May 10, 2012 Severity: Information Eugeny Brychkov, RU

Storage subsystem v.2.0a: Using and Programming

This DN0002 design note is applicable to the storage subsystem microcode version 2.0a and above.
This DN0002 design note is licensed to GR8BIT Engineering Community, and put into public access as an evaluation of the quality of product and related documentation and support services for potential customers.

Index

1. Introduction

Purpose of this document: This document is intended to provide to GR8BIT Engineering Community members, as well as other interested users and hardware, software and firmware developers who want to use GR8BIT storage subsystem v.2.0a and above, and use its API to connect and operate their own devices.

Overview of features: Starting version 2.0a GR8BIT platform has onboard IDE hard disk drive support, however initially in limited way. In general, the following media are supported with GR8BIT:

- 3.5" floppy disk drives (up to 2);
- ATA compliant IDE hard drives (up to 2).

Compatibility considerations: There're several efforts made in order to maintain compatibility with MSX standard as well as to support MS-DOS/Windows (hereinafter referred as PC systems) formats:

- GR8BIT formatted diskettes are readable and writeable by PC systems; PC systems' formatted diskettes are readable and writeable by GR8BIT, but not bootable.
- GR8BIT hard disk drive volume structures are compliant with PC systems conventions, thus GR8BIT can configure, read and write supported file systems of supported types, created by PC system. GR8BIT hard disk portion of the driver code uses LBA fields within MBR to get logical volume (LVOL) configuration, and addressing hard disk space using C/H/S mode (32-bit mathematics are embedded into microcontroller's microcode).

Supported storage media configurations:

- Diskettes formatted with 9 sectors per track (360K/720K), and diskettes formatted with 18 sectors per track (1.44MB). GR8BIT disk ROM does not follow *Media Descriptor* conventions, and it gets diskette configuration information from its boot sector. ***Important***: in order for subsystem to get diskette configuration, GETDPB or DSKCHG system calls should be invoked. Normally they are after one or two unsuccessful read operations, but some custom software (e.g. disk version of Metal Gear 2) does not invoke these calls at all, thus will not work if GR8BIT was started in Standard mode (you will need to manually reconfigure system booting in compatible mode getting default diskette configuration at the startup).
- HDDs operating in C/H/S mode; maximal configuration is 65535/15/255.
- Logical drives (up to 2) located on the HDDs with up to 65536 sectors in size (32MB) in 16-bit addressing mode and up to 8,388,608 sectors in size (4Gbytes) in 23-bit addressing mode. There's no limitation on the location of LVOL on the hard disk as all 4 bytes (32 bits) are used to address LVOL base sector. Only FAT12 partitions are supported by this microcode at this time (other types may be handled by other system software/drivers).

Limitations of functionality: Some functionality which may be desired is still missing (partially due to scarce disk driver ROM space):

- Sector interleaving during diskette format operation is not implemented;
- Floppy disk media bad block management is not implemented;
- In diskette format dialog, if you choose to write system onto diskette being formatted, all system files – DOS1 and DOS2 – will be written. If you do not need DOS2 system files, just delete them afterwards;
- Diskette format routine will always write DOS2-type boot sector, however microcontroller's API call allows choosing to write DOS1-type boot sector.

2. Operating GR8BIT storage subsystem

Storage subsystem, by default, boots with phantom drive disabled, and in diskette compatibility mode (3 FAT sectors per diskette). This deviation from the standard MSX storage subsystem behavior is intended, and performed in order to have maximal free BASIC/DOS memory after boot.

If you want to have phantom drive B:, hold CTRL key during floppy drive initialization (seek operation, a kind of “hum-hum” sound) and till ROM displays that has configured 2 logical floppy drives.

If you want storage subsystem to support Standard floppy disks (having 9 sectors per FAT copy), hold GRAPH key (or ALT key of PS/2 keyboard) till ROM displays that it is configured in Standard (9FS) mode.

2.1. Memory consumption considerations (DOS1)

Unlike in DOS2, which does not keep single full copies of file allocation tables for each drive, DOS1 allocates upper memory space for these copies. More drives you have configured, less available memory for BASIC programs and BASIC executable programs (e.g. .BIN, .GM) you have.

You may have 1 or 2 floppy devices configured, as A:, or A:/B: respectively. In *Standard* mode GR8BIT will allocate space for 9 FAT sectors per every configured floppy drive (9*512=4608 bytes), and will be capable reading 1.44 diskettes created using PC system. In *Compatible* mode GR8BIT will allocate space for only 3 FAT sectors per every configured floppy drive (3*512=1536 bytes), and you will only be able to use *Legacy 720K* disks and *Compatible 1.44M* disks (usage of *standard* PC-created 1.44 diskettes will yield unpredictable results as GR8BIT will read 9 FAT sectors into 3 sector buffer and above it).

Table below shows memory consumption for cases described above. Hold CTRL key if you wish GR8BIT not to configure second floppy drive (or phantom drive B:). Hold GRAPH (or ALT key on PS/2 keyboard) to configure floppy subsystem to work in *Compatible* mode.

CTRL key**	GRAPH key	Bytes used	Notes
Pressed	Pressed	9216	
Pressed	Released	3072	You can’t use 9 FAT sectors formatted (Standard) diskettes
Release	Pressed	4608	Logical drive B: is not available
Released	Released	1536*	You can’t use 9 FAT sectors formatted (Standard) diskettes and logical drive B: is not available

* Configuration subsystem boots into by default

** Note change of behavior of the CTRL key

This mechanism of *compatibility* does not apply to hard disk logical volumes – GR8BIT will always configure number of FAT sectors indicated in LVOL boot sector

record (e.g. for 8 sectors per FAT space consumed will equal to 8*512=4096 bytes). In addition, configuration of physical parameters of HDD (sectors per track and heads), as well as logical configuration (DPB) is performed once during system startup. This means that subsystem does not support removable and hot-pluggable HDD media, and you will need to reboot the system after you repartition the hard disk (unless repartition program updates the LVOL DPB).

2.2. Starting GR8BIT with storage subsystem installed

Before GR8BIT will get to operating system (OS) load procedure, or to the BASIC if OS in not present on the first media system finds, disk ROM will perform initialization of the storage subsystem and enumeration of the logical volumes (LVOLs) located on the hard disk drives.

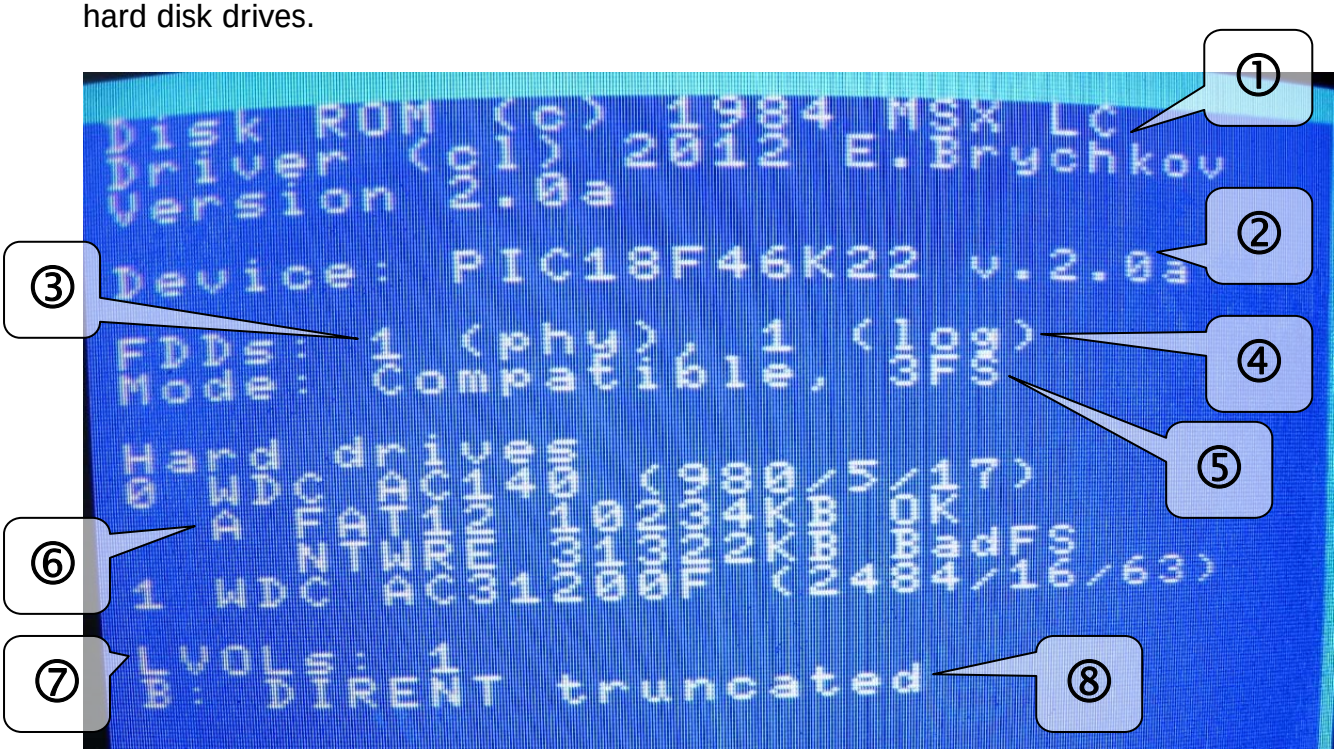

Figure 2.1. Storage subsystem initialization screen

Figure 2.1 shows messages on the initialization screen. Let's consider every part, as each provides you with important information:

1. Copyright/Copyleft notice, with GR8BIT storage subsystem ROM version (IC3);
2. Version of the microcode in the microcontroller. You should ensure that it is compatible with ROM version. In case MC is not installed, hang or malfunctions, the "none" message will be displayed;
3. Number of physical floppy disk drives. If it displays less number that you have connected, one or more floppy drives are faulty. Failure of floppy drives does not impact operation of hard disks;
4. Number of logical floppy drives configured. By default, in order to increase available memory, only one logical drive (A:) is being configured if there's only

one physical floppy drive is present. You can configure phantom drive (B:) by holding CTRL key during storage subsystem initialization.

5. Mode: either Compatible or Standard. Compatible mode refers to the handling of *floppy* media configuration of maximum 3 sectors per file allocation table (FAT) copy. These are Legacy 720K diskettes, and Compatible 1.44 diskettes. Standard mode refers to the *floppy* media configuration of maximum of 9 sectors per FAT copy. If storage subsystem is configured in Standard mode, there will be less user available upper memory space if DOS1 kernel is running (see chapter 2.1). By default Compatible mode is selected (maximum available memory). Mode can be changed by holding GRAPH key (or ALT key on PS/2 keyboards) during subsystem initialization. Compatibility mechanism described above does not apply to LVOLs configured from hard disk drives;
6. List of hard drives detected and partition entries detected on them. If trailing message is not "OK", LVOL will not be configured. If subsystem senses that there's a hard disk, but cannot get its identification information (e.g. disk is faulty, unsuccessfully completed master/slave diagnostics, or incorrect disk master/slave jumper configuration), ROM will count down approximately 8 seconds before timing out;
7. Number of LVOLs configured within drive work area. Note that having one LVOL configured does not mean you will have full access to it;
8. Any diagnostic messages from LVOL configuration microcode. Error detected during LVOL DPB (drive parameter block) configuration are:
    - *DIRENT truncated:* number of directory entries exceeds allowed 254 (like in example on fig. 2.1): directory entries will be truncated to 254 in DPB, and volume will be configured in read-only mode. Data read from such a volume *can be valid*;
    - *DPB Error:* boot record of the volume is invalid, thus DPB cannot be constructed out of it. LVOL will be configured using dummy DPB in read-only mode. Data read from such a volume will, most probably, *be invalid*.

2.3. Booting GR8BIT with storage subsystem installed

GR8BIT will boot the operating system (OS) from first storage device it finds, in case the following conditions are met:

1. Device is operational and ready for system read requests – it is attached to the controller, passes its internal diagnostics, and configured properly;
2. Partition table is valid (for hard disk logical volumes – LVOLs) – partition table can be initialized either by attaching hard disk to PC and using Disk Management utility (ensure it uses FAT12 setting), or by using GR8HDD utility on your GR8BIT;
3. Boot sector contains valid media configuration – valid boot sector is written during format operation (diskettes: *format* in OS and *call format* in BASIC; GR8HDD utility for hard disk volumes);

4. There're operating system files on the media – at least MSXDOS.SYS and COMMAND.COM (can be loaded using GR8HDD utility).

If storage subsystem detects no floppy drives, first LVOL becomes drive#0 (A:) and can be booted from . Note that DOS1 kernel does not handle no drives at all. If you detach all floppy drives, and have no LVOLs configured (due to no hard disks attached or invalid MBR/boot record), system will hang. To resolve the lockup you will remove GR8BIT storage subsystem ROM chip IC3 from the I/O board.

2.4. Configuring storage media to be used within GR8BIT

2.4.1. Configuring floppy drive and floppy disks

There's nothing about configuration of the floppy drives other than just attaching them to the respective connector on the floppy cable. Ensure that you use regular FDDs you used to use with your PC with DC (disk change) option (pre-) set. Disk drives which are configured with RDY option will not work.

Configuring diskettes involves formatting them – laying out sector information in the raw media, and filling these sectors with proper information (boot sector, FAT, directory).

There're two options for formatting diskettes:

1. Format floppy disk on your PC, using 1.44M size (for HD diskette only, PC command is "format /t:80/n:18", will create *Standard* diskette in GR8BIT terminology) or 720K size (for DD and HD diskette, PC command is "format /t:80/n:9", will create *Compatible* diskette in GR8BIT terminology). Disks created this way will be readable and writeable on GR8BIT, but will be not bootable due to PC boot code in the boot sector;
2. Format floppy disk using GR8BIT's MSX-DOS ***format*** command or BASIC's ***call format*** command. Both invoke the same formatting microcode residing in the ROM. Usage of these commands in scripts/applications is not recommended.
   When you invoke the ROM-embedded formatting command (fig. 2.2), you will select logical drive (and media in it) to format (1). If it will appear to be hard disk located LVOL, command will abort with message "Use GR8HDD".
   You will see options available (2), and dialogue string (3). Pressing respective digit key on the keyboard – "1" key to toggle format (Standard 1.44, Compatible 1.44 or Legacy720), "2" key to toggle option to write DOS files after formatting – dialogue string will be changed appropriately. You can abort the command any time until it starts formatting by pressing CTRL-STOP or CTRL-C key combination.
   If you select "System" option, MSX-DOS1 and MSX-DOS2 system files will be written onto the diskette. If you are not going to use media with DOS2, you will remove DOS2 files after format command completes.
   Note that format command will always write DOS2-type boot sector.
3. You can load image onto the diskette. Note that it can be done only after diskette was properly formatted (e.g. physically prepared for user data load).

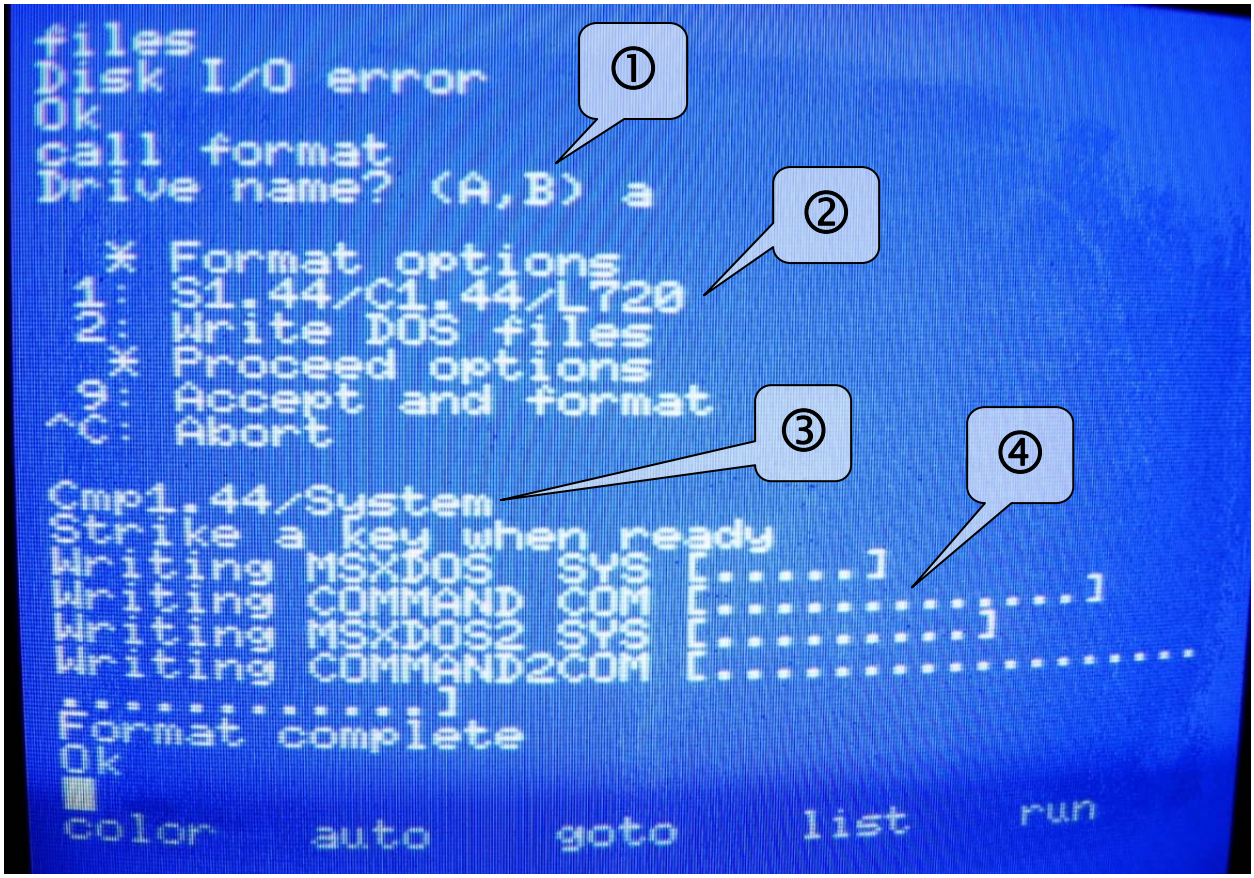

Figure 2.2. Diskette formatting dialogue

2.4.2. Configuring hard drives and logical volumes (LVOLs)

First of all, hard drive should have been during in the storage subsystem initialization (see chapter 2.2). Then you will use **GR8HDD** operating system application to partition hard disk, and prepare LVOLs for system usage. Note that this document describes initial release of GR8HDD, later versions and revisions may contain improved and enhanced functionality.

GR8HDD uses system calls provided by the GR8BIT storage subsystem ROM, and first it looks for this code, identifying the driver work area where all the definitions are stored (fig. 2.3, (1)). Then it uses hard disk enumeration call from the disk ROM and lists available hard disk drives.

At this stage you will select hard disk unit (only 0 is available choice in example on fig. 2.3), or press CTRL-C or CTRL-STOP to abort. Then you will see the list of fixed parameters, and list of changeable parameters (fig. 2.4). As you change these parameters, approximate LVOL size will be displayed at position (1). When you are satisfied with the volume size, select option 3 and confirm it by typing “Y” key (fig. 2.5). GR8HDD initializes MBR, writes control structures (including bootable boot sector), and requests you to reboot the system as LVOL parameters most probably change. Please do not select LVOL size larger than hard disk size – it may lock up HDD at the enumeration stage.

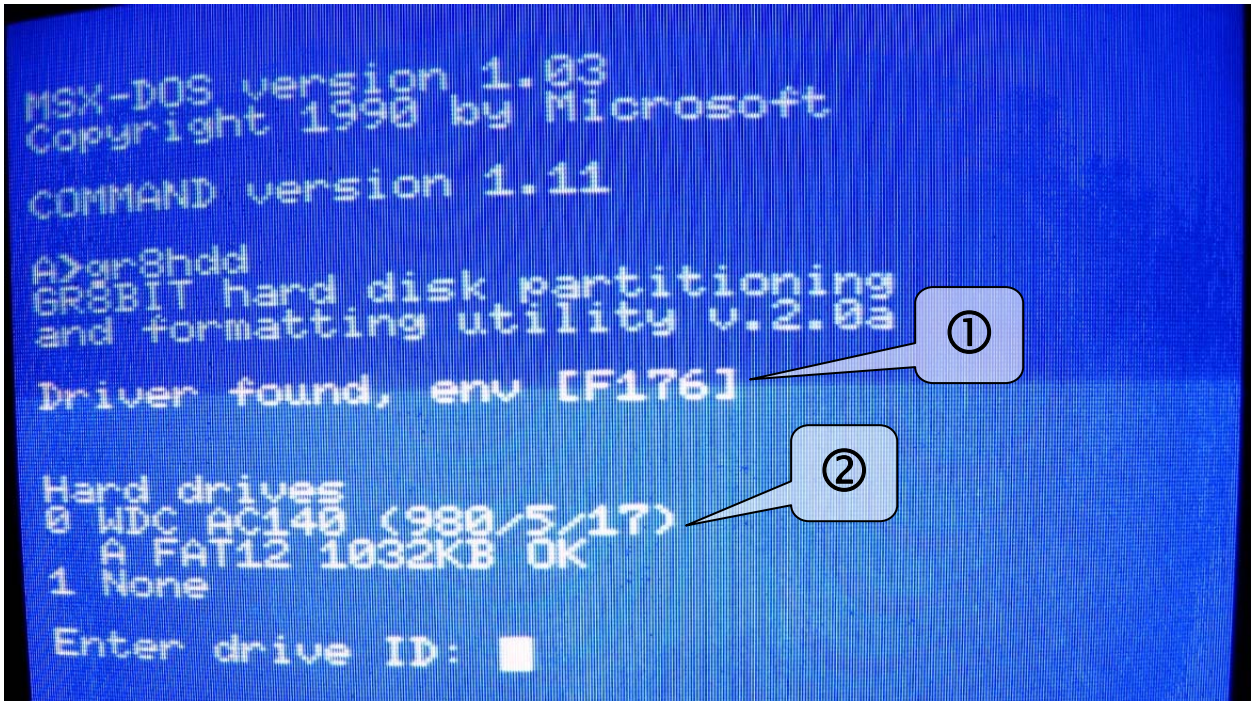

Figure 2.3. Start of GR8HDD application

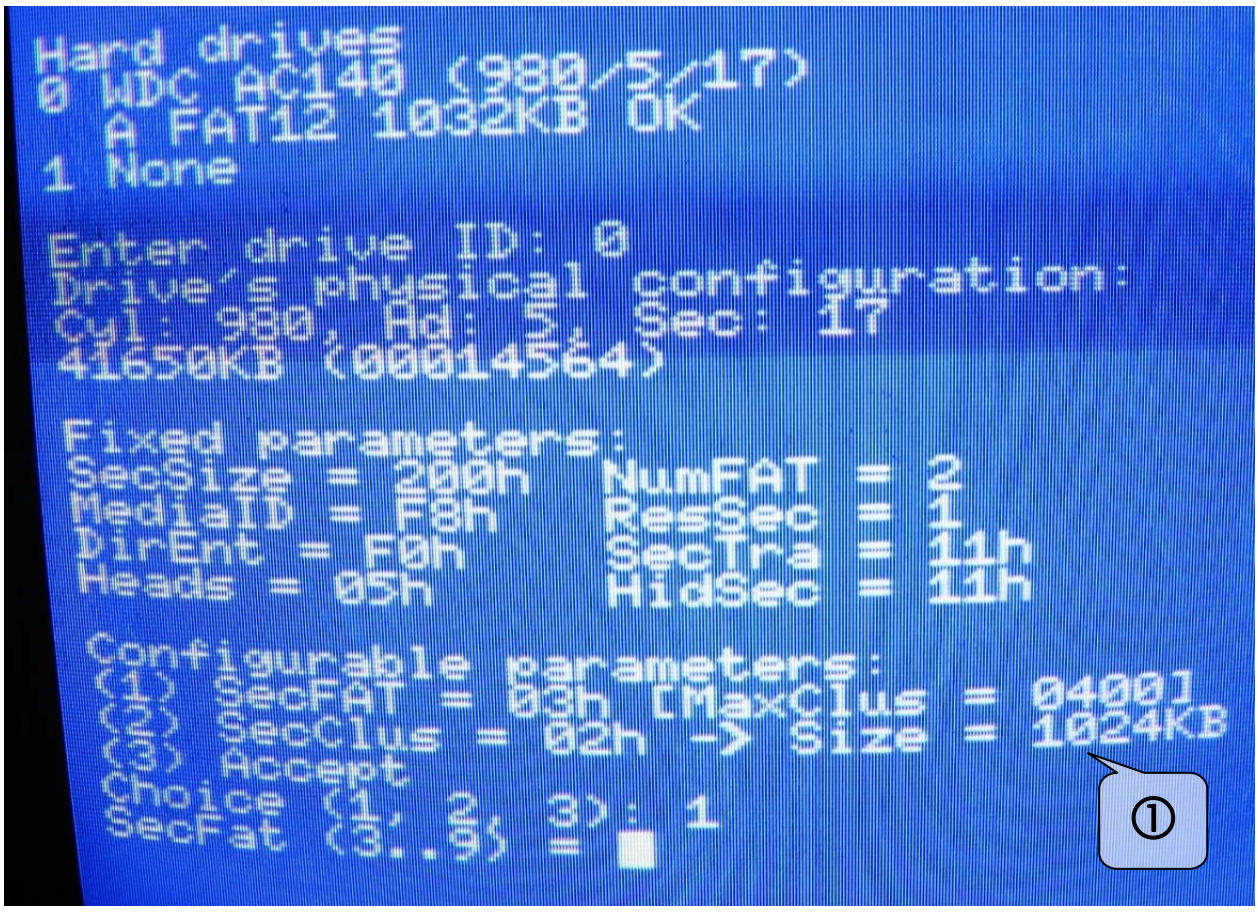

Figure 2.4. Further option selection dialogue of the GR8HDD

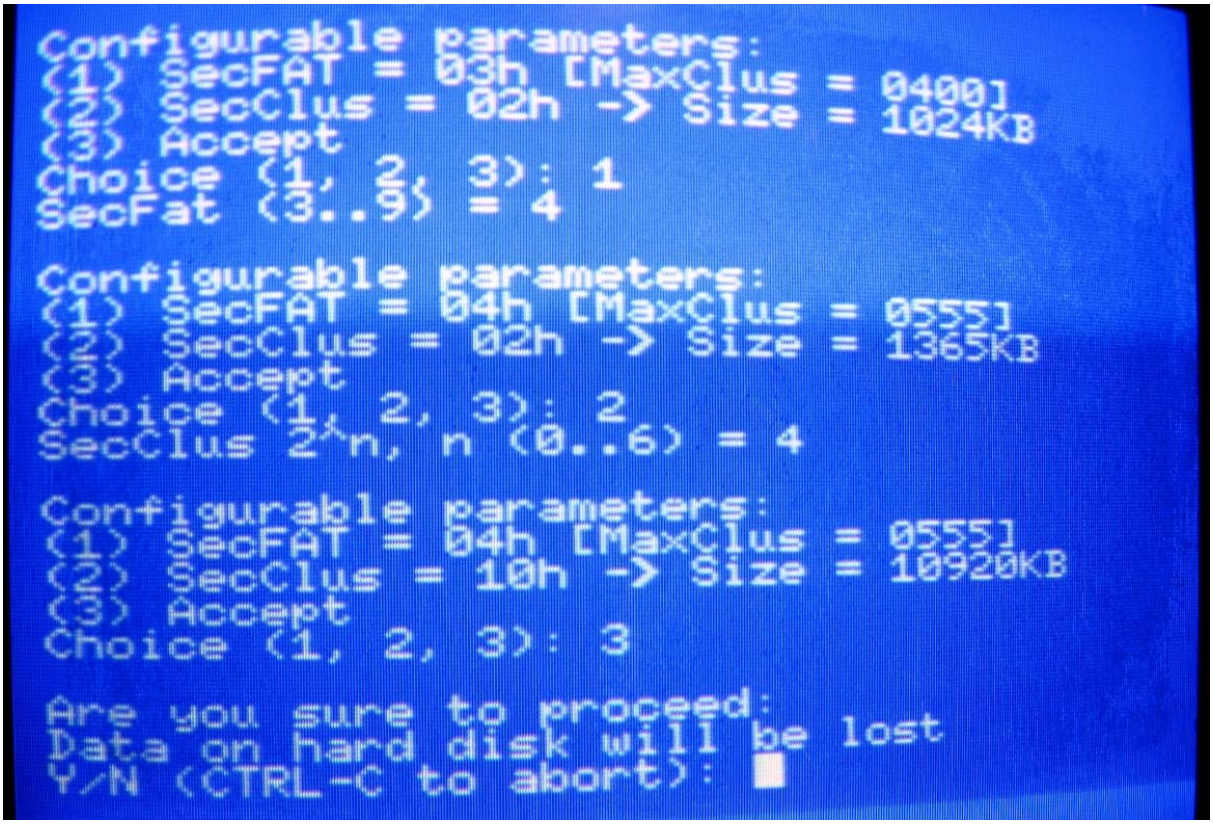

Figure 2.5. Selecting option 3

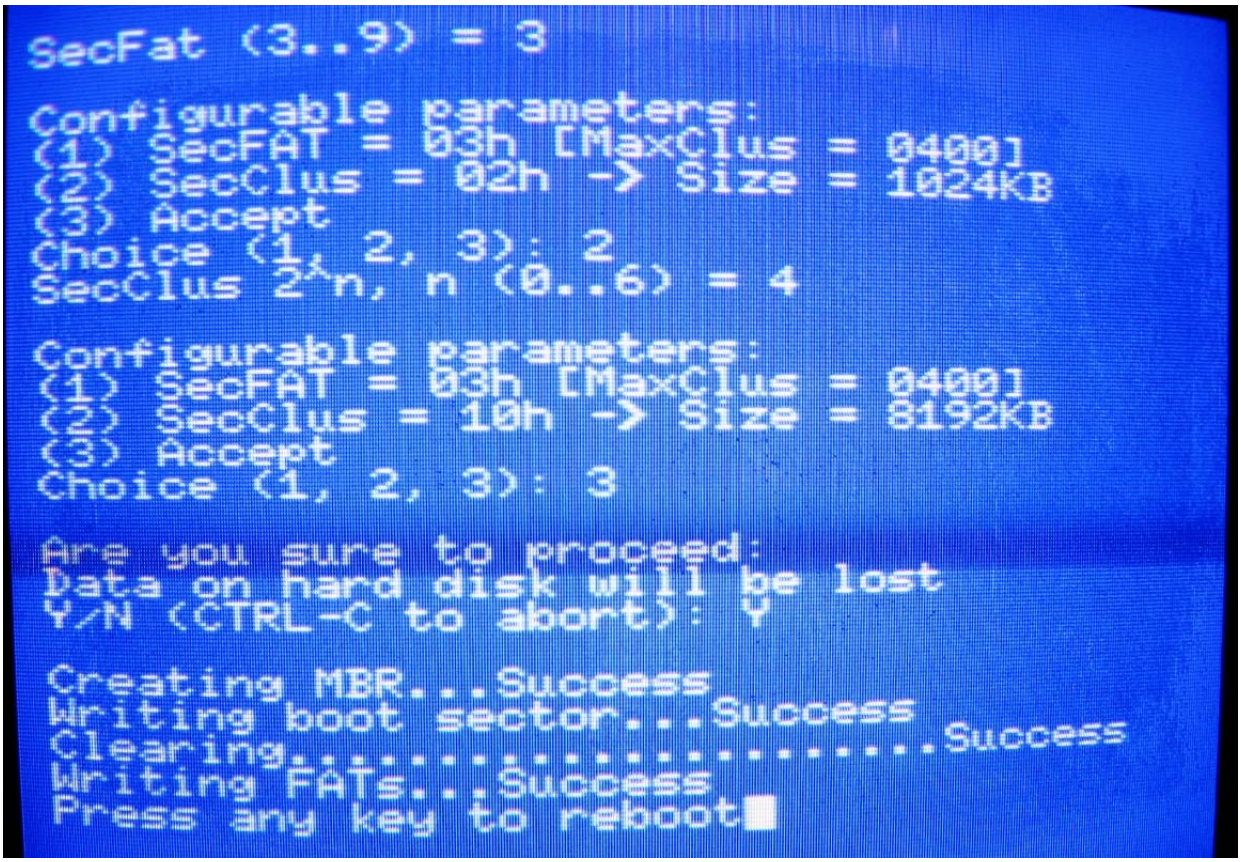

Figure 2.6. GR8HDD finished working – requires reboot

After reboot you may copy system files and application to the newly configured LVOL, and use it as any regular storage device addressing it by the respective drive letter (fig. 2.7).

Limitations: initial release of GR8HDD (a) may only configure one LVOL onto each hard drive's MBR; (b) it does not perform sanity and hard disk size versus LVOL size checks; (c) it only writes LBAs into the MBR LVOL configuration, and does not write corresponding C/H/S values of start and end of partition, (d) it can only completely overwrite current structure (no disk fixing available); (e) it does not write PC bootstrap code into MBR (PC will not be able to boot from this hard drive), however hard drive is expected to be readable if attached to the PC's IDE controller.

Figure 2.7. B: is a first LVOL of hard disk 0 (A: is a floppy drive)

3. Storage subsystem design

Subsystem is based on several components (fig. 3.1), which provide specific API for developers to extend its functionality. Microcontroller with multi-stage architecture was introduced in order to have reliable access to the media, and have unified entry point to any attached storage device. Without MC GR8BIT, and any other MSX-compatible computer, is unable to reliably control floppy disk controller data flow at 500KBit/s speed due to insufficient CPU speed at 3.58MHz system clock.

If you want microcontroller to have more functionality for floppy disk controller, you develop modifications and add-ons to the microcontroller microcode. As an example, it is not really required to have IDE disk drive connected to the 40-pin connector – you may design and build adapter to connect any device and alter microcontroller functionality appropriately.

If you want to change API between GR8BIT CPU and microcontroller, you will need first have this API defined from MC side, and then implement CPU-side routines modifying GR8BIT storage ROM.

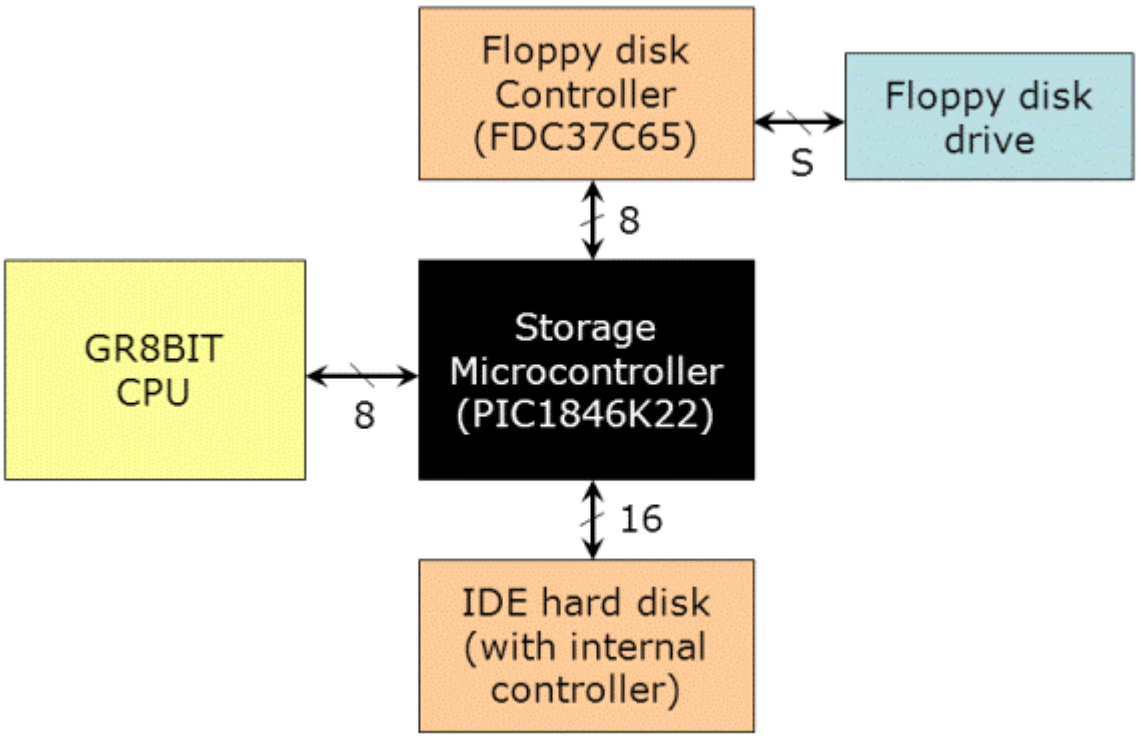

Figure 3.1. Architecture of the GR8BIT storage subsystem

Floppy disk controller was initially designed to work in PIO mode; unfortunately all the attempts to make it work reliable in PIO mode did not succeed. We decided to implement DMA transfers, and this way of interfacing succeeded. In order for your GR8BIT storage hardware to support DMA floppy disk controller interface and current release of microcode, you will need to add 4 (four) air-wires connecting IRQ, DMA, $\overline{\text{DACK}}$ and TC pins of FDC to the specified pins of port B of MC (fig. 3.2 and 3.3).

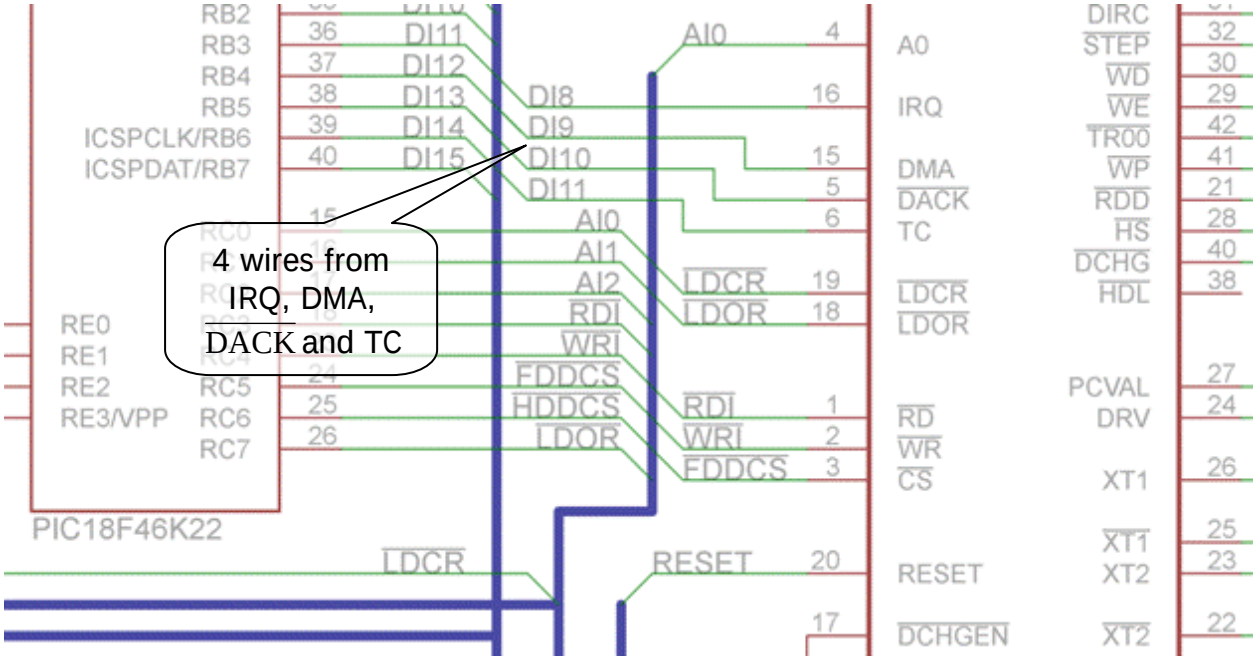

Figure 3.2. Amended circuit diagram

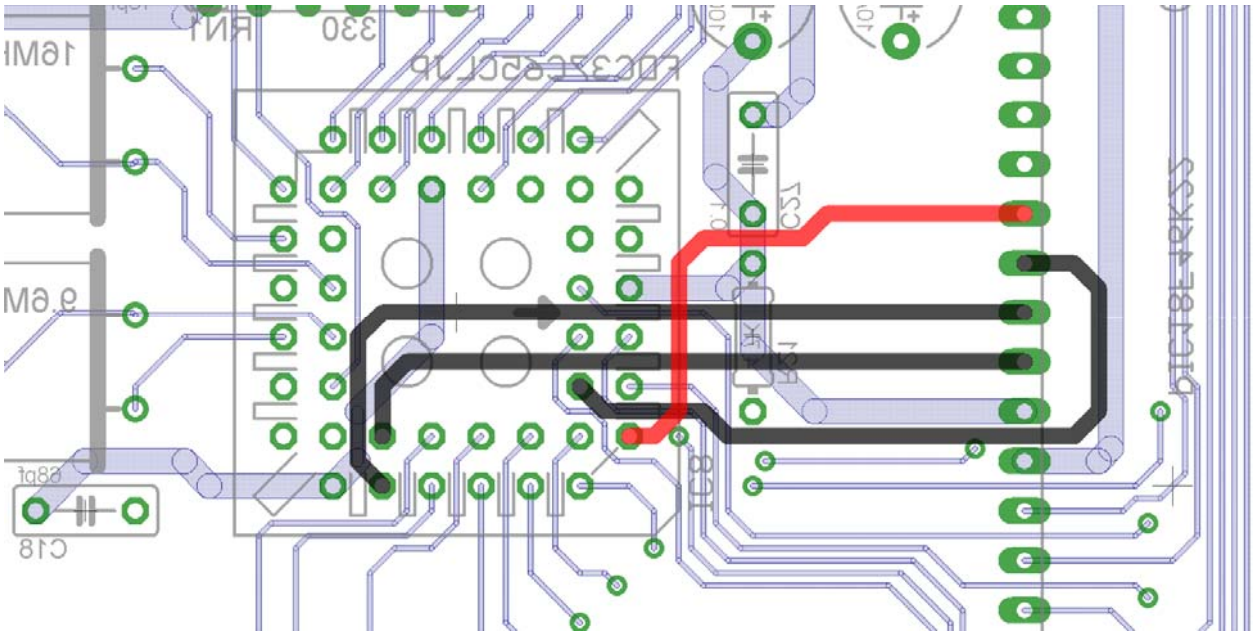

Figure 3.3. Air-wires to solder (view from PCB's solder side)

Unlike initially chosen PIC16F1939, new microcontroller PIC18F46K22 (which is a direct replacement for former) is twice faster and is able to serve GR8BIT CPU requests without additional wait stated introduced by slowdown circuit.

PIC18F46K22 has 3896 bytes of SRAM, and if API requests it to read more than one sector (DSKIO command), it can read up to 7 sectors per one call. After the read operation, MC returns number of sectors actually read for GR8BIT CPU to read valid user data from the MC's data buffer. Such a technique, coupled with DMA access method, dramatically increased floppy disk transfers speed relative to previous implementations.

IDE interface uses PIO transfers Mode 0 only to maintain compatibility with older ATA/IDE hard disk drives. If you wish you can change timing of the IDE bus modifying MC's related microcode to speed up HDD data I/O port access.

4. Programming GR8BIT storage subsystem

4.1. Application API

Application API is used by the GR8BIT and other compatible application programs to access resources attached to the GR8BIT storage subsystem; they will use standard system (BASIC or MSX-DOS) calls. Disk subsystem calls (such as INIHRD, DSKIO, DRIVES, DSKCHG, GETDPB) are not expected to be used by application software (may be used by system software).

The minor, but important change was introduced to the DSKIO system call to support 23-bit sector numbers: if bit 7 of register C (media descriptor) is reset, then remaining 7 bits are treated as upper sector number. With such a mechanism system and application software can address volumes with up to 8,388,608 sectors (4Gbytes). Note that this 23-bit sector address is a sector offset relative to the base of the LVOL, which is always 32-bit. Special thanks to Jan Wilmans for proposing this enhancement.

There's an important feature introduced in this release of storage subsystem software: GR8BIT will flash alternate screen color if application or system software tries to load data onto the stack memory location, with DSKIO returning *Write Protected* error for read operation. This functionality is very useful in DOS1 mode in multiple disk drive configurations when significant portion of memory is used by the disk buffers and programs may not fit into available free memory.

4.2. System level API

System API is used by the GR8BIT storage subsystem ROM and system utilities to control storage microcontroller (MC). Following is the list of the commands available for the system software (table 4.1). Detailed explanation of the API and CPU-microcontroller data exchange is provided in the next section.

Table 4.1. List of the available MC commands

Cmd	Name	Purpose
00	INIT	Initialize floppy drive system. Enumerates physical floppy drives, recalibrates them
01	DSKIO-R	Read of a set of sectors to the MC buffer, maximal number of sectors per call: 7 (limited by the size of internal MC's SRAM). Logical sector access (16- or 23-bit)
02	DSKIO-W	Write of a set of sectors from the MC buffer, maximal number of sectors per call: 7 (limited by the size of internal MC's SRAM). Logical sector access (16- or 23-bit)
03	DSKCHG	Get floppy drive's disk change status
04	FMTDSK	Format floppy disk. HDD format is to be performed using utility not embedded into the disk driver
05	DRIVES	Get number of physical floppy drives and currently selected drive
06	VERSION	Get MC's identification string
07	SOFTWARE	Get a chunk of the system software (MSXDOS.SYS, MSXDOS2.SYS, COMMAND.COM, COMMAND2.SYS)

Table 4.1 continued

Cmd	Name	Purpose
08	MAKEDPB	Construct DPB off the boot sector in the MC's buffer. Constructed DPB is located in MC's buffer at the offset 0x80, size = 0x12
09	HDDSTAT	Get contents of the specified HDD unit's status and error register
10	HDDID	Get HDD identification information (Identify command). Either full 512 bytes set, or selected information
11	BUFACC-R	Read from MC buffer
12	BUFACC-W	Write to MC buffer
AA	ECHO	Tests whenever MC is present and/or available within the system

In the following section **Port 0** is I/O port at address 0xA4, and **Port 1** is I/O port at address 0xA5. The following operations read or write to these ports:

	Port 0		Port 1	
Read	DB A4:	IN A,(0A4h)	DB A5:	IN A,(0A5h)
Write	D3 A4:	OUT (0A4h),A	D3 A5:	OUT (0A5h),A

4.2.1. ECHO command

This command is used to test if MC is ready to accept commands. CPU sends 0xAA to the MC, and MC replies 0xAA. Next byte read after 0xAA should be 0xFF.

Type	Description	Byte	Port	Host R/W
Command (microcontroller)	ECHO Check MC ready	0xAA	Port 0	W
	Response	0xAA	Port 1	R

4.2.2. INIT command

Initializes floppy subsystem and its variables, checks for floppy drive existence, performs their recalibration. This command is issued during storage subsystem initialization for both floppy drives, and before diskette format for specified drive.

Type	Description	Byte	Port	Host R/W
Command (floppy)	INIT Initialize floppy disk subsystem	0x00	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Reset confirmation code	0xEC	Port 1	W
	Command is being executed – controller and drives initialization			
	Completion status read	0x80	Port 1	R

4.2.3. DSKIO commands

This command reads logical sector off the specific media. Physical device is identified by the *Device identifier* parameter. Maximal number of contiguous sectors MC can read or write is 7 (512 bytes * 7 = 3584 bytes). Reads or writes are performed to/from MC's buffer, thus programmer should successfully load valid data using BUFACC-W command before writing, and get valid data from the buffer using BUFACC-R command after reading. C/H/S values for floppy drives and hard disks are calculated by the MC basing on the values of *sector per track* and *heads* supplied.

Type	Description	Byte (exm)	Port	Host R/W
Command	DSKIO* Sector read (write) with retries	0x01 (0x02)	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Device identifier (bit 6: 0=FDD, 1=HDD), and drive/LVOL ID (bit 0)	0x00	Port 1	W
	Sector** # byte 0 (LSB)	0x00	Port 1	W
	Sector** # byte 1	0x00	Port 1	W
	Sector** # byte 2	0x00	Port 1	W
	Sector** # byte 3 (MSB)	0x00	Port 1	W
	Sectors per track	0x12	Port 1	W
	Number of heads	0x02	Port 1	W
	Number of sectors to process	0x07 (maximum)	Port 1	W
	Command execution phase (read or write), filling MC buffer			
	Result phase flag	0xAF	Port 1	R
	Number of sectors successfully processed	0x01 (remaining 6 were not)	Port 1	R
	Completion status read	See table 4.2	Port 1	R

* Be very careful programming HDD write operations – user will not be able to ***eject*** hard disk like it can be done with floppy disks. You may use LVOL write protection bit to write-protect hard disk volumes
** Relative to the LVOL base in LVOL table for logical volumes, absolute for floppy disk devices.

4.2.4. DSKCHG command

This command reads status of the floppy disk change line off the specified FDD. It returns 0xFF is media was changed since last call, or 0x01 if media was not changed. Calling DSKCHG resets disk change flag.

Type	Description	Byte (exm)	Port	Host R/W
Command (floppy)	DSKCHG Check diskette change status	0x03	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Drive# (bit 0)	0x00 (d0)	Port 1	W
	Command execution phase			
	Result phase flag	0xAF	Port 1	R
	Disk change status read	0xFF (chg) 0x01 (n-chg)	Port 1	R

4.2.5. DSKFMT command

DSKFMT command formats floppy disk. It does not partition or format HDD – please use additional utility GR8HDD to partition and format HDD, or use your PC (ensure it creates FAT12 partition type [means max 32MB per partition]).

Type	Description	Byte (exm)	Port	Host R/W
Command	DSKFMT Format floppy media	0x04	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Drive# (bit 0)	0x00 (d0)	Port 1	W
	Format parameter	See table below	Port 1	W
	Confirmation code	0xEC	Port 1	W
	Format command start flag	0xAE	Port 1	R
	Drive initialization, disk format, and boot/FAT/directory sectors initialization			
	Result phase flag	0xAF	Port 1	R
	Completion status read	See table 4.2	Port 1	R

Format parameter definition

Bit	Description
2	0 = DOS1-type boot record 1 = DOS2-type boot record
1..0	00 = Standard 1.44 (9 sectors per FAT copy, 18 sectors per track) 01 = Compatible 1.44 (3 sectors per FAT copy, 18 sectors per track) 10 = Legacy 720 (3 sectors per FAT copy, 9 sectors per track) 11 = Illegal

4.2.6. DRIVES command

Returns floppy subsystem configuration: number of physical floppy disk drives connected, and last accessed (current) drive number. Current drive number is important for handling media insertion actions in phantom drive configurations.

Type	Description	Byte (exm)	Port	Host R/W
Command (system)	DRIVES Get floppy configuration	0x05	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Number of drives connected	0x01 (1drv)	Port 1	R
	Last accessed drive number (used for phantom drive configurations)	0x00 (drv0)	Port 1	R

4.2.7. VERSION command

This command returns MC's identification string, terminated with 0xFF. All the characters returned as ASCII ones, except terminating character.

Type	Description	Byte (exm)	Port	Host R/W
Command (microcontroller)	VERSION Get version of MC firmware	0x06	Port 0	W
	Command accept flag	0xAB	Port 1	R
	MC firmware ID string (read in loop)	Terminated with 0xFF	Port 1	R

4.2.8. SOFTWARE command

MC has 64Kbytes of the internal flash ROM, and it keeps copies of original MSXDOS.SYS (v1.03), COMMAND.COM (v.1.11), MSXDOS2.SYS and COMMAND2.COM (v2.20). Images of this software can be downloaded using SOFTWARE command in 512-byte chunks. By default these images are not designed to be bootable off the MC.

Chunk #0 contains package identification information: file name (11 bytes), 0x00 (initial extent #), size in bytes (2 bytes) and remaining fill bytes up to 512 bytes chunk size. Maximal size of the package (memory space which can be addressed with chunk ID) is 2^6 * 512bytes = 32Kbytes. It is practically possible to read contents of MC's flash ROM beyond the last valid chunk of the package. Package image starts at chunk #1. Bytes within last chunk beyond package size are not invalid.

Type	Description	Byte (exm)	Port	Host R/W
Command	SOFTWARE Get chunk of the system SW	0x07	Port 0	W
512 times	Command accept flag	0xAB	Port 1	R
	Package ID and chunk ID	See table below	Port 1	W
	Data exchange is required flag	0xAE	Port 1	R
	Data read from flash ROM	Data byte	Port 1	R
	Sending CRC for checking	8-bit CRC	Port 1	W
	Result phase flag	0xAF	Port 1	R
	Completion status read	0xCC (success) 0x00 (CRC err)	Port 1	R

SOFTWARE command parameter format

Bit	Description
7..6	00 Return 512 byte chunk of data from image of MSXDOS.SYS 01 Return 512 byte chunk of data from image of MSXDOS2.SYS 10 Return 512 byte chunk of data from image of COMMAND.COM 11 Return 512 byte chunk of data from image of COMMAND2.COM
5..0	Data chunk number (in 512 byte increments, thus maximal size of the software package can be 32KBytes)

4.2.9. MAKEDPB command

Before issuing MAKEDPB command, host should ensure that there's valid relevant boot sector image in the MC's buffer at the offset 0x000 (either by reading it off the media, or writing to MC's buffer). MAKEDPB constructs DPB structure at the buffer offset 0x80. If there's an issue with source data in the buffer (for example, bytes per sector not equal to 512, size of FAT is over 9 sectors etc.), GETDPB returns 0x00. The one exception is allowed – number of the directory entries, if source data has it over 254, number of directory entries in DPB is truncated to 254, and 0xCF code is returned.

After successful completion of the construction of DPB, it can be read by the host using BUFACC-R command directly to the DPB host buffer.

Type	Description	Byte (exm)	Port	Host R/W
Command	MAKEDPB Make DPB structure out of the boot sector loaded at MC's buffer offset 0x00	0x08	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Result phase flag	0xAF	Port 1	R
	Completion status read	0x00 error 0xCC success 0xCF DIRENT	Port 1	R

4.2.10. HDDSTAT command

This command allows host to identify the status of the respective HDD. It supplies status register, identifying readiness of the drive, and error register, which contains relevant information in case of status register's bit 0 ERR is set.

Type	Description	Byte (exm)	Port	Host R/W
Command (hard disk)	HDDSTAT Get HDD status information	0x09	Port 0	W
	Command accept flag	0xAB	Port 1	R
	HDD ID #	0x00 or 0x01	Port 1	W
	HDD status register	Byte	Port 1	R
	HDD error register	Byte	Port1	R

4.2.11. HDDID command

Host uses this command to identify physical configuration of the hard disk, as well as get HDD model string. It may operate in two modes according to the *data type to return* parameter – it either returns full set of inquiry data (512 bytes), or formatted defined fields. Please note that if data type bit is 0, 512 bytes transfer is finalized with transfer and checking of CRC (as it is performed with DSKIO command), but if this bit is 1 then no CRC checking is performed.

Type	Description	Byte (exm)	Port	Host R/W
Command (hard disk)	HDDID Get hard disk drive ID information	0x10	Port 0	W
	Command accept flag	0xAB	Port 1	R
	HDD ID and data type to return	Bit 7: data type Bit 0: HDD #	Port 1	W
	• If data type bit is reset, command jumps to the Result phase (0xAF), and HDD ID data (512 bytes) can be obtained by reading MC's buffer			
	• If data type bit is 1, return the following information:			
	Data exchange is required flag	0xAE	Port 1	R
	Number of cylinders LOW	Byte LOW	Port 1	R
	Number of cylinders HIGH	Byte HIGH	Port 1	R
	Number of heads	Byte	Port 1	R
	Number of sectors per track	Byte	Port 1	R
	HDD model ID string (read in loop)	Terminates with 0x00	Port 1	R
	HDD capabilities byte	Bit 0: DMA suprt Bit 1: LBA suprt	Port 1	R
	Result phase flag	0xAF	Port 1	R
	Completion status read	See table 4.2	Port 1	R

4.2.12. BUFACC commands

BUFACC command is used to access MC's internal buffer. Valid buffer starts at 0x000 and ends at 0xE00 (3584 bytes), but practically programmer can read and write above this space, altering or damaging MC's operations. In the example in the table below host reads (0x11) 18 bytes from MC buffer location 0x080 (e.g. DPB image after successful DPB construction by MAKEDPB command).

Type	Description	Byte (exm)	Port	Host R/W
Command	BUFACC Read from (write to) MC buffer	0x11 (0x12)	Port 0	W
	Command accept flag	0xAB	Port 1	R
	Byte count, LOW	0x12	Port 1	W
	Byte count, HIGH	0x00	Port 1	W
	Buffer offset, LOW	0x80	Port 1	W
	Buffer offset, HIGH	0x00	Port 1	W
	Data ready flag	0xAE	Port 1	R
Byte count	Read data bytes	Data bytes	Port 1	R/W
	8-bit CRC check byte	0xXX	Port 1	W
	Result phase flag	0xAF	Port 1	R
	Completion status read	See table 4.2	Port 1	R

4.2.13. Completion status (DSKIO, DSKFMT and HDDID)

Table 4.2. Completion status bytes for commands DSKIO, DSKFMT and HDDIO

Value	Description	Comments
0x00	Write protected	GR8BIT ROM will return "Write protected" for read operation if there's insufficient memory to read data to
0x02	Not ready	Returned when microcontroller is unresponsive
0x04	CRC error	
0x06	Seek error	
0x08	Record not found	
0x0A	Write fault	
0x0C	No diskette	No host retries to be performed in this case
0xCC	Success	You can test bit 7 of the return code (1=success)

4.3. Disk driver data structures

After initialization GR8BIT disk driver forms data structure in its work area (table 4.3, this work area is allocated by the disk subsystem kernel, and its address can be obtained using function GWRKHL). Size of the structure is 85 bytes. If GR8BIT disk kernel appears to be the first disk ROM to initialize, the byte at the offset +0 of this work area is expected to be located at address 0F17Ah, and last byte of this work area is expected to be located at address 0F1C8h.

Table 4.3. GR8BIT storage subsystem driver data

Offset	Size	Type and description
-2	2	Custom XFER routine call pointer
+0	1	Temporary argument storage
Hardware configuration		
+1	1	Bit 7: set if Compatible floppy disk mode is requested Bits 3..2: number of hard disks detected (0, 1 or 2) Bits 1..0: Number of floppy disk drives configured (0, 1 or 2)
+2	1	FDD 0 sectors per track
	1	FDD 0 heads
+4	1	FDD 1 sectors per track
	1	FDD 1 heads
+6	2	HDD 0 cylinders
	1	HDD 0 sectors per track
	1	HDD 0 heads
+10	2	HDD 1 cylinders
	1	HDD 1 sectors per track
	1	HDD 1 heads
LVOLs configuration		
+14	1	Number of logical volumes configured
		LVOL #1
+15	1	Bit 7: is set if LVOL is software write protected (DSKIO) Bit 0: Hard disk identifier (0/1)
	1	File system type (from MBR, 01h=FAT12)
	4	32-bit volume absolute starting sector (LVOL1 base, starts from 0)
	4	32-bit volume size in sectors
		LVOL #2
+25	1	Bit 7: is set if LVOL is software write protected (DSKIO) Bit 0: Hard disk identifier (0/1)
	1	File system type (from MBR, 01h=FAT12)
	4	32-bit volume absolute starting sector (LVOL2 base, starts from 0)
	4	32-bit volume size in sectors
+35	38	Custom XFER routine code
+73	10	Temporary location for bytes to copy LVOL configuration to. Cheat byte at address 0F1C1h (directly used by some applications to control speed of FDD access) is located within this area
+83	End of table – start of system executable code (0F1C9h)	

End of DN0002 “Storage subsystem v.2.0a: Using and Programming”.