but, unlike system-managed RAMTOP, this UPRAMS variable can be changed by \_NETSETMMV statement to be equal to RAMTOP, and thus, if RAMTOP is equal to DSKLPG, full space below new DSKLPG becomes available for user and \_NETBLOAD/\_NETBROWSE operations.

## DSKFMT

Initialize RAM-disk image

Format

CALL DSKFMT

### **Important notice**

Formatting RAM-disk will wipe data from it. Before performing format, ensure you backed required data up from current image of RAM disk onto another device (e.g. using \_DSKSVIMG, floppy disk, hard disk, SD-card).

Usage

This statement formats RAM disk. There're two formats available, chosen automatically:

- If RAM disk size is between 46 and 90 logical pages (368K-720K), disk is formatted as 720K disk, with missing clusters marked as bad (0FF7h) in FATs;
- If RAM disk size is between 1-45 logical pages (8K-360K), disk is formatted as 360K disk, with missing clusters marked as bad (0FF7h) in FATs;
- If RAM disk size is 0, returns *Out of memory* error.

## FORMAT

Initialize RAM-disk image: **not available for RAM disk**

Format

CALL FORMAT

Usage

**You can not** format RAM disk with this standard Disk-BASIC command. Please use \_DSKFMT statement. It is designed this way so that you do not mistakenly format real floppy or hard drive.



## DSKSTAT

Get/set state of the disk subsystem

### Format

CALL DSKSTATE (T, S)

### Arguments

T is variable or constant, and if 0, disk system is set as to be disabled, if T is 1 it is set as enabled;

S is variable getting bitmap of the disk subsystem status

### Usage

Changing T (status of disk system) will take effect on next warm boot. Hard reset or power cycle will force reloading of the default state of disk subsystem. After changing the status, you can use \_NETSAVE command to update configuration page in flash chip and make setting effective after cold start.

Variable S gets the following bitmap:

Bit	Description
7-4	Reserved
3	Set if disk image in its maximal configuration does not fit RAM disk reserved area
2	Set if DSKCHG (disk change flag) is raised
1	Set if image is mounted and is being used by the disk subsystem
0	Set if disk subsystem is enabled

## 6.2. Nextor disk subsystem

Since December 2016 GR8NET is having Nextor disk subsystem in composite mapper modes 8-14. The kernel is located in the subslot 2 of the expanded slot. Note that expansion is performed by GR8NET adapter, and thus adapter has to be installed in the primary slot. To have Nextor operational both FPGA and flash chip firmware *must be updated*, please refer to chapters [Updating GR8NET firmware](#) and [Updating FPGA firmware](#).

Nextor supports FAT12 and FAT16 file systems, and will properly work with image loaded by GR8NET DOS1 disk system (see [GR8NET Disk subsystem \(DOS1\)](#)), as well as with storage located on the SD-card.

SD-card format should be:

- Partition-less, containing volume without MBR (master boot record), starting from boot sector. Windows OS usually formats SD-cards of size less or equal to 1GB this way. When formatting you should ensure FAT file system option is selected (FAT32 or ex-FAT options must not be used);
- Partitioned cards, having MBR. Windows OS usually formats cards of more than 1GB size this way. Nextor will automatically mount only first partition;
- Partitioned by the built-in \_FDISK utility of the Nextor.

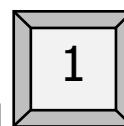
To boot your MSX into MSX-DOS2 you should put files NEXTOR.SYS and COMMAND2.COM into the root directory of the volume (RAM disk or SD-card).

To dynamically mount additional volumes use MAPDRV.COM utility for DOS or Nextor's built-in \_MAPDRV utility for BASIC.

For more information on using Nextor disk subsystem please refer to [Nextor 2.0 User Manual](#).

To disable Nextor initialization, please press and hold the following respective key until GR8NET initialization finishes completely:

GR8NET (Nextor) slot	Key scancode (hex)	International keyboard	Russian keyboard
1 (1.2)	1A		
2 (2.2)	19		



To forcefully boot Nextor in DOS1 mode press and hold key (scan code 01h).

---

## 7. GR8cloud virtual volume

Since April 2018 there's new functionality built into GR8NET: network virtual volume. This functionality is available in composite mapper modes (8-14), when Nextor storage subsystem is active.

The volume, if having proper format supported by Nextor and required files, can be booted from, and used as local storage device like SD-card.

The virtual volume appears as a storage device for MSX machine, readable and writable, but located outside of the machine on the remote server defined by the \_NETSETCLOUD command. Its index number is 2, with GR8NET SD-card's index number is 1; thus Nextor will mount SD-card first if card is present in its slot and has valid format.

Image of the volume on the remote server is just a file with defined size, having proper contents to be recognized by the Nextor as valid storage medium: MBR, boot sector, FAT, directory and user data space.

Each GR8NET is having its own volume assigned to it, secured with the password provided by the user of the GR8NET and stored on the server.

Access to the volume is having **specific level of security**, and there is low probability that there will be unauthorized access to the data without leakage of the password; however data communication is **not private**, with data flowing not being encrypted, thus subject to the interception and inspection within the intermediate devices (routers, proxies etc. appearing on the packets' path).

GR8cloud virtual volume is not a performance storage device due to the limitations of the network stream speed and nature of the file system access by the Nextor, thus do not plan to use it for data-intensive operations; use SD-card instead copying data from network volume to SD-card and back if needed.

Maximization of the virtual volume speed can be obtained by accessing it using 16KB block size (in 32-sector burst reads or writes), when authentication and other overhead is minimal. Keep this rule in mind if you are going to design applications for the GR8cloud virtual volume.

---

## 7.1. Setting up GR8cloud virtual volume

GR8cloud volume is available only in mapper modes 8-14 (when Nextor kernel is active), but you can use its setup commands in other mapper modes.

The only information you should provide to access the volume is the URI of the GR8cloud server with port number, and your GR8NET password to access the virtual volume.

The default volume image should already be present for your GR8NET on the server with default password. **As soon as you start using the virtual volume, you must contact us at [info@gr8bit.ru](mailto:info@gr8bit.ru) and supply new password to put onto the server configuration.**

### NETSETCLOUD

Set up GR8cloud virtual volume access

#### Format

CALL NETSETCLOUD (H\$, PS\$)

CALL NETSETCLOUD (S)

#### Arguments

H\$ is a host name with port number separated by colon, max string length is 70 chars

PS\$ is password to access the volume, case sensitive, max string length is 16 chars

S is enable/disable for the GR8cloud volume (see below for details)

#### Usage

Any argument may be omitted, but at least one must present. They may be variables or constants.

H\$ will have format like "network.symbos.org:684", and you *must* use port number because port number set by \_NETSETPORT will be used, and it may change during computer operation. Do not use any protocol prefixes, use just host name and port number.

PS\$ is password string, which must match the one on the server; if passwords will not match GR8NET will not be able to obtain data from the image file.

To disable the GR8cloud subsystem you set value S to 0, and GR8cloud volume becomes inaccessible immediately (even if it is still mounted by the Nextor). To enable the GR8cloud subsystem you set value to 1, but full operation of the volume is possible only after reboot. The values you define using this command are preserved by the \_NETSAVE command.

#### Example

CALL NETSETCLOUD ("network.symbos.org:684","MyPassword")

CALL NETSETCLOUD (1):CALL NETSAVE

---

A\$="MyNewPassword"

CALL NETSETCLOUD(,A\$):CALLNETSAVE

## NETGETCLOUD

Prints GR8cloud virtual volume status onto the screen

### Format

CALL NETGETCLOUD

### Arguments

No arguments are required, command is not assumed to be used from BASIC program

### Usage

You use this command to confirm the status of the GR8cloud virtual volume within GR8NET, however if you experience problems accessing it using Nextor, the issue may be in other location than GR8NET – it can be an issue with Nextor treating image data, for example, in cases if image is corrupt.

### Example

CALL NETGETCLOUD

network.symbos.org:684, enabled

Ok

## 7.2. Precautions and disclaimers

Operation of GR8cloud involves not only your GR8NET device; its operation depends on the connectivity to the internet, availability of the server which serves the volume images, and depends on your own pursuit of the security. In particular:

- The volume image is bound to the GR8NET card you use, and to the password you provide. If malefactor will get access to your GR8NET hardware, he will be able to access your GR8cloud virtual volume, and thus read, write and corrupt data on it. Be very prudent when handing your GR8NET over to people you do not trust;
- If your virtual volume is going to have important data, back the volume up, or at least important data, regularly as it changes, for example to the local SD-card;
- Data you transfer between MSX and your corresponding GR8cloud virtual volume is not private. Thus please do not store sensitive information on your virtual volume. We have implemented specific measures not to allow unauthorized access to the image's data, but data can be intercepted when transferred through the network. Of course data ciphering may be possible to implement, but it will slow down the communication significantly, so we decided not to do it at this time;
- Needless to say keep password to your virtual volume safe;
- The GR8cloud client-server solution is rather complex one, and while we will be doing our best in making service operational and available, we can not be held liable for availability, reliability of the service and consistency of data related to provision of this service, as well as for timely resolution of issues with it.

If you follow precautions above you must be safe in using the volume and data.

---

### 7.3. Acknowledgements

At the start of the GR8cloud service the server part is located within the [SymbOS](http://network.symbos.org) network server (network.symbos.org) at the port 684; we thank SymbOS team – Jörn Proda and Edo van Zanten – for hosting the images and providing the service. It is expected that SymbOS will also have the driver for GR8cloud, and you will be able to run applications from it.

---

## 8. Built-in MSX-Audio, MSX-Music and PSG

Starting January 2017 GR8NET hardware and firmware is having built-in MSX-Music (OPLL) capabilities. Starting mid of February 2017 is it having built-in MSX-Audio (limited version of Y8950). OPLL Implementation is functionally identical to the YM2413 chip, but uses 16-bit sound and finer timing than the original; it has Philips Music Module DAC implemented at port 0Ah (register is shared with Digital Waveform input). MSX-Audio is based on OPLL implementation, and has no ADPCM analysis/AD and discrete DA function.

Since version 0.9 GR8NET's MSX-Audio, MSX-Music, SCC and PSG are clocked by internal GR8NET clock of 3.579545 MHz. There's no more an option to set them to be clocked from MSX system bus.

MSX-Music and MSX-Audio are controlled by the commands `_NETGETOPL` and `_NETSETOPL`, which should be used to enable or disable built-in Music and Audio hardware, and provide their configuration.

When enabled, Audio and Music are present at their respective I/O ports (7C/7D for OPLL and C0-C1/C2-C3 for Y8950) for writing in all mapper modes, and, when in mapper mode 8, OPLL ROM BIOS will appear in subslot #3. In all other mapper modes OPLL BIOS is not available (unless faked into mapped RAM using `_NETFKOPLLR`).

`_NETSETOPL` command settings are preserved by the `_NETSAVE` command, except bits `AUDPRT` (port range where Y8950 device appears) and `AUDDEC` (deconfiguration status), which are decided by the GR8NET ROM during startup and can not be changed by the `_NETSETOPL` command.

### NETGETOPL

Gets status of built-in OPLL/Y8950, and initial setting of sample RAM size

#### Format

CALL NETGETOPL (G, CS, GRS)

#### Arguments

G is variable, a bitmap will be returned with status of the OPL subsystem

CS is variable getting size of *allocated* Y8950 sample RAM 8kByte pages

GRS is variable getting size of *requested* Y8950 sample RAM 8kByte pages

#### Usage

Variable G is having the following format:

7	6	5	4	3	2	1	0
Reserved, must be 0	AUDPRT	AUDDEC	AUDINTD	AUDDIS	OPLVLVL	OPLLDIS	

OPLLDIS 0 = OPLL output is enabled (default)

1 = OPLL output is disabled

OPLVLVL 0 = Normal volume (default)

1 = Double volume

AUDDIS 0 = MSX-Audio is enabled (default)

1 = MSX-Audio is disabled

AUDINTD 0 = MSX-Audio interrupts are enabled (default)

1 = MSX-Audio interrupts are disabled

AUDDEC 0 = MSX-Audio is deconfigured and is not / will not be available



AUDPRT 1 = MSX-Audio is configured at port indicated by AUDPRT  
 0 = MSX-Audio is configured at ports C0-C1  
 1 = MSX-Audio is configured at ports C2-C3

CS variable gets information on how much sample RAM in 8KB blocks is available to onboard Y8950. The range is within 0 to 32 (256K).

GRS variable gets size of the Y8950's sample RAM, requested before initialization and memory allocation and set by `_NETSETOPL` command. If the requested RAM size was available at the GR8NET initialization, then all this space is allocated, otherwise no sample RAM is allocated. To see the memory allocation for sample RAM, refer to [\\_NETGETMMV command](#).

Any of arguments may be omitted, but at least one must be present.

## NETSETOPL

Enables or disables built-in OPLL/Y8950, controls doubling of output amplitude, and sets sample RAM size

### Format

CALL NETSETOPL (S, SRS)

### Arguments

S is variable or constant, a bitmap of OPLL/Y8950 disable and volume settings

SRS is variable or constant setting size of Y8950 sample RAM in 8KB pages (max 32, default)

### Usage

S follows the following format:

7	6	5	4	3	2	1	0
Reserved, must be 0	N/A		AUDINTD	AUDDIS	OPLVLVL	OPLLDIS	

OPLLDIS 0 = Enable OPLL output (default)

1 = Disable OPLL output

Using this bit the command sets *OPLL disable* bit in System mode register accordingly (see [System registers](#) chapter). Please note that this bit disables built-in OPLL output, not OPLL itself. It still performs all required actions internally, and enabling OPLL output will immediately cause valid FM sound to appear

OPLVLVL 0 = Normal volume (default)

1 = Double volume

This bit sets output volume level for the OPLL/Y8950. In double volume mode audio outputs twice amplitude than in normal mode, however if many channels are generating the sound, output may occasionally appear overloaded, and output sound may be slightly distorted. Using this bit command sets *OPL 2Xvolume* bit in System mode register (see [System registers](#) chapter).

AUDDIS 0 = Enable MSX-Audio (default)

1 = Disable MSX-Audio

Using this bit the command sets *Y8950 disable* bit in System mode register accordingly (see [System registers](#) chapter). When this bit is set, Y8950 is in

---

reset mode; state of this bit does not affect ADPCM RAM space size already allocated for the Y8950.

AUDINT    0 = MSX-Audio interrupts are enabled (default)  
            1 = MSX-audio interrupt are disabled

You can disable hardware interrupts generated by the Y8950, thus, with this bits set, BASIC application may be able to manage Y8950 without getting into interrupt request deadlock, generated by Y8950.

SRS sets initial size of sample RAM size, which is used to allocate RAM when adapter initializes. After changing value using SRS, you have to reboot machine to make setting effective. If you want change to be permanent, use \_NETSAVE command.

SRS can not be larger than 32 (256KB of sample RAM).

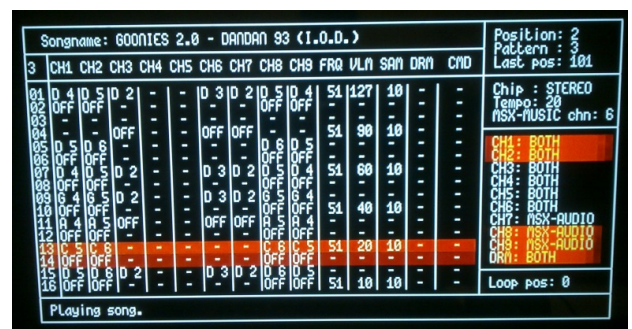
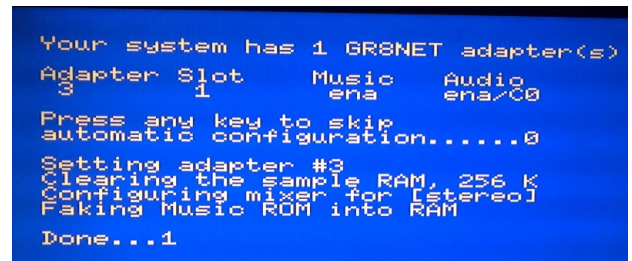
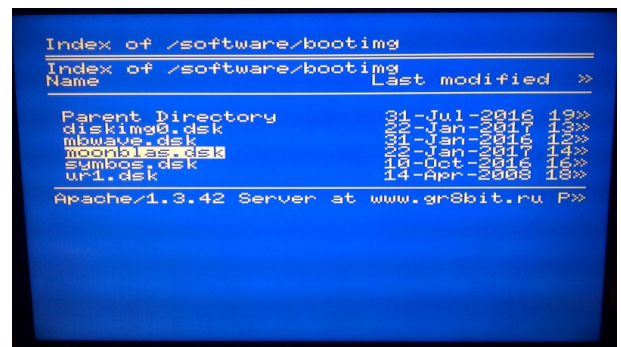
## 8.1. Starting with built-in OPLL and Y8950

OPLL (YM2413, the heart of MSX-Music) as a device is available in all mapper modes, and is enabled by default unless explicitly disabled by setting OPLL disable bit by `_NETSETOPL` command or disabled during adapter initialization after machine reset from the setting saved into the flash chip. OPLL ROM BIOS is only available in mapper mode 8 in subslot 3, if OPLL is not disabled, however there's a way to *fake* OPLL BIOS into RAM with `_NETFKOPLLR` command when using other mapper modes.

Y8950 (the heart of MSX-Audio) is also available in all mapper modes, given it was configured at available port range and was not explicitly disabled by the `_NETSETOPL` command.

You can start using both capabilities with the following simple plan:

- Connect phones to your GR8NET (if it is stereo), or any speaker device to MSX output (if your GR8NET is mono);
- Power cycle your MSX, and ensure you boot into mapper 0 mode (pressing arrow down key if needed);
- When adapter started initialization, press and hold F2 and F4 keys. Keep them pressed until "Release keys..." message is displayed;
- Release both keys, and you will be brought to the web browser. You will see GR8BIT web server in case you did not change default URI location with `_NETSETHOST`, `_NETSETPATH` and `_NETSETPORT` commands;
- Press **m** key to get to the list of files (actually links) starting with this letter, and scroll to `moonblas.dsk`. Press space key on it, and it will exit browser and start loading the image;
- The disk image you load is not original one, it has special boot program designed to properly configure your GR8NET for your best experience with it. Let program to perform configuration for you – clear sample RAM, configure mixer and fake ROM into RAM – and wait until Moonblaster application loads;
- After it starts, press F5 key to go to disk input/output screen. Press space on "Load Song" and load song named "Rotate". Then go down, and press space on "Load Samplekit", and press enter on "MBMUZAK2". Then press ESC key to return to main screen;
- Now hit F1 key to start playback, and



---

enjoy it. You can load other songs and another sample kit to hear the difference in ADPCM sound of the Y8950.

If you are an advanced user and want to play with Moonblaster application, please note the limitations of it:

- It requires MSX2 machine because it uses 80-character mode and memory mapper;
- It does not work in DOS2 mode, and it will be really hard to make it functioning properly in GR8NET mapper mode 8 (when Nextor is active) – even in DOS1 mode;
- It will not function if the size of sample RAM is 0. It must have at least 32K of sample RAM to load sample kits.

## **8.2. Playing games with built-in OPLL**

When games are loaded into their respective mapper type (for example, Aleste loads with mapper type 5 – ASCII16), OPLL ROM BIOS is not available any more, and game is unable to detect OPLL chip (unless it assumes chip should be there and uses it blindly).

There're two solutions to such situation:

1. Faking OPLL ROM into RAM so that games, when start scanning slots, see OPLL ROM signature in the RAM. To use this option your machine must have at least 64K of main RAM, at best mapped RAM. Command to use is `_NETFKOPLL`;
2. Having two GR8NETs in the system, one configured in mapper mode 8 (having GR8NET network, 512kB of RAM, Nextor and OPLL ROM BIOS), and another configured for the game. You can take the following steps to achieve the result:
  - Install GR8NET with ID=0 into slot 1, and GR8NET with ID=1 into slot 2. Connect network cable to latter GR8NET with ID=1;
  - On first boot, in BASIC, perform `_NETSETMAP(24)` and system will reset with adapter ID #0 being configured in mapper mode 8;
  - On second boot, in BASIC, you perform commands `_NETSETDA(1)` to set default adapter with ID #1 and use `_NETBROWSE` browsing for game on the internet or installed SD-card this adapter #1;
  - When you choose the game, adapter with ID #1 (in slot 2) will reconfigure to mapper type required for the game, and system will reboot with first adapter being configured in mapper mode 8, and second configured with the game.

## **8.3. Using built-in MSX-Audio**

MSX-Audio is detected differently than OPLL (YM2413), and applications supporting this hardware should find it without MSX-Audio BIOS.

When GR8NET initializes, it checks ports C0/C1 and C2/C3, and if finds free port set, sets its Y8950 to operate through those ports. If there're no ports available from these sets (for example, you have two MSX-Audio Modules installed, or two GR8NETs with MSX-Audio enabled), then GR8NET deconfigures its MSX-Audio capability.

---

To play ADPCM samples MSX-Audio needs sample RAM, its space is automatically allocated on the initialization from the available GR8NET RAM space. To see effective RAM map please use `_NETGETMMV` command.

You can disable Y8950 with AUDDIS bit using `_NETSETOPL` command, and sample RAM will not be allocated on next GR8NET initialization, and thus this space will be available for use, but in this case Y8950's audio output will also be disabled. Alternatively you can set sample RAM to lower values than 32 (like GR8NET ROM BIOS does in specific mapper modes), or to 0 to disable ADPCM function completely. Value set corresponds to number of 8KB GR8NET RAM logical pages, thus 32 pages is 256KB, and 4 pages is 32KB.

It is possible to disable interrupts from Y8950 for debugging purposes or for accessing it with BASIC programs using `_NETSETOPL` command; though please note that many applications use interrupts, and if not handled properly (not enabled after finishing debugging) machine may misbehave.



## 8.4. Alternative interface to Y8950 and OPLL registers

Both audio devices are operating through I/O, but GR8NET provides alternative read/write access to its built-in audio registers.

```
call netdump(&h40, &h6000, 272)
6000: 41 42 10 40 00 00 00 00 00 00 00 00 00 00 00 00 AB.0.....
6010: 3E 08 CD 41 01 E6 20 C8 3E FF 32 E1 5F 21 2B 40 >.MA.φ x>.2A_!+0
6020: 11 00 70 01 09 00 ED B0 C3 00 70 3E 80 32 E0 5F ..p...M≡u.p>_20_
6030: 2A 02 40 E9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 C9 ×.0Иииииииииииии
6040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
60F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
6100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
Ok
```

Picture above shows starting locations of the Boot ROM page 0F0h and highlights specific regions defined for FM generation by OPLL and Y8950.

- **Green** area shows 64 executable bytes of the boot ROM. This location is protected for writing by the CPU;
- **Red** locations are channel register settings for OPLL (YM2413), in total 9 channels, 3 bytes per channel. First byte's bits [7:4] are instrument number, bits [3:0] are volume attenuation (OPLL registers 3x), second byte's bit [5] is sustain key, [4] is key on/off, [3:1] are octave, and [0] is F-Number MSB (OPLL registers 2x), and third byte's bits [7:0] are LSBs of F-Number;
- **Fuchsia** colored area is OPLL's custom instrument properties, as defined in the datasheet's registers 0-7;
- **Yellow** locations are channel register settings for Y8950, in total 9 channels, 3 bytes per channel. First byte's bits [3:1] are feedback, bit [0] is connection type (Y8950 registers C0-C8), second byte's bit [5] is key on/off, [4:2] are octave, and [1:0] are MSBs of F-Number (Y8950 registers B0-B8), third byte's bits [7:0] are LSBs of F-Number (Y8950 registers A0-A8);
- **Light blue** locations are definitions of Y8950's instruments, 9 in total. Even locations are attributed to even operators (modulators for chained connection), and odd locations are for odd operators (carriers for chained connection). Instrument format is the same as for OPLL, except that byte at instrument offset +3 contains KSL/TL for carrier.

Two FM registers are not accessible this way: OPLL's rhythm register 0B and Y8950's rhythm register BD. They should be accessed via standard I/O.

Y8950's ADPCM registers are also accessible only through standard I/O, but there's a way to access sample RAM. To know starting page and size of the sample RAM use `_NETGETMMV` command, and then you have direct way to access these RAM pages using `_NETSETMEM` and `_NETLDBUF`.

## 8.4. Using built-in PSG

Since November 2018 GR8NET firmware is having built-in PSG (programmable sound generator). It can function as a mirror of built-in PSG, or be placed at port base 0x10 to function as second PSG, supported by applications like VGMPLAY.

PSG is being dynamically configured according to desired settings configured by the `_NETSETPSG` command, and can be reconfigured on the fly by the same command.

### NETSETPSG

Set built-in PSG properties

#### Format

CALL NETSETPSG [(F)]

#### Arguments

F is a bitmap variable or constant

#### Usage

Value of F defines initial desired state and location of the PSG:

- Bit 0 set defines if PSG should be enabled on (re) configuration;
- Bit 1 set designates port location 0x10, if reset port location 0xA0 (built-in PSG mirror).

If argument is omitted, then PSG reconfiguration is performed. If another PSG device at respective port is detected (through reading its registers), then GR8NET PSG is set into write-only mode.

### NETGETPSG

Get built-in PSG properties

#### Format

CALL NETGETPSG [(G)]

#### Arguments

G is a bitmap variable, having the following format:

7	6	5	4	3	2	1	0
Reserved			PSGRD	PSGLOC	PSGENA	PSGDLOC	PSGDENA

where:

- PSGDENA and PSGDLOC are desired initial state of the PSG (see `_NETSETPSG` command);
- PSGENA and PSGLOC are actual state (1=enabled) and location (0=0xA0, 1=0x10) of the PSG;
- PSGRD bit is set if PSG registers are readable, and reset if PSG is in write-only mode.

---

If argument is omitted, then command prints PSG state onto the screen.



```
callnetgetpsg
To be: enabled
To put: port 0xA0
Active: yes
At: port 0xA0
Read: disabled
Ok
█
```

- **To be:** desired state for PSG;
- **To put:** desired location of PSG;
- **Active:** state if PSG being enabled (configured);
- **At:** I/O base port value;
- **Read:** designation of readable (enabled) or write-only (disabled) mode



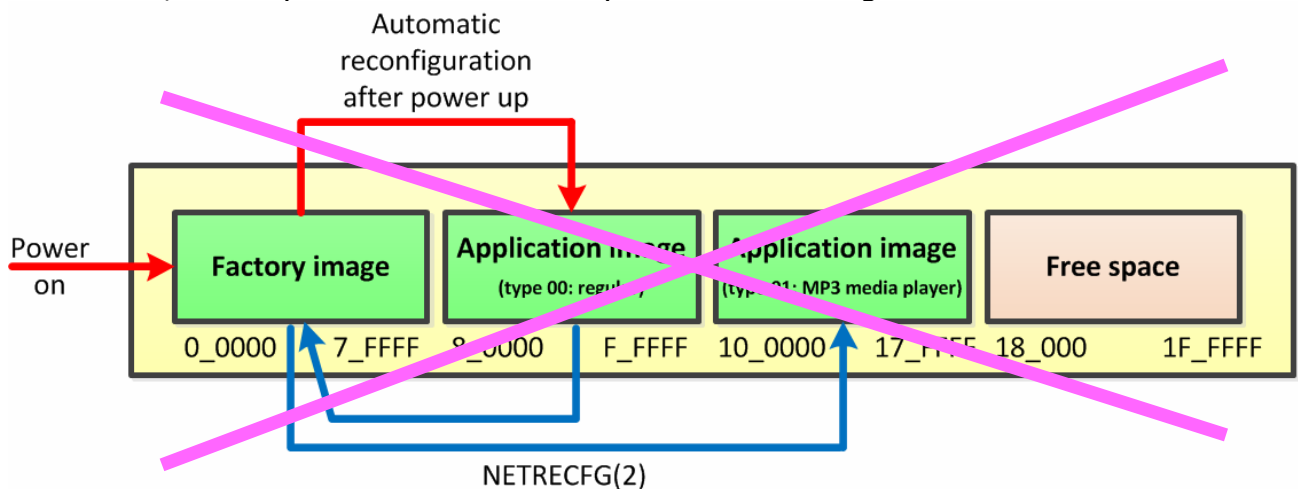
## 9. GR8NET functionality extensions

Initially GR8NET was designed to be a simple networking communication adapter, later it has grown to relatively complex device with a number of features. While regular GR8NET FPGA image is full and there's almost no possibility to add any new functionality into it, there's another direction to grow.

Original GR8NET is having 2 MB serial flash chip attached to the FPGA, as appeared later it is way more than single Cyclone III configuration requires. Original FPGA configuration file size selected was 512 KB, containing unpacked image for FPGA of about 3 million bits (375 KB). Packed image is approximately twice less, about 200 KB, which perfectly fits to the 256 KB file size.

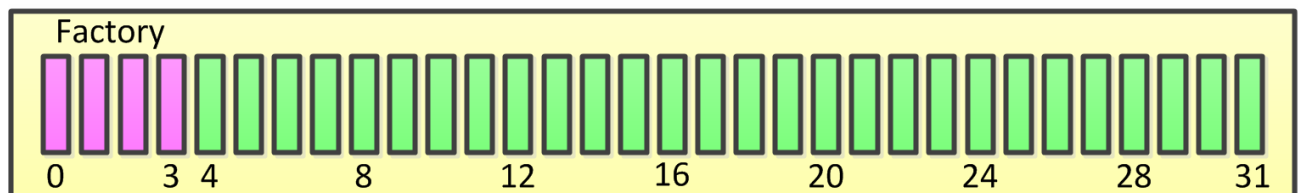
The original configuration chip, Altera EPCS16, consists of 32 sectors of 64 KB in size, thus unpacked image file occupies  $\frac{1}{4}$  of the chip, and thus chip may have had 4 FPGA configuration images.

Starting August 2017, when serial flash chip update feature was first introduced in version 0.7, the map of the serial flash chip was the following:



with only 4 *locations* available, regular image occupying first and second locations, and MP3 player image third location, and fourth, last location, reserved for further developments.

Version 0.8 has different mapping mechanism: it is based on flash chip sector of 64 KB in size, and also supports bigger serial flash chips like W25Q128FVSIQ.



The picture above shows EPCS16 chip with 32 sectors, 2 MB is total. First 4 sectors are occupied by the "factory" image, which is used for FPGA configuration when power is applied to GR8NET. All remaining space is available for FPGA configuration images or catalog data, provided in the format accepted by the `_FLUPDATE` command.

---

## 9.1. Regular GR8NET image

When GR8NET configures with regular image the cartridge starts functioning ... as GR8NET with all functionalities declared in this manual (except additionally noted). When power is applied, GR8NET must load regular image, thus this image must be written into the serial flash chip starting its sector 0.

Your GR8NET may have more than one regular image in remaining serial flash chip's space, but there's no point for having same copy of the image.

## 9.2. Embedded MP3 player

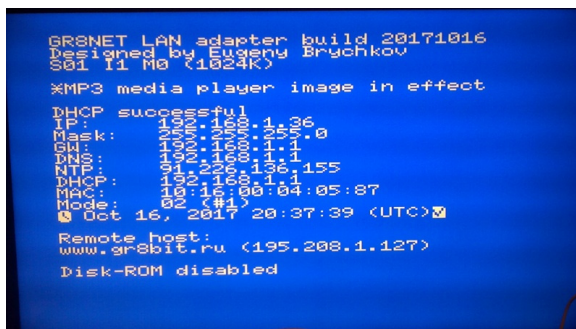
Note: the MP3 configuration image uses third party decoding engine, thus has limited supportability and maintainability as provider of the MP3 decoder does not perform support and maintenance. License and credits can be found in the license chapter at the end of the manual.

You write MP3 decoder (player) image into the serial flash at the space not occupied by any other images or data (or overwrite already existing data) after looking at the \_FLLIST command output which checks and shows the images in the chip.

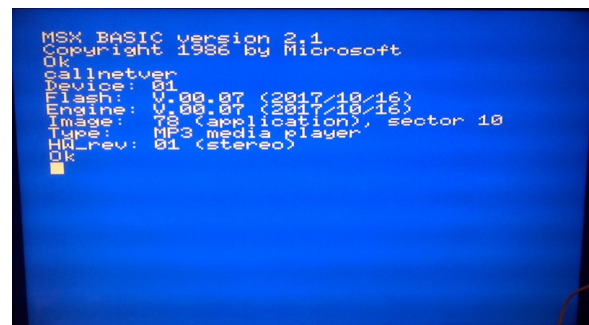
To switch GR8NET to the MP3 player mode use \_NETRECFG(n), where n is the starting sector number of the image in the serial flash chip.

After machine has reset and GR8NET re-initialization is performed, you should have Online Radio entry in the GR8NET browser and should be able to listen to online radio, as well as explore internet and inserted SD-card playing MP3 files from those locations.

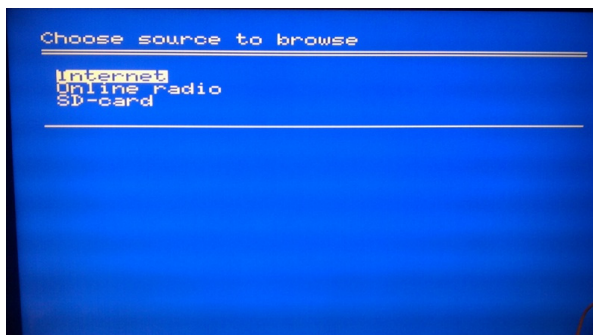
Here're several screenshots for your reference:



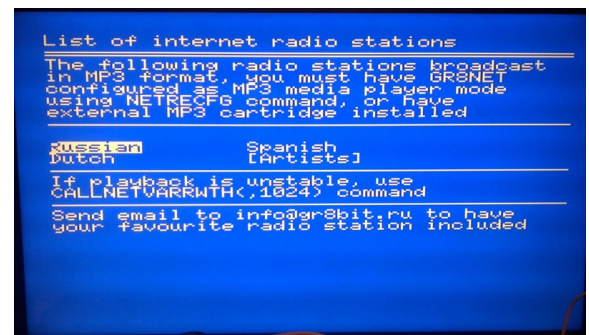
Initialization of GR8NET in MP3 media player mode



NETVER output in MP3 player mode



Online radio entry in the browser



Main online radio web page

---

### 9.2.1. Limitations of GR8NET in MP3 player mode

While MP3 player must be a great add-on to already existing GR8NET functionality, it occupies significant silicon space in FPGA, and that's why it is not within regular GR8NET firmware image, but designed as separate image. This separate image, to accommodate MP3 player, has the following functionalities removed:

- Only two mapper modes are available: 0 and 8;
- Removed sound custom chip (SCC);
- Removed Y8950 OPL (MSX-Audio);
- Removed YM2413 (MSX Music);
- Removed FPGA firmware flashing capabilities (\_NETFLUPDATE);
- Removed PCM.

The following capabilities are in this MP3 media player configuration firmware:

- SD-card circuitry (usable through \_NET commands);
- Networking;
- GR8NET built-in Disk-ROM and RAM-disk functionality;
- Digital waveform input;
- In mapper mode 8 there is 512KB of mapped RAM in subslot 1 and Nextor driving SD-card in subslot 2. Subslot 3 is empty.

There're also the following limitations

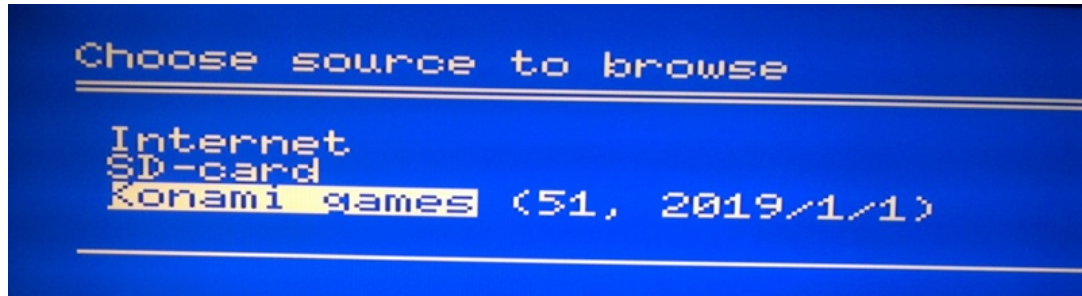
- Only MPEG-1 Layer 3 is supported. Other formats may give format error, or cause improper sound. If you are hosting a party or show, ensure that files and streams you are going to use play properly beforehand;
- It is not possible to reconfigure from MP3 media player image. Power cycle the machine to force GR8NET reloading regular image from sector 0;
- Network MP3 stream/file player requires 272 Kbytes of the GR8NET onboard RAM to operate for its buffers (256K), control data (8K), and metadata (8K) to ensure smooth playback during network interruptions. In mapper mode 0 under most conditions there will be enough RAM (e.g. with GR8NET RAM disk which takes 720K of the RAM); In mapper mode 8, if you activate RAM disk, there will be no enough RAM to run network MP3 player (512-360=152 only);
- SD-card MP3 player does not have limitation as described above for network MP3 player;
- If there's an issue with MP3 stream, MP3 decoder will hiccup. Setting \_NETVARRWTH(,1024) may help minimize issues with network streaming;
- If there's an issue with the audio stream containing ICY metadata, whole further playback will sound wrong as player will not be in sync and treat metadata as audio data and vice versa. The cure for this, if you start hearing corrupt sound, is to restart player.

The rationale of such changes is to free logic and embedded RAM space within FPGA enough to accommodate MP3 decoder, and the fact that in this mode GR8NET will be used by the operating system (BASIC, MSX-DOS or Symbos) as network/SD-card MP3 player only.

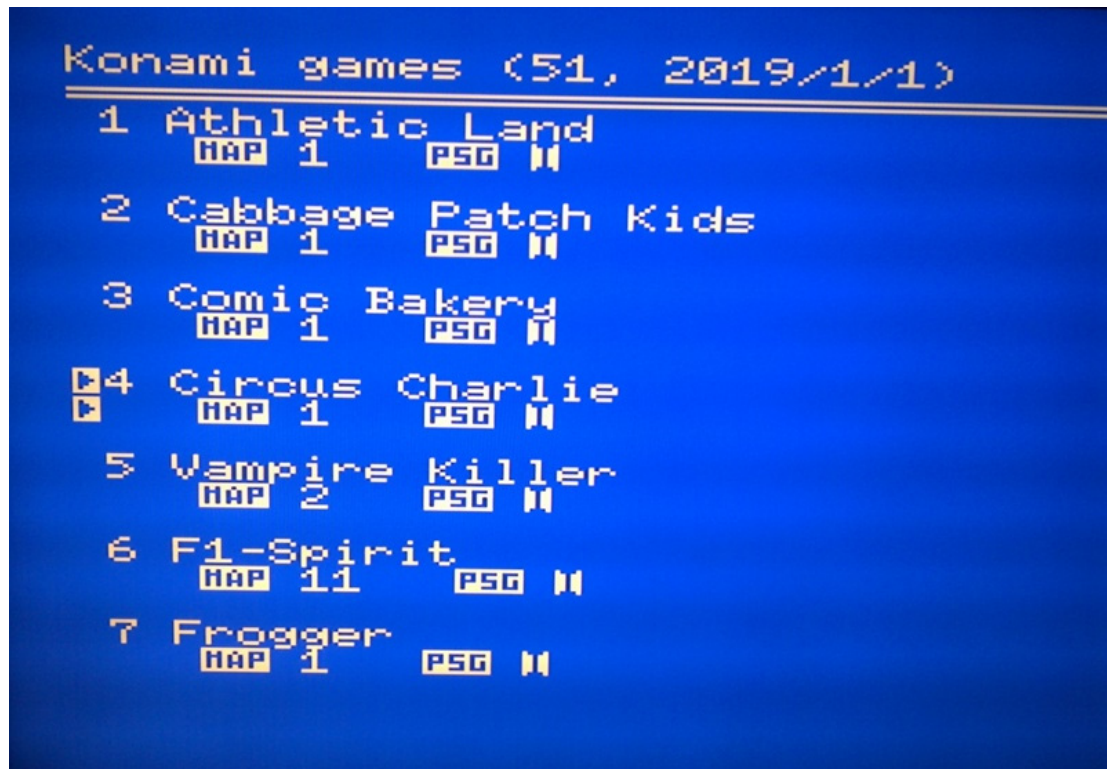
---

## 9.3. Catalogs

Another type of data which can be stored into the GR8NET serial flash chip is catalog. Catalog consists of set of compressed ROMs with specific attributes, directory and signatures. Catalog may occupy any amount of the space in the flash chip, given it is not written onto the regular image in the factory location.



When browser starts, it looks for catalog signatures, and if found, displays catalog title, number of entries and catalog build date in its source selection screen.



The catalog browser is a different application than web browser, and keys have slightly different meaning.

- Arrow up and down: previous or next entry;
- Arrow left and right: page up and down;
- Enter key: select ROM and run it;
- ESC key: exit the catalog browser;

- 
- Any other alphanumeric key: search for next entry with its name starting with the character corresponding to the key pressed. If end of catalog was reached, search continues from the beginning.

### **9.3.1. Making the catalog**

Location of the tool: <http://www.gr8bit.ru/software/romcat/>, ZIP archive with sample Konami Games catalog and tool in executable/ directory.

The creation process is straightforward: there's executable (requires .NET 2.0), which can run under Windows and Linux. There's only one argument – file name of the catalog configuration file. It is a text file, and all further instructions are provided in this configuration file. You will also need Pletter.exe file version 0.5c, which is also provided in the tool's directory.

### **9.3.2. Putting the catalog into the GR8NET serial flash chip**

Catalog maker tool creates database files, and BASIC file for performing the update. All the information – the file names and location of the database files – is specified in the configuration file used by the tool.

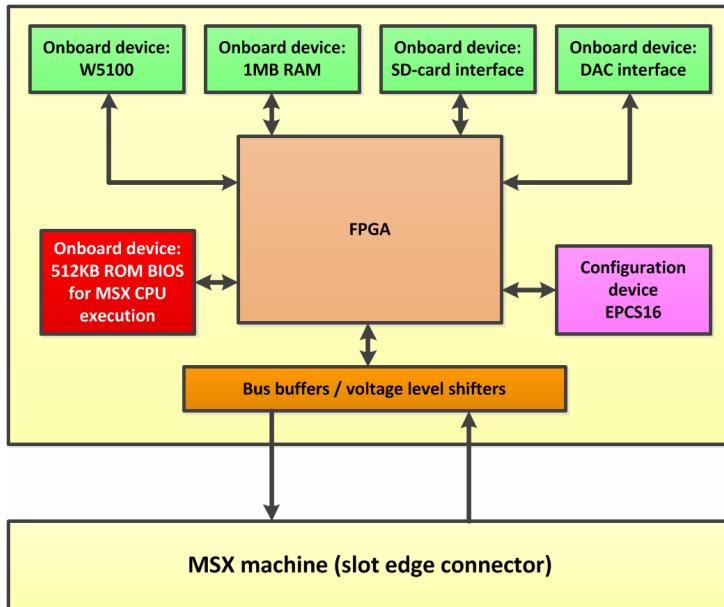
The created files – database files and BASIC program – will need to be copied into the place specified in the configuration file and which can be used by GR8NET to run BASIC program and read database files. After starting the BASIC file, it will check for space availability, perform \_FLLIST command for you and then will ask for starting sector. When choosing the sector please ensure that you do not overwrite regular image at factory location, and do not overwrite the sectors containing any other valid information (e.g. other needed images or catalogs).



## 10. Firmware upgrade

Sometimes you may find out that there're software or firmware updates available for your GR8NET adapter; to get declared bugs fixed and new declared functionalities you may need to upgrade the adapter's firmware.

To perform upgrade properly, you must understand the firmware layout, how it works, what sequence you must follow and why.

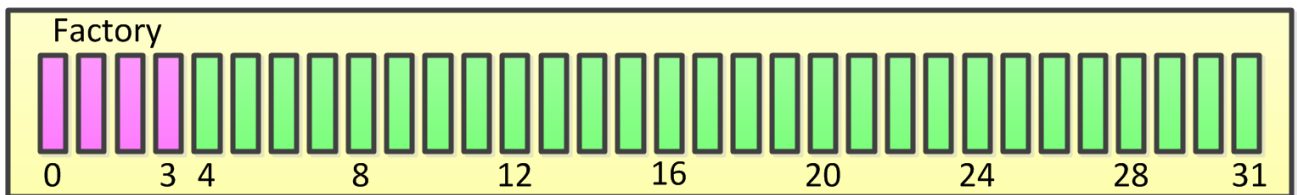


There're two devices keeping microcode (firmware) for the adapter operation –

- Configuration device: this serial flash chip contains data which loads into FPGA and makes FPGA function specific way, in other words it configures hardware devices and services FPGA provides to the system;
- 512KB ROM BIOS chip: this chip contains MSX CPU executable code so that Z80 or R800 can interface with the FPGA and get access to its onboard devices and services.

These two microcode (firmware) images are complementary to each other and strictly depend on each other's version and operation. It is highly advisable to keep their versions synchronized.

Next, FPGA configuration device is having special layout of firmware in it. Below is a map of the serial flash chip EPCS16, single bar represents single 64KB flash chip sector.



When GR8NET is powered on, it tries to load so called factory image located at the configuration chip's sector 0. If load is unsuccessful GR8NET will not appear in the system.

Information about the image being running can be obtained using \_NETVER command output:

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft
Ok
callnetver
Device: 03
Flash: V.00.08 (2019/1/12)
Engine: V.00.08 (2018/12/29)
Image: 00h (factory), sector 0
Type: regular
HW_rev: 01 (stereo)
Ok
■
```

The adapter is running factory image from sector 0 of the serial flash chip

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft
Ok
callnetver
Device: 03
Flash: V.00.08 (2019/1/12)
Engine: V.00.07 (2017/12/25)
Image: 78h (application), sector 41
Type: MP3 media player
HW_rev: 01 (stereo)
Ok
■
```

The adapter is running MP3 medial player application image from sector 41

```
Designed by Eugeny Brychkov
S81 I3RF M8 (512K+512K+Nextor+FMpak)
DHCP successful
Mask: 255.255.255.0
GW: 192.168.1.1
DNS: 192.168.1.1
```

**R stands for regular**

**F for factory location (sector 0)**

When initializing, adapter will show Lxy, where "x" is type of image (R=regular, M=MP3), and "y" is state of the image (F=factory, A=application).

The output of the \_NETVER command shows Engine (FPGA) image version and build datecode, and flash chip image version and datecode, which is Z80-executable code part of the firmware located in the flash chip.



---

## 10.1. The order of the firmware update

The proper order to update adapter's firmware will depend on the way you will be going to do it. Note that for each scenario order differs, and reasons are explained. In worst case you may need to perform all-offline update, but it is highly advised to keep your FPGA update device (e.g. GR8blaster or USB Blaster device) nearby.

### **Option 1: all-online update within the same version.**

You can use this option in case your adapter is running properly, can load firmware images into its RAM from network or SD-card, and can run `_NETFWUPDATE` and `_FLUPDATE` commands. In this case the update order is:

1. Load FPGA image into GR8NET RAM, and use `_FLUPDATE` command to update regular image in the adapter starting sector 0 (factory location). After update system will not require reboot, FPGA is still running previous version of regular image. Do not power cycle machine because power cycling will load new image into FPGA;
2. Load flash chip firmware image into GR8NET RAM, and use `_NETFWUPDATE` command to update flash chip firmware. After update system will force reboot. Please power cycle instead to allow both new images to be in effect.

### **Option 2: flash chip online and FPGA offline – within the same version.**

You can do it in case your adapter is running properly, and can load firmware image and run `_NETFWUPDATE` command, but is not capable of running `_FLUPDATE` command because regular image in FPGA is too old. In this case the update order is:

1. Load flash chip firmware image into GR8NET, and use `_NETFWUPDATE` command to update flash chip firmware. After flashing system will force reboot;
2. After updating according to step above, turn off the machine and proceed to connecting cable and performing offline FPGA update (see instructions further in this chapter).

In case you interchange the update order, you will be starting existing flash chip image with new FPGA configuration, and if there're will some incompatibilities between them, you risk to be forced to perform offline flash chip update because something may not work properly.

### **Option 3: all-offline update.**

You will do it in case adapter does not work properly and you suspect that it happens because of corrupt firmware images in serial flash chip of the FPGA or main ROM flash chip. As flash chip firmware works at higher layer than FPGA (which can be seen as hard-ware), the right order is:

1. Use cable to verify and/or update FPGA image. Note that you may need to perform *whole chip erase* in some circumstances to start from the scratch;
2. Perform offline update of the flash chip from any of the storage interfaces connected to your MSX PC.

## 10.2. Diagnosing firmware faults

If your adapter does not work as documented, there're could be several causes of it:

1. Hardware fault: it is very important to closely examine adapter (probably with magnifier) to identify if there're any visible defects, and identify what events were preceding the faulty state (e.g. physical impact like falling from table or damaged/dirty edge connector, electrical impact like electrostatic discharge etc.). Note that fault could be not at the GR8NET side, there're could be problems with MSX machine you insert GR8NET into (e.g. dirty, expanded or bent pins of the slot connector, other PC failures), it is good idea to try GR8NET in another machine if possible to see if it behaves differently;
2. If adapter looks fine, and you are sure it is not machine, then you may proceed with checking and reloading its firmware.

Identifying if firmware of FPGA level functions is relatively easy – you install adapter into machine, power machine on, and insert SD-card into its slot. If FPGA (GR8NET hardware) is functional, SD-card's red activity LED will turn on and then off as usual showing SD-card initialization. If it does not happen, then, most probably, FPGA can not configure, and its firmware is under suspicion.

Identifying if flash chip contents are the cause of malfunction is slightly more complicated – it will be required to boot into the MSX-DOS and use DBGE.COM utility (MSX debugger) to view what MSX sees in the (sub-) slot of the GR8NET location.

## 10.3. Location of the firmware update files

To perform update, you must know where firmware update files are located. Please see the picture below, the full URL of the location is <http://www.gr8bit.ru/software/firmware/GR8NET/>.

**Directory of the www.gr8bit.ru**

**Offline flash chip update files directory**

**Final v0.7 update files**

**Online FPGA update files**

**Online flash chip update file**

**Offline FPGA update file in RAR archive**

```
Index of /software/firmware/GR8NET
Personal use only - No commerce allowed
Name Last modified
Parent Directory
bloadable/
test-images/
v0.7-final/
@license.txt
@readme.txt
DBGE.rar
gr8net-bugs.txt
gr8net-chreq.txt
gr8net-fpga-mp3.bin
gr8net-fpga-reg.bin
gr8net-fpga-reg.rar
gr8net-rom.txt
gr8ntest.bin
update.bin
Apache/2.4.27 (Ubuntu) mod_ssl/2.4.27 OpenSSL/1.0.9 P>
```

---

## 10.4. Online method of firmware update

Online firmware update is only possible if adapter is fully functional, can load data from network or SD-card into its RAM and can run CALL commands described below.

There're two commands to update GR8NET ROM BIOS flash chip ("flash chip") firmware and FPGA serial flash chip ("serial flash chip"). Please note that to update/reload factory image you have to use offline method of FPGA firmware update.

### 10.4.1. Online flash chip update

Command \_NETFWUPDATE explained below allows updating flash chip contents – the code which is being executed by the MSX PC CPU. It is important that this firmware in the flash chip to be compatible with the firmware running in the FPGA, thus please plan FPGA update if there's information that this action is required for proper execution of new version of the firmware.

#### NETFWUPDATE

Update the firmware – contents of the onboard flash ROM BIOS chip

##### Format

CALL NETFWUPDATE(A)

##### Argument

If argument A is omitted or is 0, then command will just display information about current firmware level located in ROM and in RAM.

If bit 0 of argument A is set, command updates configuration area of the ROM chip.

If bit 1 of argument A is set, command updates main image of the firmware;

##### Usage

To update both areas – main image and configuration area, type \_NETFWUPDATE(3). If areas loaded into RAM will be found to be valid, update will start, otherwise error message will be displayed. To load image into RAM, you may use \_NETBROWSE or \_NETBLOAD commands locating valid image on GR8BIT website. Image is located at: <http://www.gr8bit.ru/software/firmware/GR8NET/update.bin>.

#### How exactly do I update firmware image in the flash chip using online update?

- Ensure GR8NET is connected to the internet using either DHCP or fixed IP address modes;
- Use one of the methods to load firmware image:
  - \_NETBROWSE the GR8BIT server to /software/firmware/GR8NET and press space key on update.bin file to load it into GR8NET RAM;
  - \_NETBLOAD("<http://www.gr8bit.ru/software/firmware/GR8NET/update.bin>")
  - \_NETSETHOST("www.gr8bit.ru"): \_NETSETPATH("/software/firmware/GR8NET"): \_NETSETNAME("update.bin"): \_NETBLOAD
- After successful image load run \_NETFWUPDATE(3). It will validate the image, and will request your approval to continue.
- After image is loaded into the flash chip, machine will reboot.
- New image is ready for use, and GR8NET will start using it.

---

This video by Alex Mena <https://youtu.be/05iwl8oPKW8> shows online GR8NET firmware update process.

### 10.4.2. Online FPGA chip firmware update

Since July 2017 you can update FPGA firmware through the MSX machine without using blaster device if GR8NET is operating properly.

Instructions below apply if your adapter runs version 0.8 firmware. If you have version 0.7 please refer to the migration chapter below.

The steps are very similar to the online update of the flash chip firmware, but using \_FLUPDATE command. The regular image firmware is in file "gr8net-fpga-reg.bin", and MP3 player firmware is in file "gr8net-fpga-mp3.bin". Note that MP3 image does not have hardware implemented for online serial flash chip update and for reconfiguration, thus if you mistakenly write MP3 image into the factory location (starting sector 0), you will need offline update to reload regular image into place.

### 10.5. Offline method of firmware update

Offline method should be used if you can not get adapter running properly to load firmware files into its RAM and respective CALL commands are not operational, or your GR8NET is having FPGA "engine" firmware prior September 2017 and has no functionality to update itself. To use these methods you will need additional devices:

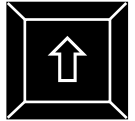
- For flash chip update you will need another storage device to load flash chip firmware image into GR8NET RAM. For example, you can use second GR8NET **in slot other than slot 1**;
- For FPGA chip update you will need GR8blaster, Byte-Blaster-II or USB-Blaster device and PC with Quartus II software installed on it. Quartus II version 13.1 is recommended – it is able to update native Altera chips (EPCS16) as well as other devices (e.g. W25Q128FVSIQ).

### 10.5.1. Offline update of the onboard flash chip

**Important:** offline update of the flash chip assumes that FPGA level firmware works properly. If it does not, you must resolve issue with FPGA before you approach troubleshooting flash chip firmware issues.

In contrast with firmware update using \_NETFWUPDATE command described above, this method assumes that GR8NET card flash chip's image is corrupt and does not initialize and does not operate. If you use Turbo-R machine please ensure that you perform update in Z80-compatible mode.

If card hangs machine during its initialization you have an option to skip GR8NET's startup. It has special logical page 0F0h which is presented by the main engine (FPGA chip) to the CPU when BIOS initialization starts. Code in this page checks for arrow up key press, and if key is pressed, skips change of the page to code of image provided by the flash chip, thus not initializing GR8NET adapter using possibly corrupt flash chip image.



Thus if you suspect that adapter's firmware stopped working properly, turn machine off, remove all the external cartridges and devices except storage device, put GR8NET adapter under question **into the slot 1**, turn machine on holding arrow up key, and wait for BASIC prompt. Then run **fwupdate.bas** utility from the attached storage device.

```
MSX BASIC version 2.1
Copyright 1986 by Microsoft

Disk BASIC version 1.0
Ok
run"fwupdate.bas
GR8NET adapter update program
Ensure GR8NET is installed
in slot 1 and press ENTER key
 0  1  2  3  4  5  6  7  8  9 10 11
12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64 65
1: 10:16:00:04:05:06
2: 10:16:00:04:05:06
3: manual FF:FF:FF:FF:FF:FF
Choice>

color auto goto list run
```

Figure 10. FWUPDATE.BAS utility screen output

---

Utility will read firmware flash image into GR8NET's RAM buffer, and request to confirm MAC address. Option 1 is sourced from configuration logical page FFh, option 2 is sourced from the current image of the flash chip (logical page BFh), and option 3 allows you entering manual value if both options 1 and 2 are invalid. You obtain valid MAC address from the serial number of the GR8NET cartridge as shown on fig. 4. In the manual prompt you should enter 12 hexadecimal digits without spaces and without colons, for example

? 101600040506←

Then you confirm MAC address choice, and utility flashes the flash chip with the image of GR8NET buffer RAM.

Firmware update utility FWUPDATE.BAS together with binary image of the flash chip and accompanying mandatory executable binaries are available for download from GR8BIT website.

Location of the files:

- <http://www.gr8bit.ru/software/firmware/GR8NET/blodable/>
  - Executable files FWUPDATE.BAS, FWUPDATE.BIN, SAVEMAC.BIN;
  - Firmware image files UPDATExx.BIN (66 files in total);
  - Or RAR archive containing all these files above;
- <http://www.gr8bit.ru/software/firmware/GR8NET/>
  - UPDATE.BIN firmware image cumulative file (528KB) to be \_NETBLOADED in case GR8NET is functional.

### 10.5.2. Offline update of the FPGA firmware

If you are supplied with the update for the GR8NET engine – FPGA chip, you can apply the update using GR8blaster (design is [here](#), or refer to [GR8programmer design](#) made by [Emil Sokolowski](#)), Byteblaster-II (was being supplied within batch #1) or USB-Blaster device (should be obtained separately to GR8NET). These devices are further referred as Blaster device.

Using Blaster device requires PC with installed programming software on it – either standalone Altera Programmer, or a bundle of Altera Quartus II software (version 13.1 is recommended). This software is available for download from Altera website for free through online registration.



**WARNING!** Connecting wires between programming device – Blaster and GR8NET card should be performed with **all the related devices disconnected from the power mains** (not just powered down).

In case of improper grounding – e.g. MSX PC having 2-wire power cord and thus NOT having protective ground connection, PC/notebook Blaster is connected to having OR not having connection to protective ground, monitor connected to the MSX computer having OR not having connection to protective ground – may cause AC voltage up to 127VAC between signals and thus damage devices being connected – GR8NET and Blaster.



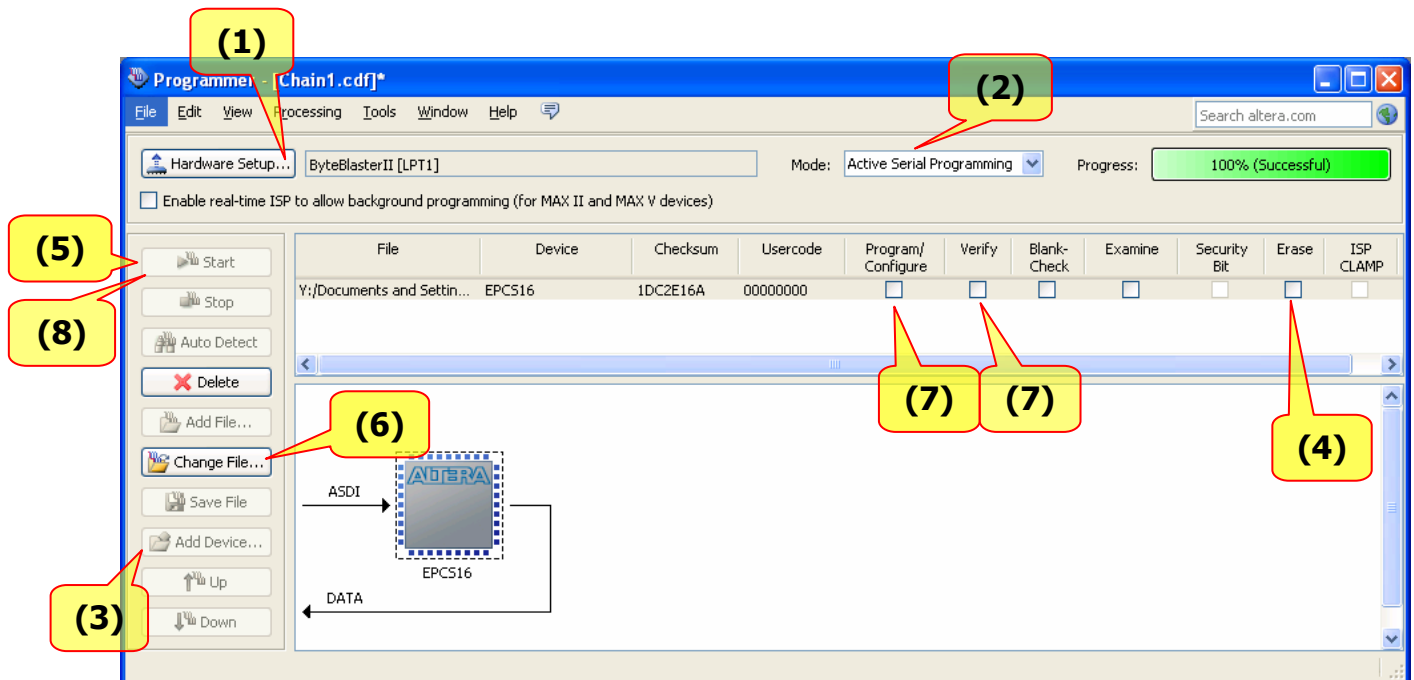
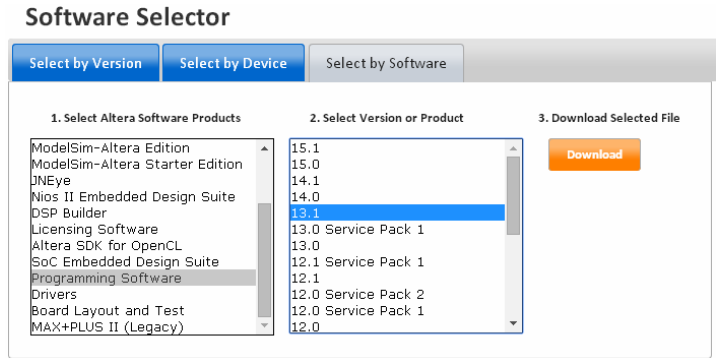
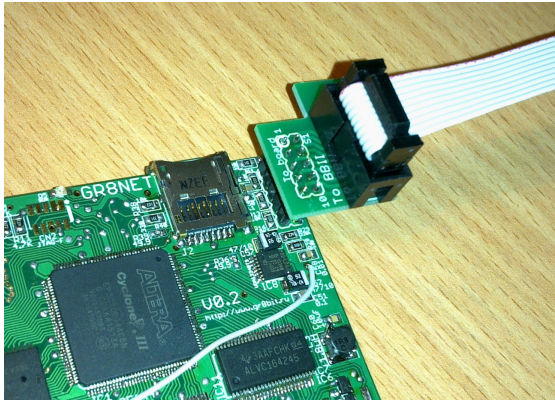
---

If you choose to use GR8blaster device for update, please follow the instructions specified in the [KB0018: GR8blaster device for MSX](#).

If you are using ByteBlaster-II device of USB-Blaster device, please follow the steps to achieve successful update:

1. Download Altera Programmer from [Altera's download center](#) and install it (fig. 11b);
2. Prepare FPGA update image, it is located in GR8NET firmware update directory being RAR archive and having POF extension;
3. Unscrew cartridge, remove board from the casing. Be very careful as device is sensitive to the ESD;
4. Disconnect MSX PC and its connected devices from the power mains. It is allowed leaving only PC with Quartus II software with parallel port powered on;
5. Connect Blaster device to its ribbon cable, and ribbon cable to the connector adapter, and connector adapter to the GR8NET as shown on the fig. 11a. Ensure you connected adapter properly and there're no hanging or misaligned pins of the adapter;
6. Insert GR8NET into any slot of MSX PC;
7. Carefully connect Blaster device to the parallel port of the PC having Altera programmer (or full Quartus software package) installed;
8. Connect MSX PC to the power mains, turn it on: programming procedure will require power from MSX PC to GR8NET. It may appear that GR8NET will not initialize because Blaster will immediately hijack operation of the FPGA (this is how Blaster was designed by Altera). Do not worry about this issue and proceed to next step;
9. Start Altera Programmer application (fig. 11c):
  - (1) set up ByteBlaster-II or USB-Blaster device and choose it;
  - (2) select Active Serial mode;
  - (3) Click *Add Device*, and select EPCS16 from the list;
  - (4) Select *Erase* checkbox (only);
  - (5) Click *Start* to start erasing entire EPCS16 chip. This step is required in order to delete all the contents and any artifacts within the chip, a kind of "start from scratch";
  - (6) After erase, click *Change file...* and select appropriate POF file;
  - (7) Now file to flash with is selected, check boxes *Program/Configure* and *Verify*;
  - (8) Press *Start* again to start programming and verification operations.
10. Device will be unavailable while programming is in progress, and if MSX was using the device, it may crash;
11. Power MSX PC off and disconnect it from the mains, disconnect Blaster device from PC's port, disassemble connection chain in backward order;
12. Turn MSX PC on and check that GR8NET initializes (e.g. looking at the SD-card activity LED when SD-card is inserted);
13. Put GR8NET board into the cartridge casing, carefully screw it, do not overtighten.





## 10.6. Online migration from version 0.7 to version 0.9

Note: you must migrate from 0.7 to 0.8 first using steps provided below, and then to 0.9 using online update methods for both FPGA chip and GR8NET flash chip.

Version starting 0.8 is different in the respect of handling FPGA serial flash chip, and version 0.7 has factory image protected from updating, and it is not that straightforward to migrate to it from version 0.7.

You will have to use intermediate update image in order to update image at factory location because engine version 0.7 is having factory image location write-protected. Thus

---

first you will load 0.8 transitional image into location 1 so that location 0 (factory) still contains image of version 0.7. After power cycle machine will configure with factory image version 0.7, and then automatically reconfigure to image at location 1, which is new version 0.8 window. And then, running version 0.8 image, you can update image at factory location.

Action plan:

1. Install GR8NET into the MSX machine, turn it on, press and hold arrow down key until GR8NET goes to BASIC;
2. Perform `_NETVER` command to ensure that Engine and system run regular image version 0.7 (both "Flash" and "Engine" report v.0.7);
3. Perform `_NETBROWSE` and load image named *gr8net-fpga-trs.bin* from the v0.7-final/ subdirectory of the firmware files location, or download this image by typing:  
`_NETBLOAD("http://www.gr8bit.ru:80/software/firmware/GR8NET/v0.7-final/gr8net-fpga-trs.bin")`
4. Write downloaded image to the location 1, issuing `_NETFPGAUPD(1)`. The command will ask for confirmation, and then write the image into sectors 8-15 of the serial flash. There will be no reboot after this operation;
5. Now update flash chip by loading update.bin file (using `_NETBROWSE` or `_NETBLOAD`), and issuing `_NETFWUPDATE(3)`, for example:  
`_NETBLOAD("http://www.gr8bit.ru:80/software/firmware/GR8NET/update.bin");_NETFWUPDATE(3)`
6. After flashing process is complete, and it will ask to press any key to reboot machine, power machine off and then on instead. GR8NET will load FPGA image version 0.7 from factory location first, then will reconfigure to location 1, which already contains version 0.8;
7. Now you load regular FPGA image into GR8NET RAM, and issue `_FLUPDATE` command with argument of sector 0:  
`_NETBLOAD("http://www.gr8bit.ru:80/software/firmware/GR8NET/gr8net-fpga-reg.bin");_FLUPDATE(0)`
8. It will ask if you really want to overwrite factory image, answer Y (capital letter Y). Ensure there will be no power interruptions during re-flash. After flashing is complete, power machine off and then on and ensure using `_NETVER` command that machine is now running both "Flash" and "Engine" of version 0.8, and image type of "factory" and sector number indicated is 0.
9. Update process is complete, now you can perform step 7 for MP3 player image *gr8net-fpga-mp3.bin* using starting sector of 4 – after this operation you will have MP3 player image in sectors 4-7, and remaining space from sector 8 to 31 (1.5MB – if you have EPCS16 chip installed) is available for other images or catalogs.

---

## 11. GR8NET technical reference

GR8NET employs several modern technologies and is a relatively complex device in its programming and management of its resources. If you experience difficulties with it, please read/search through the manual (it is searchable using ^F), or contact developer or community for more information and support.

### 11.1. Identification and detection

You MSX computer may have up to 4 GR8NET adapters installed. Each adapter is physically preset with ID number by the switch located in the edge connector's cartridge box window (fig. 5 and 12). Adapter ID is set in binary system, with switch being turned ON designating digit 1, thus both switches turned off identify adapter #0, and both switches turned on identify adapter #3.

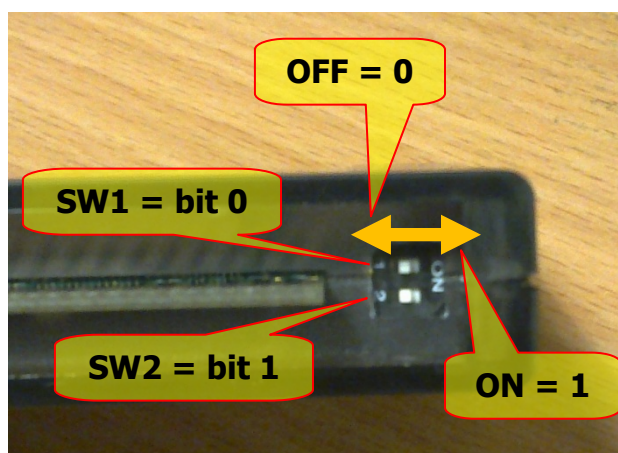


Figure 12. Configuration switches

Configuration, or *activation*, of all the GR8NET adapters in the system is performed by firmware using two I/O ports – 5Eh and 5Fh. Only one *selected* adapter can be chosen to be accessed through I/O ports at the time however application still can access adapters' features available in their respective memory (slot) space.

Port 5Eh: adapter and register selection

7	6	5	4	3	2	1	0
Selected adapter ID		Default adapter ID		Register ID			
SAID1	SAID0	DAID1	DAID0	RID3	RID2	RID1	RID0

The following fields are defined in the register 5Eh:

Field	Description
SAID	Adapter being selected, the one which will be further accessed through I/O port 5Fh
DAID	Default adapter (see <a href="#">Operating in multi-adapter environment</a> chapter). These bits should not be changed by the application
RID	Register index being selected to be accessed through port 5Fh

When adapter is not yet initialized, it accepts writes to the port 5Eh, but will not respond to port 5Eh or port 5Fh reads and perform no action on port 5Fh writes.

When adapter was initialized successfully, it will respond to port 5Eh and port 5Fh reads if its switch configuration matches field SAID, and it will write to registers indexed by RID through port 5Fh.

This mechanism allows selection of specific adapter for I/O access, and provides a way for *deactivation* of the adapter in case there's conflicting configuration between multiple adapters installed into the system.

When port 5Eh is read, adapter identified by the SAID will return previously written value. Logically, all installed adapters are expected to keep same copy of the port 5Eh contents.

Port 5Fh reads from or writes to register pointed by the value in RID field of port 5Eh register of the adapter identified by SAID field. Register allocation is the following:

Index (hex)	Op	Function
0	R	If adapter is present and activated, this register reads as the following data in loop: 'G' (47h), 'R' (52h), '8' (38h), 'N' (4Eh), following by GR8NET engine's (FPGA) build year (word, little-endian), month (byte) and day (byte). The sequence is reset by writing index 0 to port 5Eh. If adapter is inactive 0ffh is returned, and all other registers are n/a
1	R/W	This register is initialized by the GR8NET firmware, and should hold adapter's slot ID in RDSLT format. Thus application, by setting SAID bits, can identify the location of the adapter in the CPU memory space. Application should <b>NOT</b> write to this register
2	R/W	Mapper register, see <a href="#">Setting operating mode and mapper type</a> chapter
3	R	Major version of the hardware (e.g. 0)
4	R	Minor version of the hardware (e.g. 8)
5	W	PCM replenishment data byte, functions the same as <i>PCM data register</i>
	R	Prefetch data register, functions the same as <i>prefetch data register</i> in special control register set
6, 7	-	Control (6) and data (7) registers of MP3 decoder in the MP3 player FPGA image (not used in regular FPGA image)
8-E	-	Reserved
F	-	Unavailable

---

GR8NET adapters should be used in the following way:

1. User orders adapters using their identification switches from #0 to #3. Numbering order may not be contiguous;
2. User attaches specific network cables to specific adapters, thus there will be clear association between adapter # and network it is physically connected to;
3. Application is expected to know adapter # connected to each network, and performs enumeration of the adapters using port 5Eh bits SAID to see if required adapter is present ("GR8N" from register 0, or just first 'G' for easiness) and properly initialized (valid slot ID in register 1 – not 00 and not FF);
4. Application, using slot ID got from each adapters' register index 1, initializes these adapters to work with predefined subnetwork, and uses memory access to the adapters' resources;
5. If application needs to change specific adapter's mapper type (see [Setting operating mode and mapper type](#) chapter), it selects adapter using SAID bits and writes to its register 2. Note that mapper type change takes effect immediately.

## 11.2. Setting operating mode and mapper type

Now we know how to detect and manage multiple GR8NET adapters in the system, we will consider how to manage adapters individually.

Adapter management starts with its I/O mapper register 2 (see also [Identification and detection](#) chapter).

Mapper register

7	6	5	4	3	2	1	0
Special control register appearance				Memory mapper type			
Bank 3	Bank 2	Bank 1	Bank 0				

CPU visible *memory* space of the GR8NET adapter is divided into four locations – GR8NET banks: 4000-5FFF (GR8NET bank 0), 6000-7FFF (GR8NET bank 1), 8000-9FFF (GR8NET bank 2) and A000-BFFF (GR8NET bank 3). Please do not confuse these with CPU memory banks, which are 0000-3FFF (CPU bank 0), 4000-7FFF (CPU bank 1), 8000-BFFF (CPU bank 2) and C000-FFFF (CPU bank 3).

Exception to banking system explained above is ASCII16 mapper mode, however internally to GR8NET this mode is a clone of ASCII8 mode with predefined second 8K logical page number.

**Important:** here and further we will be talking about GR8NET banks. If we will need to refer to CPU banking subsystem, it will be explicitly stated as "CPU bank" or "Z80 bank".

**Special control register appearance:** special control registers is the area of the visible CPU memory which overlays the data currently visible in the specific bank. This means that having turned these registers on in bank 0 makes currently selected bank 0's page's data behind these registers inaccessible for reads and for writes.

If bit 7 of mapper register is set, then special control registers appear at the end of bank 3; if bit 6 is set, then *same* special control registers appear at the end of bank 2; if

bit 5 is set, then *same* special control registers appear at the end of bank 1; and if bit 4 is set, then *same* special control registers appear at the end of bank 0.

GR8NET mapper uses control registers in bank 0 only, together with ROM's physical page 0 (logical page 80h). This ROM page has no meaningful data or code in the ROM area shadowed by special control register.

Important: you can turn special control registers on in any mapper type except mapper type 7 "Mapped RAM". Four bank change registers present in the special control register set will reflect current pages set in banks, and are writable providing additional method for changing the pages in the specific 8KB bank.

**Memory mapper type:** 4 bits representing the mapper currently in effect. Mapper type change takes immediate effect when written to.

GR8NET may function in several mapper configurations, providing programmer a couple of standard and custom functionalities. The following mapper modes are available:

#	MM Type	Description
<b>Simple mappers, can be present in primary or expanded slot</b>		
0	GR8NET mapper	Card operates as the GR8NET internetworking adapter
1	Plain write-protected 32K ROM	Presents write-protected logical pages 0, 1, 2 and 3 (first 4 RAM pages) in GR8NET banks. Examples of software using this mapper are King's Valley, Nightmare and many more
2	Konami without SCC (K4 mapper)	256 Kbytes memory mapper. 5 significant page number bits, 32 pages, bank 0 is fixed at page logical 0 (first RAM page). Examples of the software using this mapper are Nemesis, Penguin Adventure, Treasure of Uşas, Metal Gear
3	Konami with SCC (K5 mapper)	512 Kbytes memory mapper. 6 significant page number bits, 64 pages. SCC is contained in page 63 overlaying presented logical page 63 contents. Alternatively, if SCC is switched to SCC+ mode, it appears in pages 128 and greater. Examples of the software using this mapper are Nemesis 2, King's Valley 2, Metal Gear 2, SD-Snatcher
4	ASCII 8	1 Megabyte memory mapper (all presented by onboard RAM). 7 significant page number bits, 128 pages. Examples are Auf Wiedersehen Monty and Arkanoid II
5	ASCII 16	1 Megabyte memory mapper (all presented by onboard RAM). 6 significant page number bits, 64 by 16K pages. Switching to page #x presents logical pages x*2 and x*2+1 in respective 16 Kbyte bank location
6	Mirrored ROM	GR8NET RAM buffer pages 0/1 contents are presented in CPU address space 0000-3FFF, pages 2/3 are presented in CPU address space 4000-7FFF, and pages 4/5 are presented in CPU address space 8000-BFFF.
7	Mapped RAM	Provides 1 MByte standard mapped RAM. Note than GR8NET responds to reading of memory mapper ports FCh-FFh according to the second option set by _NETSETMAP command.



Below is the list of the composite mappers; in these modes GR8NET performs slot expansion, thus GR8NET adapter itself must be installed in the primary slot (not into slot expander). These modes are specifically designed for gaming and applications which run as ROMs but require access to network (GR8NET mapper), storage (Nextor) and wants to have more RAM (512K). In these modes functions are allocated in the following way:

- Subslot 0: GR8NET mapper with 512K RAM. Note that in modes 9-14 GR8NET RAM mirrors contents of game mapper in subslot 3, thus writing to GR8NET RAM will modify information visible to CPU in subslot 3 (for example invoking \_NETBROWSE will overwrite several pages at the beginning; loading disk image using built-in Disk-ROM will also corrupt original information in specific designated logical pages);
- Subslot 1: 512KB mapped RAM (can be disabled);
- Subslot 2: Nextor storage subsystem;
- Subslot 3: see table below.

#	MM Type	Description
<b>Composite mappers, <span style="color: red;">must</span> be present in primary slot only</b>		
8	FM-Pak	Subslot 3 contains FM-Pak ROM
9	Plain write-protected 32K ROM	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 1
10	Konami without SCC (K4 mapper)	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 2
11	Konami with SCC (K5 mapper)	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 3
12	ASCII 8	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 4
13	ASCII 16	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 5
14	Mirrored ROM	Provides GR8NET RAM (seen in read/write mode through subslot 0) the same way as mapper mode 6
15	Illegal	Do not use this mapper type

Changing mapper ID through I/O:

- GR8NET mapper type is automatically initialized with logical page numbers 80h, 81, 82h and 83h;
- ASCII-8/ASCII-16 mapper types are initialized with page numbers 0;
- Other mapper types are initialized with logical page numbers 0, 1, 2 and 3 respectively.

Changing mapper ID through banking special control registers:

- The only automatic change is changing to K4 mapper when bank 0 is automatically initialized with logical page 0;
- Before changing ensure that you have set correct page numbers in special control registers to continue proper execution;



### 11.3. Logical page assignment

Every bank can be set to display specific *logical* page contents. Logical pages – from number 0 to number 255 (0FFh in hexadecimal) – are the set of contents of physical pages of the onboard GR8NET devices mapped to the CPU memory space. These devices are ROM and RAM chip, W5100 chip, and special pages provided by the GR8NET Engine (FPGA).

Page mapping is performed in the following way:

Logical page # (hex)	Number of pages (dec)	Function	Description
FF	1	System configuration and variable page	8Kbytes of RAM for GR8NET configuration data implemented using Altera Cyclone III
FE	1	SCC/SCC+ (sound custom chip)	Available in any GR8NET bank with register area start at 01800h and mode register at 01FFEh-1FFFh
FD F1	13	(Reserved area)	Empty, returns 0FFh
F0	1	GR8NET boot page	This page is presented to CPU after hard reset, called on GR8NET initialization start
EF CA	38	(Reserved area)	Empty, returns 0FFh
C9	1	SD-card status and MathPack	Presents diagnostic information on the SD-card operation and mathematical functions (MathPack)
C8	1	SD-card data buffers	4 Kbytes RAM (presented twice in 8K CPU window)
C4	4	PCM buffers	32 Kbytes of RAM implemented within Altera Cyclone III chip for PCM streaming audio
C0	4	W5100 pages	C0 is W5100 registers space, C2 is TX buffers space, and C3 is RX buffers space
BF	1	Configuration data	Adapter configuration stored in flash chip
BE B8	7	GR8NET logo	See GR8LOGO.ASC in the examples chapter
B7 B6	2	OPLL ROM	OPLL ROM appears in subslot 3 in mapper mode 8 when OPLL is not disabled using _NETSETOPLL
B5 A2	20	(Reserved area)	
A1 9A	8	GR8NET code	Routines supporting operation of the GR8NET adapter
99 8A	16	Nextor	Nextor software is available in composite mapper modes 8-14
89 88	2	Disk-ROM	MSX-DOS version 1 integrated Disk-ROM
87 81	7	GR8NET code	Routines supporting operation of the GR8NET adapter
80	1	Adapter ROM BIOS main page	Main page of the ROM presented in GR8NET bank 0 during adapter operation in mapper modes 0 and 9-14
7F  00	128	Adapter buffer RAM	Contents of GR8NET 1MB RAM chip; the space is allocated on GR8NET initialization in mapper modes 0 and 8-14 (see <a href="#">Memory manager</a> chapter), as per game mapper specification in mapper modes 1-6, and serve as mapped RAM in mapper mode 7

---

When mapper requiring 16 Kbyte page is used (ASCII-16) GR8NET engine sets two consecutive 8 Kbyte pages to values (x) and (x+1).

After hard reset (RESET signal activation due to reset button press and release, or power cycle) bank 0 is set to logical page F0h corresponding to boot page provided by the GR8NET engine (FPGA chip), which checks for arrow up key press and skips initialization of pressed, or proceeds with further initialization switching to logical page 80 (first physical ROM page) if arrow down key is not pressed.

## 11.4. Special control registers

As explained in [Setting operating mode and mapper type](#) chapter, special control registers can reside at the end of in any 8 Kbyte GR8NET banks, cause logical page's contents behind these registers accessible. Application may switch these *same* registers on in any one or more banks it needs, or turn them off in all banks (like it is done for e.g. Konami memory mappers by default).

Register address				Function	Description
B0	B1	B2	B3		
SD-card management					
5FC0	7FC0	9FC0	BFC0	Status	Shows SD-card access machine status and allows manual reset of the card
5FC1	7FC1	9FC1	BFC1	Sector buffer position	Defines the place in the buffer where sector data (will) reside
5FC2 5FC3 5FC4 5FC5	7FC2 7FC3 7FC4 7FC5	9FC2 9FC3 9FC4 9FC5	BFC2 BFC3 BFC4 BFC5	Sector # (32-bit value)	512-byte Sector number to perform operation on
5FC6	7FC6	9FC6	BFC6	Sector count	Number of sectors to process (0=no op, max 8)
5FC7	7FC7	9FC7	BFC7	Command/status register	Starts SD-card R/W process
5FC8 ... 5FCB	7FC8 ... 7FCB	9FC8 ... 9FCB	BFC8 ... BFCB	FPGA flash chip interface	Used to access FPGA serial chip (read, erase and write sectors)
5FCC ... 5FCF	7FCC ... 7FCF	9FCC ... 9FCF	BFCC ... BFCF	SD-card size	32-bit size of SD-card in 512-byte sectors
5FD0 ... 5FD2	7FD0 ... 7FD2	9FD0 ... 9FD2	BFD0 ... BFD2	Reserved	
5FD3	7FD3	9FD3	BFD3	PSG volume	Linear volume from 80h (full sound) to 0 (mute)
System registers and audio settings					
5FD4	7FD4	9FD4	BFD4	System mode register 1	Controls deconfiguration of Y8950, its hardware interrupt generation and I/O port location
5FD5 5FD6	7FD5 7FD6	9FD5 9FD6	BFD5 BFD6	Mixer	Identifies channel audio from GR8NET devices go into
5FD7	7FD7	9FD7	BFD7	Y8950 volume	Linear volume from 80h (full sound) to 0 (mute) (see also Volume section of this table)

Register address				Function	Description
B0	B1	B2	B3		
5FD8	7FD8	9FD8	BFD8	MSX-Audio settings (ADPCM RAM)	Starting RAM logical page number
5FD9	7FD9	9FD9	BFD9		Current RAM size in 8kB pages
5FDA	7FDA	9FDA	BFDA	OPLL volume	Linear volume from 80h (full sound) to 0 (mute) (see also Volume section of this table)
System registers					
5FDB	7FDB	9FDB	BFDB	System mode register 0	Mapper read, clock source, OPLL and Y8950, 2xVolume and others
5FDC	7FDC	9FDC	BFDC	Error register	Register keeping error code for last network subsystem operation
5FDD	7FDD	9FDD	BFDD	P5EDATA	Contents of the register written to port 5Eh
5FDE	7FDE	9FDE	BFDE	Mapper ID	Mapper ID register, same as appears in index register 2 through I/O
5FDF	7FDF	9FDF	BFDF	Adapter ID	Adapter identification
Bank switching					
5FE0	7FE0	9FE0	BFE0	Switchable bank 0	Identify logical page numbers presented in respective banks within GR8NET mapper. Game mappers have different register set
5FE1	7FE1	9FE1	BFE1	Switchable bank 1	
5FE2	7FE2	9FE2	BFE2	Switchable bank 2	
5FE3	7FE3	9FE3	BFE3	Switchable bank 3	
Configuration					
5FE4	7FE4	9FE4	BFE4	My slot ID	Slot ID of the adapter in RDSLT format. Appears in I/O index register 1
5FE5	5FE5	9FE5	BFE5	Bank2 slot ID	Storage area to preserve bank 2 slot identifier when switching bank 2 to GR8NET slot
Volumes					
5FE6	7FE6	9FE6	BFE6	Master DAC volume	GR8NET DAC linear volume. All volumes are in range of 080h (maximal, unmodified sample data) and 0h (mute)
5FE7	7FE7	9FE7	BFE7	SCC volume	SCC linear volume
5FE8	7FE8	9FE8	BFE8	Digital waveform/MM DAC input volume	Linear volume for the digital data in the register space 5FEF:5FEE
5FE9	7FE9	9FE9	BFE9	PCM volume	Linear volume for the PCM playback

Register address				Function	Description
B0	B1	B2	B3		
Controlled interrupt generator with watchdog					
5FEA	7FEA	9FEA	BFEA	Control register	Start/stop, run, mode
5FEB	7FEB	9FEB	BFEB	24-bit timer counter value/Frequency (Must read LSB first, must write MSB last)	Controlled interrupt generator with watchdog
5FEC	7FEC	9FEC	BFEC		
5FED	7FED	9FED	BFED		
Waveform input					
5FEE	7FEE	9FEE	BFEE	Digital waveform input (16-bit) Shared with MM DAC, signed integer (Must write LSB first, for 8-bit samples writes MSB only)	This register is used to mix within internal GR8NET DAC
5FEF	7FEF	9FEF	BFEF		
PCM function					
5FF0	7FF0	9FF0	BFF0	Control register	Start/stop, mode, status
5FF1	7FF1	9FF1	BFF1	PCM buffer bytes free (Must read LSB first)	PCM function pointers
5FF2	7FF2	9FF2	BFF2		
5FF3	7FF3	9FF3	BFF3	PCM buffer write pointer (Must write MSB last)	
5FF4	7FF4	9FF4	BFF4		
5FF5	7FF5	9FF5	BFF5	PCM data register	Sequential replenishment of the PCM memory
5FF6	7FF6	9FF6	BFF6	(also available @ port index 5)	
Prefetch function					
5FF7	7FF7	9FF7	BFF7	Control register	Control and status
5FF8	7FF8	9FF8	BFF8	Low 16 bytes of the start/current memory pointer	Prefetch function pointers
5FF9	7FF9	9FF9	BFF9		
5FFA	7FFA	9FFA	BFFA	Low 16 bytes of the end memory pointer or mask	
5FFB	7FFB	9FFB	BFFB		
5FFC	7FFC	9FFC	BFFC	High nibble: high 4 bytes of end pointer, low nibble: high 4 bytes of start/current pointer	
5FFD	7FFD	9FFD	BFFD	Two locations (16-bit word) for prefetched memory data	Prefetch data word
5FFE	7FFE	9FFE	BFFE		
Adapter mode					
5FFF	7FFF	9FFF	BFFF	Adapter flags	Used by the firmware to identify the behavior of specific functions

---

## 11.5. Hardware-accelerated functions

There's a set of useful functions programmer can use to accelerate access to the card's resources, and implement specific multimedia solutions. These functions, as well as some card's configuration parameters, are accessed through special control registers.

### 11.5.1. Controlled interrupt generator with watchdog

Controlled interrupt generator provides the feature to generate controlled timing to PCM function, controlled *hardware* interrupt and timing to the applications using polling of generator's status register. Interrupt counter can use 100 nanoseconds clock period (10 MHz frequency), or use the same clock feeding audio devices in GR8NET (internal 3.579545 MHz generator). The timing properties of the interrupts are calculated using the following formulae:

- at 10 Mhz period  $T_{\text{int}} = 10^{-7} * n$  ;
- at 3.579545 Mhz period  $T_{\text{int}} = 279.365 * 10^{-9} * n$  ;
- frequency  $f_{\text{int}} = \frac{1}{T_{\text{int}}}$  .

There're two modes generator can operate in: countdown timer and frequency generation.

In countdown timer mode timer counter value is 24-bit and thus can have values from 0 to 16,777,215. Sample values are listed below.

Countdown timer @10 Mhz (CLS bit=0)

Counter	0	10	128	227	453	907	1250	1000000	16777215
$T_{\text{int}}$ , sec	1.6777216	1 $\mu$ s	12.8 $\mu$ s	22.7 $\mu$ s	45.3 $\mu$ s	90.7 $\mu$ s	125 $\mu$ s	1	1.67772215
$f_{\text{int}}$ (Hz)	0.596	1000000	78 125	44 053	22 075	11 025	8 000	1	0.596

Countdown timer @3.579545 Mhz (CLS bit=1)

Counter	0	10	46	81	162	325	448	3580000	16777215
$T_{\text{int}}$ , sec	4.6863732	2,79 $\mu$ s	12,85 $\mu$ s	22,63 $\mu$ s	45,25 $\mu$ s	90,79 $\mu$ s	125,14 $\mu$ s	1	4.6863729
$f_{\text{int}}$ (Hz)	0.2133846	1000000	77 826	44 198	22 099	11 015	7 991	1	0.2133846

In frequency generation mode value is 16-bit, with most significant byte (bits 23-16) always being 0.

Frequency generator (always uses 10MHz clock, bit CLS has no effect)

Frequency	0	10	50	8000	11025	22050	44100	65535
$T_{\text{int}}$ , sec	15.259 $\mu$ s	0.1	20 ms	125 $\mu$ s	90.7 $\mu$ s	45.4 $\mu$ s	22.7 $\mu$ s	15.26 $\mu$ s
$f_{\text{int}}$ (Hz)	65535	10	50	8 000	11 025	22 026	44 052	65 531

---

In order to make new counter/frequency value effective, you have to write 24-bit's MSB, even if it is the same as was written before. On this write whole 24-bit value is latched into the generator.

In case you need longer interrupt period, you can implement interrupt counting in software. For example, in case you need event to occur each minute, you set interrupt generator to 1 second, and count 60 interrupts.

Reading 24-bit counter will give application the current countdown value at the time when LSB was read. Application should read LSB first, and MSB last to get stable correct value. Programmer should make adjustment for counter value reading and processing operation time to identify more or less definitive countdown stage of the timer/generator.

The GR8NET card as a source of the interrupt is not standard for the MSX PC, and MSX PC standard interrupt service routine does not clear the interrupt from GR8NET. It presents specific risk when interrupt generator is started but for some reason is not serviced, GR8NET will keep interrupt wire active, causing MSX interrupt service routine constantly being called. It will look like machine hangs.

Watchdog mechanism is introduced preventing failures arising from recursive interrupt service routine calling. It will keep hardware interrupt line active for another countdown period after interrupt line activation. If this period expires and application does not clear interrupt condition, if hardware interrupts are enabled, watchdog deactivates hardware interrupt line and suspends generator; however if hardware interrupts are disabled watchdog does not suspend generator. In both cases watchdog sets generator error bit in control/status register. Main code of application will need to sense this error condition and restart generator if it has shut down.

If hardware interrupts are enabled, when counter reaches zero (in frequency mode countdown number is internally calculated within generator circuit) hardware interrupt is generated. In IM1 (interrupt mode 1) call to physical address 0038h is performed in case CPU's maskable interrupts are enabled.



Interrupt generator control register (5FEA) has the following format:

Bit	Name	Description
0	OP	Generation operation control W: set to 1 resume counting and reset to 0 to suspend counting R: is set to 1 if interrupt generator counter is counting
1	RST	Reset control W: Set to 1 to reset generator internal counter. If you set OP bit simultaneously with setting RST bit, counter will reset and generator restart counting using new values. Resetting counter also resets ER and INT bits R: n/a
2	HZP	Period of Hz mode W: If reset to 0, generator treats 24-bit value in counter register as timer countdown counter; if set to 1 then counter register's low 16-bits (first two bytes) are treated as frequency in Hertz R: Value written before
3	WDE	Watchdog enable bit. W: set this bit to 1 to enable watchdog R: Value written before
4	IGE	Hardware interrupt enable bit. W: set this bit to 1 in order to enable hardware interrupt line activation R: Value written before
5	CLS	Generator clock source. W: If reset to 0, generator uses 10 MHz clock (100 ns period) for countdown, if set to 1 generator uses built-in 3.579545 MHz clock (~279 ns period) R: Value written before
6	ER	Interrupt generator error. W: write 1 to clear this error, writing 0 has no effect R: is set to 1 if counter stopped counting because its interrupt request was not cleared in time and timing is disabled by watchdog. Condition is cleared by writing 1 into this bit. Writing to bit 1 (RTS) also clears this condition
7	INT	Interrupt occurrence flag. W: write 1 to clear this bit (mean interrupt was serviced), writing 0 has no effect R: Set to 1 if interrupt generator is the source of the hardware interrupt. The condition is being cleared by writing 1 to this bit

To use controlled interrupt generator:

1. Disable generator by writing RST=1 and OP=0;
2. Choose source and mode you will be using: source can be 10 MHz clock or 3.579545 MHz clock, and mode can be countdown count or frequency in Hertz. Calculate 24-bit value accordingly, and write 24-bit generator counter register. MSB of this register should be written last, even if it is 0, because whole 24-bit value is internally latched

---

on the write to this MSB byte location. Note that in HZP=1 (Hertz) mode MSB is not effective, and should always be written as 0;

3. Decide if you will need hardware interrupt. If you do not need it, skip to step 5.
4. Set up interrupt handling routine. In IM1 mode Z80 jumps to address 0038h. In DOS mode you can change the jump instruction at location 0038h, but in BASIC mode the jump is in ROM and you will have to find another way to your interrupt handling code – use standard hooks (HKEYI or HTIMI), switch slot in the bank 0 to the RAM and process interrupt code in this RAM, or use different CPU interrupt mode – IM2. In case you will use standard hooks ensure timing requirements are met because standard service routine procedure starts with a number of push instructions, keyboard and other device handling. In case you will write your own interrupt service routine from the scratch do not forget to poll for VDP interrupt and clear its condition otherwise machine will fail due to improper handling of the VDP interrupt;
5. Decide if you need to disable watchdog (WDE bit). You may need it disabled to have hardware interrupts line being active until they are served (e.g. if you use code which massively uses interrupt masking, for example in disk I/O).
6. Start counter writing RST=1, OP=1, WDE, HZP, IGE and CLS bits simultaneously;
7. Your interrupt handling routine should write 1 to INT bit to clear interrupt condition and reset hardware interrupt line (if IGE is 1) in case this bit INT reads 1. Application code should monitor condition of ER bit, if it is set, then one of more interrupt periods are missed. In case watchdog is enabled, it is task of main code to monitor ER bit, because watchdog suspends interrupt and interrupt service routine will not be called any more. To clear ER flag write 1 to this bit. If application (or programmer) notices that ER flag is set, they may choose to decrease frequency of interrupts, or redesign the code;
8. When generator is not needed any more, perform step 1, restore all hooks and slot assignments.

Important notes:

- Code should clear interrupt condition **before enabling maskable interrupts**;
- When application reads interrupt generator control register to obtain status information, this reading takes time, and further processing of the information read from this register also takes time, and value you have read may have expired every nanosecond after you have read it. Thus if ER bit is not set, you are *not* guaranteed that it is not got set when you started processing the status. The certain condition is when you read the status register and it has ER bit set – it 100% means that there was and is an error condition. Therefore, please design you code in the way that it services generator's interrupt request in time, and ensure main application code regularly monitors status of generator and restarts it if needed.
- Implementation of the interrupt service routine may look the following way:

3A	EA5F	LD	A,(05FEAh)
07		RLCA	
38	xx	JR	C,xx

This code, instead of masking/testing bit in register A, shifts INT bit in A into the CY flag and tests this flag.

---

### 11.5.2. PCM function

GR8NET provides feature of streaming PCM data into its mixer basing on the contents of the PCM buffer. This buffer is 32 Kbytes long, and is being written into the mixer using controlled interrupt generator event. At interrupt counter equal to 99 (44 053 Hz) full buffer 16-bit contents will be sent to mixer and DAC within 0.3719 seconds, if counter is 1122 (8 kHz) full buffer 8-bit contents flush will take 4.096 seconds. After the hardware reset PCM function is in reset state.

In contrast with Z80-interrupt-based implementations, when sample output time is not precise due to Z80 command execution flow and maskable interrupt management, PCM function does not have any sample output time instability, and *always* outputs samples within approximately 10 nanoseconds delay after generator's countdown counter reaches zero.

There're four registers related to the PCM function operation:

- **PCM buffer bytes free:** the 16-bit value indicates the free space in the PCM buffer *in bytes*. Check this value to identify how much data PCM buffer needs to be buffered with to continue waveform output without delays and interruptions. Important: the value is dynamically calculated during execution of PCM playback thus it is constantly being changed; to read correct value you **must** read least significant byte first, and then most significant byte. 16-bit load commands like **LD HL,(05FFBh)** are suitable for getting correct value;
- **PCM buffer write pointer:** the 16-bit value identifying the location in the PCM buffer where application should write PCM data. This pointer is tightly coupled with bytes free pointer: application should use this pointer to buffer data into PCM buffer of the *PCM buffer bytes free* count. After writing data into the PCM buffer, you should update PCM buffer write pointer.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N/A		PPN		Address within respective 8 Kbyte page (0000-1FFF)											

PCM buffer write pointer points to *physical location* within the PCM buffer, with PPN being physical page # of the PCM buffer (00b to 11b) and 13-bit address (0000-1FFF) being the position within respective physical page. When accessing PCM pages through Z80 visible banks, programmer should set up respective logical page in specific bank (page# is 0C4h+PPN), and then access 13-bit address within this page.

- PCM function control register (5FF0) has the following format:

Bit	Name	Description
0	RP	PCM reset control W: reset to 0 (default) to put PCM function into the reset state R: reset to 0 if PCM function is in reset state
1	SP	Start or suspend PCM processing. W: if set starts or continues PCM processing R: reset if PCM function was not started, suspended or ended its operation; set to 1 if PCM is processing
2	DS	PCM data size. Reset to 0 for 8-bit sample data, set to 1 for 16-bit sample data. W: data size R: data size
3	PE	Playback ended. W: n/a R: if set to 1 means that playback has ended
4	AS	Auto-stop. W: If set to 1, when PCM function plays all the data available (write pointer equals to internal read pointer), it stops, deactivating SP bit, and application will need to start further playback manually. If reset to 0, PCM function waits for new sample, and as soon as new sample arrives, it puts it into its DAC. R: Value written before
5	STE	Stereo W: sets PCM engine into stereo mode, with first sample for the left channel and second sample for the right channel R: Value written before
6		
7	HSP	R: Bit is set to 1 if PCM buffer has at least one space for sample of DS size W: n/a

- PCM function data register is write-only register allowing application to replenish PCM memory without using pointers. Application reads bit 7 of the control register to see if there's at least one space for the sample of the type set in control register's DS bit, and if bit is set to 1, writes sample to the PCM data register. Sample is being written to the position currently pointed by PCM buffer write pointer, and pointer is increased by number of bytes written. Both byte-load and word load commands are allowed on the PCM data register, but application should ensure information it writes remains aligned with sample data size set in DS;
- PCM function data register is also available through I/O index port #5 for write only. This functionality is very useful when performing OUTI or OTIR commands from the memory to PCM data register with source memory pointer increase but no change to output location (port number in CPU register C);
- When PCM is set into stereo mode, it will output first sample (8-bit or 16-bit) into left channel, and second sample into right channel.

---

Please follow the steps to use PCM function:

1. Reset PCM function by writing 0 into PCM control register;
2. Set RP bit to get PCM function out of reset condition;
3. Perform data buffering, ensuring you buffer 8- or 16-bit and single or two channel data according to further control register setting.
  - a. Using PCM bytes free and buffer write pointer: read PCM buffer bytes free (maximal value is 32766 bytes – even value to be aligned with 16-bit sample data), read PCM buffer write pointer, then set logical page number in required bank to 0C4h+PPN, and sequentially fill buffer with data. You will have to rotate PCM logical pages (0C4h-0C7h), for example after page 0C5h address 1FFF goes page C6h and address 0000, after page 0C7h and address 1FFF goes page 0C4h and address 0000; alternatively you can set two consecutive banks with consecutive pages, and do not care about crossing the boundary *between them*. After you buffered the data, increase 16-bit PCM buffer write pointer by the number of bytes you wrote. *Do not* write more data than bytes free register indicates otherwise PCM sound may become corrupt;
  - b. Using index I/O register #5 and HSP bit: read control register, and check for HSP bit. If it is set, then perform single OUTI to the index register #5. Continue doing so until data depletes or HSP bit becomes zero;
  - c. You can combine two techniques described above: use index I/O register a number of times indicated by PCM bytes free.
4. Set controlled interrupt generator for required frequency by writing 0 into IG bit and then setting values in registers 5FEB-5FE9.
5. Decide if you need generator to generate hardware interrupts, and start the generator by writing 1 to IG bit (and IGE bit as decided);
6. Start PCM playback by simultaneously writing 1 to SP bit, and setting DS bit to respective sample size and STE bit to indicate number of channels;
7. In your application monitor PCM buffer bytes free register, and if it is not zero, go to the step 3. If you preserved logical page # value and address within page, you can reuse them to write further data, but *you have to use* new bytes free count;
8. If you need to stop/suspend PCM data performance, write 0 to SP bit. Reading SP bit will give you information about if PCM is playing or had finished the playback. This status will also be available in PE bit – if it is set, playback has ended.

Note: if you decide to disable interrupts from controlled interrupt generator (bit IGE is set to 0), then watchdog will not function.

---

### 11.5.3. Prefetch function

For complex DSP (digital signal processing) implementations, involving big amounts of memory, MSX system may not be able to keep pace with interrupt servicing. Servicing PCM content from the GR8NET 1MByte RAM to the digital waveform input involves managing 20-bit address, banking mechanism, 20-bit sample counter. Within 8-bit machine such tasks present enormous overhead in computing and register load-testing, even if all values are stored in the primary and alternate register sets.

Consider the following (non-optimized) code which gives understanding in level of processing complexity:

```
plyint:                ; PCM playback interrupt service routine
    push    af
    ld      a,(05feah)    ; read generator status register
    rlca
    jr      c,service    ; INT bit set, go to service
    rlca
    jr      c,error      ; ER bit is set, go set flag ending DSP
svcend:
    in      a,(099h)      ; clear VDP interrupt if it was raised
    ei
    reti

service:
    ld      a,099h
    ld      (05feah),a    ; clear generator's interrupt condition

    ld      bc,1
    and     a
    sbc     hl,bc
    ex      de,hl
    ld      c,b
    sbc     hl,bc
    ex      de,hl        ; decrease sample counter - pair DE:HL
    jr      c,plyend     ; counter is 0, go end playback

    exx
    ld      a,e
    ld      (05fe1h),a    ; E is RAM page in bank 1
    ld      a,(hl)
    ld      c,a
    inc     hl
    ld      a,(hl)
    ld      b,a          ; get 16-bit sample into BC from (HL) which is within
    inc     hl            ; 6000-7FFF window, and increase HL
    ld      (05feeh),bc  ; place sample into digital waveform input

    bit     7,h
    jr      z,nonb        ; test if HL, pointer to data to read, went over 8000h
    inc     e             ; if yes, increase page # and reset pointer to start
    ld      h,060h        ; of the bank 1

nonb:
    exx
    jr      svcend
```

This code is even unable to perform PCM output from the GR8NET RAM at 11025 kHz. Needless to say, machine is busy with stupid unneeded tasks it is not designed to perform.

In order to unload such tasks off the Z80 CPU and increase the speed, prefetch function was designed. It has two 20-bit pointers, which designate start of the contiguous RAM addressing area, and end of the area (thus each pointer is within 00000-FFFFFh) – or

looping address and looping mask. No memory banking is involved; address is plain address within onboard physical device (RAM, ROM chips or W5100).

Let's consider the design of the prefetch control register.

Bit	Name	Description
0	PE	Prefetch ended W: n/a R: is set to 1 if start and end pointers are equal
1	PSRC	W: Source for the prefetch. 00 onboard 1 MByte RAM (00000-FFFFFF)
		01 onboard 512 Kbyte ROM (00000-7FFFFF)
2		10 onboard W5100 chip (00000-07FFF)
		11 n/a R: Previously written value
3	CYC	W: set to 1 for cyclical prefetch mode. End counter acts as mask, not as physical end address R: Previously written value
4-6	N/A	Reserved
7	HHE	HTTP header end. W: n/a R: bit is set to 1 if 4 previous bytes read from prefetch data register equal to 0D, 0A, 0D and 0A designating end of HTTP header and start of data

Prefetch function may work in two different modes:

- CYC = 0: in this mode prefetch function presents one-shot pass within address space. Application writes start and end *absolute* memory device pointers, and reads prefetched data from prefetch data register until bit 0, PE, becomes 1.
- CYC = 1: in this mode prefetch cycles through the specific address space. Application writes start *absolute* address, and writes address *mask* into end register. When reading prefetched data, next address following maximal masked address will be reset to 0. Let's illustrate it on the example: addressing W5100 chip's socket 2 RX space through prefetch. Application writes 0707A into the start register (assuming data to be read starts at address 07A within 2K memory space of socket 2); then writes 07FFh into end register (2K memory space mask). Application gets information from W5100 chip how many data bytes are available at the socket 2's buffer, and starts reading this count of bytes from prefetch data register. If data being read is received using HTTP protocol, application checks HHE bit, and finalizes HTTP header when this bit is set, and then continues with HTTP data. When start pointer (current pointer) reaches 077FF (077FF & 07FF equals to 07FF), next pointer is calculated using 077FF & not(07FF), and equals to 07000. In cyclical mode PE flag has no valuable meaning.

Application can read either 8-bit or 16-bit value from the prefetch data register area. When prefetch data register is read, the read is physically performed from the target device set by PSRC bits, thus prefetch will always return up-to-date value.



**Important:** if application uses 16-bit word reads, it must ensure that both start and end counter are odd or even, otherwise end of prefetch will be missed because end-of-prefetch event will happen between reads of 16-bit word and will be reset on 16-bit load command completion, resulting with start pointer being 1 byte ahead of end pointer.

When using prefetch, interrupt service routine may look the following way:

```

...
ld    de,05ffah    ; pointer to prefetch status register
ld    bc,05feah    ; pointer to the generator status register
contply:
ld    a,(plyend)
or    a
jr    z,contply
...

plyint:                ; prefetch interrupt service routine
ex    af,af         ; preserve A which is being used in main code
ld    a,(bc)         ; load generator's status register, BC is preset to 05FEAh
rlca
jr    c,service      ; service the interrupt
rlca
jr    c,intstop       ; end playback with error: generator error occurs
ex    af,af         ; restore A
ei
reti

intstop:
ld    a,1            ; generator error occurs: end playback
ld    (endflag),a    ; set some flag to signal parent code that playback ended
jr    intexit

service:
ld    hl,(05ffbh)    ; read prefetched 16-bit sample, a byte into L and next byte into H
ld    (05feeh),hl    ; write it to digital waveform input
ld    a,(de)         ; read prefetch status register, DE is preset to 05FFAh
rrca
jr    c,intstop       ; if end bit it set, go set the flag
ld    a,99h          ; WDE=1, IGE=1, clear INT bit, continue interrupt generation
ld    (bc),a         ; clear interrupt condition

intexit:
in    a,(099h)       ; clear VDP interrupt if it's raised
ex    af,af
ei
reti

```

It is clear that this (slightly optimized) interrupt handler is much faster and uses less CPU time.

To use prefetch function please follow the following steps:

1. Decide if you need one-shot or cyclical prefetch operation, and choose source device.
2. Write start absolute address and either end absolute address (CYC=0) or mask (CYC=1) into pointer registers;
3. Write prefetch control register;
4. Read 8-bit byte or 16-bit word from the prefetch data register. If you read a byte, start pointer will be increased by 1, if word – it will be increased by 2;

- 
5. You can know current prefetch pointer by reading start pointer's locations;
  6. In one-shot mode, to know if start (current) and end pointers match, read prefetch flags register. Bit 1 is set if pointers match. Be careful with 16-bit word prefetch access: pointer matching may happen during execution of the 16-bit data load instruction – between this instructions reading bytes;
  7. You can continuously read through whole onboard device (one-shot operation) or through specific memory window (cyclical operation).

#### **11.5.4. Combining functionalities of generator, prefetch and PCM**

All three functions can be combined in order to unload useless pointer calculations and page operations off the CPU. Example could be playback of the PCM data located in the GR8NET card's RAM. After loading audio file into the RAM, application knows its start address and end address. Before anything else, application resets generator. Application sets these addresses up in the prefetch function. Then application resets PCM function, and while bit 7 of PCM control register is set to 1, reads sample (8-bit or 16-bit) from prefetch data register, and writes this sample, byte by byte, to PCM data register, thus buffering samples into PCM memory. Application can buffer in bulk by examining PCM bytes free, and write this count of bytes – it will be much faster operation. After PCM control register's bit 7 becomes 0 (PCM buffer is full), or source data exhausted (bit 0 of prefetch flags register is set), application starts PCM playback function, but it does not play yet. To start operation, application sets up controlled interrupt generator (without hardware interrupt generation), and launches it. Then application regularly checks bit 7 of PCM control register, and quickly replenishes samples from prefetch register, until PCM control register bit 7 becomes 0 or prefetch data ends.

Employing such technique allows programmer to know beforehand the frequency of PCM buffer's need for sample replenishment, and thus is able to schedule other useful tasks between replenishments. At the same time replenishment task becomes most easy – checking two status bits and moving 8-bit or 16-bit data from one fixed memory location to another, without any pointer operations and bank switching.

#### **11.5.5. Sound custom chip (SCC/SCC+)**

GR8NET provides SCC/SCC+ functionality through:

- Logical page 0FEh in GR8NET mapper modes 0 and 8-14;
- Through K5 mapper (mapper modes 3 and 11);
- In mapper mode 8 in subslot 3 in the same way as in K5 mapper mode, but with no actual K5 mapper.

There's only one SCC implementation within GR8NET, thus if the device appears in two different locations – for example in K5 mapper and in GR8NET mapper in mapper mode 11 – it will be the same device with same registers.

Since February 2018 SCC was extended into SCC+, with the implementation working in SCC compatibility mode after hardware reset or power cycle. Important to know that if

application switches SCC into SCC+ mode and does not switch back to SCC mode, other applications (like games) not knowing about existence of SCC+ will not detect compatible SCC – the proper ways to cope with these situations are (1) design applications returning SCC+ into SCC compatibility mode after exiting (including emergency exit), (2) switching SCC+ manually through custom assembler code or using \_NETSETMEM command, or (3) resetting/power cycling machine to reset implementation into SCC compatibility mode.

SCC/SCC+ (here and further just SCC) occupies some space within 8KB page, with all other space filled with 0FFh bytes (GR8NET mappers) or showing game mapper contents (in K5 mapper mode).

Below is logical page layout in SCC compatibility mode; in this mode SCC will appear in GR8NET mapper logical page 0FEh, and in K5 game mapper bank 2 logical page 03Fh. Mode register bit 5 is reset.

<b>SCC (compatibility mode)</b>		
<b>Address range</b>		<b>Function</b>
0	17FF	Filled with 0FFh within GR8NET mappers logical page 0FEh; representing game mapper ROM contents for K5 mapper (mapper mode 3 and 11, mapper page register bits [5:0])
1800	187F	Waveform registers, channels 1-4 (32 bytes per channel), with channel 5 waveform data equal to channel 4 waveform (R/W access)
1880	188F	SCC control registers (write only)
1890	189F	Mirror or SCC control registers (write only)
18A0	18BF	Mirror of channel 4 waveform data (read only). Note that in GR8NET implementation these locations show waveform data of channel 4, not channel 5; however from functional perspective writing to e.g. address +1860 will cause same value appearing at address +18A0. But writing waveform 5 in SCC+ mode, and then switching to SCC compatibility mode will reveal waveform data of channel 4 in this location
18C0	18DF	No function. This memory area is reserved for the test/deformation register in original SCC+, but GR8NET does not use this register
18E0	18FF	No function
1900	19FF	Same as 1800-18FF
1A00	1AFF	Same as 1800-18FF
1B00	1BFF	Same as 1800-18FF
1C00	1CFF	Same as 1800-18FF
1D00	1DFF	Same as 1800-18FF
1E00	1EFF	Same as 1800-18FF
1F00	<b>1FFD</b>	Same as 1800- <b>18FD</b>
1FFE	1FFF	Mode register (write only); this register is used to switch implementation between SCC and SCC+ modes. The register is having bit 5 implemented only, thus writing 0 in these locations will cause implementation functioning in SCC mode, and writing 020h will cause it functioning in SCC+ mode

Below is logical page layout in SCC+ mode; in this mode SCC+ will appear in GR8NET mapper logical page 0FEh, and in K5 game mapper bank 3 logical pages 080h-0BFh (bit 8 set causes SCC+ appearance in the page). Mode register bit 5 is set.

<b>SCC+</b>		
<b>Address range</b>		<b>Function</b>
0	17FF	Filled with 0FFh within GR8NET mappers logical page 0FEh; representing game mapper ROM contents for K5 mapper (mapper mode 3 and 11, mapper page register bits [5:0])
1800	189F	Waveform registers, channels 1-5 (32 bytes per channel) (R/W access)
18A0	18AF	SCC+ control registers (write only)
18B0	18BF	Mirror of SCC+ control registers (write only)
18C0	18DF	No function. This memory area is reserved for the test/deformation register in original SCC+, but GR8NET does not use this register
18E0	18FF	No function
1900	19FF	Same as 1800-18FF
1A00	1AFF	Same as 1800-18FF
1B00	1BFF	Same as 1800-18FF
1C00	1CFF	Same as 1800-18FF
1D00	1DFF	Same as 1800-18FF
1E00	1EFF	Same as 1800-18FF
1F00	<b>1FFD</b>	Same as 1800- <b>18FD</b>
1FFE	1FFF	Mode register (write only); this register is used to switch implementation between SCC and SCC+ modes. The register is having bit 5 implemented only, thus writing 0 in these locations will cause implementation functioning in SCC mode, and writing 020h will cause it functioning in SCC+ mode

Actual location of the SCC is defined by the mapper type application accesses it in, and by the mode SCC is currently in:

- When using SCC/SCC+ within GR8NET mapper, logical page 0FEh will always contain either SCC or SCC+ device, and application can change mode by writing mode register at the end of the logical page;
- When application wants to use SCC mode in K5 mapper mode, it writes 0 to the location 9FFEh or 9FFFh (into mode register), forcing implementation using SCC compatible mode. Game mapper page number does not matter – mode register is always in these locations. Then application switches K5 mapper page in bank 2 to page 03Fh, and SCC appears in the space 9800-9FFF (mode register remains at 9FFE-9FFF). Page contents not overlaid by SCC will present data from K5 mapper page 03Fh;
- When application wants to use SCC+ mode in K5 mapper mode, it writes 020h (bit 5 set) to the location 0BFFEh or 0BFFFh (into mode register), forcing implementation using SCC+ mode. Game mapper page number does not matter – mode register is

---

always in these locations. Then application switches K5 mapper page in bank 3 to page numbers 080h-0BFh (bit 7 set), and SCC+ appears in the space B800-BFFF (mode register remains at BFFE-BFFF).

As you can see mode register is always at the locations 09FFE-09FFF *and* 0BFFE-0BFFF irrespective K5 mapper page number switched for these locations; it is the same register, thus application can write to any of these locations to switch between SCC/SCC+ modes.

**Limitations and deviations** of the GR8NET SCC+ implementation:

1. No test/deformation register, writing to respective locations has no effect;
2. Mode register is located in 1FFE/1FFF offset within logical page using GR8NET mapper, and at addresses 9FFE/9FFF *and* BFFE/BFFF in K5 mapper (in contrast with original chip where this register is accessible only through locations BFFE/BFFF);
3. When writing waveform data to channel 4, data of waveform channel 5 is not written to in SCC compatibility mode; thus writing waveform to channel 4 in SCC mode and then switching to SCC+ mode and assume that waveform 5 will contain the same waveform it not correct;
4. Memory mode and bank select bits are not implemented in the mode register.

#### **11.5.6. Digital waveform input and Philips Music Module DAC**

GR8NET provides functionality to mix custom waveforms into its output signal by writing digital sample values into digital waveform input (16-bit, 5FEF:5FEF). Sample modification may be timed by the GR8NET controlled interrupt generator's interrupt service routine. Important: for proper update of the 16-bit sample write low byte first, and then write high byte. Actual update of the sample for mixing takes place when high byte is written. 16-bit load commands like **LD (05FFEh),HL** are suitable for storing correct value within single CPU instruction.

The same set of registers is available through I/O port 0Ah in case bit 0 of Y8950 GPIO port is set to 1. The value written into the port 0Ah represents higher byte of the waveform, centered around 080h (thus 080h represents zero level voltage, 0 represents  $-V_{\max}$  and 0ffh represents  $+V_{\max}$ ). When writing to port 0Ah, Digital input waveform's lower byte will be cleared to 0.

#### **11.5.7. Volume registers**

GR8NET has seven volume registers, allowing application modifying digital volumes of the internal GR8NET devices. Modifying (decreasing) the volume of available channels may be useful because mixture of the input digital signals may overflow 16-bit physical DAC, presenting sound overload effect and corruption of the output analog waveform. Another use is to level the volume with other devices installed in your MSX machine.

---

Each volume register may have values from 80h (128 decimal) marking maximal volume (unmodified digital sample), and 0 marking muting the digital sample. Sample volume scaling down is performed in linear way.

- SCC volume register (5FE6): performs scaling of the digital sample going from SCC module;
- Master DAC volume (5FE7): performs scaling of the digital sample after final digital mixing;
- Digital waveform input volume (5FED): performs scaling of the digital sample written to the digital waveform input register and Y8950 Philips MM DAC;
- PCM volume (5FF5): performs scaling of the digital sample going from the PCM playback. In MP3 mode player serves as MP3 waveform volume;
- OPLL volume (5FDA): performs scaling of the built-in YM2413 (MSX-Music) device;
- Y8950 volume (5FD7): performs scaling of the built-in Y8950 (MSX-Audio) device;
- PSG volume (5FD3): performs scaling of the built-in PSG device;

After hardware reset all volume registers are initialized to the maximum (080h), and reloaded from the ROM configuration page when during GR8NET ROM BIOS initialization.

### 11.5.8. Micro-SD card interface

GR8NET is having micro-SD-card slot and supports SC/HC cards. Independently of the card architecture, application will access card data on 512-byte sector level. GR8NET is having half of one RAM buffer page (4KB) dedicated for the storage of the data read from or to be written to the card, thus may hold maximum 8 sector contents at a time.

Let's consider registers related to the SD-card operation:

#### SD-card status register

Bit	Name	Description
7	INSLT (R)	Set if SD-card is physically present in the slot
6	INIP (R/W)	Initialization is in progress. W: Write 1 to restart SD-card initialization process. The bit will become 0 when initialization complete (see status in bits 3:0) R: Is set if SD-card is initializing
5-4	CTP (R)	SD-card type. Valid only after successful initialization of the card. 00 = SC V1 card, 01 = SC V2 card, 10 = HC card, 11 = MMC
3-2	SDE1 SDE0 (R)	Contains error code if SDE bit is set. 00=no error, 01=SD-card is busy, 10=R1 response timeout, 11=data token timeout
1	SDE (R)	This bit is set if SD-card finished initialization with error, and SD-card interface can not be used to access the SD-card. Application may try re-initialization of the card using INIP bit of this register
0	SDRDY (R)	Is set if SD-card interface is ready to accept requests

**Sector buffer position:** identifies the position in the SD-card buffer memory which will be used to read data from (write operation) or write data to (read operation). Sector buffer position may have values ranging from 0 to 7, register is organized in the way application can read it and calculate absolute CPU address for access:

```
ld    de,(05fc0h)    ; sector buffer position goes to D
ld    e,0             ; align to 512 byte boundary
ld    hl,06000h       ; base address for GR8NET bank 1 (example)
add   hl,de           ; now HL contains physical CPU address to access data
```

7	6	5	4	3	2	1	0
0	0	0	0		SBP		0

After successful read or write command completion buffer position is increased by the number of sectors successfully processed. When writing this register, ensure you write position into bits 3:1, and not to 2:0.

**Sector #:** represents 32-bit value, little-endian, which is used to identify the sector (block) number to be accessed. Application can not access SD-card in byte mode, even if card supports it.

After successful completion sector # is increased by the number of successfully processed sectors.



**Sector count:** this register has the following format:

7	6	5	4	3	2	1	0
NOSP <sub>(R)</sub> CHKPT1 <sub>(W)</sub>				NOSR <sub>(R/W)</sub>			

**NOSR** is Number Of Sectors Requested to be accessed. Value of 0 means no operation and 8 means 8 sectors to process. After read/write operation completion NOSR contains number of sectors remaining to process;

**NOSP** is Number Of Sectors Processed, and *after the operation* is set to the number of sector successfully read or written during last SD-card interface access. This is read-only property.

**CHKPT1** is Check Pattern value which is written together with NOSR. CHKPT1 is equal to 9, thus to instruct SD-card interface to read or write 7 sectors application should write 097h into the sector count register. Such check pattern write should be performed each time read or write operation is requested, because check pattern is reset after completion of the request. If check pattern written is not equal to 9, operation returns error and SDCCMD (command) register has CAINV bit set.

**Command register (SDCCMD):**

7	6	5	4	3	2	1	0
DSKCHG	0	0	R2VAL	$\overline{R/W}$	CAINV	CERR	COPST
	CHKPT2 <sub>(W)</sub>						

Setting **COPST** starts the SD-card access operation; the access will be read if  $\overline{R/W}$  is reset to 0 or write if  $\overline{R/W}$  is set to 1. When access is complete, COPST bit is reset to 0. If **CERR** bit is set then request processing completed with error, and application should check SD-card status register's bits 3:2 to know the cause of the error. Application also has access to the advanced troubleshooting information, available in the MathPack logical page C9h.

If after execution of the command **CAINV** bit is set, then command was not processed because it was passed with invalid argument (e.g. card is in byte-addressing mode and block requested is outside of 32-bit range or check patterns are invalid).

When writing command register bits, application should ensure writing check pattern 2 **CHKPT2** being 06h, thus to start card write operation application should write 069h into the SDCCMD register.

Bits CAINV and CERR are reset by writing 0 into them, or by issuing new read or write command.

**R2VAL** is set when R2 response following write command is valid (see C9h page contents).

Bit **DSKCHG** indicates if SD-card was removed *since last time this bit was cleared*. When GR8NET initializes, it sets this bit into 1 to signal to driver performing full initialization of its working structures. To reset the flag, driver writes 1 to this bit when accessing SD-card for read or write operations. It is possible to write 080h into the command register; this will not trigger any operation, but it will clear all error bits. If card is not present in the slot this bit DSKCHG will be steady 1 and will not be changeable. Please sense card insertion status by using INSLT bit of status register.

Logical page C9h contents (see SD-card “physical layer” specification for more detail)

Offset		Size	Description
Dec	Hex		
0	0	1	R1 response from the card given for the last request
1	1	1	R2 response if bit R2VAL is set
2	2	4	R3 response given during initialization for CMD58 command
6	6	4	R7 response given during initialization for CMD8 command
10	0A	1	Data token value detected during last read or write command
11	0B	16	Card identification register (CID) given during its initialization
27	1B	5	Last command given to the SD-card
128	80	16	128-bit CSD register reported by the SD-card

To use SD-card interface please follow the steps below:

1. (optional) Reset the SD-card subsystem by writing 1 into INIP bit of SD-card command register. Wait until bits SDRDY or SDE become set. If SDRDY is set proceed to next step, or if SDE is set investigate the reason by looking into bit SDE1/SDE0, R1/R3/R7 responses in the logical page 0C9h;
2. Read command register and check for its bit 7 being 1. If it is set, check INSLT bit of status register. If INSLT is reset, card is not in the slot. Halt operations until user inserts card into the slot. Note that reseating of the SD-card in the slot resets sector buffer position register to 0;
3. Set up RAM buffer position (note to use bits 3:1), and sector #;
4. Write 090h+number of sectors to process, not more than 8 sectors. If you set up more than 8 sectors, or write something else than 9 into high nibble, command will immediately return with CAINV bit set in status register;
5. Write 0E0h+flags into the command register, having but 7 set to reset DSKCHG state bit. Meaningful flags to start the processing are bit 0 COPST (should be set to 1) and bit 3  $\bar{R}/W$  (0=read, 1=write). If you write something else than 6 into high nibble, command will immediately return with CAINV bit set in status register. If you write 0 into COPST bit no operation will be performed;
6. Wait for COPST bit to become 0, then examine bits CERR and CAINV. If CAINV is set, then you made a mistake in the arguments to the command (also you have set sector # out of the SD-card memory space), if bit CERR is set then there was problem with SD-card operation – you will need to look into bit 4 R2VAL and if set analyze R2 response available in logical page 0C9h, and look into SDE1/SDE0 bits of status register to see at which stage error have happened (card did not become ready, command was not accepted by card or proper data token was not received);
7. Note that after successful operation buffer position (SBP) and sector # automatically increase by the number of sectors processed, if you need to retry operation you will have to reload both fields.

### 11.5.9. Math-Pack

Working with modern storage and data processing devices is not an easy task using 8-bit CPU, and GR8NET provides support for Z80 to complete relatively complex tasks. While addition and subtraction of 32-bit values does not take much CPU power if it is not main task of the application, multiplication and division is almost a killer for any optimized code.

Functionalities described below are originally designed to be complementary for the SD-card support however application is free to use them for any other purpose. All values are little endian.

Logical page C9h, explained in previous section, has additional fields:

Offset		Size	Op	Description
Dec	Hex			
File-system specific math				
32	20	1	R/W	Sectors per cluster [boot sector: +0Dh]
33	21	2	R/W	Reserved sectors [boot sector: +0Eh]
35	23	1	R/W	Number of FAT copies [boot sector: +10h]
36	24	4	R/W	Volume starting LBA [MBR: +1C6h+10h*MBR_entry#, or 0 if there's no MBR]
40	28	4	R/W	Sectors per FAT *
44	2C	4	R	LBA of the start of the first FAT copy
48	30	4	R	LBA of the start of the second FAT copy (valid if available)
52	34	4	R	LBA of the starting cluster / root directory **
User-defined cluster number calculations				
56	38	4	R/W	Cluster # ***
60	3C	4	R	LBA for the first sector of cluster
64	40	4	R	LBA of the FAT sector to get next cluster # in the chain
68	44	2	R	Offset in the FAT sector (see previous item) to get next cluster#
70	46	1	R/W	File system type and cluster number flags

Notes for the table above:

\* This field has different layout for FAT32 and FAT16 file systems:

- FAT32: it is 32-bit value of the sectors per FAT from EBPB [boot sector: +24 (dword)];
- FAT16: it is divided into two logical registers – bits [15:0] designate 16 bits of sectors per FAT [boot sector: +16 (word)], bits [31:16] designate number of root directory entries [boot sector: +11 (word)].

\*\* For FAT32 it has LBA number for first data cluster #2, for FAT16 it has LBA number of root directory start.

\*\*\* For FAT16 only lower 16 bits of the cluster number are used for calculations.

Application copies volume data into the fields (sources of data are indicated in the square brackets), and obtains sector address for FAT and first data cluster. Calculation is performed within tenths of nanoseconds, thus from Z80 point of view – immediate. Following formulas are used:

- First FAT = (reserved sectors) + (volume starting LBA)
- Second FAT = (first FAT) + (sectors per FAT)

- Starting cluster (root dir) = (first FAT) + (number of FATs) \* (sectors per FAT)

In order to simplify file operations, application may use calculations for specific cluster number (they are based on the values stored into and calculated by the formulae above): LBA (absolute sector number) of the beginning of the cluster, FAT sector LBA and offset within this FAT sector to get next cluster number in the file chain.

- Cluster LBA = (LBA of cluster #2) + (cluster # - 2) \* (sectors per cluster) + (root directory size if FAT16)
- LBA of FAT sector to get cluster chain = (first FAT) + (cluster #) \* (2 - FSTYP) \* 2 / 512
- Offset in FAT sector to get next cluster = ((cluster #) \* (2 - FSTYP) \* 2) AND 01FFh

File system type and cluster number flags location (+70) has the following format:

7	6	5	4	3	2	1	0
0	0	0	CLNFLG (R)				FSTYP (R/W)

If FSTYP is 0, FAT32 file system is assumed (FAT chain offset\*4), if FSTYP is 1, FAT16 file system is assumed (FAT chain offset\*2). This flag also affects internal calculations using *sectors per FAT* (+28) and *cluster #* (+38) fields. If FSTYP is 1 (FAT16), *sectors per FAT* register has different purpose, and *cluster #* higher 16-bits, while writable and readable by CPU, are discarded in calculations.

Cluster number flags CLNFLG may have the following values:

#	FAT32 (FSTYP=0)	FAT16 (FSTYP=1)	Description
0	*000_0002...*FFF_FFEF	0002...FFEF	Normal data cluster
1	*FFF_FFF8...*FFF_FFFF	FFF8...FFFF	End of chain
2	*000_0000	0000	Zero cluster
3	*FFF_FFF7	FFF7	Bad cluster
4	*000_0001	0001	Special reserved
5	*FFF_FFF0...*FFF_FFF6	FFF0...FFF6	Other reserved

The following code may be used to identify type of the cluster:

```
ld      a,(MTHBASE+FSTYP)      ; get FS flags, MTHBASE points to BASE address of MathPack, FSTYP is 046h
rrca                    ; flags now are in bits 3:0
and     0fh              ; not only flags bits are in A, ZF is set if flags are 0 (normal data cluster)
jr      z,datacl          ; go there if we have data cluster # in CLUNUM (CLUNUM is +38h in MathPack)
dec     a                 ; is flags nibble equal to 1? (end of cluster chain)
jr      z,endchn          ; go there for end of chain
dec     a                 ; is flags nibble equal to 2? (empty file, no cluster was allocated in directory entry)
jr      z,empfil          ; go there for processing empty file contents
jr      fserr             ; otherwise bad or reserved cluster = file system error
```

Logical page C9h has binary/BCD conversions **removed**:

Offset		Size	Op	Description
Dec	Hex			
Binary/BCD type conversions - <b>removed</b>				
Integer to BCD				
71	47	4	R	Returns FF FF FF FF (hexadecimal)
75	4B	8	R	Returns 4A 42 94 96 72 95 00 00 (hexadecimal)
BCD to integer				
83	53	8	R/W	Returns FF FF FF FF FF FF 00 00 (hexadecimal)
91	5B	4	R	Returns FF FF FF FF (hexadecimal)

**Important:** Binary/BCD conversion hardware acceleration was removed starting version 0.9 from the GR8NET engine: it was occupying a space in the silicon, but was used in very few, non-performance-critical internal applications like \_NETRESST, \_NETGETMD, \_NETSETDM and \_NETSDCRD. These commands perform software conversion now instead.

Logical page C9h has the following 32-bit math extensions:

Offset		Size	Op	Description
Dec	Hex			
Generic 32-bit math operations				
95	5F	1	R/W	Division operation control bits and flags
Multiplication				
96	60	4	R/W	Operand A for multiplication, 32-bit, <b>Signed</b>
100	64	4	R/W	Operand B for multiplication, 32-bit, <b>Signed</b>
104	68	8	R	Product of above A*B, 64-bit, <b>Signed</b>
Division				
112	70	4	R/W	Numerator
116	74	4	R/W	Denominator
120	78	4	R	Quotient (numerator \ denominator)
124	7C	4	R	Remainder (numerator MOD denominator)

Multiplication operation is expected to be executed within 200 nanoseconds; Division operation is expected to be executed within 320 nanoseconds – both from Z80 standpoint – immediately.

Location +95 (+5Fh) has special format to control division operation:

7	6	5	4	3	2	1	0
DIV0 (R)	DLOCK (R/W)	SIGD (R/W)	SIGN (R/W)	0	0	0	0

SIGN and SIGD are bits which define state of the numerator and denominator respectively. Bit is set to 1 if value in respective field is signed, and reset to 0 if unsigned. Special care was taken to handle most negative value, designing actual computing space as 33 bits instead of 32 bits.

---

DIV0 is a flag being set if denominator is equal to 0 meaning that calculated values of quotient and remainder are not correct.

DLOCK, if set, suspends division calculation, and quotient and remainder remain the same even if application changes numerator and denominator. This feature is very useful if application performs wave division – e.g. when quotient is going to be used as numerator for next wave of division. Application sets DLOCK flag, performs LDIR or \_NETGETMEM/\_NETSETMEM from quotient to numerator, and then resets DLOCK flag to obtain new division result.

Quotient is returned with the sign according to the setting of bit 31 of numerator and denominator and SIGN/SIGD flags. Remainder is *always* returned positive.

GR8NET performs measurement of the clock speed of the slot it is installed in, presenting the frequency in Hertz in the page C9:

Offset		Size	Op	Description
Dec	Hex			
Clock speed				
144	90	3	R	24 bits (in little endian format) representing the clock speed of the GR8NET audio, maximal properly measured speed is 16777215 Hz

The source being measured is identified by the MCLKSRC bit of [System mode register](#).

### 11.5.10. Mixer and DAC (digital to analog converter)

GR8NET features 16-bit physical DAC circuit. Before final 16-bit sample is formed, all channels (SCC, digital waveform input, PCM, OPLL, Y8950 and PSG) scaled using respective volume registers, and then master DAC volume is applied. Mixer has 21-bit adder, and in case value overflows – sample value  $< -2^{15}$  or  $> 2^{15}-1$  – sample is clipped to the minimum or maximum respectively.

Mixer is having a register, controlling the appearance of specific signal in its left and right channels (or left channel only for monophonic GR8NET adapter).

MSB (e.g. 5FD6)								LSB (e.g. 5FD5)							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
(reserved, should be 0)				PSG		Y8950		OPLL		Wave		SCC		PCM	

The register consists of two bytes, with 12 bits – 6 groups by 2 bits – defined. The value written in each group has the following effect:

Value (binary)	Effect
00	Mute, no sound in any channel
01	Left, sound goes only to left channel
10	Right, sound goes to right channel only
11	Both, sound goes to both channels

For monophonic version of GR8NET only right channel is available, thus setting sound output to left channel only will not be heard through MSX machine audio output.

### 11.5.11. System registers

- **System mode register 0**

7	6	5	4	3	2	1	0
PSGRD	MMDR	MMDPR	AUDDIS: Y8950 disable	VOL2X: 2Xvolume	OPLLD: OPLL disable	MCLKSRC: measurement clock source	MRF: Mapper read flag

**Mapper read flag:** if this flag (bit 0 of register) is set, then, when CPU reads mapped RAM ports 0fch-0ffh, GR8NET mapped memory mapper will respond with respective values appropriate for mapper in effect: in mapper mode 7, 6 bits will identify one of 64 RAM pages, in mapper mode 8-14, 5 bits will identify one of 32 RAM pages.

**Warning!** GR8NET can only output *whole* byte to the data bus, it can not output only significant bits (6 or 5 bits) leaving remaining bits in high-impedance state. By enabling mapped RAM mapper read flag, you may cause electrical conflict on the data bus in case there're other mapped memory mappers putting their whole byte output onto the bus, or MSX machine is designed in the way (using internal data bus





---

buffers) it may have signal conflicts internally to it. This issue should not cause damage to the system, but please use this feature prudently. Check [Multiple memory mappers in MSX system](#) discussion for more information.

**Measurement clock source:** if this flag (bit 1 of register) is set, then registers in locations +144 (decimal) of MathPack page C9 present speed of the GR8NET internal clock in Hertz; if flag is reset, then these registers present speed of the MSX system bus in Hertz.

**OPLL disable:** if this flag (bit 2 of register) is set, then internal OPLL analog output is disabled, and OPLL ROM BIOS will not appear in subslot 3 in mapper mode 8.

**2Xvolume:** if this flag (bit 3 of register) is set, then internal OPLL/Y8950 output twice amplitude of the waveform, increasing volume of the device approximately 1.5 times. However setting this bit has a downside: if many channels are producing the sound, output may become overloaded and output sound may occasionally appear distorted.

**Y8950 disable:** if this flag (bit 4 of register) is set, then internal Y8950 output (analog and digital) is disabled, sample RAM is not allocated on startup, respective ports C0-C1 or C2-C2 are not configure and not occupied.

**MMDR** (mapped memory disable register) and **MMDPR** (mapped memory disable pending register) are two related registers. When MMDR is set to 1, mapped RAM is disabled – not visible in its slot/subslot and does not respond for I/O port mapper register reads even if *Mapper read flag* is set; thus MMDR disables mapped RAM immediately. MMDPR, if set, will be assigned to MMDR when GR8NET mapper type I/O register write is performed; thus it is a kind of pending change so that mapped RAM disable happens only when mapper type is changed, and not before it (because immediate mapped RAM disable may cause machine to lose its RAM if GR8NET mapped RAM was set as primary RAM).

**PSGRD** (PSG read flag) is set causing built-in PSG responding to the data reads at PSG read port (0A2h or 12h). This bit is set during initialization of built-in PSG or its reconfiguration, and must be preserved and not be changed by the software. Wrong setting of this bit may cause machine malfunction or even physical damage due to electrical conflict between two PSG devices in the system.

#### • System mode register 1

7	6	5	4	3	2	1	0
FPGA image type (R)	SCCM	PSGLOC	PSGENA	Y8950 int disable	Y8950 port select	Y8950 configured	

**Y8950 configured:** if this flag is set, then Y8950 device is in *configured* state, and will function normally if enabled through System mode register 0. If this flag is reset, then Y8950 is held in reset state not depending on the Y8950 disable bit of System mode register 0.

**Y8950 port select:** if this flag is set, then Y8950 device, of configured and enabled, will be present in I/O ports C2 and C3; if reset, then in I/O ports C0 and C1.

**Y8950 int disable:** if this flag is set, then Y8950 device, of configured and enabled, will never generate hardware interrupt, and its state should be polled using status Y8950 register.

**FPGA image type:** identifies current GR8NET Engine (FPGA) image type running: 00 means regular image, 01 means MP3 media player image. These two bits are read-only.

**SCCM:** is an SCC/SCC+ mode register bit, if set to 1 means that GR8NET built-in SCC is in SCC+ mode, if reset to 0 means that it is in SCC compatibility mode;

**PSGENA:** this flag being set means that built-in PSG is enabled at the location of PSGLOC;

**PSGLOC:** location of the built-in PSG – if this bit is reset, then PSG is located at base port 0A0h (mirror for machine's internal PSG), if this bit is set, then PSG is located at base port 010h.

- **Error register**

This register is having special behavior, If ERRST bit is set, Error code in the register is reset to 0. If ERRST bit is not set, then, if current Error code is 0, new code being written will be stored in it; however, if Error code is not 0, it will not be modified.

7	6	5	4	3	2	1	0
ERRST	Error code						

- **Adapter flags**

This is status register, identifying operation mode as shown below.

7	6	5	4	3	2	1	0
<u>Interface</u> 1: Error 0: OK	<u>Diag mode</u> 1: On 0: Off	Reserved	Reserved	<u>GR8cloud</u> 1: Enabled 0: Disabled	<u>Disk ROM</u> 1: Enabled 0: Disabled	<u>Network</u> 1: DHCP 0: Fixed	<u>Boot state</u> 0: Cold 1: Warm

### 11.5.12. FPGA flash chip interface

Since June 2017 GR8NET is having built-in FPGA flash chip interface allowing updating FPGA firmware without Blaster devices, writing to and reading user data from the onboard serial flash chip.

Note that under specific circumstances – when FPGA “factory” (boot) image is corrupt – you will anyway need to use GR9blaster, USB-Blaster or Byte-Blaster-II device in order to recover the board’s functionality.

Registers related to the FPGA flash chip interface are located in special register set, and in Math-Pack page.

Special register set registers

Address	Purpose	
5FC8	Data register, location where application reads fetched data, and writes data to be written into the flash chip. This register is also used for access preparation during writing the special sequence of bytes	
5FC9	Control and status register. At any time it can be read to obtain current state of the flash chip access machine. This register is also used for access preparation during writing the special sequence of bytes	
5FCA	Low	Access address's low 16 bits, it points to the specific location in the sector of serial flash chip
5FCB	High	

Math-Pack related locations

Offset	Byte	Op	Description
Dec	Hex		
Serial flash device byte location address (current for write, next for read)			
147	93	LSB	Full 24-bit address of the current (for write and erase) and past (for read) access to the flash chip
148	94		
149	95	MSB	

Flash chip can be read at any time; all its contents are available within 24-bit space from address 000000h to FFFFFFFh (16MB), however actual size of the chip may differ (e.g. EPCS16 is 2MB). Chip can be written byte-by-byte, but, if the location was written to before, whole sector containing the location should be erased.

This access is targeted for 64KB sector access, in other words, space in serial flash chip is designated for bulk data writing (e.g. GR8NET FPGA application image, or user data), and not at the single byte level.

In the following subchapters we will cover all three operations – read, erase and write.

#### 11.5.12.1. FPGA flash chip access control/status register

When reading, application will identify state machine status depending on the values in the fields of this register.

7	6	5	4	3	2	1	0
SFL_busy	SFL_error	SFL_access_type	State machine				

---

State machine will have the following states:

State	Actions to perform
0	Ready to accept start of sequence, byte 0AAh into control register
1	Ready to accept second byte of sequence, 055h into data register
2	Ready to accept third byte of sequence (depends on operation)
3	Ready to accept fourth byte of sequence (depends on operation)
4	Waiting for serial flash sector # to be written (0...255)
5-15	Operation in progress, cancellation depends on command running

SFL\_access\_type identifies the operation currently running, or is about to be run.

Access type	Actions to perform
0	FPGA reconfiguration
1	Erase operation is in preparation or in progress
2	Write operation is in preparation or in progress
3	Read operation is in preparation or in progress

SFL\_error is set when error – either invalid sequence, or hardware error occurs.

SFL\_busy is set when state machine is busy performing required operation; no writes should be performed to data, control or address registers, read from data register will return invalid data, but control/status register returns up-to-date information.

#### 11.5.12.2. Data read

To perform the read, the steps below must be followed:

1. Read control/status register to see its state machine value. If it is 0, then write 0AAh to this register, and read it again. If state now changed to 1, go to step 2, otherwise write another 0AAh, and check again – state must change to 1 (with previous operation cancelled);
  - 1a. When state changes to 1, access address within page is reset to 0, but you have an opportunity to change it by writing target address into access address register. You can write address in any order (HI-LOW or LOW-HI);
2. Write byte 055h into the data register; read control/status register, state machine value must change to 2. You still have the opportunity to change access address register;
3. Write byte 088h into control register; read control/status register, state machine value must change to 3;
4. Write byte 0eh into data register; read control/status register, state machine value must change to 4;
5. Write serial chip sector number into data register, for example 010h for page 16. Remember than sectors are 64 Kbyte in size, and if chip is smaller than address then spare address bits will be discarded;

- 
6. Interface performs read from the target location, and increases address, thus in address register (in special register set and in Math-Pack page) you will see incremented address;
  7. Read byte of data from data register;
  8. You can read consecutive bytes, reading chip in loop from its address 0 to 0FFFFFFh. To stop reading operation, write anything to the control/status register.

If there's error condition, state machine raises SFL\_error bit and goes to state 0, thus aborting the operation. Causes of the error could be: hardware error, or you did not write proper sequence to the state machine to identify read command.

Note that you can change address in address register in steps 6 and 7 (while reading consecutive bytes), but you should keep in mind that previous value was already read, thus next read after you change the address will have invalid value, and you need to read data one more time to obtain correct value.

### **11.5.12.3. Data write**

To perform the write, the steps below must be followed:

1. Read control/status register to see its state machine value. If it is 0, then write 0AAh to this register, and read it again. If state now changed to 1, go to step 2, otherwise write another 0AAh, and check again – state must change to 1 (with previous operation cancelled);
  - 1a. When state changes to 1, address within page is reset to 0, but you have an opportunity to change it by writing target address into address register. You can write address in any order (HI-LOW or LOW-HI);
2. Write byte 055h into the data register; read control/status register, state machine value must change to 2. You still have the opportunity to change access address register;
3. Write byte 0E3h into control register; read control/status register, state machine value must change to 3;
4. Write byte 0C9h into data register; read control/status register, state machine value must change to 4;
5. Write serial flash chip sector number into data register, for example 010h for page 16. Remember that sectors are 64 Kbyte in size, and if chip is smaller than address then spare address bits will be discarded. Also remember that writing to byte which was previously written (having 0s in it) is incorrect operation, and before it you should erase whole sector (thus erase all 64 Kbyte of data);
6. State machine must change to state 7, and is waiting for data byte. Write data byte to data register, and it will be written to the target address within the chip, and address register value will increment;
7. Check SFL\_busy for write operation completion. You can write consecutive bytes, checking for SFL\_busy bit being 0 and SFL\_error bit being 0, until address reaches end of sector (0FFFFFFh), and then command finishes and state machine goes to state 0 waiting for next sequence.

---

If there's error condition, state machine raises SFL\_error bit and goes to state 0, thus aborting the operation. Causes of the error could be: hardware error, write error because location was not erased, or you did not write proper sequence to the state machine to identify read command.

Note that you can change address in address register in steps 6 (before you write data byte to the data register), and next data byte will be written to the new location.

#### **11.5.12.4. Sector erase**

Sector erase is required if you plan to write data bytes in place which were previously written to. Generally if byte is having 0 bit in it at any place, it can not be overwritten and must be erased. Reading whole sector, byte by byte, and check them all being 0FFh, or, alternatively, just erase the sector preserving required information from it.

Note that when sector is erased, all information in it will be lost, thus if there's valuable information in the serial flash chip sector, you must read it to the machine's RAM, and rewrite it back to the serial flash chip.

Serial flash chip may have maximum 256 sectors 64 Kbyte each, but actual chip may have less sectors (e.g. EPCS16 is having 32 sectors only). First 4 sectors (for compressed FPGA image – uncompressed FPGA image may occupy more space, use \_FLLIST command to identify) are reserved for the factory boot image, and GR8NET will stop functioning if this image will get corrupt.

To perform sector erase follow the steps:

1. Read control/status register to see its state machine value. If it is 0, then write 0AAh to this register, and read it again. If state now changed to 1, go to step 2, otherwise write another 0AAh, and check again – state must change to 1 (with previous operation cancelled);
  - 1a. When state changes to 1, address within page is reset to 0, but you have an opportunity to change it by writing target address into address register. You can write address in any order (HI-LOW or LOW-HI);
2. Write byte 055h into the data register; read control/status register, state machine value must change to 2;
3. Write byte 076h into control register; read control/status register, state machine value must change to 3;
4. Write byte 0F1h into data register; read control/status register, state machine value must change to 4;
5. Write serial flash chip sector number into data register, for example 010h for page 16. Remember that pages are 64 Kbyte in size, and if chip size is less than 256 sectors, spare address bits are discarded (e.g. erasing sector 20h for EPCS16 will, most probably, erase sector 0 of it and make GR8NET inoperable because of factory image corruption);
6. Most probably control/status register's state will immediately change to 096h, meaning busy (erasing), operation type 2, and state machine value 6. You must

---

poll control/status register to finish operation – state machine value becomes 0, busy flag clears, and no error bit is set after completion.

If there's error condition, state machine goes to state 0 and raises SFL\_error bit, thus aborting the operation. Causes of the error could be: hardware error, or you did not write proper sequence to the state machine to identify read command.

Value within lower 16-bit address register does not matter for erase function, only sector number supplied within the sequence matters.

#### **11.5.12.5. FPGA reconfiguration**

The reconfiguration is to be performed through the same interface as read, write and erase for serial flash chip. The only argument required is the starting sector number for the valid image.

If GR8NET is running in application mode (bit 5 on MSM state mode is 1, see [Current FPGA image properties](#) chapter), then reconfiguration must be performed through the reconfiguration to the factory mode image at sector 0 first.

Thus algorithm is:

1. Identify the mode current image is running in by checking MSM state mode bit 5.  
If this bit is reset, go to step 3;
2. Perform reconfiguration to the factory image starting with sector 0;
3. Perform reconfiguration to the desired application image at non-zero sector – into application mode.

Note that image loaded and configured from the sector 0 may also run in application mode (while being loaded from the factory image location). In this case bit 5 will be set, and then step 2 is required to re-configure to the same image at sector 0 to get from application into the factory mode.

To perform the re-configuration follow the steps:

1. Read control/status register to see its state machine value. If it is 0, then write 0AAh to this register, and read it again. If state now changed to 1, go to step 2, otherwise write another 0AAh, and check again – state must change to 1 (with previous operation cancelled);
2. Write byte 055h into the data register; read control/status register, state machine value must change to 2;
3. Write byte 021h into control register; read control/status register, state machine value must change to 3;
4. Write byte 090h into data register; read control/status register, state machine value must change to 4;
5. Write serial flash chip starting sector number into data register;
6. Reconfiguration starts. Please note that during reconfiguration all resources provided by the GR8NET to the machine will become unavailable, and you must ensure that you run code in safe place, no interrupt servicing of GR8NET is involved, and stack is also in safe place – as well as contents of the system



- variables in the CPU bank 0 and bank 3 (e.g. GR8NET provided mapped RAM set up as main system RAM disappears leading to all MSX system variables being lost).
7. After reconfiguration GR8NET adapter's resources are not initialized (they are in the initial state), thus its functionality must be further configured by the user code. For example, when reconfiguring to GR8NET standard MP3 or regular image using `_NETRECFG`, code reboots machine letting MSX system, and GR8NET reinitialize properly through their ROM BIOS.

#### 11.5.12.5. Serial flash chip information

When GR8NET starts in regular image mode, it queries serial flash chip attached to the FPGA, and provides this information to the MSX computer. This information is used by the `_FLINFO` command to display flash chip information.

There're following related locations in the MathPack logical page (0C9h):

Offset		Op	Description
Dec	Hex		
152	98	R	Bits [1:0] identify board hardware revision (0=mono, 1=stereo); bits [7:3] identify mask for the serial flash chip address bits [23:19]
153	99	R	Silicon ID for the serial flash chip
154	9A	R	Manufacturer ID
155	9B	R	Chip type identification bits, so called ID[15:8]
156	9C	R	Capacity ID, in most cases identify size of chip as $2^N$

All the information is read-only, allowing identification of what chip is installed, and its size.

#### 11.5.13. Current FPGA image properties

As described in [GR8NET functionality extensions](#) there could be several FPGA images in the serial flash chip and the image at the location 0 is called "factory" image because it is started when GR8NET is powered.

FPGA keeps track of image it is currently running and its mode, and there're two registers in the Math-Pack (logical page 0C9h) identifying the image and the mode.

Offset		Op	Description
Dec	Hex		
"Remote update" Master state machine properties			
150	96	R	FPGA image status register
151	97	R	Image starting sector number

Image starting sector number is just a serial flash chip sector number the FPGA image was loaded and configured from. Sector 0 means that current image is a factory image.

---

FPGA image status register has the following format

7	6	5	4	3	2	1	0
0	MSM state mode		nCONFIG	CRCerr	nSTATUS	WDTIMER	RUCONFIG

Please refer to the Altera Cyclone III handbook and Remote Update IP manual for specific meaning of the status bits [4:0]. Bits [6:5] identify *Master State Machine Current State Mode*: bit 5 set means that image is in *application* mode. It is very important during the reconfiguration as if FPGA is in application mode, it can reconfigure to another image only through *factory* mode.

---

### 11.5.14. MP3 player interface

While studying MP3 player interface, keep in mind that GR8NET uses third party design, and it was not natively designed for GR8NET, and interface, as well as some special tasks to be performed on it, may look inconvenient. This is life, and fortunately there're ways making things working.

All MP3 player access is performed through I/O ports 5Eh and 5Fh, thus application may not even need to know the slot # GR8NET performing MP3 playback is installed in; the only thing it should know is the instance # (ID) of the GR8NET identified by the two most significant bits of the port 5Eh contents.

Two index registers are used – 06h and 07h – as shown in the chapter [Identification and detection](#). Register 06 is control register (R/W), and register 07 is data register (write only).

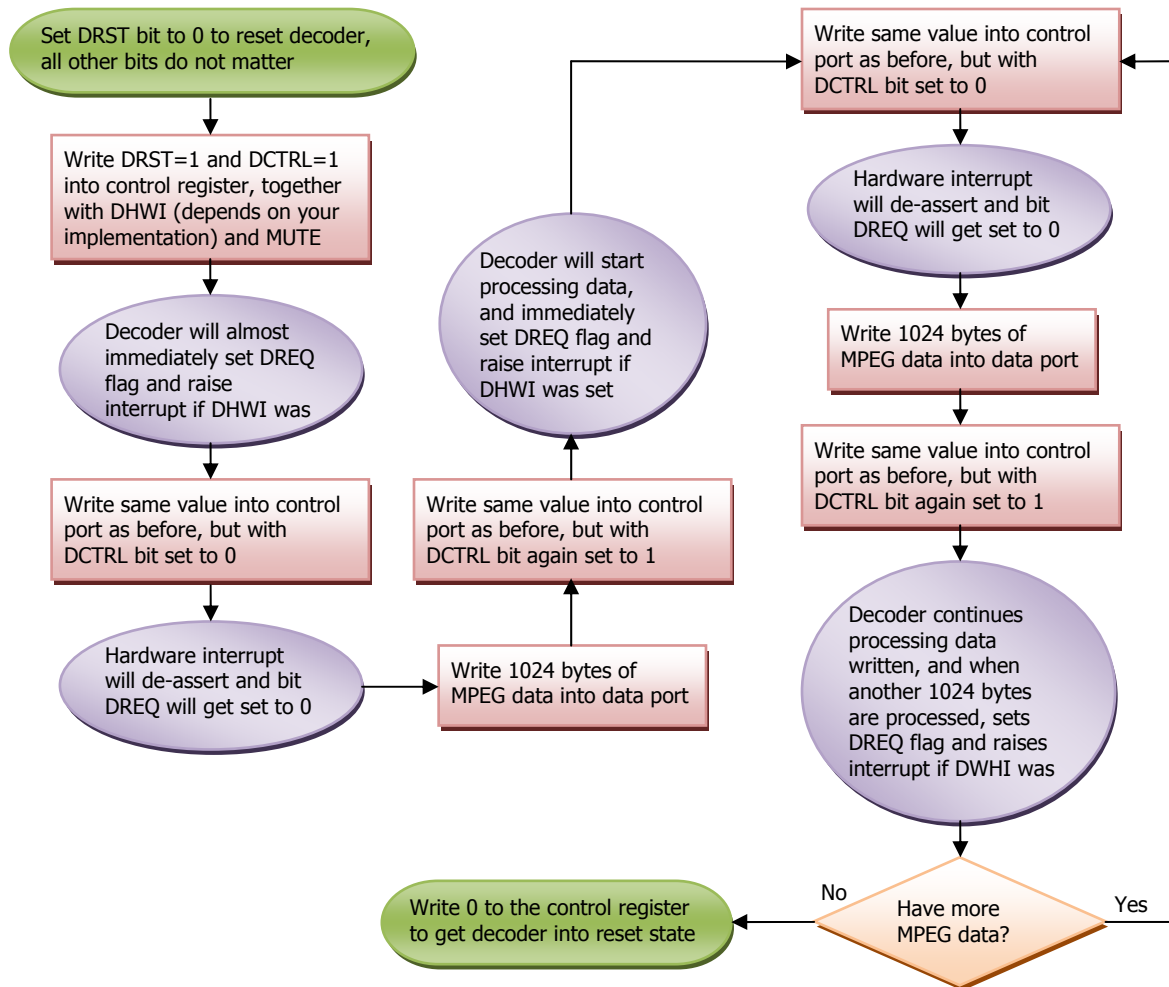
MP3 decoder control register (index 06h)

7	6	5	4	3	2	1	0
DREQ (R)	DHWI (R/W)	N/A (0)	N/A (0)	N/A (0)	DRST (R/W)	DCTRL (R/W)	MUTE (R/W)

Description of the control register

Bit	Purpose
DREQ	If this bit is set by the decoder, it is time to replenish decoder's data buffer through data port. The size of data is always 1024 bytes. DREQ will automatically reset when DCTRL bit is reset
DHWI	Writing 1 to this bit will enable hardware interrupt when DREQ gets set. Hardware interrupt will stay active while DREQ is active
DRST	If this bit is having value of 0, decoder is in reset state. Note that its buffers are not got reset, thus they will need to be purged before decoder starts producing proper sound when reused
DCTRL	This bit, together with DREQ, controls interface state machine. See the explanation below the table
MUTE	If this bit is set, output of the decoder is muted. Very useful when you purge decoder's buffers (so that user would not hear anything while purge is in progress)

Decoder data register (index 07h) is just a register which must have data bytes written at the specific time. Below is the explanation of the workflow. If write is performed in wrong time, it may cause MP3 player sound to corrupt (temporarily or permanently within current MP3 playback session).



The process is relatively simple: first you ensure that decoder is reset, then you bring it out of reset, with DCTRL bit set meaning that decoder's buffer RAM is unblocked for decoder access. At this time decoder has no data, and it immediately asks for first transfer of 1024 bytes. You set DCTRL to 0 (blocking decoder accessing RAM) and quickly write 1024 bytes of data into decoder, and then unblock it. Now decoder has some data to start, it starts with this data, but as it has 2048 bytes of data buffer, it can immediately ask for another 1024 bytes. So you do the same way – block RAM, write 1024 bytes, unblock it. And next time, when decoder has 1024 bytes freed, it raises DREQ and interrupt line to signal that it wants more 1024 bytes of data. At the end of data, you just reset decoder, possibly losing about 20 ms of the sample playback (as it is not possible to know the exact position of the decoder and if it is finished).

### 11.5.14.1. Identification of the MP3 GR8NET in the system

Code GR8NET ROM BIOS uses is shown below. It scans through all adapters, 0-3, and tries to find adapter with respective number (by reading 'G' and 'R' from index register 0), and then checks if index register 6 returns 0ffh. If it does, even if adapter is present in the system, then it does not have MP3 decoder in it. Otherwise routine returns with CY reset and adapter number set as active and its index register set to 6 (so it is ready for decoder control port access).

```
;*****  
; FNDMP3: find MP3 services within all the installed adapters  
;*****  
; in:      nothing  
; out:     CY is set if no GR8NET adapter with MP3 services is found  
; CY is reset if found, and then index register is set to #6  
; regs:    DE  
fndmp3:  
    ld     de,P10DATA  
    ld     a,(de)  
    and    030h  
fndout:  
    out    (05eh),a          ; select adapter 0, index reg #0  
  
    ; checking for adapter presence  
    in     a,(05fh)  
    cp     'G'  
    jr     nz,nomp3a  
    in     a,(05fh)  
    cp     'R'  
    jr     nz,nomp3a  
  
    ; adapter found  
    ld     a,(de)  
    or     06h                ; (CY is reset)  
    out    (05eh),a          ; go to index reg #6  
    in     a,(05fh)          ; get byte from index register 6, MP3 decoder control reg  
    inc    a                  ; is it 0ffh? (CY is not affected)  
    ret    nz                 ; found adapter with MP3 services, return with CY reset  
                                ; by previous OR  
nomp3a:  
    ; we are here to skip adapter  
    ld     a,(de)  
    and    0f0h                ; reset index reg #  
    add    a,040h              ; activate next adapter  
    jr     nc,fndout          ; of adapter is <=3, go for it  
  
    ld     a,(de)  
    and    030h  
    ld     d,a  
    ld     a,(ADAPID)  
    and    0c0h  
    or     d  
    out    (05eh),a          ; revert my adapter as active with index reg #0  
  
    scf  
    ret
```

---

### 11.5.14.2. Issues with MP3 decoder

Now to the issues with the decoder implementation:

1. If MPEG data is not in time, decoder's audio output continues playing the same existing samples from the output sample RAM, and it sounds weirdly. This is clearly a bug, but the way to fix it was not found so far;
2. Decoder is not designed for multiple run. It requires full state reset for proper audio operation: just setting DRST to 0 resets logic, but does not reset RAM contents. Thus when you re-run the decoders after it was reset, it still outputs remainders of the previous decoded data for about 20 ms, but it causes very negative experience. The workaround exists for it though: before you start decoder with your data, you start it muted, and supply 5 MP3 frames of silence onto its output. This way decoder's output RAM is purged with zeroes. The format of silence frames is the following:

First frame:

```
00000000: FF FB 90 64-00 0F F0 00-00 69 00 00-00 08 00 00
00000010: 0D 20 00 00-01 00 00 01-A4 00 00 00-20 00 00 34
00000020: 80 00 00 04-... followed by 381 bytes of 'U' (055h)
```

Further frames:

```
00000000: FF FB 90 64-40 8F F0 00-00 69 00 00-00 08 00 00
00000010: 0D 20 00 00-01 00 00 01-A4 00 00 00-20 00 00 34
00000020: 80 00 00 04-... followed by 381 bytes of 'U' (055h)
```

In total single frame is 417 bytes, thus to fill whole decoder's buffer with frames of silence you will need  $2048/417=4$  frames plus 380 bytes of 5<sup>th</sup> frame, or 5 full frames. This is approximation because when second 1024 bytes will go to buffer, first will start to be decoded and all buffers start to be purged, and as each stereo frame will decode into  $2*1152$  16-bit samples even single frame should be enough for clean-up, but that would be a good idea to feed several frames in order to ensure that job is done properly.

Below is the code used by the GR8NET firmware.

```
*****
;
; SILENC: play 5 MP3 frames of silense
*****
; in:      MP3 control register set up in the port 05eh
silenc:
    push    hl
    push    de
    push    bc
    xor     a
    out     (05fh),a        ; reset decoder
    ex      (sp),hl
    ex      (sp),hl        ; small delay
    ld      a,07h
    out     (05fh),a        ; start decoder, no interrupts, output muted

    ; wait until decoder wants more data
silnod:
```

```

in      a,(05fh)          ; get status
rlca
jr      nc,silnod         ; wait while no data wanted

ld      a,05h
out     (05fh),a          ; suspend decoder, muted

ld      a,(P10DATA)
inc     a
out     (05eh),a          ; now at decoder's data register

; we will write whole 2K decoder's buffer at once, it will be:
; - frame 1 header 36 bytes
; - frame 1 contents 'U' 381 bytes, in total 417 bytes, complete frame 1
; - frame 2 header. 36 bytes
; - frame 2 contents 'U' 381 bytes, in total 834 bytes, complete frame 2
; - frame 3 header. 36 bytes
; - frame 3 contents 'U' 381 bytes, in total 1251 bytes, complete frame 3
; - frame 4 header. 36 bytes
; - frame 4 contents 'U' 381 bytes, in total 1688 bytes, complete frame 4
; - frame 5 header. 36 bytes
; - frame 5 contents 'U' 344 bytes, in total 2048 bytes, partial frame 5
ld      de,0400h          ; D=4 complete frames, E=frame #
ld      c,05fh            ; output port #
wriFra:
call    wrihdr
ld      hl,0381d
call    wridat
inc     e
dec     d
jr      nz,wriFra         ; continue with 4 frames

; now send partial frame 5
call    wrihdr            ; its header
ld      hl,0344d
call    wridat

; we are finished writing whole buffer to decoder, starting it
ld      a,(P10DATA)
dec     a
out     (05eh),a          ; back to decoder control register
ld      a,07h
out     (05fh),a          ; start decoder, muted

; wait until decoder requests for second chunk of data (data is already in place)
silno1:
in      a,(05fh)          ; get status
rlca
jr      nc,silno1         ; wait while no data wanted

ld      a,05h
out     (05fh),a          ; suspend decoder, muted
ex      (sp),hl
ex      (sp),hl
ld      a,07h
out     (05fh),a          ; tell decoder that data is already in there, muted

; wait until decoder requests for second chunk of data (to finish)
silno2:
in      a,(05fh)          ; get status
rlca
jr      nc,silno2         ; wait while no data wanted

; finished, reset decoder
xor     a
out     (05fh),a          ; reset decoder
pop     bc
pop     de

```

```

    pop    hl
    ret

; write header to the decoder
; in:      C=05fh, E=frame #
wrihdr:
    ld     hl,silhrd
    ld     b,04h
    otir                    ; write first 4 bytes of the header
    ld     a,e
    or     a
    jr     z,firfra         ; first frame

; we are here if we are not at the first frame
    inc    hl
    inc    hl                ; move pointer to consecutive frame side info bits

firfra:
    outi                    ; start og side info
    outi                    ; continuation of side info
    ld     hl,silhr3
    ld     b,silhre-silhr3
    otir                    ; write remaining of the side info
    ret

; write same char to decoder
; in:      HL=char count, C=05fh
wridat:
    ld     b,'U'
contwr:
    out    (c),b
    dec    hl
    ld     a,h
    or     l
    jr     nz,contwr
    ret

silhrd:
; 00000000: FF FB 90 64-(00 0F)/(40 8F) F0 00-00 69 00 00-00 08 00 00
; 00000010: 0D 20 00 00-01 00 00 01-A4 00 00 00-20 00 00 34
; 00000020: 80 00 00 04-... followed by 381 bytes of 'U' (055h)
    db     0ffh,0fbh,090h,064h                ; followed by 0,0fh for first frame and 40,8f for following
silhr1:   db     0,0fh
silhr2:   db     040h,08fh
silhr3:   db     0f0h,0,0,069h,0,0,0,8,0,0
          db     0dh,020h,0,0,1,0,0,1,0a4h,0,0,0,020h,0,0,034h
          db     080h,0,0,4
silhre:

```



---

## 11.6. Mapper modes

When system is powered on, GR8NET starts in mapper mode 0. Then you can change mapper type in BASIC using `_NETSETMAP` command. After the change, system will reboot to make new mapper type effective.

Mapper modes 1-6 are pure game mappers, as soon as GR8NET is switched to these modes, the ROM image is not writable any more, please ensure to copy all the required ROM data into GR8NET RAM before switching the mapper mode.

For composite mapper modes 9-14, game mapper ROM contents are also accessing through the GR8NET RAM in subslot 0, thus it is possible to modify ROM contents when slot 3 is switched to game mapper.

### 11.6.1. Mode 0: GR8NET internetworking adapter

To change to mapper mode 0 in BASIC please use the following command:

**CALL NETSETMAP(16)**

In this mode application may enjoy whole range of functionalities described in this document, depending on FPGA image being run (e.g. regular or MP3 player). The important requirement for proper software operation is having logical page 80h (first ROM page) in the bank 0.

In mapper mode 0 GR8NET is mapped into Z80 visible memory space in the following way:

Address range	Function	
4000 ... 5FFF	Bank 0 (default is beginning of the ROM chip space)	Regs
6000 ... 7FFF	Switchable bank 1	Regs
8000 ... 9FFF	Switchable bank 2	Regs
A000 ... BFFF	Switchable bank 3	Regs

Special control registers must be available in bank 0 only. If application will switch registers on in another bank(s), it should switch them off before calling any API of the card as card does not expect these registers to be in banks 1, 2 and 3.

---

### 11.6.2. Mode 1: plain 32kByte write-protected memory chunk

To change to mapper mode 1 in BASIC please use the following command:

**CALL NETSETMAP(1)**

In this adapter represents its first 32kBytes of onboard RAM as plain contiguous space starting 4000 and ending BFFF. After machine reboot, MSX BIOS only turns bank 1 (4000-7FFF) in the cartridge's slot location, if software needs to use bank 2 (8000-BFFF), it should turn this bank in cartridge slot manually. Software must load ROM image (data) starting logical page 0 before switching mapper mode to 1.

### 11.6.3. Mode 2/3: Konami memory mappers

To change to mapper mode 2 or 3 in BASIC please use one of the following commands:

**CALL NETSETMAP(2)** or

**CALL NETSETMAP(3)**

These memory mappers feature in the games of size more than 32kBytes like Vampire Killer, King's Valley 2 or Metal Gear 2: Solid Snake. The difference between them is that in mode 2 mapper has fixed mapper page 0 in its 4000-5FFF location, does not have SCC in its page 3F in location 9800-9FFF, and mapper type 2 (K4 mapper) has size of 256 KB while mapper type 3 (K5 mapper) has size of 512 KB.

To eliminate conflict between K5 game mapper memory space and MSX-Audio sample RAM space, when mapper mode change is performed by \_NETSETMAP command, the start page of sample RAM is moved to logical page 60h (thus in maximal configuration of 32 pages occupies logical pages 60h-7Fh at the end of GR8NET 1MB buffer RAM).

After reset or power cycle SCC implementation initializes in SCC compatibility mode (since 27 Feb 2018). For more information about SCC/SCC+ refer to the [Sound custom chip \(SCC/SCC+\)](#) chapter and Albert Beevendorp's tech pages.

After mapper type is changed to 2 or 3 through GR8NET I/O register, banks are assigned with pages 0, 1, 2 and 3.

### 11.6.4. Mode 4: ASCII-8 memory mapper

To change to mapper mode 4 in BASIC please use the following command:

**CALL NETSETMAP(4)**

This mode is very similar to the Konami mapper mode 2 with slightly different bank switching addresses, and availability of the 7 address bits which allow 1MB of total addressable RAM. When switching to this memory mapper through port I/O or \_NETSETMAP command, all 4 game mapper banks are initialized to page 0.

---

### 11.6.5. Mode 5: ASCII-16 memory mapper

To change to mapper mode 5 in BASIC please use the following command:

**CALL NETSETMAP(5)**

This mapper assumes changing contents of only two CPU 16KB banks, 4000-7FFF and 8000-BFFF. GR8NET adapter emulates this behavior using its 8K banking system: for example if application switches 16KB CPU bank 1 to page number 5, GR8NET engine will switch two of its banks – bank 0 and bank 1 – to pages 10 and 11. Thus each 16KB page X is represented by the two logical GR8NET pages equal to  $X*2$  and  $X*2+1$ . When switching to this memory mapper through port I/O or `_NETSETMAP` command, all 2 game mapper banks are initialized to page 0.

### 11.6.6. Mode 6: Mirrored ROM

To change to mapper mode 6 in BASIC please use the following command:

**CALL NETSETMAP(6)**

This mode is similar to mode 1, but first 64 Kbytes of GR8NET RAM are presented from the CPU address 0m thus ROM “AB” header must appear at the ROM absolute address of 4000.

### 11.6.7. Mode 7: 1 Megabyte mapped memory

To change to mapper mode 7 in BASIC please use the following command:

**CALL NETSETMAP(7)**

In this mode card represents its available RAM to the MSX machine as normal mapped RAM, available in all banks. Mapper is having its internal memory mapping registers at ports 0FCh-0FFh, its readability will be controlled by second argument of `_NETSETMAP` command (respective bit in System mode register 0).

Cartridge is fully dedicated for mapped RAM operation, there's no GR8NET ROM initialization, and no `_NET`, `_DSK` or `_FL` commands present, and BASIC will return *Syntax error* if you try using them.

If card will be the one with largest memory installed into the *non-Turbo* machine, its space will become main memory for MSX. This may affect behavior of some applications and games which by default expect RAM to be in slot 3.2.

### 11.6.8. Modes 8-14: Composite mappers

*To work in these mapper modes GR8NET adapter must be in primary slot.*

These modes represent mixture of the mapper modes 0 to 6, they are very useful in case your MSX machine is having small amount of RAM, or you want extra control over data in the read-only game mapper. When booting, and if machine is not a Turbo-R, GR8NET will be chosen as main RAM with 512K in size (if there're no other larger memory mappers). In addition, game mappers present in subslot 3, will have access to GR8NET's Nextor system with SD-card.

The following diagrams show memory mapping assuming GR8NET is installed in primary slot X.

Slot X.0	Slot X.1	Slot X.2	Slot X.3
Contains GR8NET ROM with full functionality. Available RAM size will be 512K, otherwise 1MB if mapped RAM is explicitly disabled using MMDR bit of system mode register 0. Running _NET and _DSK commands (browser and other accessing GR8NET RAM buffer) will alter this RAM contents.	If not disabled by MMDR bit of the system mode register 0, contains 512K mapped RAM, the second half of the available GR8NET onboard RAM space. This mapped RAM can be disabled by setting 3 <sup>rd</sup> argument of _NETSETMAP to 1, or setting third argument in brackets of file name to 1, e.g. {311} when using browser.	Contains <b>Nextor</b> ROM. Note that if application being run is game, or any other not supporting FAT16 volumes, you must insert SD-card with first partition formatted in the supported way for game (e.g. standard diskette image 720K/FAT12), and boot GR8NET's Nextor in DOS1 mode holding '1' key during its initialization.	Contains game mapper or FM-Pak ROM depending on mapper type selected: 8: FM-Pak ROM 9: Plain 32K 10: Konami 4 11: Konami 5 / SCC 12: ASCII-8 13: ASCII-16 14: Mirrored  See details below.

Subslot 3 will have respective game mapper type identified by mapper number minus 8, thus mapper mode 11 will have Konami 5 (8+3) mapper in place. If you select mapper type 8, then FM-Pak ROM will appear in subslot 3.



**Important note:** game mappers in subslot 3 share their space with GR8NET buffer RAM located in subslot 0. As soon as RAM in the location is writable, you can alter (or corrupt) ROM image in game mapper in subslot 3 through subslot 0. While you can load ROM images on the fly using GR8NET browser, and they will appear in subslot 3 immediately, using specific GR8NET commands (e.g. starting browser or using LDBUF) may corrupt current data in subslot 3.

---

When changing to modes 8-14, do not forget that configurations having GR8NET functionality in them require *special registers* to be set in GR8NET bank 0, and take special care about mapper read flag, thus to perform change you may do the following:

CALL NETSETMAP(16+8, 2)      to switch to mapper mode 8 with "auto" mapped RAM mapper read flag

CALL NETSETMAP(16+8+3,,1)      to switch to mapper mode 11 with Konami SCC mapper in subslot 3, also with "auto" mapped RAM mapper read flag, and have mapped RAM in subslot 1 disabled (e.g. special case for Metal Gear 2 because it can not run in the same slot with main RAM).

Another tradeoff of having mapped RAM together with GR8NET mapper is that RAM disk can only be of 360K or 256K in size (see [Memory manager](#) chapter), and, together with RAM disk enabled, will have only 152K memory available for system and user. In case 720K-sized disk is loaded into the RAM disk space, GR8NET firmware will throw warning, and all reading or writing sectors exceeding configured space limit will return *Not ready* error.

#### **Examples of mapper switching commands**

**CALL NETSETMAP(24)** switches to mapper mode 8

**CALL NETSETMAP(25)** switches to mapper mode 9

**CALL NETSETMAP(26)** switches to mapper mode 10

**CALL NETSETMAP(27)** switches to mapper mode 11

**CALL NETSETMAP(28)** switches to mapper mode 12

**CALL NETSETMAP(29)** switches to mapper mode 13

**CALL NETSETMAP(30)** switches to mapper mode 14

#### **11.6.8.1. RAM allocation conflicts in composite mappers**

GR8NET is having 1MB of its buffer RAM, and in specific circumstances there could be the cases when this RAM is not enough to accommodate workspace of all the devices activated at the specific moment.

These three devices include:

1. Game mapper: mappers affected are 11 (K5/512KB), 12 (ASCII8/1024K) and 13 (ASCII16/1024K);
2. Mapped RAM: 512KB (half of the GR8NET buffer RAM);
3. MSX-Audio sample RAM is enabled takes up to 256KB space.

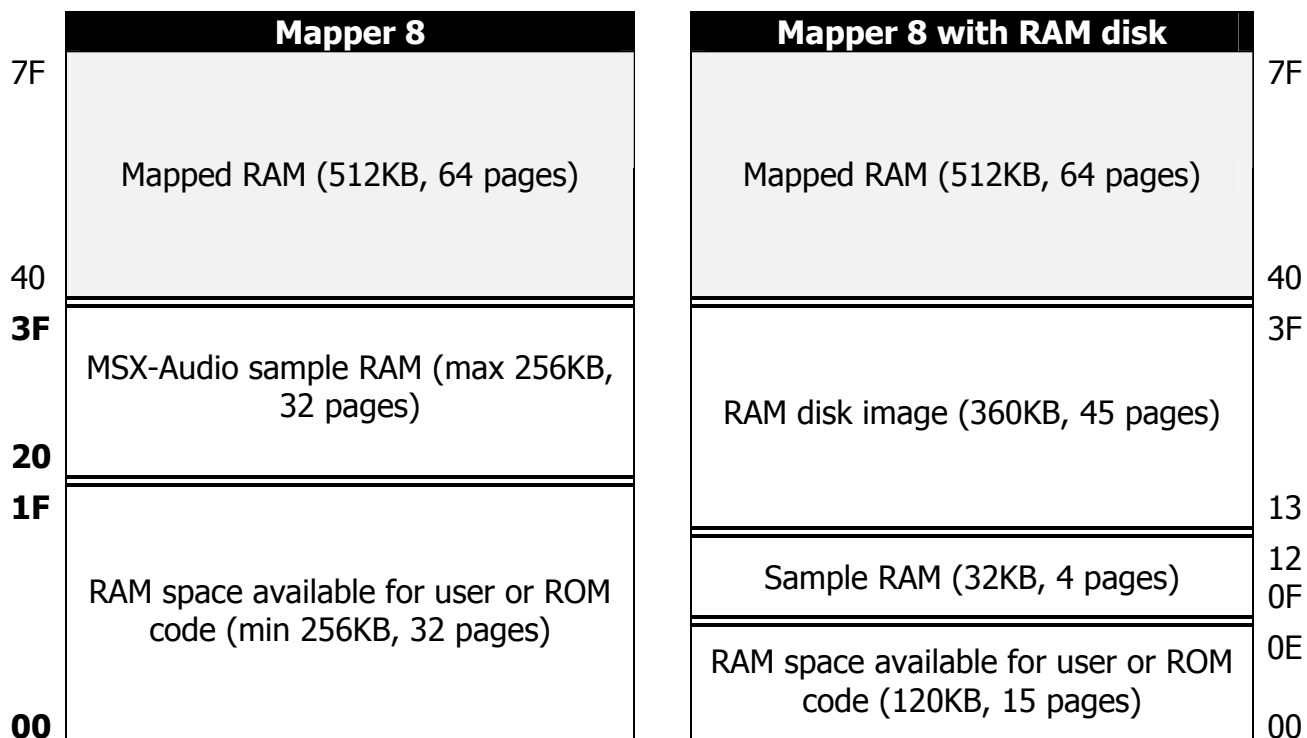
The exact GR8NET RAM allocation is explained in detail in the chapter [Memory management](#).

Let's consider the picture below, showing typical GR8NET RAM allocation in the mapper mode 8, which has no game mapper.

On the left diagram you can see that mapped RAM takes half of the RAM at the top pages 40-7F, then MSX-Audio sample RAM, if MSX-Audio is enabled, takes ¼ of the RAM in the pages 20-3F, and there's only ¼ of the remaining space (256 KB) available for the

user code and data (e.g. using `_NETBLOAD`). Actual RAM allocation can be obtained using command `NETGETMMV`.

On the right diagram you can see that GR8NET is smart by allocating only 32KB (maximum) if RAM disk is enabled – 32K is usually enough for utilities like VGMPLAY or MoonBlaster application, and there is still 120KB space available for browsing, bloading and user data.



**Here're the methods to ensure such conflict does not occur:**

1. There's no issue if ROM image size you load into GR8NET is  $\leq 256\text{KB}$  (see left diagram above);
2. If ROM image size you load is  $> 256\text{KB}$  but  $\leq 512\text{KB}$ , you have the following options –
  - a. Disable MSX-Audio completely using `_NETSETOPL(4)`; it will also disable sample RAM. In this case built-in Y8950 will not be available at all;
  - b. Set MSX-Audio sample RAM to 0 using `_NETSETOPL(,0)`; then system will output FM sound generated by built-in Y8950, but will have sample RAM unavailable for reading and writing;
  - c. Disable mapped RAM using third argument of mapper change command, e.g. `_NETSETMAP(27,,1)`; in this case GR8NET ROM BIOS will move sample RAM to the page 60h, and system will have fully functional MSX-Audio, but no mapped RAM in subslot 1.
3. If ROM image size you load into ASCII8 and ASCII16 mapper types is  $> 512\text{KB}$ , the only way to keep ROM intact is to perform both disable of mapped RAM using e.g. `_NETSETMAP(28,,1)` and disable sample RAM by either `_NETSETOPL(4)` or `_NETSETOPL(,0)`.

---

### 11.6.8.2. Limitations of setting target mapper to composite mappers

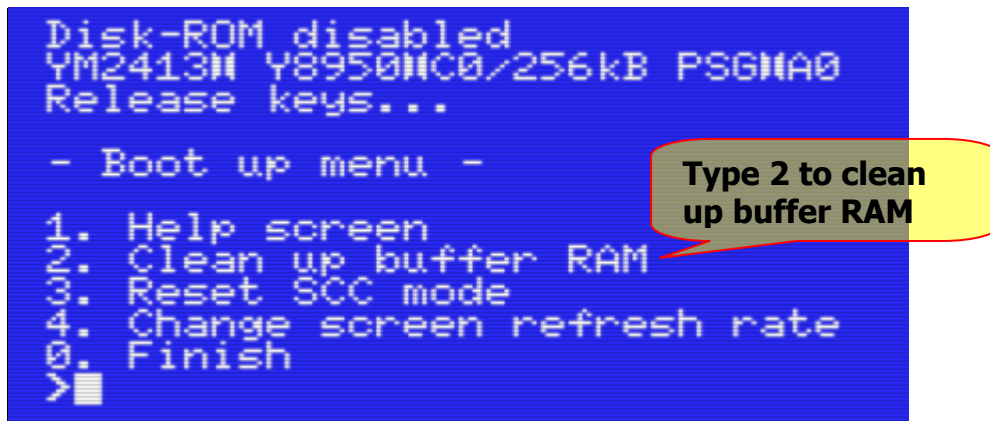
There're cases when you would want to set target mapper to one of the composite mappers containing game mapper, for example using `_NETTGTMAP(27)` setting it to mapper 11. While it is convenient making GR8NET reconfiguring from mapper 0 to mapper 11 on the boot-up, it has one big drawback you must know about.

In the game mapper mode, the contents of the GR8NET buffer RAM appear in two places:

- It appears in the GR8NET subslot, and is available for read and write. This means that any command you run altering GR8NET RAM buffer – for example `_NETBROWSE` or `_NETBLOAD` – will corrupt the contents of ROM;
- And it appears in the subslot 3 as respective mapper type set by the `_NETSETMAP`. These contents are read-only, and subject to functionality of specific game mapper.

The second point provides uncontrolled risk: if GR8NET RAM contents start with valid ROM header (characters "AB" followed by valid call table), MSX ROM BIOS will call respective ROM address, and ROM will start. Resetting the machine will not help – contents will still be there, and after reconfiguration to the target mapper machine will always start the ROM. The probable way could be power cycling the machine in the hope that after power cycle contents will change and will not provide valid ROM header to the MSX BIOS any more.

There's better way to deal with this issue (since March 2018): there's boot-up menu allowing you cleaning up the GR8NET buffer RAM space. When GR8NET initializes, press and hold **TAB key** until it instructs to release the key, and you will get the following menu:



You select option 2, and GR8NET automatically identifies the size of buffer RAM to clean up (so that it do not clean up mapper RAM which is potentially already initialized as main RAM by the MSX ROM BIOS).

---

## 12. Programming API

GR8NET in its mapper mode 0, and in composite mappers 8-14 in subslot 3, has predefined page allocation: bank 0 is always logical page 80h (ROM start with calling points), bank 1 is always RAM (by default configuration logical page 0FFh), bank 2 is always switchable ROM page, and bank 3 is always W5100. When application starts, it may expect such bank allocations; when exiting, application should revert back to the original bank allocations.

Some calls will modify pages visible in banks, for example TCPEST will switch to W5100 registers in bank 3, and thus programmer should pay attention to the changing pages in the banks after calling GR8NET API.

If you are going to use BDOS (0005h) call in MSX-DOS environment you should know that if you have GR8NET slot switched on in CPU banks 1 or 2 BDOS call may change them back to RAM slot.

Your application will interface with GR8NET directly – identifying card using ports 5Eh and 5Fh, and accessing GR8NET adapter's RAM, ROM and W5100.

### 12.1. Identification of the adapter

It is user's task to properly enumerate adapters within the system by their configuration switches, and connect respective network cables to each adapter. Applications may provide choice of the adapters and their functionalities. Applications may perform specific auto-detection actions to find out network adapters are connected to – e.g. using DHCP requests to see which IP address adapter is given, or performing ping to the predefined remote host.

To identify if adapter #2 is installed, application can use the following execution flow:

```
...
ld      a,080h          ; adapter #2, register 0
out     (05eh),a        ; select adapter and register
in      a,(05fh)        ; get register 0 value from adapter #2
cp      'G'
jr      nz,noadap       ; no adapter #2

ld      a,081h          ; adapter #2, register 1
out     (05eh),a        ; select adapter and register
in      a,(05fh)        ; get adapter #2's register 1 (slot ID)
cp      0ffh
jr      nz,inierr       ; adapter did not initialize properly

ld      (slotid),a      ; otherwise preserve adapter #2 slot ID

ld      a,082h          ; adapter #2 register 2
out     (05eh),a        ; select adapter and register
in      a,(05fh)        ; get adapter #2's register 2 (mapper type reg)
...
```

After identifying slot ID adapter #2 is located in, application will switch to this slot and operate the adapter.

If application will want to detect GR8NET device using slot and subslot scan, it will be able to find GR8NET adapter by the string "GR8NET" at the address 5FB8h followed by 2 bytes identifying major and minor version of the adapter's software.

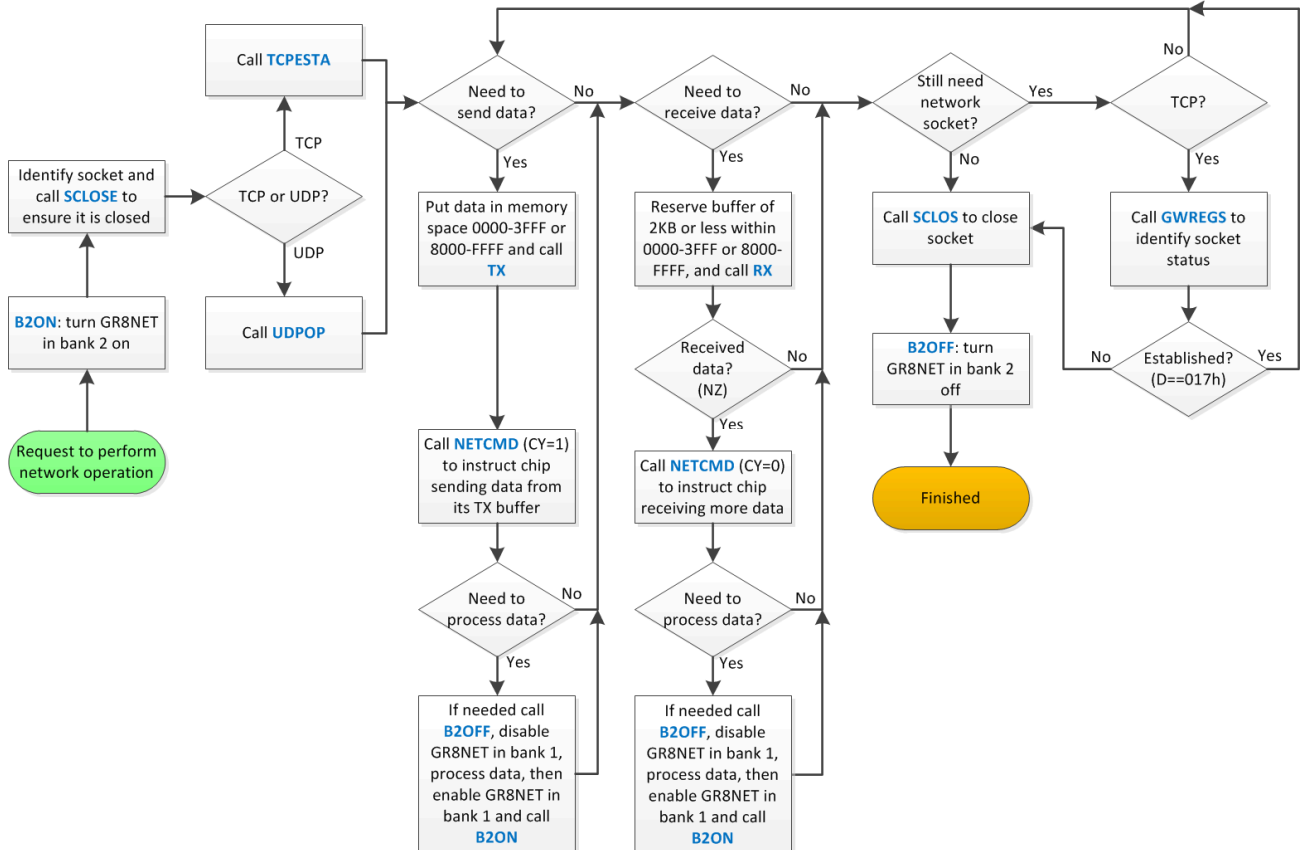


## 12.2. Direct firmware calls

When you have GR8NET adapter slot switched on in CPU bank 1 and GR8NET mapper bank 0 is having page 80h visible, you have access to the direct call API, which provides specific very useful functionalities to your application.

Some firmware calls require turning GR8NET in CPU bank 2 on, use B2ON and B2OFF routines to turn it on and off respectively. Ensure not to call B2ON more than once to enable GR8NET slot in bank 2: the routine uses single variable for previous slot assignment storage, and calling it write will overwrite original slot value with slot value of GR8NET.

You should keep in mind that during execution of these calls executing code should still have access to the supplied data, do not place strings to print to the screen into main PC RAM in CPU bank 1, because executing code is located in this bank. If you need temp storage, you can use GR8NET's RAM in GR8NET bank 1 (6000-7FFF) *configuration* page 0FFh from addresses 7800h to 7FFFh (do not use 6000-77FF as this space contains GR8NET firmware control data and can be used for temporary data storage).



Example of network operation workflow using GR8NET direct firmware calls

**URISOP** URI structure operations (added 20171111)

Address 5F6Dh

Operation 0: Disassemble host name string into URI structure

Input A=0

HL'=pointer to URI structure

HL=host name string to disassemble

Output CY is set if input string is empty or too long to fit into host name field(s) of the current URI structure version (max 31 for v0 and max 63 for v1)

Registers AF, AF', BC, DE, HL

Notes Instrumental function for URI structure v1 when host name is split into two parts.

Operation 1: Assemble host name from URI structure's fields into string

Input A=1

HL'=pointer to URI structure

DE=target location for host name string

Output DE=pointer after the string, string is not terminated with null character

Registers AF, BC

Notes Instrumental function for URI structure v1 when host name is split into two parts. If URI structure is v0 maximal string length is 31 bytes, if v1 this length is 63. Host name string will not contain device/protocol name (without HTTP:// or SDC:// identifiers).

Operation 2: Make full URI string from the URI structure

Input A=2

HL'=pointer to URI structure

DE=target location for URI string, string is terminated with null

Output HL=pointer to the terminating null character

Registers AF, BC, DE

Notes Maximal size of the target location is 7 ("HTTP://"), plus 31 or 63 (host name, depending on the URI structure version), plus 6 (port ":xxxxx"), plus 240 (path), plus 63 (name) and plus 63 (query string) with terminating null, and equals to 411 for URI structure version 0, and 443 for URI structure version 1.

**DATCOD** Get year, month and day of flash chip firmware build

Address 5F70h

Input -

Output DE=year (e.g. 2016), B is month (1-12), C is day (0-31)

Registers -

Notes You can use this call to identify firmware build date and thus if capabilities application requires are available or not. To find out if this call is available or not check address of (05F70h) to contain JP instruction (0C3h).

**UDPOP** Open socket for UDP communication

Address 5F73h

Input A = socket number (0 or 1)

DE = UDP port number

---

Output	CY is set if error
Registers	AF, BC, HL, IX, IY
Notes	Before calling this routine, GR8NET should be turned on in CPU bank 2 with B2ON

---

**GCURL** Get current slot ID information

Address	5F76h
Input	D = CPU bank number (0, 1, 2, 3)
Output	A = slot ID in RDSLT format
Registers	AF, BC, DE, HL
Notes	For bank 1 A will return own GR8NET slot information, or subslot 0 of own slot if adapter is mapper mode 8.

---

**B2OFF** Turn GR8NET adapter off in CPU bank 2 (8000-BFFF)

Address	5F79h
Input	Nothing
Output	Nothing
Registers	-
Notes	Restores visibility of slot (subslot) before B2ON is called

---

**B2ON** Turn GR8NET adapter on in CPU bank 2 (8000-BFFF)

Address	5F7Ch
Input	Nothing
Output	Nothing
Registers	-
Notes	Turning GR8NET on in CPU bank 2 is required to call several firmware calls (TX, RX, TCPEST, UDPOP) allowing access to GR8NET banks 2 (expansion ROM pages) and 3 (LAN chip). This routine should NOT be called recursively, as it stores previous slot ID information in the single memory cell, and if called second time, information in this cell will be rewritten by the GR8NET slot ID

---

**GWREGS** Get W5100 socket status registers

Address	5F7Fh
Input	A = socket number (0 or 1)
Output	D = socket status register E = socket interrupt register
Registers	IX, IY
Notes	For the information about meaning of the information returned in DE please refer to the W5100 datasheet. In most cases application will need <i>socket status register</i> to identify the state socket is in (open, established, closing, closed) to perform appropriate operation in it and its data exchange.

---

**NETCMD** Instruct W5100 to perform send or receive command

Address	5F82h
Input	A = socket number (0 or 1)

	CY is set to instruct socket to send
	CY is reset to instruct socket to receive
Output	CY is set if socket error occurs
Registers	IX, IY, HL, BC
Notes	This routine should be executed after calling TX or RX: after TX in order to instruct W5100 to send data transferred to its buffer, after RX for acknowledging previously received data and get ready receiving new data (TCP) or free space to receive UDP packet(s). This routine need <i>not</i> be executed in the loop, once it is executed, W5100 is performing requested task. In case of receive command subsequent call to this routine in TCP mode will cause W5100 to re-acknowledge previously received data to the remote host (a kind of network retry)

<b>RX</b>	Get received data from the socket
Address	5F85h
Input	A = socket number (0 or 1) HL = pointer to RAM buffer (must be outside of CPU bank 1) DE = maximum buffer size (data will be $\leq$ 2KB)
Output	ZF is NZ if there's more data in network chip's RX buffer BC = size of data copied into the buffer (if ZF is NZ) HL = at the end of data in the RAM buffer (if ZF is NZ)
Registers	All registers in alternate register set, IX, IY
Notes	GR8NET should be switched on in CPU bank 2 with B2ON. This routine does not issue RECV command to the W5100 chip's socket, use NETCMD call with CY reset for this purpose

<b>MMVAR</b>	Get / set memory manager variables
Address	5F88h
Input	CY is set to set user protected area starting page A=user protected area starting page (UPRAMS) CY is reset to read memory manager variables
Output	CY is set if UPRAMS can not be set (in case of input CY being set) H = RAMMAX (total number of RAM pages available) L = DSKLPG (RAM disk image starting page) D = RAMTOP (maximal page number available for user) E = UPRAMS (first page number of user-protected RAM area) B = number of RAM pages available within user protected area For more information please refer to <a href="#">Memory manager</a> chapter.
Registers	AF, C
Notes	Gives information about availability of the GR8NET buffer RAM, and reservation of the user space which will not be used by system tools like NETBLOAD.

<b>DEV8RW</b>	DEV_IO routine for built-in Nextor
Address	5F8Bh
Input	CY=0 to read, 1 to write

---

	A = Device number, should be 1
	C = Logical unit number, should be 1
	B = Number of sectors to read or write
	HL = Source or destination memory address for the transfer
	DE = Address where the 4 byte sector number is stored
Output	A = Error code
Registers	All
Notes	Reads specified number of sectors from SD-card into memory location. IMPORTANT: routine does not implement XFER method, thus space pointed by HL <b>must</b> be located in RAM. Size of SD-card in sectors can be obtained from <i>SD-card size</i> register from special register set.

---

<b>PARURI</b>	Parse URI string into the URI structure
---------------	---

Address	5F8Eh
Input	HL points to the URI string BC points to the URI structure
Output	-
Registers	All
Notes	Function populates fields which are present in the URI string. For example, if destination port is not listed in URI string, this field will not be updated. Thus before calling network access routines ensure that all fields in URI structure are correctly populated

---

<b>TCPEST</b>	Establish TCP/IP connection
---------------	-----------------------------

Address	5F91h
Input	A socket number (0, 1) BC points to the URI structure
Output	CY is set if error occurs
Registers	All
Notes	Do not use sockets 2 or 3 as they may be used by system routines like BLOAD and DHCP/DNS requests. This routine performs DNS query on the host name in the URI structure if its flag is set to 0. Before calling this function, GR8NET bank 2 should be turned on with B2ON

---

<b>PUTSTR</b>	Print inline string
---------------	---------------------

Address	5F94h
Input	Null-terminated string follows CALL instruction
Output	-
Registers	AF
Notes	This routine uses CHPUT function of the MSX BIOS

---

<b>PRSTR</b>	Print formatted string pointed by HL
--------------	--------------------------------------

Address	5F97h
Input	HL points to the null terminated text BC for %c

Output	-
Registers	All
Notes	<p>Uses MSX BIOS CHPUT routine. String should not be located in CPU bank 1.</p> <p>Format:</p> <ul style="list-style-type: none"> <li>• “%%” prints % sign</li> <li>• “%c” prints contents of BC register in hexadecimal representation</li> <li>• “%H” followed by byte count [1 byte] and address [2 bytes] prints bytes in hexadecimal format of byte count from location pointed by address (i.e. performs memory dump)</li> <li>• “%A” is similar to “%H”, however %A is followed by termination character [1 byte] then maximal number of characters to dump [1 byte] and then address [2 bytes].</li> </ul> <p>In %A and %H options 0 byte count dumps 256 bytes.</p>

<b>GHCODE</b>	<b>Get HTTP response code from the HTTP header</b>
---------------	--

Address	5F9Ah
Input	HL = pointer to HTTP header
Output	<p>CY reset then HL = HTTP response code</p> <p>CY set if error</p>
Registers	All, except BC is preserved
Notes	The header string should be null-terminated

<b>MGETRQ</b>	<b>Construct GET HTTP request</b>
---------------	-----------------------------------

Address	5F9Dh
Input	<p>HL points to the User Agent string</p> <p>BC points to the URI structure with all fields filled in</p> <p>DE points to the destination memory area</p>
Output	<p>CY set if error</p> <p>CY reset then HL = pointer to created request string, BC = byte count</p>
Registers	All
Notes	This function provides HTTP request text to be sent using TX call

<b>TX</b>	<b>Put packet into TX buffer of W5100</b>
-----------	---

Address	5FA0h
Input	<p>A = socket # (0 or 1)</p> <p>HL = pointer to data to send within CPU banks 0, 2 and 3</p> <p>DE = number of bytes to send (<math>\leq 2\text{KB}</math>)</p>
Output	<p>If CY is set there's no space in socket's buffer, and</p> <ul style="list-style-type: none"> <li>• HL is unchanged</li> <li>• BC = free space available within socket's TX buffer</li> </ul> <p>If CY is reset then</p> <ul style="list-style-type: none"> <li>• HL points to the end of data transferred to TX buffer</li> <li>• BC equals to DE</li> </ul>
Registers	IX, IY, F, BC
Notes	Application should successfully establish connection with remote host before

calling this function. Before calling this function, GR8NET bank 2 should be turned on with B2ON. Data and its parts should not be located within CPU bank 1 (4000-7FFF). Minimal number of bytes to send is required for UDP communication so that routine would not send partial UDP packet. To instruct network chip sending the data to the wire perform call to NETCMD call with CY flag set.

#### **GETDEC** Get decimal representation of the number from the text

Address 5FA3h  
 Input DE = text in memory, can be preceded with leading spaces  
 Output CY set if error, DE is not changed  
 CY reset then BC:HL = 32-bit number, DE points to byte after recognized number  
 Registers All

#### **CHKHIP** Check host name to be an IP address

Address 5FA6h  
 Input BC = URI structure  
 Output -  
 Registers All  
 Notes This routine should be called before TCPEST; it checks host name to be an IP address, and if it is, moves this IP address into resolved IP address field and sets flag that IP address is valid

#### **CHPUT** Put character onto the screen

Address 5FA9h  
 Input A = character to print  
 Output -  
 Registers None  
 Notes This routine uses MSX BIOS CHPUT routine

#### **PRDE16** Print 16-bit number in decimal representation

Address 5FACH  
 Input HL = value to print  
 Output -  
 Registers AF  
 Notes This routine uses MSX BIOS CHPUT routine

#### **PRIPA** Print IP address in decimal notation onto the screen

Address 5FAFh  
 Input HL = pointer to 4 octets of IP address  
 Output -  
 Registers All  
 Notes This routine uses MSX BIOS CHPUT routine

#### **SCLOSE** Closes socket

---

Address 5FB2h  
Input A = socket #  
Output -  
Registers IX, IY, AF

**PRHEX** Print hexadecimal representation of A (2 characters)

Address 5FB5h  
Input A = byte  
Output -  
Registers AF

**GR8RSG** GR8NET adapter ROM signature

Address 5FB8h  
This address contains characters "**GR8NET**" followed by major (byte) and minor (byte) versions of the flash chip firmware.



## 12.3. URI structure

You will use URI structure to provide input for several firmware calls like TCPEST, PARURI, MGETRQ or CHKHIP. Before performing any of these calls, however, you should ensure that URI structure is set up with information required for successful execution of the call. For example, TCPEST requires source and destination ports, as well as proper setting of the flag.

+#	Len	Network URI structure	SD-card URI structure
+0	1	URI structure type (see below the table)	
+1	32 (31+'\0')	Host name	32-byte directory entry (see below)
+33	4	IP address. If it corresponds to host name (or host name is empty and IP address should be effective) if bit 1 of byte in offset +0 is set	Starting cluster of the file (32 bits), valid if bit 1 of byte in offset +0 is set
+37	2	Remote (destination) port (e.g. 80 for HTTP)	File size in bytes (32 bits), valid if bit 1 of byte in offset +0 is set
+39	2	Local (source) port. Do not use port numbers below 49153. Special case is port value 0 means dynamic port number selected by the firmware	
+41	240 (239+'\0')	Path on the server	Path for the file
+281	64 (63+'\0')	File name of the resource on the server	File name
+345	64 (63+'\0')	Query string for the server's resource	N/A, is not used
+409	EOS0	End of URI structure v0	End of URI structure v0
+409	32 (31+'\0')	Extended host name for URI structure v1	
+441	EOS1	End of URI structure v1	End of URI structure v1

Since November 2017 URI structure was extended to second host name field. This new structure is called URI structure version 1, while original is called URI structure version 0. Version of the structure (and thus firmware handling of its space and its contents) is identified by the bit 6 of URI structure type. This extension format was chosen to keep new firmware compatible with applications designed for the initial URI structure version 0.

With URI structure version 1, host name (at +33) and extended host name (at +409) form the host name, and are considered as single null terminated string, leading to 63 characters plus null character.

To identify if the current GR8NET firmware revision supports URI structure v1, perform call to PARURI having URI string more than 32 host name characters, for example HL pointing to "http://012345678901234567890123456789012", if PARURI will return CY flag set then firmware does not support v1 structures.

Special byte at the location +0 in the structure identifies type of the URI structure, its version and its state. The byte is a bitmap of several flags, with bits having different meaning for network and SD-card URI structures.

Bit	Value or meaning for network URI structure	Value or meaning for SD-card URI structure
0	0 (identifies resource as network)	1 (identifies resource as SD-card located)
1	This bit is set to 1 if host name was resolved to IP address at offset +33	This bit is set to 1 in case structure is valid, SD-card resource exists and its starting cluster and file size contains actual information
2	Bit set to 1 forces NETBLOAD command to perform HTTP HEAD request rather than GET	These two bits identify SD-card partition to be used, 0 for partition 0 (identified by SDC://) to 3 for partition 3 (identified by SDF://)
3	Bit set to 1 forces termination of data transfer for GET request method after HTTP header is received	
4	N/A, must be 0	N/A, must be 0
5		
6	If set identifies URI structure version 1, otherwise version 0	
7	Reserved	Reserved

URI type is identified by the parsing code (PARURI call) by the prefix. If URI string start with HTTP:// then structure type is (changed to) network structure, if with SDC:// then to SD-card structure type. Type is indicated by the bit 0 of byte in offset +0 of the structure. Please note that for SD-card type of structure host name is skipped, thus to identify path from the root you should use SDC://path/filename.

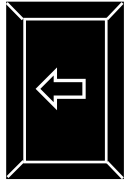
Host name field at offset +33 (of 32 bytes in size), when URI structure identifies SD-card resource, will contain directory entry corresponding to path + file name if bit 1 of byte in offset +0 is set. Exception is root directory – this field will contain all nulls except one field of byte offset +0bh will contain 010h ("subdirectory").

---

## 12.4. TCP/IP UNAPI implementation

Since September 2017 GR8NET firmware is having TCP/IP UNAPI version 1.0 implemented, *with several limitations*. Since May 2019 most of these limitations are removed, and almost all functionality is present in GR8NET TCPIP UNAPI implementation.

If you want to disable UNAPI implementation hooking onto the BIOS extension mechanism (EXTBIO) then press and hold arrow left key during GR8NET initialization, and it will print message that UNAPI was not initialized.



This chapter gives brief overview of the UNAPI implemented, and notes the deviations from the published standard. Two documents describing UNAPI and specifically TCP/IP UNAPI are:

- The [TCP/IP UNAPI specification](#) (version 1.0);
- [MSX-UNAPI](#): Unified procedure for API definition and discovery on MSX computers (version 1.1).

There're several programming guidelines for making reliable software with the implemented TCP/IP UNAPI:

1. Usage of UNAPI calls are not allowed within the interrupt service routines; there must be no concurrent calls to the UNAPI routines. If your application must be responsive to the events driven by interrupt, raise flag in the interrupt routine and then call UNAPI from the main code;
2. You can implement networking API routine calling in the following ways within the MSX-DOS environment:
  - a. By using far calls with RST 30h service routine. This routine will ensure that after API call is exited, all slot configuration is restored;
  - b. By enabling UNAPI slot in respective CPU bank with ENASLT (bank 1 – 4000-7FFF for GR8NET) and performing direct calls, but switching RAM page in bank 1 back before calling any MSX-DOS function. Not following this rule may result in malfunction on specific circumstances (e.g. when CTRL-STOP is pressed and exception is generated, see [this forum thread](#));
3. Data passed to the API calls must not be located in the CPU bank 1, you must arrange it in bank 0 (0000-3FFF) or banks 2 and 3 (8000-FFFF).

Exception:

- TCPIP\_TCP\_CLOSE and TCPIP\_TCP\_ABORT, they function the same way trying disconnecting from the remote device and then closing and freeing the socket, thus after any of them application will not be able to send or receive the data.

List of the API functions comprises the following entries:

#	Name	Description and conditions
0	UNAPI_GET_INFO	Provides API version supported (1.0)
1	TCPIP_GET_CAPAB	Provides capabilities supported bitmapped into Z80 registers. Three sockets are used for any of TCP, UDP or IP RAW modes, shared with BASIC network file I/O; datagram size is set to 1024 bytes
2	TCPIP_GET_IPINFO	Get IP addresses within the networking environment. Peer IP address and secondary DNS IP address are not available in the context of GR8NET, and return ERR_INV_PARAM
3	TCPIP_NET_STATE	Get state of the network. As W5100 does not have any mean to know the physical connection state of PHY, API will always report network state as Open
4	TCPIP_SEND_ECHO	This routine not only sends ICMP packet, it wait for reply within time identified by value set with NETVARUDTO command (default is 2 seconds), thus if remote interface will not reply please allow several seconds to complete this API call
5	TCPIP_RCV_ECHO	This routine will extract result of the TCPIP_SEND_ECHO and pass it to caller
6	TCPIP_DNS_Q	Performs querying of the DNS server, and waits for the response within the time defined by the NETVARUDTO command (default is 2 seconds). Please design your software the way internally caching resolved host names needed for your application to operate; note that respective capabilities bit "Checking network state requires sending a packet in loopback mode, or other expensive (time consuming) procedure" returned by the TCPIP_GET_CAPAB is set
7	TCPIP_DNS_S	This routine extracts results obtained by the TCPIP_DNS_Q, and will always return status that name was resolved using DNS query if name does not represent direct IP address
8	TCPIP_UDP_OPEN	Allocates one of the sockets for the UDP operation. Before termination, your application must explicitly close respective sockets otherwise system risks allocated sockets out of the available pool, affecting UNAPI as well as BASIC networking I/O operation
9	TCPIP_UDP_CLOSE	Closes the socket; this routine must be called before exiting to free allocated sockets for further usage by the system or other applications
10	TCPIP_UDP_STATE	This routine will return 1 datagram if there's one or more in the RX buffer, thus you must extract datagrams one by one with respective speed unless datagrams will get lost because of RX buffer overflow.
11	TCPIP_UDP_SEND	This routine sends UDP datagram.
12	TCPIP_UDP_RCV	This routine extracts single UDP datagram from the RX buffer if there's one.
13	TCPIP_TCP_OPEN	Opens socket for TCP communication, and tries to connect to remote device (for active connection) or listens for incoming connection (in case of passive connection).
14	TCPIP_TCP_CLOSE	Both calls invoke the same routine, which first disconnects from remote device, and then closes the socket. Thus after calling any of these functions application will not be able to send or receive any data
15	TCPIP_TCP_ABORT	
16	TCPIP_TCP_STATE	Returns state of the TCP connection, and information about remote device if needed
17	TCPIP_TCP_SEND	Sends data using TCP connection. Connection must be in established state, otherwise result is not guaranteed
18	TCPIP_TCP_RCV	Receives TCP data. After socket exiting established state, remaining data will still be available until application explicitly calls TCPIP_TCP_CLOSE or TCPIP_TCP_ABORT

#	Name	Description and conditions
19	TCPIP_TCP_FLUSH	Not implemented. TCP data is always automatically flushed to the network
20	TCPIP_RAW_OPEN	Opens socket in IP RAW mode. Ensure your application closes the socket when it is not needed any more to free it for other applications
21	TCPIP_RAW_CLOSE	Closes the socket and frees it to the pool of the sockets available through TCP/IP UNAPI and BASIC network I/O
22	TCPIP_RAW_STATE	Returns state information. If there will be one or more datagrams waiting in the RX buffer, will always return 1, thus application should obtain datagrams one by one
23	TCPIP_RAW_SEND	Sends RAW IP datagram
24	TCPIP_RAW_RCV	Gets one RAW IP datagram if there's one in the RX buffer
25	TCPIP_CONFIG_AUTOIP	Configures GR8NET in DHCP or in fixed IP address mode. Bit 1 of input register C has no meaning: if bit 0 is set then all IP information is provided by the DHCP, if bit 0 is reset then all IP addresses are in fixed IP address mode. In case adapter was unable to configure in DHCP mode, it will fall back to fixed IP configuration
26	TCPIP_CONFIG_IP	Manually configure an IP address
27	TCPIP_CONFIG_TTL	Get/set the value of TTL and ToS for outgoing datagrams: default are TTL=128, ToS=0. Calling this routine will change these settings for user sockets 0-2, thus BASIC network file I/O will be affected. System socket 3 (used for e.g. _NETBLOAD) is unaffected and will always get default values of TTL and ToS
28	TCPIP_CONFIG_PING	Sets respective flag in networking chip to respond or not to the ICMP ping requests on the network
29	TCPIP_WAIT	This routine does nothing, and always returns ERR_OK

## 12.5. Video file formats

Provision of the video file formats helps users and programmers creating their own tools for making the videos for MSX. Please refer to the chapter [Playing video from SD-card](#) for how to run the video playback.

SCREEN 2 videos are usually having file extension of .SC2 and can be of two formats:

- Format **version 0**, the older format (non-interlaced), is having fill frame color data following the full frame pattern data, thus the only way to update the VRAM will be to fully load pattern data into pattern VRAM location first, and then load full color data. While it is fastest way to proceed, this format is not suitable for displaying decent image on MSX1 and MSX1.5 machines having only one video page due to heavily visible artifacts between time of the update of pattern data and color data when pixels display correct pattern and incorrect color information for them. This format is also missing frame synchronization markers, causing player to display and sound a garbage at the very end of the video if SD-card is having high number of sectors per cluster;
- Format **version 1**, new format (interlaced), with 256 bytes of pattern data interlaced with 256 bytes of color data, thus allowing video player to quickly update the frame by the set of regions of 256\*8 pixels with minimal visible artifacts. This

format also features synchronization marker, allowing for proper detection of the end of video, and thus correct playback termination.

### **.SC2 version 0 video file format**

Name	Value, hex	Size	Purpose
ID	FC	1	File type identifier
MOD	02	1	Screen mode identifier, with bit 7 used for target vertical retrace rate
X	20	1	Raster X size in bytes
Y	18	1	Raster Y size in character places
AUI		16384	Initial 8-bit 22 kHz audio data for pre-buffering
FRP		6144	Frame pattern data, 256 * 24 bytes
FRC		6144	Frame color data, 256 * 24 bytes
PAL		32	Palette data in VDP palette register format
AUS		2	16-bit word for the size of following audio chunk
AUD		[AUS]	Audio data to be further buffered into PCM engine
FRP	Next frame, and frames continue till the end of the file (cluster on the storage media)		

### **.SC2 version 1 video file format**

Name	Value, hex	Size	Purpose
ID	FB	1	File type identifier
MOD	02	1	Screen mode identifier, with bit 7 used for target vertical retrace rate
X	20	1	Raster X size in bytes
Y	18	1	Raster Y size in character places
AUI		16384	Initial 8-bit 22 kHz audio data for pre-buffering
FSY	FB	1	Frame synchronization byte
FRP0		256	Frame pattern data @ position (0;0) of (256;8) in size
FRC0		256	Frame color data @ position (0;0) of (256;8) in size
FRP1		256	Frame pattern data @ position (0;8) of (256;8) in size
FRC1		256	Frame color data @ position (0;8) of (256;8) in size
...	...	...	...
FRP23		256	Frame pattern data @ position (0;184) of (256;8) in size
FRC23		256	Frame color data @ position (0;184) of (256;8) in size
PAL		32	Palette data in VDP palette register format
AUS		2	16-bit word for the size of following audio chunk
AUD		[AUS]	Audio data to be further buffered into PCM engine
FSY	Next frame, and frames continue till the value of FSY is 0, meaning correct video file ending		

To have a proper playback on the TMS99xx VDPs, video player employs adaptive frame synchronization technique, using PCM engine state to identify if audio buffer runs

out and when it is required to skip the specific frame display. Thus technique allows uninterruptible audio playback, however video may get slightly out of the synchronization with audio, up to several hundreds of microseconds.

Video file formats for SCREEN 8 and SCREEN 12 are the same, but due to different VRAM data interpretation by the VDP, the data itself will be different.

### **.SC8/.SCC file formats**

<b>Name</b>	<b>Value, hex</b>	<b>Size</b>	<b>Purpose</b>
ID	FC	1	File type identifier
MOD	08/0C	1	Screen mode identifier, with bit 7 used for target vertical retrace rate
X	88	1	Raster X size in pixels (e.g. 136, for SCREEN 12 must be divisible by 4)
Y	66	1	Raster Y size in pixels (e.g. 102, <b>must be even</b> )
AUI		16384	Initial 8-bit 22 kHz audio data for pre-buffering
FSY	FC	1	Frame synchronization byte
VRP	0	1	VRAM page identifier (A16...A14)
VAH	8E	1	VRAM port write identifier, VDP R#14
VR0	3C, 6D	2	Starting pointer within active VRAM page (e.g. 2D3C)
AU0		4	4 bytes of the audio data to buffer
L0D		X	Color data for the row 0
VRP	0	1	VRAM page identifier (A16...A14)
VAH	8E	1	VRAM port write identifier, VDP R#14
VR1	3C, 6E	2	Starting pointer within active VRAM page (e.g. 2E3C)
AU1		4	4 bytes of the audio data to buffer
L1D		X	Color data for the row 1
...	...	...	...
VRP	02	1	VRAM page identifier (A16...A14)
VAH	8E	1	VRAM port write identifier, VDP R#14
VR191	3C, 52	2	Starting pointer within active VRAM page (e.g. 923C)
AU191		4	4 bytes of the audio data to buffer
L191D		X	Color data for the row 191
PTL	1F	1	Pattern table location switching
PTP	82	1	Pattern table register location, R#2
AUS		2	16-bit word for the size of following audio chunk
AUD		[AUS]	Audio data to be further buffered into PCM engine
FSY	Next frame, and frames continue till the value of FSY is 0, meaning correct video file ending		

Notes on the .SC8/SCC format:

- VRP identify currently active page write is performed into. After write is complete, player waits for next interrupt, and switches visible page into location of PTL. On next frame second 64K bank is selected using VRP, and at the next interrupt

---

following completion of data transfer, second that 64K bank is made visible using PTL;

- As you can see some values are presented exactly the way they are sent to VDP (register values and identifiers). This format allows for faster processing, while slightly grows the video file;
- Actual VRAM address the write is performed to is comprised of VRP:VR, with VRP representing format of R#14 and VR the format of VRAM pointer (with bit 14 being set to signify following write to the VRAM);
- Handling of machines with additional wait states (having T97-A/B/C chip like Turbo-R or Panasonic FS-A1WX/FX) is performed by GR8NET identifying value of additional wait state during its startup initialization, and then video player using this information to cut out top and bottom 3 lines (for 1 additional wait state) or 6 lines (for 2 additional wait states) from the display image, thus saving time to keep up with the interrupt/frame rate. The same performance system information is displayed by the \_NETSYSINFO command.

## 12.6. Networking libraries for Fusion-C (SDCC-based)

You can download source code and object files from this location: <http://www.gr8bit.ru/software/programming/>. Developed libraries, while state that they are developed for GR8NET, support any MSX networking adapter implementing TCP/IP UNAPI.

The software package contains:

- Assembly source file (.s) with TCP/IP libraries;
- C header source file (.h) for Fusion-C;
- Test-bench C source file;
- Console I/O routine source files for test-bench (header .h and assembly .s);
- Test-bench application file (.com).

Instructions on how to compile the test-bench project are located in the test-bench source file.

Test-bench application tests almost all capabilities to the best extent it can. The only command line input parameter is IP address of the host on the network which can answer PING requests (used for ICMP/IPRAW testing), accepts HTTP requests through port 80 (used for TCP testing), can process DNS requests (used for UDP testing) – ideally local gateway device which used to have all these services up and running.



Below are several screenshots of the operation of the test-bench application.

```
Ok
system
NEXTOR.SYS version 2.01
Copyright (2014) Konamiman

A>topipstat 192.168.1.1
topip_enuerate = 0x0001
Target IP address: 0xC0.A8.01.01

topip_impl_getinfo = 0x0000 (ERR_OK Operation completed successfully)
GRONET LAN adapter build 20190530
API spec: 0x0100
API impl: 0x0009

topip_get_capab_flags_llproto = 0x0000 (ERR_OK Operation completed successfully)

Link level proto: 0x03
Feat. flags: 0x001C
Capab. flags: 0x7C3D

Press any key for the next test■
```

Capabilities

```
topip_get_capab_flags_llproto = 0x0000 (ERR_OK Operation completed successfully)

Link level proto: 0x03
Feat. flags: 0x001C
Capab. flags: 0x7C3D

Press any key for the next test

topip_get_capab_connections = 0x0000 (ERR_OK Operation completed successfully)
max_tcp_conn: 0x03
max_udp_conn: 0x03
max_raw_conn: 0x03
free_tcp_conn: 0x03
free_udp_conn: 0x03
free_raw_conn: 0x03

topip_get_capab_dtg_sizes = 0x0000 (ERR_OK Operation completed successfully)
max_outgoing_dtg_size: 0x0400
max_incoming_dtg_size: 0x0400

topip_net_state = 0x0000 (ERR_OK Operation completed successfully)
net_state: 0x02

Press any key for the next test■
```

Capabilities

```
Test of the TTL/ToS configuration routines

topip_config_ttltos_get = 0x0000 (ERR_OK Operation completed successfully)
TTL=80, ToS=00

topip_config_ttltos_set = 0x0000 (ERR_OK Operation completed successfully)

topip_config_ttltos_get = 0x0000 (ERR_OK Operation completed successfully)
New TTL=7E, ToS=FF

topip_config_ttltos_set = 0x0000 (ERR_OK Operation completed successfully)

TTL/ToS test successful

Press any key for the next test■
```

TTL/ToS

```
peer_ip: 0x00.00.00.00
subnet_mask: 0xFF.FF.FF.00
gateway_ip: 0xC0.A8.01.01
dns_ip_pri: 0xC0.A8.01.01
dns_ip_sec: 0x00.00.00.00

Resetting to original settings..
topip_config_autoip_set = 0x0000 (ERR_OK Operation completed successfully)

topip_config_autoip_get = 0x0000 (ERR_OK Operation completed successfully)
autoip: ip=yes, dns=yes

topip_config_autoip_set = 0x0000 (ERR_OK Operation completed successfully)

topip_get_ipinfo = 0x0000 (ERR_OK Operation completed successfully)
local_ip: 0xC0.A8.01.2D
peer_ip: 0x00.00.00.00
subnet_mask: 0xFF.FF.FF.00
gateway_ip: 0xC0.A8.01.01
dns_ip_pri: 0xC0.A8.01.01
dns_ip_sec: 0x00.00.00.00

Press any key for the next test■
```

AUTOIP/CONFIG\_IP

```
Test of the PING and DNS routines

topip_send_echo = 0x0000 (ERR_OK Operation completed successfully)
Waiting for incoming ICMP data...
topip_rcv_echo = 0x0000 (ERR_OK Operation completed successfully)
response from: 0xC0.A8.01.01
ttl: 0x00
icmp_id: 0x0001
seq_number: 0x0002
data_length: 0x0040

topip_dns_q = 0x0000 (ERR_OK Operation completed successfully)
topip_dns_s -> State/substate/IP: 0x02/00/C3.D0.01.7F
Press any key for the next test■
```

PING and DNS

```
Test of the UDP communication routines

topip_udp_open = 0x0000 (ERR_OK Operation completed successfully)

topip_udp_state = 0x0000 (ERR_OK Operation completed successfully)
Dtg/size/port: 0x0000/0000/0035
topip_udp_send = 0x0000 (ERR_OK Operation completed successfully)
Waiting for incoming UDP data...
topip_udp_state = 0x0000 (ERR_OK Operation completed successfully)
Dtg/size/port: 0x0001/002F/0035
topip_udp_rcv = 0x0000 (ERR_OK Operation completed successfully)
response from: C0.A8.01.01
port: 0035
dtg size: 002F/002F
00 01 81 80 00 01 00 01 00 00 00 00 03 77 77 77 06 67 72 ..www.gp
38 62 69 74 02 72 75 00 00 01 00 01 C0 0C 00 01 00 01 00 8bit.ru.....w.....
00 0E 10 00 04 C3 D0 01 7F .....un..
topip_udp_close = 0x0000 (ERR_OK Operation completed successfully)

Press any key for the next test■
```

UDP

```
IP:ports: 0xC0.A8.01.01/0050/C008
State: 0x07/00
Sizes: 0x0040/0000/0800

topip_top_rcv = 0x0000 (ERR_OK Operation completed successfully)
40 54 54 50 2F 31 2E 30 20 35 30 30 20 6E 6F 20 73 75 63 HTTP/1.0 500 no suc
60 20 63 65 61 64 65 72 3A 20 48 6F 73 74 0D 0A 53 65 72 h header: Host..Ser
70 65 72 3A 20 41 87 20 5B 3A 37 5D 0D 0A 43 6F 6E 74 65 ver: Ag [47]..Conte
6F 74 2D 74 79 70 65 3A 20 74 65 78 74 2F 68 74 6D 6C 0D nt-type: text/html;
0A 0D 0A 3C 68 74 6D 6C 3E 0A 3C 68 65 61 64 3E 0A 3C 2F ...<html>.<head>.</
68 65 61 64 3E 0A 3C 62 6F 64 79 3E 0A 3C 68 31 3E 35 30 head>.<body>.<h1>50
30 3A 20 6E 6F 20 73 75 63 68 20 6F 65 61 64 65 72 3A 20 0: no such header-
40 6F 73 74 3C 3E 68 31 3E 0A 3C 2F 62 6F 64 79 3E 0A 3C Host</h1>.</body>.<
2F 68 74 6D 6C 3E 0D 00 /html>..
topip_top_state = 0x0000 (ERR_OK Operation completed successfully)
IP:ports: 0xC0.A8.01.01/0050/C008
State: 0x07/00
Sizes: 0x0000/0000/0800

TCP session finished
topip_top_close = 0x0000 (ERR_OK Operation completed successfully)

Press any key for the next test■
```

TCP

```
Test of the IPRAW communication routines

topip_ipraw_open = 0x0000 (ERR_OK Operation completed successfully)

topip_ipraw_state = 0x0000 (ERR_OK Operation completed successfully)
Proto: 0x01
Sizes: 0x0000/0000

topip_ipraw_send = 0x0000 (ERR_OK Operation completed successfully)
Waiting for incoming IPRAW/ICMP data...response from: C0.A8.01.01
data size: 0x028
00 00 55 00 01 00 06 61 62 63 64 65 66 67 68 69 6A 6B ..UU....abodefghijk
6C 6D 6E 6F 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 1mnopqrstuvwxyzabodefg
68 69 hi
Press any key for the next test■
```

IPRAW

```

Test of the PING parameter configuration routines
tcpip_config_ping read = 0x0000 (ERR_OK Operation completed successfully)
tcpip_config_ping enabled = 0x0000 (ERR_OK Operation completed successfully)
Perform ping of this device 0x0.0.0.20. Press any key to continue
tcpip_config_ping disabled = 0x0000 (ERR_OK Operation completed successfully)
Perform ping of this device 0x0.0.0.20. Press any key to continue
tcpip_config_ping restored = 0x0000 (ERR_OK Operation completed successfully)

Operations 0x0032, errors 0x0000
A>

```

## PING response

Header file is enclosed below.

```

/*
GR8NET internetworking adapter
MSX TCP-IP UNAPI implementation version 1.0 for Fusion-C / header file
Initial release by Eugeny Brychkov 2019-05-31
License: keeps the same as TCP/IP UNAPI v.1.0 by Konamiman, plus requirement
to cite original author when this work or its derivatives are used

The workflow is the following:
1. Perform enumeration of the implementations using tcpip_enumerate(), it will reset active_implementation to 0. The routine
will check for maximum 4 implementations (e.g. 4 GR8NETs installed in the system), and return this value;
2. You set appropriate implementation number in the active_implementation, and work with this implementation getting its
capabilities, network properties etc.
3. If you need to work with another implementation, change active_implementation, and then all calls will be performed
to the respective implementation. Note that structures must be different for each connection-implementation as they
are socket-specific. Any UNAPI-related routine will return integer error code
*/

/* TCP/IP UNAPI management */
extern int tcpip_enumerate(void); /* performs checking for installed implementation and gets calling information
for 4 of them, returns number of implementations (max 4). Resets
active_implementation to 0 */
extern int active_implementation; /* currently active implementation: 0 to 3, being reset to 0 by tcpip_enumerate() */
extern int implementation_count; /* number of available implementations. Populated by tcpip_enumerate(),
initially set as 0 */
extern int ram_helper_call_address; /* RAM helper jump table address in CPU bank 3, populated by tcpip_enumerate() */
extern int tcpip_impl_getinfo(char** impl_string, /* UNAPI_GET_INFO: Obtain the implementation name and version. The implementation */
int* api_spec_version, /* is identified by the active_implementation (0..3). If active_implementation number */
int* api_impl_version); /* is wrong (not within 0..number of implementations-1) then ERR_INV_IMPL is returned */

/* Error codes, integer values */
#define ERR_OK 0 /* Operation completed successfully */
#define ERR_NOT_IMPL 1 /* Capability not implemented */
#define ERR_NO_NETWORK 2 /* No network connection available */
#define ERR_NO_DATA 3 /* No incoming data available */
#define ERR_INV_PARAM 4 /* Invalid input parameter */
#define ERR_QUERY_EXISTS 5 /* Another query is already in progress */
#define ERR_INV_IP 6 /* Invalid IP address */
#define ERR_NO_DNS 7 /* No DNS servers are configured */
#define ERR_DNS 8 /* Error returned by DNS server */
#define ERR_NO_FREE_CONN 9 /* No free connections available */
#define ERR_CONN_EXISTS 10 /* Connection already exists */
#define ERR_NO_CONN 11 /* Connection does not exist */
#define ERR_CONN_STATE 12 /* Invalid connection state */
#define ERR_BUFFER 13 /* Insufficient output buffer space */
#define ERR_LARGE_DGRAM 14 /* Datagram is too large */
#define ERR_INV_OPER 15 /* Invalid operation */
#define ERR_INV_IMPL 255 /* Invalid implementation set by the active_implementation variable */

/* Network media states */
#define MEDIA_STATE_CLOSED 0
#define MEDIA_STATE_OPENING 1
#define MEDIA_STATE_OPEN 2
#define MEDIA_STATE_CLOSING 3
#define MEDIA_STATE_UNKNOWN 255

/* TCP communication states */
#define TCP_STATE_UNKNOWN 0
#define TCP_STATE_LISTEN 1
#define TCP_STATE_SYN_SENT 2
#define TCP_STATE_SYN_RECEIVED 3
#define TCP_STATE_ESTABLISHED 4
#define TCP_STATE_FIN_WAIT_1 5
#define TCP_STATE_FIN_WAIT_2 6
#define TCP_STATE_CLOSE_WAIT 7
#define TCP_STATE_CLOSING 8
#define TCP_STATE_LAST_ACK 9
#define TCP_STATE_TIME_WAIT 10

/* TCP communication close reasons */
#define TCP_CLOSER_UNKNOWN 0
#define TCP_CLOSER_NO_CONN 1 /* This connection has never been used since the implementation was initialized */
#define TCP_CLOSER_CLOSED 2 /* The TCP_IP_TCP_CLOSE method was called */
#define TCP_CLOSER_ABORTED 3 /* The TCP_IP_TCP_ABORT method was called */
#define TCP_CLOSER_RST 4 /* A RST segment was received (the connection was refused or aborted by the remote host) */
#define TCP_CLOSER_US_TOUT 5 /* The user timeout expired */
#define TCP_CLOSER_CE_TOUT 6 /* The connection establishment timeout expired */
#define TCP_CLOSER_LOST 7 /* Network connection was lost while the TCP connection was open */
#define TCP_CLOSER_UNREACH 8 /* ICMP "Destination unreachable" message received */

/* Connection types */
#define CONNTYPE_TRANSIENT 0

```

```

#define CONNTYPE_RESIDENT 1

/* Operation definitions for tcpip_config_ping */
#define CONFIG_PING_GET 0
#define CONFIG_PING_SET 1

typedef struct {
    char proto_used; /* Link level protocol used */
    int feat_flags; /* Features flags */
    int capab_flags; /* Capabilities flags */
} tcpip_unapi_capab_flags_llproto;

typedef struct {
    char max_tcp_conn; /* Maximum simultaneous TCP connections supported */
    char max_udp_conn; /* Maximum simultaneous UDP connections supported */
    char max_raw_conn; /* Maximum simultaneous raw IP connections supported */
    char free_tcp_conn; /* Free TCP connections currently available */
    char free_udp_conn; /* Free UDP connections currently available */
    char free_raw_conn; /* Free raw IP connections currently available */
} tcpip_unapi_capab_connections;

typedef struct {
    int max_outgoing_dtg_size; /* Maximum outgoing datagram size supported */
    int max_incoming_dtg_size; /* Maximum incoming datagram size supported */
} tcpip_unapi_capab_dtg_sizes;

typedef struct {
    char local_ip[4]; /* Local IP address */
    char peer_ip[4]; /* Peer IP address */
    char subnet_mask[4]; /* Subnet mask */
    char gateway_ip[4]; /* Default gateway */
    char dns_ip_pri[4]; /* Primary DNS server IP address */
    char dns_ip_sec[4]; /* Secondary DNS server IP address */
} tcpip_unapi_ip_info;

typedef struct {
    char dest_ip[4]; /* +0 (4): IP address of the destination machine */
    char ttl; /* +4 (1): TTL for the datagram */
    int icmp_id; /* +5 (2): ICMP identifier */
    int seq_number; /* +7 (2): ICMP sequence number */
    int data_length; /* +9 (2): Data length, 0 to maximum datagram size - 28 */
} tcpip_unapi_echo;

typedef struct {
    char flags;
    char state; /* note that state (reg B) and substate (reg C) have different meaning for tcpip_dns_q */
    char substate; /* and tcpip_dns_s. Please refer to the UNAPI description for more information */
    char host_ip[4]; /* Resolved IP address (only if no error occurred and state of 1 or 2 is returned) */
} tcpip_unapi_dns_q;

typedef struct {
    char num_of_pend_dtg; /* Number of pending incoming datagrams */
    int size_of_oldest_dtg; /* Size of oldest pending incoming datagram (data part only) */
    int port_number; /* Local port number */
} tcpip_unapi_udp_state;

typedef struct {
    char dest_ip[4]; /* +0 (4): Destination IP address */
    int dest_port; /* +4 (2): Destination port */
    int data_length; /* +6 (2): Data length to send or actual data length when receiving */
} tcpip_unapi_udp_dtg_parms;

typedef struct {
    char dest_ip[4]; /* +0 (4): Remote IP address (0.0.0.0 for unspecified remote socket) */
    int dest_port; /* +4 (2): Remote port (ignored if unspecified remote socket) */
    int local_port; /* +6 (2): Local port, 0FFFFh for a random value */
    int user_timeout; /* +8 (2): Suggestion for user timeout value */
    char flags; /* +10 (1): Flags */
    char conn_state; /* +11 (1): Connection state updated by TCPIP_TCP_STATE */
    char close_reason; /* +12 (1): Close reason (only if ERR_NO_CONN is returned) by TCPIP_TCP_STATE */

    int incoming_bytes; /* +13 (2): Number of total available incoming bytes updated by TCPIP_TCP_STATE and TCPIP_TCP_RCV */
    int urgent_incoming_bytes; /* +15 (2): Number of urgent available incoming bytes updated by TCPIP_TCP_STATE and TCPIP_TCP_RCV */
    int send_free_bytes; /* +17 (2): Available free space in the output buffer (0FFFFh = infinite) updated by TCPIP_TCP_STATE */
} tcpip_unapi_tcp_conn_parms;

typedef struct {
    char ip[4]; /* +0 (4): Destination IP address */
    int data_length; /* +4 (2): Data length */
} tcpip_unapi_ipraw_parms;

typedef struct {
    char proto_code; /* +0 (1): Associated protocol code */
    int num_of_pend_dtg; /* +2 (2): Number of pending incoming datagrams */
    int size_of_oldest_dtg; /* +4 (2): Size of the oldest pending incoming datagram */
} tcpip_unapi_ipraw_state;

/***** Network capabilities and state routines *****/

/* TCPIP_GET_CAPAB: Get information about the TCP/IP capabilities/features/link level protocol,
connection pool and datagram sizes */
extern int tcpip_get_capab_flags_llproto(tcpip_unapi_capab_flags_llproto* capab_flags_llproto);
extern int tcpip_get_capab_connections(tcpip_unapi_capab_connections* capab_connections);
extern int tcpip_get_capab_dtg_sizes(tcpip_unapi_capab_dtg_sizes* capab_dtg_sizes);

/* TCPIP_GET_IPINFO: Get IP address. Always returns ERR_OK, if was unable to acquire
specific IP address respective field is left as 0.0.0.0 */
extern int tcpip_get_ipinfo(tcpip_unapi_ip_info* ip_info);

/* TCPIP_NET_STATE: Get network state */
extern int tcpip_net_state(int* net_state);

/***** ICMP routines *****/

/* TCPIP_SEND_ECHO: Send ICMP echo message (PING). All values of the structure
must be populated before performing the call */
extern int tcpip_send_echo(tcpip_unapi_echo* echo_param_block);

/* TCPIP_RCV_ECHO: Retrieve ICMP echo response message. The call populates
the structure at the address passed */
extern int tcpip_rcv_echo(tcpip_unapi_echo* echo_param_block);

```

```

/***** Host name resolution routines *****/

/* TCPIP_DNS_Q: Start a host name resolution query. Flags field in the structure
must be defined before performing the call:
- bit 0: Only abort the query currently in progress, if there is any (other flags and registers are then ignored)
- bit 1: Assume that the passed name is an IP address, and return an error if this is not true
- bit 2: If there is a query in progress already, do NOT abort it and return an error instead */
extern int tcPIP_dns_q(char* hostname, tcPIP_unapi_dns_q* dns_q);

/* TCPIP_DNS_S: Obtains the host name resolution process state and result. Flags field
in the structure must be defined before performing the call:
- bit 0: Clear any existing result or error condition after the execution (except if there is a query in progress) */
extern int tcPIP_dns_s(tcPIP_unapi_dns_q* dns_q);

/***** UDP protocol related routines *****/

/* TCPIP_UDP_OPEN: Open a UDP connection */
extern int tcPIP_udp_open(int port_number, int conn_lifetime, int* conn_number);

/* TCPIP_UDP_CLOSE: Close a UDP connection */
extern int tcPIP_udp_close(int conn_number);

/* TCPIP_UDP_STATE: Get the state of a UDP connection. Populates local port number,
number of pending datagrams and size of oldest pending datagram fields */
extern int tcPIP_udp_state(int conn_number, tcPIP_unapi_udp_state* udp_state);

/* TCPIP_UDP_SEND: Send an UDP datagram. Destination IP address, destination
port number and data size fields must be populated */
extern int tcPIP_udp_send(int conn_number, tcPIP_unapi_udp_dtg_parms* dtg_parms,
char* data);

/* TCPIP_UDP_RCV: Retrieve an incoming UDP datagram. Populates IP address,
source port and actual received data length fields */
extern int tcPIP_udp_rcv(int conn_number, char* buffer, int dtg_maxsize,
tcPIP_unapi_udp_dtg_parms* dtg_parms);

/***** TCP protocol related routines *****/

/* TCPIP_TCP_OPEN: Open a TCP connection. Remote IP address, port numbers,
timeout value and flags fields must be populated before performing the call */
extern int tcPIP_tcp_open(tcPIP_unapi_tcp_conn_parms* tcp_conn_parms, int* conn_number);

/* TCPIP_TCP_CLOSE: Close a TCP connection */
extern int tcPIP_tcp_close(int conn_number);

/* TCPIP_TCP_ABORT: Abort a TCP connection */
extern int tcPIP_tcp_abort(int conn_number);

/* TCPIP_TCP_STATE: Get the state of a TCP connection. Puts information into connection state,
close reason, both incoming and free bytes fields. It will also put IP address and port
numbers in */
extern int tcPIP_tcp_state(int conn_number, tcPIP_unapi_tcp_conn_parms* tcp_conn_parms);

/* TCPIP_TCP_SEND: Send data a TCP connection. Flags field must be defined
before performing the call (bit 0: Send the data PUSHed; bit 1: The data is urgent) */
extern int tcPIP_tcp_send(int conn_number, char* data, int data_length, int flags);

/* TCPIP_TCP_RCV: Receive data from a TCP connection. Updates both received byte numbers
within the structure */
extern int tcPIP_tcp_rcv(int conn_number, char* buffer, int maxsize,
tcPIP_unapi_tcp_conn_parms* tcp_conn_parms);

/* TCPIP_TCP_FLUSH: Flush the output buffer of a TCP */
extern int tcPIP_tcp_flush(int conn_number);

/***** Raw IP connections related routines *****/

/* TCPIP_RAW_OPEN: Open a raw IP connection. IPRAW protocol list is here:
http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml */
extern int tcPIP_ipraw_open(int protocol_number, int conn_lifetime, int* conn_number);

/* TCPIP_RAW_CLOSE: Close a raw IP connection */
extern int tcPIP_ipraw_close(int conn_number);

/* TCPIP_RAW_STATE: Get the state of a raw IP connection */
extern int tcPIP_ipraw_state(int conn_number, tcPIP_unapi_ipraw_state* ipraw_state);

/* TCPIP_RAW_SEND: Send a raw IP datagram. Structure passed must contain
destination IP address and data size set before calling */
extern int tcPIP_ipraw_send(int conn_number, char* data,
tcPIP_unapi_ipraw_parms* ipraw_parms);

/* TCPIP_RAW_RCV: Retrieve an incoming raw IP datagram */
extern int tcPIP_ipraw_rcv(int conn_number, char* buffer, int maxsize,
tcPIP_unapi_ipraw_parms* ipraw_parms);

/***** Configuration related routines *****/

/* TCPIP_CONFIG_AUTOIP: Enable or disable the automatic IP addresses retrieval */
extern int tcPIP_config_autoip_get(char* ip_mode);
extern int tcPIP_config_autoip_set(char* ip_mode);

/* TCPIP_CONFIG_IP: Manually configure an IP address */
extern int tcPIP_config_ip(tcPIP_unapi_ip_info* ip_info);

/* TCPIP_CONFIG_TTL: Get/set the value of TTL and TOS for outgoing datagrams */
extern int tcPIP_config_ttltos_get(unsigned* ttltos_get);
extern int tcPIP_config_ttltos_set(unsigned ttltos_set);

/* TCPIP_CONFIG_PING: Get/set the automatic PING reply flag */
extern int tcPIP_config_ping_get(char* flag_value_get);
extern int tcPIP_config_ping_set(char flag_value_set);

/* End of header file */

```



## 13. Applications

This chapter lists several applications you can use with GR8NET, and describes how you should use them, and their limitations. In case you are willing to write application for GR8NET, please contact us at [info@gr8bit.ru](mailto:info@gr8bit.ru).

### 13.1. MSX webserver

Type: MSX-DOS executable

Location: <http://www.gr8bit.ru/software/binaries/ws.rar>, password "webserver"

Related video: [https://youtu.be/0tQw\\_Xuq900](https://youtu.be/0tQw_Xuq900)

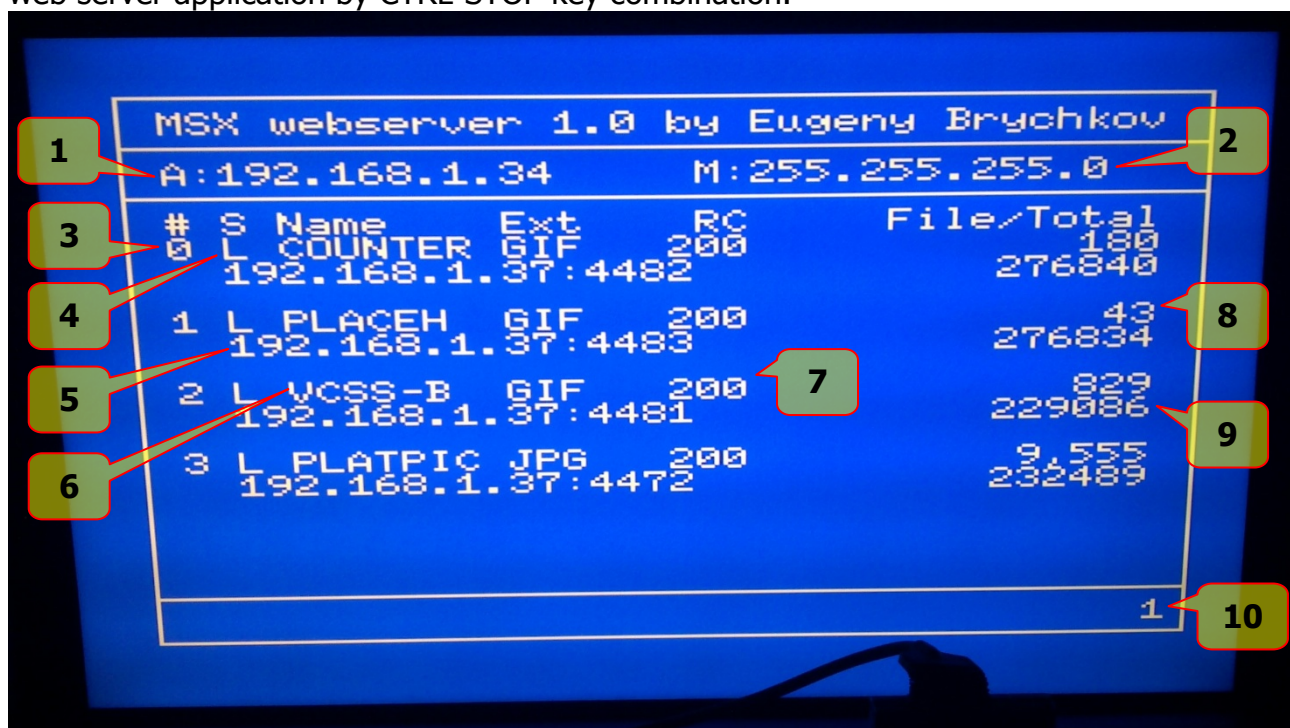
Root of the web server should be located on the SD-card, in any subdirectory. To start web server, you type

**ws [GR8NET adapter number] /[web server root]**

for example

ws 0 /www

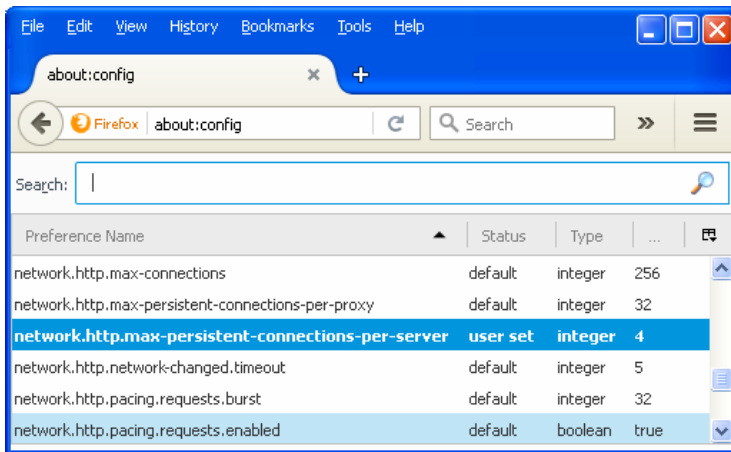
In case resource you identify as webserver root is not a valid subdirectory, or it does not exist, application will exit with the error. At any moment you can terminate execution of the web server application by CTRL-STOP key combination.



Webserver application displays the following information when operates:

1. Webserver IP address;
2. Webserver network mask;
3. Socket number, in total 4 sockets per GR8NET adapter;

4. State of the socket: L (listening), R (receiving request), C (closing), S (sending response header) and D (sending data);
5. Requestor IP address and port;
6. File name in 8:3 format;
7. HTTP response code;
8. Size of the file;
9. Total *data* bytes sent through the socket;
10. Visualization mode: 0 or blank (display everything), 1 (do not display total data bytes), 2 (do not display total data bytes and file sizes).



When creating web server pages for GR8NET, please ensure files are named in standard 8:3 DOS format otherwise web server will not find files and return *500 Bad request* error.

It is advisable to use user agent which allows customization of the number of concurrent connections to the web server. Mozilla Firefox is a good choice; as GR8NET adapter has only 4 sockets, you need to ensure

that there will be no more than 4 concurrent requests made to the adapter otherwise *Connection refused* error will be returned to the requestor.

In Firefox, open new tab and go to **about:config** address. Accept warning about modification of the default configuration, find option **network.http.max-persistent-connections-per-server** and decrease it to 4 or even to 2, this will minimize issues with refusal of the GR8NET to serve HTTP requests.



Changing visualization level from 0 (default) to 1 or 2 will speed up communication because webserver will not spend time displaying numbers onto the screen. With visualization level 0 local download speed is about 25 Kb/s, with level 1 is about 100 Kb/s. To change visualization level press and release F5 key, change takes effect on release of the key.

## 13.2. FTP client

Type: BASIC program

Location: <http://www.gr8bit.ru/software/basic/ftp.asc>

Load application from the local storage device, or use HTTP: network device to get it from the abovementioned URI location.

```
run
FTP server? ftp.gr8bit.ru
Connecting to FTP server...success
220 nichosting.ru ftp server
user gr8bit_
331 Password required for gr8bit_ftp
pass
230 User gr8bit_ logged in
cwd /
250 CWD command successful
pasv
227 Entering Passive Mode (178,210,84,141,201,45).
* Streaming from B2.D2.54.8D/C92D *
retr nishisan.asc
150 Opening BINARY mode data connection for nishisan.asc (210 bytes)
* Command: [RETR] *
* Performing data download [nishisan.asc] *
* Target is [nishisan.asc] *
* Download finished *
226 Transfer complete

color      auto      goto      list
```

Sample download dialog

FTP client is able to download to and upload files from the local storage device. Please examine sample download dialog above carefully and note that you should use raw FTP commands, you can find full list [here](#).

FTP client works with server putting it into passive mode by PASV command. In passive mode FTP server waits for active connection by the client to exchange requested information through the data connection.

For uploading files onto the FTP server, use STOR (store) command instead of RETR (retrieve); for listing directory contents onto the screen use LIST command (also preceded by PASV command).

---

### 13.3. Video player

Status: discontinued, please use NETPLAYVID built-in command

Type: MSX-DOS executable program

Location: <http://www.gr8bit.ru/software/video/>

Video player application consists of the several parts: video maker and video player.

Video player is GR8VIDEO.COM application which should be executed under MSX-DOS. It requires several input parameters, and command line should be as follows:

GR8VIDEO X[!] /*absolute\_path*

where X is adapter number (0-3) to be used for video playback;

! is an optional character which causes player not to perform waiting input when error condition is detected;

/*absolute\_path* is an absolute path to the video file on the SD-card, starting from the root. Note that leading slash character is mandatory.

The following modes are supported:

Mode	VDPs	Scan	Video configuration	Audio configuration
SCREEN 2	T, 3, 5	60 Hz	256 x 192 pixels	22050 Hz, 8-bit, mono
SCREEN 8	3, 5		X*Y to be not greater than 13,872, but not less than 11,098 (for example 136*102)	
SCREEN 12	5			

T: TMS99xx, 3: V9938, 5: V9958

If there's a condition, which may cause player to display video improperly, application will display warning message and wait for key press in case ! character is not specified in the command line. The following issues may occur:

- No enough video RAM: modes 8/12 will refuse to be played on 16K VRAM, and mode 2 will play using single video page with 16K VRAM, causing visible artifacts on the screen;
- Unsupported mode: videos created for specific video mode will not play on VDPs which do not support this mode.

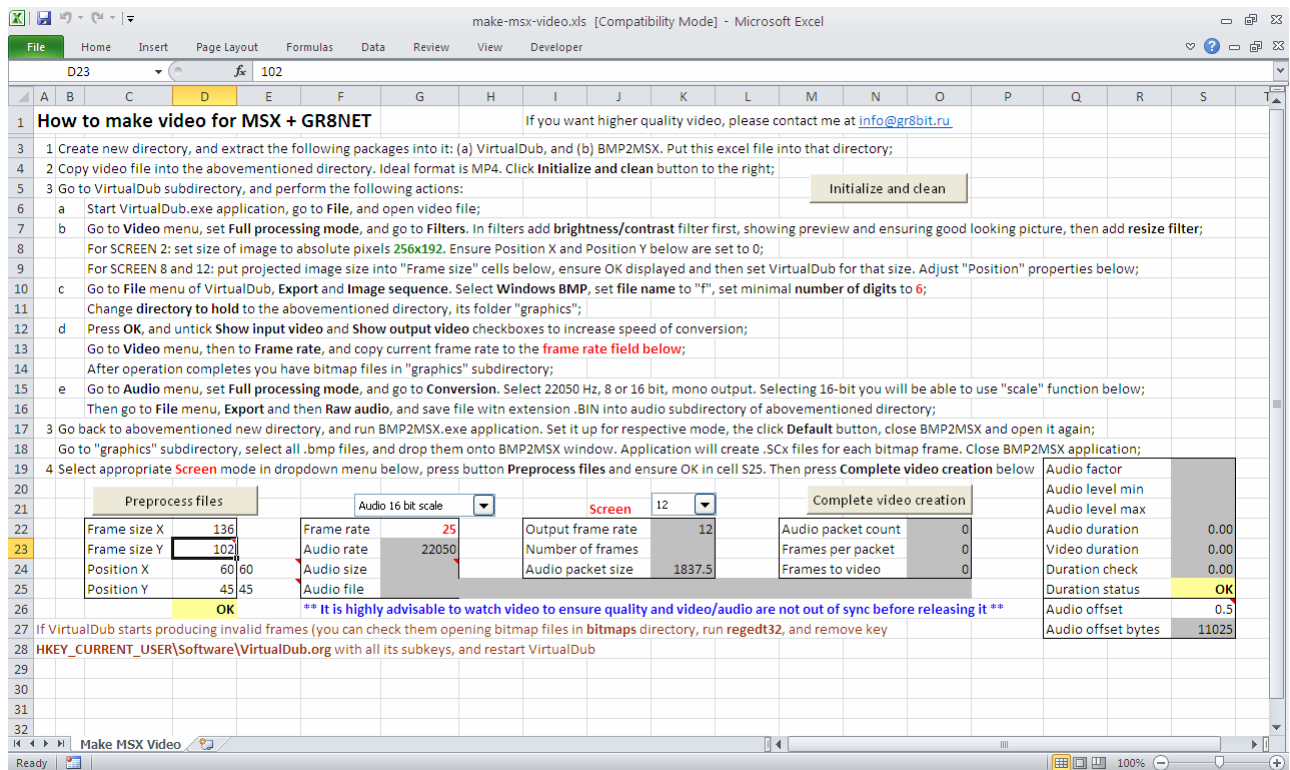


### 13.3.1. Making videos for MSX

In general making video is relatively simple, but you should strictly follow the process and fulfill very specific conditions.

Creation is performed by using Excel 97/2003 macro-enabled spreadsheet. The following steps are required:

- Obtain video file, the best it should be the MP4 format;
- Use VirtualDub tool to slice video file onto the frames of required size: 256\*192 for SCREEN2 mode, or number of pixels (X\*Y) between 13872 and 11098 for SCREEN 8 and 12 modes.
- Use VirtualDub to convert video's audio into 22050 Hz, 8- or 16-bit mono;
- Use BMP2MSX to convert bitmap frames created by VirtualDub into MSX screen representation files;
- Run script in Excel spreadsheet to complete video creation.



You will see detailed explanation on video creation when you open the spreadsheet. All required resources – VirtualDub, BMP2MSX, GR8VIDEO player and MSX video creation spreadsheet are located at <http://www.gr8bit.ru/software/video/>.

### 13.3.2. Converting .SC2 file from version 0 to version 1 format

As explained in the chapter [Playing video file from SD-card](#), there's new format for .SC2 file called version 1, and only files of this new format can be played on the MSX1

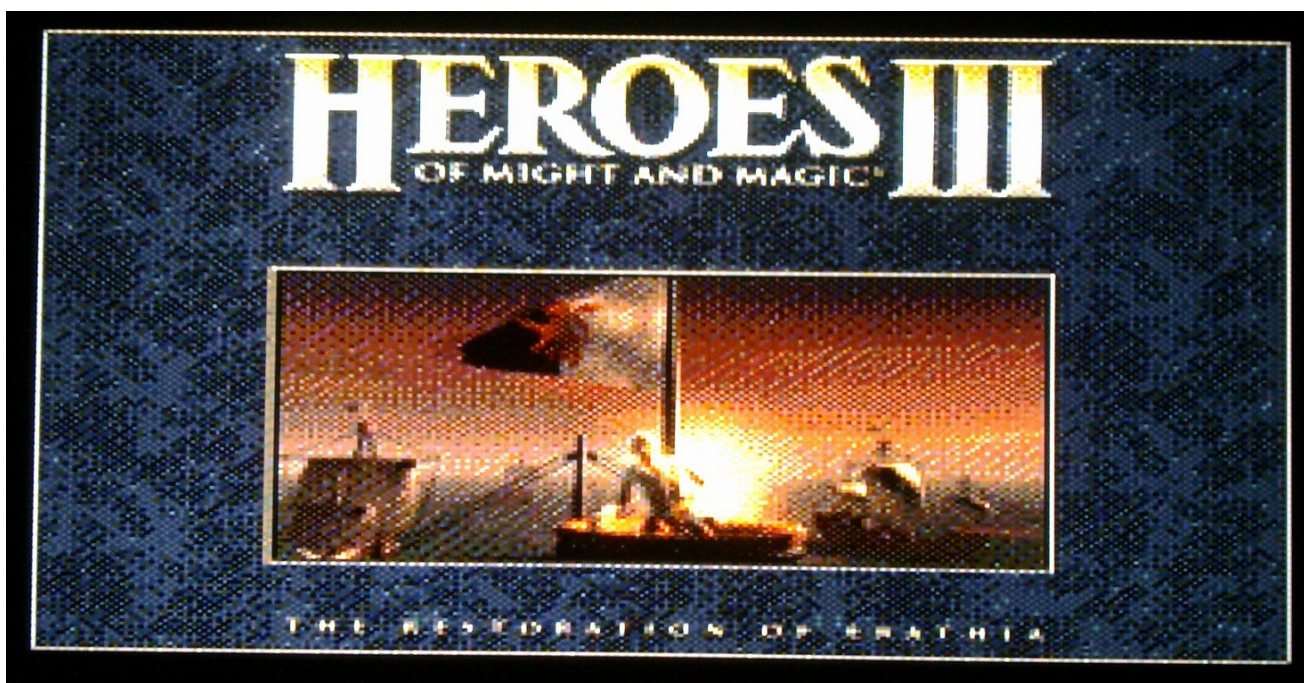
and MSX1.5 machines. The format is only about the layout of data in the file, and it is extremely easy to convert version 0 file to version 1 file. To convert files, please use **convert-sc2-video.xls** Microsoft Excel file available in the location <http://www.gr8bit.ru/software/video/>. Follow instructions in the file for successful completion of the task.

### 13.4. Heroes of Might and Magic III demo

Please watch the video about the demo here: <https://youtu.be/neIQXmLb6bw>.

This short demo with very limited play capabilities was designed to showcase the capabilities of the GR8NET in terms of video, audio, graphics, BASIC statement support, and confirm the easiness of programming the advanced multimedia applications using GR8NET. The location of the demo is <http://www.gr8bit.ru/software/video/>, download the archive, and extract its contents onto SD-card into directory /hmm3/. Go to \_NETBROWSE, SD-card, navigate to the directory, and start hmm3.asc file by pressing Enter key on it. Below are the several screenshots. Application supports keyboard as well as MSX mouse.

The demo will only run on MSX2 machines and above, as it uses SCREEN 8 video mode.

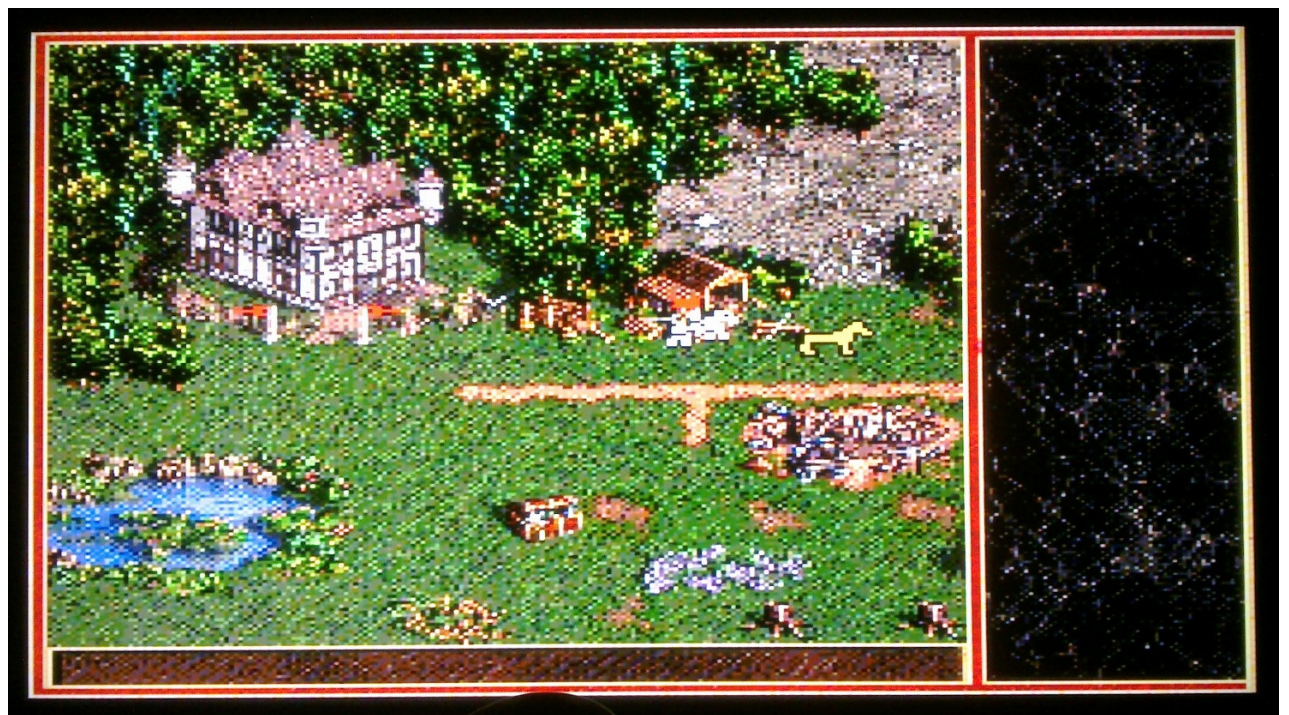


Intro video





Main menu screen



Game screen (demo version)

---

## 13.4. Card game "DURAK"

Please watch the video about the demo here: <https://youtu.be/R5fJk83PvLc>.

Play popular Soviet game called DURAK ("The Fool") with multimedia content provided by the GR8NET. The code itself is a BASIC program, and it can be found here: <http://www.gr8bit.ru/software/basic/durak/>, file name is durak.asc. Go to the server with \_NETBROWSE, locate this file and press Enter key on it; program will automatically download all required data from the internet.

The game is single player; there were plans to make multiplayer game through the network, but development took more time than expected and was interrupted by other competing priorities.



Shot of the Youtube video about the game



---

## 13.5. GR8cloud server

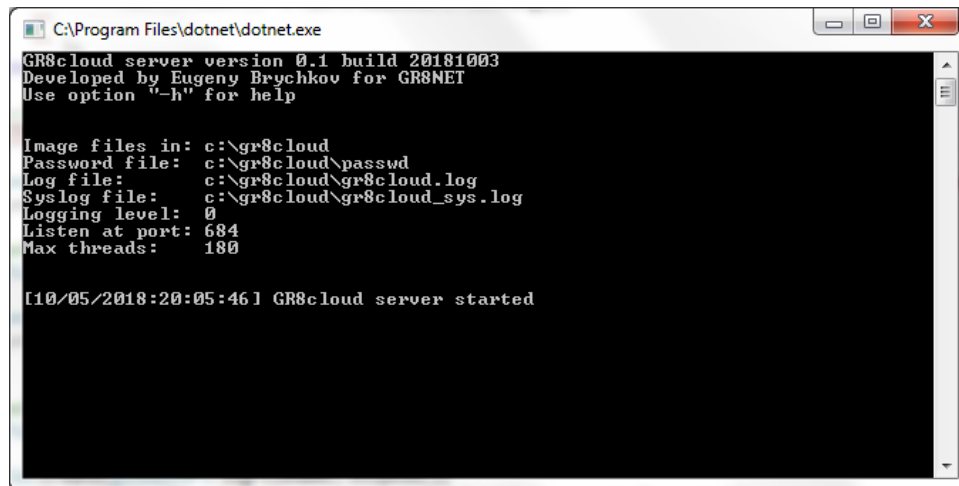
Download page is here: <http://www.gr8bit.ru/software/gr8cloudserver>.

Since October 2018 you can set up your own GR8cloud servers using provided .NET DLL application. This application runs on Windows OS as well as on Linux OS (tested on Ubuntu), and requires .NET Core Framework version 2.1 or higher installed.

Before starting the application, its data file must be prepared. The default location of files is defined by the %ApplicationData% path within the environment, appended with *gr8cloud*, thus on Windows it can be *c:/ProgramData/gr8cloud*, and on Linux systems will be */usr/share/gr8cloud*. This path can be modified by the command line keys.

Command to run the server is


**Windows**      dotnet.exe gr8cloudserver.dll [keys]  
**Linux**          [sudo] dotnet gr8cloudserver.dll [keys]



```
C:\Program Files\dotnet\dotnet.exe
GR8cloud server version 0.1 build 20181003
Developed by Eugeny Brychkov for GR8NET
Use option "-h" for help

Image files in: c:\gr8cloud
Password file:  c:\gr8cloud\passwd
Log file:       c:\gr8cloud\gr8cloud.log
Syslog file:    c:\gr8cloud\gr8cloud_sys.log
Logging level:  0
Listen at port: 684
Max threads:    180

[10/05/2018:20:05:46] GR8cloud server started
```



```
eb@eb: ~
Syslog file:  /home/eb/gr8cloud/gr8cloud_sys.log
Logging level: 0
Listen at port: 684
Max threads: 180

[10/05/2018:21:39:09] GR8cloud server started
^Ceb@eb:~$ sudo dotnet gr8cloudserver.dll -d /home/eb/gr8cloud
GR8cloud server version 0.1 build 20181003
Developed by Eugeny Brychkov for GR8NET
Use option "-h" for help

Image files in: /home/eb/gr8cloud
Password file:  /home/eb/gr8cloud/passwd
Log file:       /home/eb/gr8cloud/gr8cloud.log
Syslog file:    /home/eb/gr8cloud/gr8cloud_sys.log
Logging level:  0
Listen at port: 684
Max threads:    180

[10/05/2018:21:39:37] GR8cloud server started
```

Keys accepted by the application are listed in the table below. Unknown keys will cause application error message and exit.

CLI key	Description
-L <loglevel>	Set logging level. 0 sets minimal logging (errors), and 1 sets maximal logging (every access to the GR8cloud will be logged into <i>gr8cloud.log</i> file. Default log level is 0
-P <portnumber>	Set network port for the server, must be 16-bit value. When setting the port ensure you do not take ports used for another applications on the computer server is going to run at (e.g. port 80, 23 or 443). Default port is 684
-D <path>	Path to the data base for the server. Path must be notated according to operating system used. Overrides default path. Use quotes for strings containing spaces
-h	Help screen displaying list of the keys
-?	

You can run more than one instance of the gr8cloudserver, given the following:

1. Ports they listen on are different;
2. They use different directories, configuration and image files.

Examples of commands running the gr8cloud server:

```
dotnet.exe gr8cloudserver.dll -p 684 -d c:\myvolumes1
dotnet.exe gr8cloudserver.dll -p 685 -d c:\myvolumes2
dotnet.exe gr8cloudserver.dll -p 686
```

will start three GR8cloud server applications with one listening on port 684, another on 685 and another on 686, and with their data base directories located in c:\myvolumes1, c:\myvolumes2 and c:\ApplicationData\gr8cloud respectively.

The following preparation is required for proper application startup:

1. Data base directory must exist on the target path – either defined by the default %ApplicationData%\gr8cloud, or by the key -D;
2. File *passwd* must exist in the data base directory;
3. Images for respective GR8NET adapters must exist in the data base directory.

Log and statistic files are being created on the first run if they are not present, and they do not need to be created manually.

Format of the password *passwd* file is very simple: it is a text file containing lines of GR8NET adapter MAC address with access password separated by the space. Password must be 16 characters long max, and not blank. It is highly recommended to end last line in the password file with new line. Example of the password file:

```
101600040501_MyPassword↵
10160004051E_i*love+gr8net↵
10160004057A_msx_is_the_best↵
<empty line>
```

Password file is portable between Windows and Linux (no issue with CRLF/LF). Passwords must be printable characters, properly identified by the MSX platform and server platform's OS. Use uppercase hexadecimal characters for the GR8NET adapter MAC addresses.

---

Password file can be edited or replaced without stopping or restarting the gr8cloudserver. The only issue could be is access error returned to the GR8NET(s) while password file is being written (in general – locked), but there's unlock wait mechanism in the server thus if file is not locked for more than 100 ms then there's high probability that GR8NET requests will be properly fulfilled when password file is unlocked.

Images served by the gr8cloudserver application are simply the images of the volumes of any size supported by the Nextor in multiple of 512 bytes. However special precaution must be taken when creating large images – they have big FAT tables, and volume initialization (when GR8NET starts), DIR and other commands performing reading of the FAT may take significant time.

Images should be named with GR8NET MAC address, and extension *img*. For example, 1016000405A4.img Use uppercase hexadecimal characters for the GR8NET MAC address, and lowercase characters for *img* extension.

Images can be added without restarting gr8cloudserver. Application does not mount images, it just accesses them as files when respective GR8NET adapter connects and requests image data for reading or writing. Ideally image is added when respective GR8NET is not accessing its GR8cloud.

Operational log file: ***gr8cloud.log*** contains access errors (for logging level 0) or all access records (for logging level 1) in the following format:

<IP address> <Timestamp> <finish code> <Run time> <Key code> <Decoded response> <Write data size>

where Timestamp is local server time; Run time is server processing time in milliseconds, and Write data size is number of bytes requested to be written into the image.

Finish codes are listed below:

Finish code	Explanation
0	No error, successful (will not appear for logging level 0)
6	Authentication failed. Remote device is not the GR8NET
9	Password file access issue, ensure it is not locked by any other process
10	Blank password in password file
11	Password is too long (must be max 16 characters)
12	Authentication failed. Respective entry must exist in the <i>passwd</i> file, and passwords must match between password file and one configured in GR8NET
13, 14	Image access error. Ensure it is not locked by another process. Locking by the same GR8cloud server should not normally occur
15	Sector requested is out of the image sector range
16, 21	Error reading or writing to the image file

If you encounter other finish codes, report them back to find out the cause of the issue.

Another log file, *gr8cloud\_sys.log*, will contain messages about server start and its startup properties.

---

There will be directory *stat* created in the *gr8cloud* directory, which will contain statistics on the usage of the respective images. It will contain file named with GR8NET MAC address with extension .txt, for example 1016000405A4.txt, and it will contain only one line with four fields, for example

16 200 19 "10/03/2018:22:51:12"

with first number indicating number of queries to get GR8cloud volume size, second number indicating number of sectors read, third number indicating number of sectors written, and last field is the timestamp of last access to the respective GR8cloud volume.



---

## 14. Troubleshooting

There could be several sources of the problems, internal and external to the adapter, and before performing major modifications like re-flashing the flash chips you have to work out all possible obvious issues – for simple reason – if there's some failure, re-flashing may make situation worse.

### **When GR8NET is inserted, MSX machine does not start, or behaves weirdly**

- Ensure adapter is not damaged and does not have symptoms of being poured in with any solutions, and does not have any foreign objects inside it;
- Check edge connector of the GR8NET and clean it with spirit, not applying spirit onto the casing. Check slot connector of the machine for bent pins, foreign objects or dirt. Remove objects and clean connector with spirit. Wait until spirit and its water component evaporates before inserting GR8NET and trying it again.

### **GR8NET is installed, but there is no its initialization screen**

- Check connectors and clean them;
- Ensure Byteblaster-II/USB-Blaster programming adapter is not connected to the GR8NET;
- Ensure that arrow up key is not pressed during GR8NET initialization;
- Use Leonid Baraz's debugger to switch to slot where GR8NET is installed and investigate (a) if special control registers are present in location 5FC0-5FFF, and (b) if boot logical page F0h is not corrupt and/or starting page of ROM image (logical page 80h) is not corrupt.

### **Every networking command (NETLOAD, NETBROWSE) returns *Device I/O error***

There're several things to check:

- \_NETSTAT to ensure IP addresses are correct;
- \_NETGETHOST/GETPATH to ensure URI points to existent and accessible remote resource, and \_NETGETNAME for \_NETLOAD command;
- \_NETGETPORT to ensure that remote port address is properly set (e.g. 80 for HTTP server communication);
- Perform re-initialization of network configuration, typing \_NETDHCP.

### **Networking operations work unreliably or adapter does not connect at all**

- Check RX LED of the adapter: if it lit or very frequently flashes without TX LED flashing, then subnetwork is most probably having broadcast storm or DoS attack, and you will need to investigate your subnetwork for computers infected with malware or viruses. See "how to get network traffic log" section;
- Use PC on the same subnetwork to access remote resources – like web pages on the remote host or issuing nslookup request. If PC also fails such requests (e.g. web browser not loading the page, displaying messages like "internet unavailable" or "DNS bad config", or nslookup displaying that there's no DNS server available on the subnetwork or it is unreachable) – investigate problems with the network. The first

---

action, if it is possible, may be to reboot router/restart network services like DHCP/DNS;

- Use Wireshark to investigate what is going on with the network. Please refer to the “how to get network traffic log” section.

### **Adapter is unable to initialize in DHCP mode**

- When GR8NET initializes after power on or reset, if it does not display “DHCP successful” or “DHCP unsuccessful”, and just displays IP addresses, then GR8NET is configured in fixed IP address mode by default, and you must issue `_NETDHCP` manually, and then, after successful DHCP command completion, perform `_NETSAVE` to make adapter write flag that it must try DHCP first before falling back to fixed IP configuration mode;
- Check if there’s DHCP server on the network;
- Check if you have two adapters with the same MAC addresses on the network;
- Use `_NETVARUDTO` command’s bits [11:8] increasing number of DHCP request retries; or try placing network hub between GR8NET and your router. Some *too clever* routing devices (e.g. Cisco) may need, or be configured, to wait some time after connected peer to its port goes up (e.g. you turn MSX on or perform its reboot); placing device in between will keep router’s port always up and configured. The dumber intermediate hub device is, the better. For more information see related Cisco’s article [Using PortFast and Other Commands to Fix Workstation Startup Connectivity Delays](#), in particular (added 01-Jun-2017):



Cisco added the PortFast or fast-start feature. With this feature, the STP for this port assumes that the port is not part of a loop and immediately moves to the forwarding state and does not go through the blocking, listening, or learning states. This command does not turn STP off. This command makes STP skip a few initial steps (unnecessary steps, in this circumstance) on the selected port.

Most probably your router is having similar command to set up the port GR8NET is connected to;

- Use Wireshark to capture UDP packets.

### **Adapter is unable to perform DNS queries**

- Check if there’s DNS server on the network;
- Use PC on the same network to query DNS using web browser or `nslookup`. If they fail, you may need to restart DNS service on the server and will need to investigate networking problems.

### **How to get network traffic log?**

In order to get network traffic log you will need to install Wireshark application (available for free from [www.wireshark.com](http://www.wireshark.com)). Note that in standard configuration it will be able to capture only the following packets: packets from the workstation application is installed on; packets to the workstation application is installed on; UDP broadcast packets.

Wireshark is very useful in troubleshooting issues with DHCP because this protocol uses UDP broadcast packets. Start Wireshark before GR8NET initializes its network subsystem,

---

and stop capture after GR8NET finishes, and you will be able to see full log of packet exchange between GR8NET adapter and DHCP server under "DHCP" protocol type.

To troubleshoot issues with TCP you will need to install Wireshark on the machine running web server GR8NET is trying to connect to (or install test web server on the machine with Wireshark installed). Another alternative for whole communication capture – install Windows or Linux PC in network bridge mode between GR8NET and outer world, and run Wireshark on this PC.

### **How can I test mappers in GR8NET?**

- If card displays its firmware initialization screens and initializes as expected (in DHCP or fixed IP address mode) you may consider that mapper mode 0 is functioning properly;
- Use `_NETBROWSE` statement to connect to remote server (by default it is <http://www.gr8bit.ru>) and browse directory `/software/roms`. If this statement will display contents of remote web page, you can be 100% sure that mapper mode 0 is functioning properly;
- To test mapper #1 (plain write-protected 32K) click Enter on *Nightmare*;
- To test mapper #2 (Konami without SCC) run *The treasure of Uşas* or *Metal Gear*;
- To test mapper #3 (Konami with SCC) run *Metal Gear 2* or *King's Valley 2*;
- To test mapper #4 (ASCII-8) run *Auf Wiedersehen Monty*.
- To test mapper #7 (Mapped RAM) use `_NETSETMAP(7)` and after reboot examine bytes at addresses F341-F344 by `PRINT HEX$(PEEK(&HF341))`. It is very probable that GR8NET will appear having biggest RAM space in your machine, thus the BIOS will set it up as main RAM and abovementioned memory cells will contain slot number associated with the GR8NET adapter (e.g. 01h – slot #1 adapter may be installed in, or 89h if you installed GR8NET into the slot expander's slot #1.2). Please note that TESTRAM.COM does not identify non-native *mapped* RAM properly.

### **Audio output of the adapter is too loud or too quiet**

By default adapter is having all its audio channel volumes set to maximum – 80h (see `_NETSNDVOL` command). Application can change volume level of each device separately.

- Sound is too loud: decrease volume level for the involved device or master volume in range of 80h (&H80, full sound) and 0 (mute);
- Sound is too quiet: increase volume level for the involved device or master volume in range of 80h (&H80, full sound) and 0 (mute). However, if volume is still low, it is possible to increase master volume to 0ffh (&HFF, 255 decimal), causing GR8NET 2x digital amplification of the total audio signal level. Further volume increase is not possible without hardware changes to the adapter.

To set maximal possible volume of all the audio in GR8NET, you can use `_NETSNDVOL(255,128,128,128)`. Remember that these values are preserved by `_NETSAVE` command.

---

### Getting "Wrong version of MSX-DOS" with SD-card

This error appears when you try to run .COM program, and current media (SD-card) does not have VOL\_ID (volume ID) signature in it (e.g. SD-card was formatted on Windows OS). To confirm to COMMAND2.COM continue using the volume, type the following command:

```
SET EXPERT=ON
```

### SD-card works with Nextor, but gives *Device I/O error* with \_NETBROWSE

There's something wrong with file system on your SD-card. It must have valid boot sector and/or master boot record (MBR) with valid volume record.

Most probably, if you formatted SD-card with Nextor's \_FDISK, card is missing special compatible "55 AA" signature in the boot sector. To fix it run the following file from the web browser <http://www.gr8bit.ru/software/basic/nxtbootf.asc>.

### Machine hangs / reboots when loading software from RAM disk

If software is operational in general, this situation may happen when you enable GR8NET's disk subsystem, but your machine is having internal disk-ROM. In this case there will be extra high memory space allocated for extra logical drives, and some software (e.g. Psycho World, Noise Disk) will not work because it is designed to load its code, data and stack into predefined areas in RAM but with number of drives above 2 this specific area may happen to be reserved and used by the system. Consider the following actions to remediate the problem:

- When machine starts device initialization (after it displays the MSX logo), and till it finishes device initialization, press and hold CTRL key to force all available Disk-ROMs initializing single logical drive per controller. If you have GR8NET RAM disk enabled and another disk controller in the system (e.g. built-in FDC), this action will result in system having two logical drives, and this configuration of two logical drives is usually the one your software should work with (and developed for). **Note:** you can combine multiple keys pressed for desired result, for example you press and hold CTRL, F4 and F2 keys, and when GR8NET instructs to release keys, you only release F4 and F2, keeping CTRL key pressed until machine device initialization finishes.
- If above action did not work, then you either load software from internal storage with GR8NET's disk subsystem disabled, or remove internal machine's disk-BIOS chip from the system, and then software will run properly from the GR8NET RAM-disk.

### I open network file, but when trying I/O I get "*File not open*" BASIC error

You must look for commands which implicitly close the files –

- CLEAR
- MAXFILES (which implicitly calls CLEAR)

The fix should be execute these commands before you open network (or any other) file.

### GR8NET SCC does not output any sound

Some application has switched the device into SCC+ mode and did not return it back into SCC mode. The easiest way to reset to SCC mode is perform machine reset.

---

### **Further technical information about GR8NET**

- Release notes of the last version: <http://www.gr8bit.ru/software/firmware/GR8NET/gr8net-rom.txt>;
- Pending change requests: <http://www.gr8bit.ru/software/firmware/GR8NET/gr8net-chreq.txt>;
- Bug reports: <http://www.gr8bit.ru/software/firmware/GR8NET/gr8net-bugs.txt>.

---

## 15. Examples of the code

Samples of the following code can be downloaded from <http://www.gr8bit.ru/software/basic>.

- **Reading remote text file with HTTP header**

TCP.BAS – program to display contents of the @licence.txt file located at <http://www.gr8bit.ru/software/basic>.

```
10 CALLNETSETHOST("www.gr8bit.ru")
20 OPEN"TCPA:"AS#1
30 PRINT#1,"GET /software/basic/@license.txt HTTP/1.0"
40 PRINT#1,"HOST: www.gr8bit.ru":PRINT#1,""
50 IF EOF(1) THEN PRINT:PRINT"RECEIVED":CLOSE#1:END
60 LINEINPUT#1,A$:PRINTLOC(1);" ";A$
70 GOTO 50
```

- **Reading remote text file – contents only, no headers**

HTTP.BAS – program to display contents of the @licence.txt file located at <http://www.gr8bit.ru/software/basic>.

```
10 CALLNETSETHOST("www.gr8bit.ru"):A$="/software/basic/@license.txt"
40 OPEN"HTTPA:A$"AS#1
50 IF EOF(1) THEN PRINT:PRINT"RECEIVED":CLOSE#1:END
60 LINEINPUT#1,A$:PRINTLOC(1);" ";A$
70 GOTO 50
```

- **GR8NET disco**

PLAYLIST.ASC – program sequentially playing wave files

```
10 'Sample playlist file for GR8NET
20 CALLNETSETHOST("www.gr8bit.ru"):CALLNETSETPATH("/software/audio/")
30 RESTORE 70
40 CLS:READ A$:IF A$="" THEN PRINT"Finished":END
50 READ B$,C$:PRINT A$:PRINT B$:PRINT:CALLNETPLAYWAV(C$+".wav")
60 GOTO 40
70 DATA "Bad Boys Blue","Queen of Hearts","qoh-22-8"
80 DATA "Genesis","Home by the Sea","hbts-22-8"
90 DATA "Billy Idol","Eyes without a face","ewoaf-22-8"
100 DATA ""
```

- **Displaying BSAVED pictures**

SCR.ASC – program displaying image previously saved with BSAVE in SCREEN8 mode

```
10 'Displaying photo of Nishi-san
20 callnetsethost("www.gr8bit.ru")
30 callnetsetpath("/software/images/")
40 callnetsetname("nishisan.sc8")
50 callnetbload
60 screen 8
70 callnetbtov
80 a$=input$(1)
```

- **Get file from internet and save it to local storage device**

SAVE.ASC – prompts for remote file URI, local file name, and copies the file. File size should be ≤1MByte.

```
10 'File copy from internet to local storage device using GR8NET adapter
20 'Developed 02 Jan 2016 by Eugeny Brychkov
30 DEFINT A-P:DEFDBL S-U:S=0#:T=0#:U=0#
40 PRINT"Default server: ";CALLNETGETHOST:PRINT:PRINT"Default path: ";CALLNETGETPATH
50 PRINT:PRINT"Default file: ";CALLNETGETNAME:PRINT"asd":INPUT"Enter URI to the remote file";F$
60 CALLNETBLOAD(F$):CALLNETCODE(A,B):IFA<>0THENPRINT"Bload error: ";A:END
70 CALLNETGETMEM(255,&H6054,A,B,C,D):S=A+B*256+C*65536# +D*16777216#:PRINT"Size: ";S:P=0
80 INPUT"Output file";O$:OPENO$AS#1LEN=128:FIELD#1,128ASS$:LSET$=STRING$(128,"!")
90 U=&H6000:L=64:IFS<128THEN130ELSEIFS<4096THENL=FIX(S/128)
100 FORM=1TOL:IFS<128THEN130
110 A=VARPTR(S$):CALLNETGETMEM(0,A+1,B,C):T=B+C*256:CALLNETLDRAM(P,U,128,T)
```

```

120 PUT#1:U=U+128:S=S-128:NEXT:P=P+1:GOTO90
130 CLOSE:IFS=0THEN160
140 OPENO$AS#1LEN=1:FIELD#1,1ASS$:LSET$=STRING$(1,"!"):T=LOF(1)+1
150 FORC=1TOS:CALLNETGETMEM(P,U,A):LSET$=CHR$(A):PUT#1,T:T=T+1:U=U+1:S=S-1:NEXT:CLOSE
160 PRINT"Finished":END

```

## • Display GR8NET logo

GR8LOGO.ASC – displays GR8NET logo onto the screen (MSX2 and above)

```

10 'Display GR8NET logo, requires MSX2 machine which supports SCREEN 8
20 'Developed 13 Apr 2016 by Eugeny Brychkov
30 SCREEN 8:COLOR ,,0
40 CALLNETBTOV(369):SET PAGE(1):A$=INPUT$(1):'Image is located in the GR8NET ROM

```

## • Simple FTP client

FTP.ASC – allows downloading and uploading files using FTP. To authenticate use USER and PASS commands; to list remote directory use PASV and LIST commands; to download use PASV and RETR commands; to upload use PASV and STOR commands.

```

10 'Simple FTP client
20 'Developed 08 Jan 2016 by Eugeny Brychkov
30 CLEAR2048:MAXFILES=3
40 INPUT"FTP server";S$:IFS<>""THENCALLNETSETHOST(S$)
50 CALLNETSETPORT(21):PRINT"Connecting to FTP server...";
60 OPEN"TCPA:"AS#1:PRINT"success":'open control connection
70 LINEINPUT#1,M$:PRINTM$:'Initial FTP server message
80 LINEINPUT Q$:PRINT#1,Q$:'command for FTP server
90 LINEINPUT#1,R$:'response from FTP server
100 PRINTR$
110 I=1:GOSUB440:'get FTP code at the beginning of response string
120 IFN<>227THEN180:'not a response for PASV command
130 PRINT"* Streaming from ";L=LEN(R$):FORI=1TOL:IFMID$(R$,I,1)<> "("THENNEXTI:PRINT"Parse error":STOP
140 I=I+1:GOSUB440:A=N:GOSUB440:B=N:GOSUB440:C=N:GOSUB440:D=N:GOSUB440:P1=N:GOSUB440:P0=N
150 CALLNETSETHOST(A,B,C,D):PO=P1*256+P0:CALLNETSETPORT(PO)
160 PRINTHEX$(A);".";HEX$(B);".";HEX$(C);".";HEX$(D);".";HEX$(PO);" *"
170 OPEN"TCPB:"AS#2:GOTO 380
180 IFN<>150THEN380:'not a data exchange response
190 CO$="":FORI=1TO4:A$=MID$(Q$,I,1):A=ASC(A$):IFA=32THEN210ELSEIFA>96AND A<123THENA=(A AND &HDF)
200 CO$=CO$+CHR$(A):NEXTI
210 PRINT"* Command: [";CO$;"] *":IFCO$="LIST"THEN350ELSEIFCO$<>"RETR"ANDCO$<>"STOR"THENPRINT"* Illegal command *":STOP
220 FM$="":L=LEN(Q$):FORI=LTO1STEP-1:A$=MID$(Q$,I,1):IFA$<>" "AND A$<>"\"AND A$<>"/"THENFM$=A$+FM$:NEXTI
230 IFCO$="STOR"THEN400;'otherwise it is RETR
240 'RETR command: get file from remote server
250 PRINT"* Performing data download [";FM$;"] *"
260 L=LEN(FM$):FK$="":FORI=1TO9:C$=MID$(FM$,I,1):FK$=FK$+C$:IFC$<> "."THENNEXTI
270 EX$="":FORJ=LTO1STEP-1:C$=MID$(FM$,J,1):IFC$=" "THENEX$=MID$(FM$,J,4)ELSENEXTJ
280 FM$=LEFT$(FK$,LEN(FK$)-1)+EX$
290 PRINT"* Target is [";FM$;"] *":OPENFM$AS#3LEN=1:FIELD#3,1ASFO$
300 ON ERROR GOTO 470
310 IF EOF(2) THEN CLOSE#3:CLOSE#2:ON ERROR GOTO 0:PRINT"* Download finished *":LINEINPUT#1,M$:PRINTM$:GOTO80
320 A$=INPUT$(1,2)
330 LSETFO$=A$:PUT#3:GOTO 310
340 'LIST command: file listing to the screen
350 PRINT"* File list *"
360 IF EOF(2) THEN CLOSE#2:LINEINPUT#1,M$:PRINTM$:GOTO 80
370 LINEINPUT#2,A$:PRINTA$:GOTO360
380 IFN<>221 THENGOTO 80 ELSE PRINT"* FTP session finished *":CLOSE#1:END
390 'STOR command
400 PRINT"* Performing data upload [";FM$;"] *":OPENFM$AS#3LEN=1:FIELD#3,1ASFO$
410 FS=LOF(3):FORI=1TOFS:IF EOF(2) THEN CLOSE#3:CLOSE#2:PRINT"Communication error":GOTO80
420 GET#3:PRINT#2,FO$;NEXTI:CLOSE#3:CLOSE#2:PRINT"* Completed *":LINEINPUT#1,M$:PRINTM$:GOTO80
430 'get number from the list; I=position in string; R$ as string
440 N$="":FORJ=0TO2:C$=MID$(R$,I,1):R=ASC(C$):IFR>&H2F AND R<&H3A THEN N$=N$+C$:I=I+1:NEXTJ
450 N=VAL(N$):I=I+1:RETURN
460 'EOF char error handling
470 IF ERL=320 AND ERR=55 THEN A$=CHR$(&H1A):RESUME NEXT ELSE PRINT"Error";ERR;"in";ERL:END
480 'find substring TS$ in T$, return TO non-zero if found

```



```

490 TO=0:TR=LEN(TS$):TL=LEN(T$)-TR+1:IFTL<=0THENRETURN
500 FORTC=1TOTL:IFMID$(T$,TC,TR)=TS$THENTO=1:RETURNELSENEXTTC:RETURN

```

## • Perform DNS query using UDP: device

TEST-UDP.ASC – asks for host name and performs query to Google's 8.8.8.8 DNS server

```

0 ' Developed by Eugeny Brychkov 06 Sep 2017
1 ' Requires firmware datecode 20170905 or later
2 ON STOP GOSUB 190:STOP ON:ON ERROR GOTO 190
10 PRINT"This program performs DNS query":PRINT"to the google's server 8.8.8.8 using"
20 PRINT"UDP: device":PRINT"Press any key to continue":A$=INPUT$(1):PRINT:CALLNETGETHOST(,H$)
30 INPUT"Remote host name":A$:OPEN"UDPA:"AS#1
40 ' Make up DNS query packet
50 RESTORE 1000:GOSUB 990:P=1
60 GOSUB900:IF P<>-1 THEN 60
70 RESTORE 1010:GOSUB 990
80 ' Send the packet to 8.8.8.8
90 CALLNETSNDTG(1,8,8,8,53):PRINT"Waiting for reply from DNS"
100 ' Wait for reply
110 IF EOF(1) GOTO 110
120 ' Getting packet header
130 A$=INPUT$(8,1):PRINT"Reply from ";:FOR I=1 TO 4:Q$=STR$(ASC(MID$(A$,I,1))):L=LEN(Q$):PRINT RIGHT$(Q$,L-1);:IF I<>4 THEN PRINT".";
140 NEXT I:P=ASC(MID$(A$,5,1))*256+ASC(MID$(A$,6,1)):PRINT"port";P;:S=ASC(MID$(A$,7,1))*256+ASC(MID$(A$,8,1)):PRINT"size";S
150 ' Read packet contents into GR8NET temp buffer
160 FOR I=0 TO S-1:A$=INPUT$(1,1):CALLNETSETMEM(0,&H6000+I,ASC(A$)):NEXT I
170 ' Display packet contents' dump
180 PRINT"DNS reply packet contents":CALLNETDUMP(0,&H6000,S)
190 CLOSE #1:ON ERROR GOTO 0:STOP OFF
200 PRINT:PRINT"Application finished":END
900 ' Parse host name up to . and put into buffer
910 L=LEN(A$):Q$="":FOR I=P TO L:Z$=MID$(A$,I,1):IF Z$="." THEN P=I+1:GOTO 930
920 Q$=Q$+Z$:NEXT I:P=-1
930 L=LEN(Q$):PRINT#1,CHR$(L);:PRINT#1,Q$;:RETURN
980 ' Put preferined data into the TX buffer
990 READ A:FOR I=1 TO A:READ B:PRINT#1,CHR$(B);:NEXT I:RETURN
1000 DATA 12,0,1,1,0,0,1,0,0,0,0,0
1010 DATA 5,0,0,1,0,1

```

## • Simple chat program based to TCS: device

TEST-TCS.ASC – waits for connection, receives lines and sends reversed lines back

```

0 ' Developed by Eugeny Brychkov 05 Sep 2017
1 ' Requires firmware datecode 20170905 or later
2 ON STOP GOSUB 300:STOP ON:ON ERROR GOTO 300
10 PRINT"This program emulates line-based"
20 PRINT"telnet peer using TCS: device":PRINT"Press CTRL-STOP to end application"
30 PRINT"Press any key to continue":A$=INPUT$(1):PRINT
40 PRINT"Perform telnet to port 23"
50 PRINT"to IP address ";:CALLNETIP:PRINT
60 ' Set source port 23 and open in server mode
70 CALLNETSETPORT(,23):OPEN"TCSA:"AS#1
80 ' Check if there's a connection, loop if not
90 PRINT:PRINT"Waiting for connection"
100 IF EOF(1) THEN 100
110 PRINT"Connected":PRINT:PRINT#1,"Type ";CHR$(34);"quit";CHR$(34);" to end session":PRINT#1,""
120 ' Wait for line from the remote connected peer
130 LINEINPUT#1,A$:PRINT"Remote: ";A$
140 ' Perform reversion of received string
150 A=LEN(A$):B$="":IF A>0 THEN FOR I=A TO 1 STEP -1:B$=B$+MID$(A$,I,1):NEXT I
160 ' Replying to remote peer with reversed string
170 PRINT"Reply: ";B$
180 PRINT#1,B$
190 ' Checking if we are asked to drop connection
200 IF A$="quit" THEN PRINT"Session finished":CLOSE #1:GOTO 70
210 ' Checking if connection is dropped, then close and go reopen
220 IF EOF(1) THEN PRINT"Connection ended":CLOSE #1:GOTO 70
230 ' If connection is still there, continue getting input

```

```

220 GOTO 130
300 CLOSE #1:CALLNETSETPORT(,80):ON ERROR GOTO 0:STOP OFF
310 PRINT:PRINT"Application finished":END

```

- **PING request/reply using IPRAW: device**

TEST-IPRAW.ASC – asks for IP address to ping, sends ICMP request and displays reply

```

0 ' Developed by Eugeny Brychkov 07 Sep 2017
1 ' Requires firmware datecode 20170905 or later
2 'ON STOP GOSUB 300:STOP ON:ON ERROR GOTO 300
10 PRINT"This program sends ICMP ping":PRINT"request and receives reply using"
20 PRINT"IPRAW: device"
30 PRINT"Press any key to continue":A$=INPUT$(1):PRINT
40 INPUT"IP address to ping";A$:A$="http://" + A$ + ".1"
40 ' Open IPRAW device
50 OPEN"IPRAW:A$"AS#1
60 ' Making ping packet in the buffer
70 P=&H6000:S=0:RESTORE 1000:GOSUB 990
80 N=FIX(RND(-TIME)*65535):N1=FIX(N/256):N2=N-N1*256:CALLNETSETMEM(0,P,N1,N2):S=S+2:P=P+2
90 FOR I=0 TO 31:CALLNETSETMEM(0,P,65+(I AND 15)):P=P+1:S=S+1:NEXT I
100 ' Calculating checksum, data size is even
110 P=&H6000:C=0:CY=0:FOR I=1 TO S/2:CALLNETGETMEM(0,P,A1,A2):A=A1*256+A2:C=C+A:P=P+2:NEXT
I:C1=FIX(C/65536):C=65535*(C1+1)-C
120 C1=FIX(C/256):C2=C-C1*256:CALLNETSETMEM(0,&H6002,C1,C2)
130 ' Putting packet to output buffer
140 FOR I=0 TO S-1:CALLNETGETMEM(0,&H6000+I,A):PRINT#1,CHR$(A);:NEXT
I:PRINT:PRINT"Sending...":CALLNETDUMP(0,&H6000,S):PRINT:PRINT
150 ' Sending ICMP packet, IP and proto are already defined by OPEN
160 CALLNETSNDTG(1)
170 ' Waiting for reply
180 IF EOF(1) GOTO 180
190 ' Getting packet header
200 A$=INPUT$(6,1):PRINT"Reply from ";:FOR I=1 TO 4:Q$=STR$(ASC(MID$(A$,I,1))):L=LEN(Q$):PRINT RIGHT$(Q$,L-1);:IF I<>4
THEN PRINT" ";
210 NEXT I:S=ASC(MID$(A$,5,1))*256+ASC(MID$(A$,6,1)):PRINT" size";S
220 ' Read packet contents into GR8NET temp buffer
230 FOR I=0 TO S-1:A$=INPUT$(1,1):CALLNETSETMEM(0,&H6100+I,ASC(A$)):NEXT I
240 ' Display packet contents' dump
250 CALLNETDUMP(0,&H6100,S)
260 CLOSE #1:ON ERROR GOTO 0:STOP OFF
270 PRINT:PRINT"Application finished":END
990 READ A:FOR I=1 TO A:READ B:CALLNETSETMEM(0,P,B):P=P+1:S=S+1:NEXT I:RETURN
1000 DATA 6,8,0,0,0,4,0

```

- **HTTP access using TCP: and HTTP: devices**

TEST-TCP-DS.ASC – displays contents of file sending request in delayed-send mode (\*)

```

0 REM Developed by Eugeny Brychkov 05 Sep 2017
1 REM Requires firmware datecode 20170905 or later
10 PRINT"This program gets file using"
20 PRINT"TCP: device in non-delayed":PRINT"send mode"
30 PRINT"Press any key to continue":A$=INPUT$(1):PRINT
40 CALLNETSETHOST("www.gr8bit.ru")
50 OPEN"TCPA:*"AS#1
60 PRINT#1,"GET /software/basic/@license.txt HTTP/1.0"
70 PRINT#1,"Host: www.gr8bit.ru":PRINT#1,"Connection: close"
80 PRINT#1,"":CALLNETSNDTG(1)
90 IF EOF(1) THEN PRINT:PRINT"RECEIVED":CALLNETRESST:CLOSE#1:END
100 LINEINPUT#1,A$:PRINTLOC(1);" ";A$
110 GOTO 90

```

TEST-HTTP.ASC – opens and displays two remote files simultaneously

```

0 REM Developed by Eugeny Brychkov 05 Sep 2017
1 REM Requires firmware datecode 20170905 or later
10 PRINT"This program gets two files using"
20 PRINT"HTTP: device concurrently"
30 PRINT"Press any key to continue":A$=INPUT$(1):PRINT
40 CALLNETSETHOST("www.gr8bit.ru"):MAXFILES=3

```

---

```
50 A$="/software/basic/@license.txt":OPEN"HTTP:A$"AS#1
60 B$="/software/basic/capabilities/test-http.asc":OPEN"HTTPB:B$"AS#2
70 IF NOT EOF(1) THEN LINEINPUT#1,A$:PRINTLOC(1);" ";A$
80 IF NOT EOF(2) THEN LINEINPUT#2,A$:PRINTLOC(2);" ";A$
90 if EOF(1) AND EOF(2) THEN PRINT:PRINT"RECEIVED":CALLNETRESST:CLOSE:END
100 GOTO 70
```

---

## 16. References

- Albert Beevendorp, *Megabit ROM Cartridges*, available online at <http://bifi.msxnet.org/msxnet/tech/megaroms> (accessed on 10-Jun-2015)
- WIZnet, *W5100*, available online at <http://www.wiznet.co.kr/product-item/w5100/>, (accessed on 10-Jun-2015)
- SD Association (2015), *Simplified specifications*, available online at <https://www.sdcard.org/downloads/pls/> (accessed on 30-Oct-2015)
- Altera Corporation (2009), AN 521: Cyclone III Active Parallel Remote System Upgrade Reference Design, available online at <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an521.pdf> (accessed on 18-Jan-2019)

---

## 17. Legal statements

### 17.1. MP3 audio decoder legal statements

This is mandatory chapter to satisfy related open source licensing agreement.

#### 17.1.1. Citation of the work being used

<b>Authors</b>	Ko, Ho Fai and Nicolici, Nicola
<b>Institution</b>	McMaster University, Canada
<b>Title</b>	MAC_MP3: A Low Energy Implementation of an Audio Decoder
<b>URL</b>	<a href="http://www.ece.mcmaster.ca/~nicola/cadt.html">http://www.ece.mcmaster.ca/~nicola/cadt.html</a>
<b>Year</b>	2007
<b>Copyright</b>	Copyright © 2007, McMaster University, Ho Fai (Henry) Ko, Nicola Nicolici

The source code was not modified, and was compiled for Cyclone III FPGA without issues. The software is distributed free of charge. Executable portions of the code are provided free of charge. Nicola Nicolici did not respond to inquiries.

#### 17.1.2. McMaster University Open Source Software License

##### MCMaster UNIVERSITY OPEN SOURCE SOFTWARE LICENCE

This software licence is a legally binding agreement between McMaster University ("MCMaster"), with principal place of business at 1280 Main St. W., Hamilton, Ontario L8S 4L8 and the end user ("You") in relation to MAC\_MP3 ("Software").

USE OF THE SOFTWARE AND ANY DISTRIBUTION OF AND MODIFICATIONS TO THE SOFTWARE INDICATES YOUR ACCEPTANCE OF THIS LICENCE. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT INSTALL, USE, MODIFY OR DISTRIBUTE THE SOFTWARE.

TERMS OF USE 1. MCMaster grants You a royalty-free licence to use, copy and distribute royalty-free verbatim copies of the Software in executable code and source code provided that: a) Each copy carries with it a copy of this Licence and the warranty disclaimer in paragraph nine ("Disclaimer"), as well as the following copyright notice: Copyright (c) 2007, McMaster University, Ho Fai (Henry) Ko, Nicola Nicolici; b) Each copy, when distributed in executable code, must contain an offer by You to provide the source code.

2. You may modify copies of the Software and create derivative works based on the Software and distribute these modifications and derivative works provided that such distribution is royalty-free and that each copy: a) Carries a prominent notice stating that the work has been modified; b) Carries with it a copy of this Licence, Disclaimer and the following copyright notice: "This derivative work/publication includes MAC\_MP3. Copyright (c) 2007, McMaster University, Ho Fai (Henry) Ko, Nicola Nicolici"; and c) Contains an offer by You to provide the source code.

3. You must not impose further restrictions on any recipient of any verbatim copies, modifications or derivative works of the Software.

4. You must not re-distribution the Software for commercial purposes without express written authorization from MCMaster. Contact McMaster University - Office of Research Contracts and Intellectual Property (Tel: 905-525-9140 ext 22873; Fax: 905-540-8019; email: [orcp@mcmaster.ca](mailto:orcp@mcmaster.ca)) for such approval which may be conditioned on the payment of royalties or such other terms as MCMaster may,

in its sole discretion, decide.

5. If You use or reference the Software in any publication (including scientific publications, electronic documents and websites) or derivative work, You must give appropriate reference to the Software and MCMaster.

OWNERSHIP 6. MCMaster retains all right, title and interest in the Software, including, but not limited to, all trademarks, copyright, and patents and other intellectual property. All trademarks, trade names, logos, customarily used symbols and other designations as used or adopted by MCMaster, including without limitation the designation of the Software, will at all times be and remain the

property of MCMaster and cannot be used in conjunction with the distribution by You of the Software.

THIRD PARTY RIGHTS 7. THE SOFTWARE MAY BE DEPENDENT ON PROGRAMS, OPERATING SYSTEMS OR OTHER INTELLECTUAL PROPERTY OF THIRD PARTIES WHICH MAY REQUIRE THIRD PARTY LICENCES. COMPLIANCE TO TERMS AND PAYMENT OF FEES ASSOCIATED WITH THESE THIRD PARTY LICENCES ARE THE SOLE RESPONSIBILITY OF YOU AND ARE NOT INCLUDED IN THIS LICENCE. In particular, MCMaster does not represent or warrant that the Software is free of infringement of any third-party patents. Commercial implementations of MPEG-1 and MPEG-2 audio/video, including shareware, are subject to royalty fees to patent holders.

---

McMASTER advises You that these patents are general in scope and may affect your implementations regardless of your implementation design.

TERM AND TERMINATION 8. This Licence commences on the date the Software is electronically or physically delivered to You. This Licence will terminate immediately without notice if You fail to comply with any of the terms and conditions of this Licence.

WARRANTY DISCLAIMER 9. THIS SOFTWARE IS PROVIDED "AS IS" WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS, OR IMPLIED. ALL WARRANTIES AND REPRESENTATIONS, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, ACCURACY AND FITNESS FOR A PARTICULAR PURPOSE, ARE DISCLAIMED AND EXCLUDED. MCMaster DOES NOT WARRANT THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE SOFTWARE WILL OPERATE UNINTERRUPTED, OR ERROR FREE, OR THAT ANY DEFECTS IN THE SOFTWARE WILL BE CORRECTED. YOU ASSUME ALL RISK AND RESPONSIBILITY FOR THE SELECTION, INSTALLATION, USE, QUALITY, PERFORMANCE, AND RESULTS OBTAINED FROM THE SOFTWARE.

LIABILITY DISCLAIMER 10. IN NO EVENT WILL MCMaster OR ANY OF ITS REPRESENTATIVES BE LIABLE TO YOU FOR ANY DAMAGES OF ANY KIND WHATSOEVER, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, IN ANY WAY ARISING OUT OF OR IN CONNECTION WITH EITHER THE SOFTWARE, ITS DISTRIBUTION, USE OR PERFORMANCE, OR THIS LICENCE, AND WITHOUT REGARD TO WHETHER ANY CLAIM FOR DAMAGES IS BASED IN CONTRACT, WARRANTY OR TORT, OR ARISES UNDER STATUTE, COMMON LAW, OR OTHERWISE, OR WHETHER INJURY WAS SUSTAINED BY PERSONS, PROPERTY, OR OTHERWISE, OR WHETHER LOSS WAS SUSTAINED FROM THE LOSS OF USE, DATA, INFORMATION, BUSINESS INTERRUPTION, OR PROFITS. MCMaster SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY FOR DAMAGES OF WHATSOEVER KIND AND NATURE CAUSED BY THE INABILITY OF THE SOFTWARE TO OPERATE EFFICIENTLY OR EFFECTIVELY. BY

AGREEING TO THESE TERMS AND CONDITIONS, YOU WILL INDEMNIFY MCMaster FOR ANY AND ALL THIRD PARTY CLAIMS FOR ANY THEORY OF LIABILITY.

JURISDICTION 11. This Licence will be construed in accordance with the laws of the Province of Ontario, Canada and the courts of Ontario will have jurisdiction over all claims, disputes and actions related to this Licence.

REPORTING ERRORS [optional] 12. MCMaster does not warrant that Software will meet Your requirements or that its use will be uninterrupted or error free. Notwithstanding the foregoing, You acknowledge that if problems are encountered with the Software, You may notify the Authors at MCMaster [Nicola Nicolici <nicola@ece.mcmaster.ca>]. MCMaster makes no warranty that reported errors will be fixed. If You transmit source code improvements or modifications to MCMaster, You agree to provide to MCMaster a non-exclusive, royalty-free licence to use, copy and modify such improvements or modifications.

ENTIRE AGREEMENT 13. This Licence is the complete and exclusive agreement between both parties in relation to the Software and supersedes any proposal or prior agreement, oral or written, and any other communication between the parties relating to the subject matter of this Licence.

## **17.2. Pletter legal statement**

Pletter v0.5b - XL2S Entertainment 2008

Have fun with it, just do not harm anyone :)

Keep in mind pletter is based on software with the following license:

Copyright (c) 2002-2003 Team Bomba

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

## 18. Document revision history

02 Feb 2016

- MSX-DOS integration chapter is added;
- Firmware call addresses have changed;
- Added cold/warm boot notice into introduction.

10 Mar 2016

- Added output port for PCM function: index register #5 for OUTI/OTIR implementation.

15 Apr 2016

- NETBTOV now supports logical page number within its first argument;
- NETBLOAD now supports two more arguments PG:ADDR identifying start of data load;
- Added NETPLAYBUF family of commands.

11 May 2016

- Inserted *Applications* chapter; described MSX webserver and FTP BASIC program operation.

09 Jun 2016

- Added SELECT, CLS/HOME and alphanumeric key functionality to browser;
- Added mappers 6 and 8, and memory manager chapter;
- Added statements: GETMMV, SETMMV, VARRWTH statements;
- Added ERRREG into special control registers list;
- Completely rebuilt DSK interface;

23 Jun 2016

- Added NTP section and commands;
- Enhanced URI structure to 63 characters for file names, and added query string of max 63 characters. Added GETQSTR and SETQSTR statements.

04 Jul 2016

- Enhanced URI structure to support SD-card URI;
- Changed NETSETHOST, NETGETHOST including SD-card access;

17 Aug 2016

- Index register #5 can now be input for prefetch function for INI/INIR operation.

09 Oct 2016

- Added more information about mappers, mapper type 8, mapped RAM read flag;
- Expanded browser with input URI argument, input/output flags, added NETVARBRSTR.

05 Nov 2016

- Added TGTMAP command to set target mapper;
- Added value 2 for mapped RAM register read flag for SETMAP and TGTMAP commands to be default value and causing GR8NET to auto-detect if, within current configuration, it should set up mapper registers to be readable;
- Split part 2 onto two parts explaining physical design and initialization.

20 Dec 2016

- Corrected PARURI firmware call's address (correct is 5F8E)
- Added Nextor as a chapter and into the description of mapper mode 8
- Added direct firmware calls DEV8RW, MMVAR, RX, NETCMD, GWREGS, B2ON, B2OFF, GCURL, UDPOP, added network operation workflow chart in firmware calls chapter

29 Dec 2016

- Added support for 7 Mhz overclocked MSX machines;
- Moved *SD-card CSD* information from special register set to the logical page C9;
- Added bit 1 in system mode register as source of the SCC clock (later renamed to *SCC, OPL and OPLL clock source*);
- Added MSX slot clock speed identification in logical page C9;
- Added two commands NETGETCLK and NETSETCLK.

22 Jan 2017

- Added F1 and F2 keys for browser, Disk-ROM initialization diagram;
- Inserted chapter 7 *Built-In OPLL*;
- Added NETOPLL command, added 5<sup>th</sup> parameter into NETSNDVOL command.



---

13 Feb 2017

- Added Y8950 (MSX-Audio) support;
- Replaced NETOPLL command with two – NETGETOPL and NETSETOPL;
- Added MSX-Audio settings in the special register set (5FD8/5FD9), added Y8950 disable bit in system mode register;
- Updated memory manager chapter to reflect MSX-Audio sample RAM allocation;

28 Feb 2017

- Added section 5.1 on opening files in the browser

16/21 Mar 2017

- Added NETSYSINFO command to get system information and system performance data for troubleshooting
- Added NETVARUDTO command to control DHCP and DNS timeouts.

30 Mar 2017

- Added command NETPLAYVID and error code 2E into NETCODE, discontinued GR8VIDEO.COM application (however it will still run);
- Added CTRL-V key combination into browser to play video files from SD-card using it.

26 Apr 2017

- Added *Quick user guide* chapter;
- Redesigned WAV player chapter to be media player, and added MP3 streaming support subchapters;

03 May 2017

- Added DSKSVIMG command;
- Totally reworked built-in Disk-ROM to work from CPU bank 1;
- Flash chip firmware version is changed from 0.4 to 0.5.

09 May 2017

- Added mapper modes 9-14;
- Added NETGETMAP command, added third argument to NETSETMAP command;
- Added mapped RAM disable functionality (to disable GR8NET mapped RAM in mapper modes 9-14 because some games can not run if main RAM is in the same slot as game);
- Changed adapter identification through index I/O port 0 from 'G' to "GR8N" + GR8NET engine date code.

15 May 2017

- Expanded NETPLAYVID command to initialize screen without running video;
- Added {} modifier to disk image naming to force 2 logical drive configurations when {2} is in place.

21 May 2017

- Added NETPLAYVID(SM) flag bits 7 and 6 in addition to screen mode in bits [3:0];
- Redesigned NETPLAYBUF family of the commands.

18 August 2017

- Added chapters about FPGA chip interface and remote update functionality;
- Added Y8950 volume, mixer registers, *Managing audio mixer* chapter;
- Added bit 5 of the PCM control register (stereo mode), now NETPLAYWAV can play stereo files;
- Added NETFPGAUPD command and related subchapter;
- Added support for SD-card with multiple partitions (SDC://, SDD://, SDE:// and SDF://)
- Inserted NETRESST section (3.13).

13 Sep 2017

- Redesigned BASIC I/O access, inserting three new chapters at the beginning;
- Network file does not need socket identification any more (socket ID will be just discarded and socket will be system allocated);
- Added 5 more arguments to NETSNDTG command to override remote IP address and remote port;
- Redesigned BASIC I/O chapter, separating UDP and IPRAW devices, adding device string format subchapter for OPEN statement;
- Changed BASIC I/O devices to be binary ones with receiving EOF character (&H1A) not causing *Input past end* error any more;
- Corrected output of mono GR8NET through the right channel (not left as was stated before);
- Added subchapter about TCP/IP UNAPI implementation;
- (16 Sep 2017) Removed subchapters about firmware update from chapter 3, put information from them into expanded chapter 8.

27 Sep 2017

- Added Y8950 interrupt disable bit into NETSETOPLL/NETGETOPLL commands, added system mode register 1 and associated bits (Y8950 configuration state bit, port selection, interrupt disable).

16 Oct 2017

- Added chapter 8 about embedded MP3 media player and NETRECFG command, shifting all following chapters' numbers;
- Added argument to NETFPGAUPD command;

- 
- Added image type into NETVER, and added NETVER command at the beginning of chapter 3 "Using GR8NET in BASIC";
  - Added FPGA image type into system mode register 1;
  - Added legal chapter for *MAC\_MP3: A Low Energy Implementation of an Audio Decoder*;
  - Added notice to MSX-Music/MSX-Audio chapter that NETSETCLK sets clocking source for OPL/OPLL (not just SCC), and that NETSETCLK state is saved by the NETSAVE command.

22 Oct 2017

- Now NETSETMAP may be run without arguments to start the ROM loaded into the GR8NET RAM buffer;
- Added Philips Music Module DAC at port 0Ah shared with Digital Waveform input.

11 Nov 2017

- Extended URI structure to version 1 with additional extension field for host name;
- Corrected firmware call flow picture (was missing NETCMD call when sending), added USIRSOP system call;
- Now NETSETMIX command accepts lower case characters in the mixer setting string;
- Default remote port number is set to 80, if URI string does not have port # in it, PARURI resets it to 80;
- Now media player (NETPLAYWAV) follows network redirects (but not NETBLOAD) the same as in NETRESST command;
- Added bit 3 for network URI structure (20171113);
- Added NETCFG command (it was there for quite a long time but it was forgotten to be included).

18 Nov 2017

- Added error code 23 "Unable to redirect";
- Slightly updated NETRESST command description.

25/28 Dec 2017

- MP3 FPGA firmware image now can work in mapper 8 mode (to support Sympos), network MP3 player now displays buffer status bar and supports ICY metadata;
- NETTELNET command functionality was removed, NETTERM (terminal) put instead.

07 Feb 2018

- Documented NETBITOV command.

02 March 2018

- Added SCC+ support.

07 March 2018

- Added subchapter *RAM allocation conflicts in composite mappers* and improved NETSETMAP subchapter;
- Updated *Initialization sequence and messages* chapter.

18 March 2018

- Corrected error in the description of slot connector bus speed measurement, only 3 bytes (24 bits) represent the counter.

06 Apr 2018

- Added bits [11:8] to the NETVARUDTO command to adjust number of DHCP request retries on GR8NET initialization;
- Inserted chapter 7 "GR8cloud virtual volume", with all subsequent chapters increasing their index number;
- Enhanced description of the NETTERM command per Fabio Roncolato's findings.

13 Apr 2018

- Returned the reworked TELNET application back (graphics-based, SCREEN 7, thus MSX2 only).

07 May 2018

- Added SCCM bit into system register 1, and added '+' character display after adapter flags in "Initialization messages and sequence" and \_NETSTAT command synopsis sections;
- Added "How Do I...?" section.

13 September 2018

- Modified chapter *Playing video from SD-card* with MSX1 information;
- Added subchapters *Video file formats* and *Converting .SC2 file from version 0 to version 1 format*;
- Updated \_NETSYSINFO picture and comments;
- Added chapter about HMM3 demo and card game "DURAK".

20 September 2018

- Added CS variable argument to the NETGETOPL statement;
- Rewrote *Starting with built-in OPLL* chapter to reflect updates to the "moonblas.dsk" disk image.

10 October 2018

- Added *GR8cloud server* subchapter into *Applications* chapter.

---

15 November 2018

- Added *Exporting and importing GR8NET configuration* subchapter;
- Added *Using built-in PSG* subchapter;
- Updated NETSND, NETSETMIX, NETCFG, NETGETCLK, NETSETCLK commands to reflect addition of built-in PSG;
- Updated special register set table with PSG volume, and updated descriptions of System mode registers 0 and 1 to reflect new built-in PSG control bits;
- Added *BASIC command reference* subchapter (yet without links to individual commands);
- Added "How do I" SCC+OPLL entry (25 Nov 2018).

08 January 2019

- Reworked and updated *Memory manager* chapter with the memory allocation diagrams for each mapper mode;
- Added *Boot-up* menu subchapter and *Change refresh rate* option to this menu;
- Added *Serial flash chip information* subchapter, and *Serial flash chip and configuration image management* subchapter;
- Added \_FL\* commands into and updated \_NETFPGAUPD command in *BASIC command reference* chapter;
- Added *Pletter legal statement* subchapter;
- Updated *SCC/SCC+* section to mention that SCC/SCC+ is now also available in mapper mode 8;
- Added subchapter "Online migration from version 0.7 to version 0.8";
- Changed MathPack's audio clock source from MSXBUS clock measurement to the measurement of the GR8NET multiplexed audio clock (either MSXBUS or internal 3.579545 MHz precision clock);
- Corrected consistency about MSX internal mixer connection for mono and stereo versions of GR8NET;
- And in overall proof-read whole manual ensuring there're no obvious errors and typos.

30 May 2019

- Changed how NETGETCLK and NETSETCLK work: they do not set internal device's clock source any more (these devices are always clocked from internal clock now), but set the source to measure and return: internal 3579545 clock or external MSX BUS clock speed;
- Modified descriptions of NETGETMD, NETSETDM and NETSDCRD digesting all types of variables and not using MathPack any more;
- Added SHIFT+ENTER key combination into \_NETBROWSE;
- Changed controlled generator specification because it can not be clocked by MSX bus any more – only options are 10 MHz or internal precise 3.579545 MHz;
- Changed bit CLKSRC to MCLKSRC in system mode register 0, and its function (now selects measurement source only);
- Added section "Networking libraries for Fusion-C (SDCC-based)".