Nextor 2.0 Alpha 2 User Manual

By Konamiman, 8/2011

Index

1. Introduction 2
   1.1. Background 2
   1.2. Goals 3
   1.3. System requirements 3
2. Features 3
   2.2. Standardized and documented driver development system 4
   2.3. Drive to device/partition mapping management 4
   2.4. Drive lock 4
   2.5. Reduced allocation information mode 5
   2.6. Z80 access mode 5
   2.7. Fast STROUT mode 5
   2.8. Extended mapper support routines 6
   2.10. Built-in partitioning tool 6
   2.11. Embedded MSX-DOS 1 6
   2.12. Enhanced Disk BASIC 6
3. Using Nextor 7
   3.1. Installing Nextor 7
   3.2. Booting Nextor 7
   3.3. Managing media changes 9
   3.4. Using the tools 10
   3.5. The built-in partitioning tool 14
   3.6. Extensions to Disk BASIC 15
   3.7. New BASIC error codes 18
4. Change history 19
   4.1. Alpha 2 19
5. Contact 19

1. Introduction

Nextor is an enhanced version of MSX-DOS 2, the disk operating system for MSX computers. It is based on MSX-DOS 2.31, with which it is 100% compatible.

This document provides a description of the features that Nextor adds to MSX-DOS 2 and is intended primarily for end users, but it explains basic concepts that will be useful for developers as well. There are however two other documents aimed specifically at developers: *Nextor 2.0 Alpha 2 Programmers Reference* and *Nextor 2.0 Alpha 2 Driver Development Guide*. The reader of this document is assumed to have experience with MSX-DOS 2 at least at the user level.

WARNING: Nextor is in alpha state. This means that it has not been thoroughly tested, so please backup your valuable data before using it. Also, not all the planned features are implemented yet.

1.1. Background

MSX-DOS is the only official disk operating system for MSX computers. The last version, labeled 2.31, appeared in 1990 accompanying MSX Turbo-R computers.

MSX-DOS was developed in a time in which the only option for massive storage in MSX computers was the floppy disk, and when used as a “floppy disk only operating system” MSX-DOS works indeed just fine. Over the years, however, more modern massive storage options have appeared in the form of amateur-made hardware -- from the early 90’s SCSI and IDE hard disk controllers to today’s multimedia card readers. MSX-DOS has been used to manage these devices, but not without some problems:

- MSX-DOS handles sector numbers as 16 bit entities, and the only filesystem it supports is FAT12. This limits the size of a single filesystem volume to 32MB. Unofficial patches have been developed to add support for the FAT16 filesystem.

- The actual device driver (the code that interacts with the massive storage controller hardware) is embedded within the operating system kernel ROM, present in computers with built-in floppy disk drives and in external floppy disk controllers. There is no officially documented way to embed a custom device driver within the kernel ROM; developers of custom storage controller hardware have to reverse-engineer the kernel code in order to embed a custom driver.

- There is a fixed direct, one-to-one correspondence between the drive letters as seen by the user and the device units exposed by the device driver API. For example, in order to access drive A:, MSX-DOS asks the driver to access its first device; while the second device is queried when accessing drive B:. This is OK for floppy disks, but when using more complex devices that have one or more partitions, it is up to the driver (and usually also to external tools made by the driver developer) to manage the drive to device and partition assignment.

- Managing non-block devices (such as CD-ROMs) is extremely difficult, as it implies a hard work of reverse-engineering on the kernel code.

1.2. Goals

The primary goal of Nextor is to solve the aforementioned problems, by using MSX-DOS 2 as the basis for implementing the features that are needed for a MSX computer equipped with 21th century storage devices. More specifically, the main goals that Nextor development efforts aim to are:

- Provide native support for the FAT16 filesystem.
- Provide a standardized, well documented system for developing custom storage device drivers and either embedding them within the OS kernel ROM or loading them dynamically from RAM.
- Provide a device-based driver system (in contrast with the MSX-DOS drive-based system), so that the driver developer must only worry about enumerating and accessing storage devices and it is the operating system who manages the device- and partition-to-drive assignment.
- Provide support for non-block devices, and for block devices with filesystems other than FAT12/16.

Aside from the main goals, Nextor offers other secondary but also useful new features not present in MSX-DOS. Keep on reading for more details.

1.3. System requirements

Nextor will run on any MSX computer (from MSX1 onwards) having at least 128K of mapped memory. In computers with no mapped memory or having less than 128K in the largest mapper, Nextor will boot in MSX-DOS 1 mode (the DOS prompt is available only if the computer has 64K of RAM).

You can simply burn a standalone version of Nextor (with a dummy device driver) and use it together with storage controllers associated to a MSX-DOS kernel. You will then benefit from features such as the FAT16 filesystem support or the Z80 access mode; note however that the drive to device/partition mapping management feature requires a device driver specifically made for Nextor.

2. Features

This section overviews the features that actually make Nextor an enhanced version of MSX-DOS 2. Operational details are provided in further sections.

2.1. FAT16 filesystem support

Nextor provides built-in support for the FAT16 filesystem. There is no need to install any patch, and it is perfectly possible to boot the system from a FAT16 volume. Volumes up to 2GB in size can be used.

Additionally, standard boot sectors (those present in factory-formatted or partitioned devices, or in devices formatted or partitioned by PC computers) are fully supported as well. In contrast, MSX-DOS 2 treated all disks not formatted by itself as MSX-DOS 1 disks.

2.2. Standardized and documented driver development system

Developers of custom storage controller hardware now have a standardized and well-documented system for developing custom drivers. The driver structure, the details about the routines to be implemented and the “recipe” for embedding the driver within the Nextor kernel are provided so that no more reverse-engineering is needed.

The driver main purpose is to enumerate and access storage devices, but it also contains some extensibility points to add custom BASIC statements (via CALL command), extended BIOS commands, or a timer interrupt service routine.

The following resources are available for Nextor device driver developers:

- The *Nextor 2.0 Alpha 2 Driver Development Guide* document.
- A template driver file, DRIVER.ASM, that can be used as the skeleton for developing custom drivers.
- A command line utility, MKNXEROM, that will do all the work of embedding a device driver within the Nextor kernel ROM. It is provided as a Windows executable (MKNEXROM.EXE) and as a standard C source file (MKNEXROM.C).

Note: at this time it is not possible to invoke the FORMAT command on a drive mapped to a device controlled by a Nextor device-based driver. This will change in a future version of Nextor.

2.3. Drive to device/partition mapping management

Driver developers can choose between two driver styles when developing a Nextor device driver: the *drive-based driver* and the *device-based driver*. The former mimics the MSX-DOS drivers by providing a one-to-one mapping between OS drive letters and driver units; it is still the responsibility of the driver to manage the drive to device and partition mapping. The later is way more interesting.

Device-based drivers do not work in terms of driver units but directly in terms of devices. This means that the driver has no routines like “Read sector X of unit N” but rather “Return information of device X” and “Read raw data from device X”. A device-based driver can handle up to seven devices, and each device can have from one to seven logical units.

The best part is that when using device-based drivers, Nextor will handle the assignment of devices and partitions to drive letters, both automatically (at boot time) and manually (by using a mapping utility that in turn invokes a new function call). The driver developer only needs to implement raw access to the device.

2.4. Drive lock

Nextor allows marking drives as *locked*. When a drive is locked, the kernel code will not ask the driver if the media in the drive has changed; instead, it will assume that the user will never change the media. This is useful when a removable device such as a multimedia card is used as the main storage device, as it prevents the kernel to waste time executing media verification code.

The drive from within NEXTOR.SYS is loaded (both at boot time and via CALL SYSTEM command) will be automatically locked. Other drives can be locked manually by using the supplied tool LOCK.COM.

All drives can be locked, even those belonging to MSX-DOS drivers (including floppy disk drives).

2.5. Reduced allocation information mode

Nextor allows putting drives in *reduced allocation information mode*. When in this mode, the ALLOC function, which returns information about the total and free space available in a drive, will return fake information if necessary, so that the calculated total or free sector count will always fit in 16 bits. In other words, on drives with the reduced allocation information mode active, when the total or free space is greater than 32MB (which is possible in FAT16 volumes), ALLOC will return 32MB.

This feature is intended to avoid compatibility issues with applications that assume the underlying filesystem to be always FAT12 and therefore expect a total or free space information of up to 32MB. However it is also useful for the end user, since calculating the free space on a device (at the end of a DIR command, for example) takes a lot of time on large devices. When the reduced allocation information mode is active for a drive, Nextor will stop calculating free space as soon as it reaches a count of 32MB. Of course, when the space information is less than 32MB, then the actual value will be returned.

The reduced allocation information mode can be activated for all drives, however it makes sense only for drives mapped to a FAT16 filesystem (FAT12 drives will never have a size of more than 32MB).

2.6. Z80 access mode

In MSX Turbo-R computers MSX-DOS 2 always switches to the Z80 CPU when accessing a disk driver. Nextor will never change the CPU when accessing drivers attached to a Nextor kernel, but when accessing drivers attached to a MSX-DOS kernel it is possible to have the *Z80 access mode* active or not. When active, Nextor will switch to Z80 before accessing the driver, as MSX-DOS does.

The Z80 access mode is active by default for all MSX-DOS drivers. It is possible to switch it on or off on a per driver basis (it is not possible to change it for specific drive letters).

2.7. Fast STROUT mode

The MSX-DOS function STROUT prints a string terminated with a “$” character. What this function actually does is to perform one separate call to the CONOUT function (which prints one single character) for every character of the string.

Nextor introduces the *fast STROUT* mode. When this mode is active, the string will be copied to a 512 byte buffer in page 3 and then it will be printed in one single call to the kernel code, which increases the speed of the printing process. The drawback is that the string length is limited to 511 bytes when this mode is active; longer strings will be truncated before being displayed.

2.8. Extended mapper support routines

MSX-DOS 2 provides a set mapper support routines, which allow applications to allocate 16K RAM segments. Nextor maintains the original routines, but provides two new ones that allow allocating a contiguous block of memory (from 1 byte to 16K) inside a given segment.

2.9. Boot keys

MSX-DOS 2 provided the boot keys SHIFT (to disable the disk system) and CTRL (to disable the second floppy disk drive emulation). Nextor provides the following additional keys:

1: Force boot in MSX-DOS 1 mode. This key was already available in MSX Turbo-R computers, but with Nextor it can be used in all MSX computers.

3: Force boot to the BASIC prompt, ignoring any existing boot code (that is, do not try to load and run NEXTOR.SYS, AUTOEXEC.BAS or the code in the boot sector).

2.10. Built-in partitioning tool

The Nextor kernel has a built-in device partitioning tool that can be started by just executing CALL FDISK in the BASIC prompt. It can be used to create partitions of any size between 100KB and 2GB on devices controlled by Nextor device based drivers.

2.11. Embedded MSX-DOS 1

The Nextor kernel contains the MSX-DOS 1 kernel, so that it is possible to boot in this environment when necessary. The Nextor version of MSX-DOS 1 does not provide any additional functionality to users or developers relative to the original version, however it has been modified internally so that it can access devices attached to Nextor drivers.

2.12. Enhanced Disk BASIC

Disk BASIC has been extended with new commands. Also, some of the existing commands have been improved.

3. Using Nextor

This section explains the operational details of Nextor and the associated utilities.

3.1. Installing Nextor

Nextor consists of the following components:

- The Nextor kernel ROM. It must contain a device driver, although a “standalone” version is provided which contains a dummy driver exposing no devices.

- The NEXTOR.SYS file, which is necessary in order to boot in the DOS prompt. This file has the role that MSXDOS2.SYS had in MSX-DOS 2 (in fact, NEXTOR.SYS is just an extended version of MSXDOS2.SYS).

- The COMMAND2.COM file. There is no special command interpreter for Nextor; instead, the same command interpreter of MSX-DOS 2 is used (any version of COMMAND2.COM from 2.20 will do), and new features are handled by using external commands.

In order to boot in the MSX-DOS 1 prompt, you need the usual MSXDOS.SYS and COMMAND.COM files. Also, if you have just the kernel and no NEXTOR.SYS or MSXDOS.SYS files, Nextor will boot in the BASIC prompt (running AUTOEXEC.BAS if present).

Therefore, in order to “install” Nextor, you have two options:

1. Burn a ROM with the appropriate Nextor driver directly in the storage device controller.

2. Burn a standalone version in a flash ROM cartridge, and use it together with your storage device controller in another slot.

Also, you need to copy at least NEXTOR.SYS and COMMAND2.COM to your boot device (it is recommended to have the associated utilities available as well) unless you are happy in the BASIC prompt. More details about the boot procedure follow.

3.2. Booting Nextor

The Nextor booting procedure is similar to the one performed by MSX-DOS 2. However, if Nextor device-based drivers are present, things are a little different since it is necessary to perform a drive to device and partition mapping for all the drives attached to Nextor drivers (if you are not using any Nextor kernel with a device-based driver attached, then the booting procedure is identical to MSX-DOS 2).

At boot time, Nextor will assign **two** drives to each Nextor device-based driver found. Then it will perform a device and partition to drive automatic mapping procedure for each of these drives. The procedure is as follows:

1. Start with the first drive associated to the first Nextor device-based driver found.

2. Start with the first logical unit on the first device available.
3. Scan all the existing primary partitions. If one of them has a FAT12 or FAT16 filesystem which has a file named NEXTOR.DAT in the root directory, map it to the drive. (If the drive has no valid partition table, search for one single filesystem at device sector zero)
4. If step 3 fails, repeat it with the next logical unit available in the device.
5. If step 3 fails for all the logical units on the device, repeat it with the next device.
6. If step 3 fails for all the devices, repeat from step 1 but this time do not search a NEXTOR.DAT file (that is, map the first FAT12 or FAT16 partition available).
7. If step 6 fails (there are no usable partitions available), leave the drive unmapped.
8. Repeat steps 2-7 for the second drive assigned to the driver, but skipping the logical unit already assigned to the first drive (this step is skipped if the first drive was left unmapped).

In short: the first two available devices having a FAT12 or FAT16 partition are assigned to the two drives, but partitions having a NEXTOR.DAT file in the root directory have preference. Note that the contents of the NEXTOR.DAT file is irrelevant, it may even be an empty file (in future versions of Nextor this file is likely to contain some system configuration information). Also, note that only primary partitions are examined in the automatic mapping procedure.

After the automatic mapping is finished, the boot procedure will continue with the following steps:

1. If the “3” key is being pressed, the system displays the BASIC prompt.

2. Otherwise, if the NEXTOR.SYS and COMMAND2.COM files are present in the boot drive (the first drive that is not unmapped), the DOS prompt is shown after AUTOEXEC.BAT is executed (if present).

3. Otherwise, if the boot drive has a MSX-DOS 1 or MSX-DOS 2 boot sector, its boot code is executed as in the case of MSX-DOS: first in the BASIC environment with the carry flag reset, then in the DOS environment with the carry flag set. This will usually cause MSXDOS.SYS and COMMAND.COM to be loaded if present.

4. If the previous step returns, then the BASIC environment is activated, and AUTOEXEC.BAS is executed if present.

Note that step 3 will not be done if the disk has a standard boot sector (not created by MSX-DOS 1 or MSX-DOS 2). The built-in disk partitioning tool will create MSX-DOS 2 boot sectors for all partitions of 32MB or less, and standard boot sectors for larger partitions.

3.2.1. Booting in DOS 1 mode

Nextor kernel can boot in MSX-DOS 1 mode. This will happen if anything of the following conditions is met:

- The computer has no mapped memory, or the largest mapper has less than 128K.

- The boot drive has a MSX-DOS 1 boot sector (boot sectors not having standard format or MSX-DOS 2 format will be considered MSX-DOS 1 boot sectors).

- The “1” key is kept pressed during boot.

The boot procedure for MSX-DOS 1 mode is the same as for the normal (MSX-DOS 2 compatible) mode, with the following differences:

- During the automatic mapping procedure, only the MSX-DOS 1 compatible partitions will be examined. These are FAT12 partitions with three or less sectors per FAT.

- After the automatic mapping procedure, the NEXTOR.SYS and COMMAND2.COM search step is omitted.

Partitions of 16MB or less created with the built-in disk partitioning tool will have three sectors per FAT or less, so these can be used in MSX-DOS 1 mode.

Remember that MSX-DOS 1 can boot the DOS environment (MSXDOS.SYS and COMMAND.COM) if the computer as 64K of RAM. Otherwise, only Disk BASIC can be used.

At this time there is no way to change the drive mapping performed at boot time in MSX-DOS 1 mode. This capability will be added in a future version of Nextor.

Note: when booting directly in the BASIC prompt in MSX-DOS 1 mode, it is no longer necessary to execute “POKE &HF346,1” prior to CALL SYSTEM.

3.3. Managing media changes

Before trying to read or write data from a device, MSX-DOS asks the device driver if the media has changed, in order to update its internal information about the accessed filesystem. Nextor does the same, but if the drive being accessed is mapped to a device-based driver, things get a little trickier because disk partitioning is involved.

When Nextor detects a media change in a drive mapped to a device-based driver, the following procedure is performed:

- The drive is mapped to the first available valid primary partition found on the device. Valid partitions are FAT12 and FAT16 partitions. If the device has no partition table, the drive is mapped to its absolute sector zero.

- All the other drives mapped to other partitions of the same device will be left unmapped.

It is recommended to lock drives mapped to removable media in order to avoid unnecessary media checks. Also, if the driver always returns “unknown change status” for a device, locking is the only way to keep the desired partition mapped to the drive.

The behavior of a media change for a drive mapped to a device-based driver in MSX-DOS 1 mode is undefined at this time; this will change in a future version of Nextor.

3.3.1. Media changes in MSX-DOS 1 mode

When Nextor is running in MSX-DOS 1 mode, media changes are not managed for drives mapped to device-based drivers. For these drives, Nextor will assume that the medium does never change, and therefore will never ask for change status information to the driver; if the medium is changed, it is necessary to manually inform Nextor about the change by issuing a CALL MAPDRV command from the BASIC prompt.

3.4. Using the tools

Nextor is supplied with a set of tools that allow managing the new capabilities available. All of these tools are .COM files intended to be executed from within the DOS prompt.

This section explains how to use these tools. Note however that you can also get help by displaying the desired file directly with the TYPE command (for example: TYPE MAPDRV.COM).

All the tools rely on the new function calls provided by Nextor for its behavior. If you are a developer and want to know more details, please refer to the *Nextor 2.0 Alpha 2 Programmers Reference* document.

Please note that none of these tools work in MSX-DOS 1 mode.

3.4.1. MAPDRV: the drive mapping tool

MAPDRV.COM is a tool that allows mapping a drive letter to a partition on a device controlled by a Nextor device-based driver. It is possible to map any drive, even those initially unmapped or associated to a MSX-DOS driver or a Nextor drive-based driver.

The usage syntax for MAPDRV is:

```
MAPDRV [/L] <drive>: <partition>|<primary>-<extended>|d|u
       [<driver slot>[-<driver subslot>] <device index> [<LUN index>]]
```

The partition number can be specified in two ways:

- As a <*primary*>-<*extended*> pair, where <*primary*> ranges from 1 to 4. If 0 is specified in <*extended*>, then the specified primary partition is mapped. If <*extended*> is 1 or more, then the primary partition is assumed to be extended.

- As a logic partition number. Number 1 is primary partition 1; numbers 2 onwards refers to extended partitions contained within the primary partition 2. For example, logical partition 2 is equivalent to 2-1; logical partition 3 is equivalent to 2-2; etc.

Note that if partitions 2 to 4 are not extended, the only way to map then is to use the syntax <*primary*>-*0*.

If logical partition number 0 is specified, then the drive is mapped to the absolute sector zero of the device.

There are two options for specifying the device where the partition is located:

- Do not supply any parameter after the partition number. In this case, the partition is assumed to be in the same device already mapped to the drive (this works only if the drive is currently mapped to a device-based driver).

- Supply a slot number and a device index. In this case, the slot corresponds to the Nextor kernel that contains the driver that handles the device. Optionally, a logical unit number can be supplied too; default value is 1.

If “d” is specified instead of a partition number, then the drive will be mapped to its default state, which can be one of the following:

- If the drive was unmapped at boot time, then it is left unmapped.
- If at boot time the drive was assigned to a MSX-DOS driver unit, or to a Nextor drive-based unit, then it is mapped to the same unit.
- If at boot time the drive was assigned to a Nextor device-based driver, then an automatic mapping procedure (equal to the one performed at boot time) will be performed. This may or may not result in the drive having the same mapping it had at boot time, depending on the mapping state of the other drives.

If “u” is specified instead of a partition number, then the drive will be left unmapped

The optional parameter “/L” locks the drive immediately after doing the mapping (recommended for removable devices that will not be changed).

3.4.2. DRIVERS: the driver information tool

The DRIVERS.COM utility, which is ran without parameters, displays information about the available MSX-DOS and Nextor drivers. It will display the name and version (for Nextor drivers only), the slot number, and the assigned drives at boot time. MSX-DOS drivers will be identified as “Legacy driver”.

This tool is useful mainly to get the slot numbers of the drivers, in order to supply them as parameters to the other tools.

3.4.3. DEVINFO: the device information tool

The DEVINFO.COM utility displays information about the devices controlled by a given Nextor device-based driver. The information displayed includes the device name and manufacturer (when available), the device index, and the associated logical units types and sizes.

The usage syntax for DEVINFO is:

```
DEVINFO <driver slot>[-<driver subslot>]
```

This tool is useful mainly to get the device and logical unit indexes, in order to supply them as parameters to the MAPDRV tool.

3.4.4. DRVINFO: the drive information tool

The DRVINFO.COM utility, which is ran without parameters, displays information about all the available drive letters (those that are not unmapped). The displayed information includes the associated driver slot and other information that depends on the associated driver type (driver name and version for Nextor drivers; device and logical unit numbers for Nextor device-based drivers; relative unit for MSX-DOS and Nextor drive-based drivers). MSX-DOS drivers are identified as “Legacy driver”.

3.4.5. LOCK: the drive lock and unlock tool

The LOCK.COM utility allows locking and unlocking drive letters. The usage syntax for LOCK is:

```
LOCK [<drive letter>: [ON|OFF]]
```

When ran without parameters, a list of the drive letters currently locked is shown. If only a drive letters is specified, the current lock status for the drive is shown.

When a drive is marked as locked, Nextor will never check the media change status for the drive; instead, the inserted media is assumed to never change. This speeds up media access, but be careful since data corruption may happen if the media is changed while it is locked.

The drive from within NEXTOR.SYS is loaded (both at boot time and via CALL SYSTEM command) will be automatically locked.

Any disk error which is aborted will automatically unlock the involved drive. Other than that, drives will be unlocked only when the LOCK utility is ran with the OFF parameter.

3.4.6. RALLOC: the reduced allocation information mode tool

The RALLOC.COM utility allows activating or deactivating the reduced allocation information mode for a drive. The usage syntax for RALLOC is:

```
RALLOC [<drive letter>: ON|OFF]
```

If no parameters are specified, a list of drives currently in reduced allocation information mode will be shown.

When a drive is in this mode, the ALLOC function, which returns information about the total and free space available in a drive, will return fake information if necessary, so that the calculated total or free sector count will always fit in 16 bits. In other words, on drives with the reduced allocation information mode active, when the total or free space is greater than 32MB (which is possible in FAT16 volumes), ALLOC will return 32MB.

Nextor will never modify the reduced allocation information mode status for a drive automatically, it is the user who always controls this behavior. Disk errors or media changes do not modify the reduced allocation information mode status either.

3.4.7. Z80MODE: the Z80 access mode tool

The Z80MODE.COM utility, which works on MSX Turbo-R computers only, allows activating or deactivating the Z80 access mode for a MSX-DOS driver. The usage syntax for Z80MODE is:

```
Z80MODE <driver slot>[-<driver subslot>]] [ON|OFF]
```

If only a driver slot is specified, the current Z80 access mode state for the driver will be shown. The Z80 access mode is set or unset on a per driver basis (it is not possible to change it for specific drive letters).

The Z80 access mode can be set or unset on MSX-DOS drivers only (Nextor will never switch the current CPU when accessing a Nextor driver). When set, Nextor will switch the current CPU to Z80 prior to performing any operation with the driver. When not set, Nextor will not change the current CPU when accessing the driver.

Whether a given MSX-DOS driver needs the Z80 access mode to be set or not depends on each driver; when in doubt, look at the driver documentation or ask the driver developer if at all possible. Floppy disk drives are likely to need the Z80 access mode to be active.

At boot time Nextor will activate the Z80 access mode for all MSX-DOS drivers. Other than that, Nextor will never automatically change the Z80 access mode for any driver, it is the user who always controls this behavior.

3.4.8. FASTOUT: the fast STROUT mode tool

The FASTOUT.COM utility allows to switch on an off the fast STROUT mode. The usage syntax for FASTOUT is:

```
FASTOUT [ON|OFF]
```

When invoked without parameters, it will show the current status of the FASTOUT mode.

The MSX-DOS function STROUT prints a string terminated with a “$” character. What this function actually does is to perform one separate call to the CONOUT function (which prints one single character) for every character of the string.

When the fast STROUT mode is active, the string will be copied to a 512 byte buffer in page 3 and then it will be printed in one single call to the kernel code, which increases the speed of the printing process. The drawback is that the string length is limited to 511 bytes when this mode is active; longer strings will be truncated (only the first 511 characters will be displayed).

3.4.9. DELALL: the partition quick format tool

The DELALL.COM utility will perform a quick format on the filesystem visible on a given drive letter. The usage syntax for DELALL is:

```
DELALL <drive letter>:
```

What this tool does is to clean the FAT and root directory areas of the filesystem, thus effectively deleting all the information on the filesystem. There is no way to undo the operation; the files will be permanently lost so please use with care.

This tool can be used on any drive, even those attached to MSX-DOS drivers. Note that the drive must be mapped to a valid FAT12 or FAT16 filesystem, otherwise this tool will not work.

3.5. The built-in partitioning tool

The Nextor kernel has an embedded utility for partitioning storage devices attached to Nextor device-based drivers. To start it, just invoke CALL FDISK from the BASIC prompt. It works properly on both 40 columns and 80 columns mode.

The tool has a user interface based on menus, so anyone should be able to use it by just following the indications provided in the screen (when in doubt, look for an indication on what to do next in the lower line of the screen). There are however some points of interest to consider that are not mentioned in the tool itself:

- The tool allows creating up to 256 FAT12 and FAT16 partitions on any block device attached to a Nextor device-based driver. MSX-DOS drivers and Nextor drive-based drivers are not supported.

- With this tool it is not possible to add new partitions to an already partitioned device. All existing partitions must be removed before defining new partitions.

- Partitions from 100KB (the minimum supported partition size) up to 32MB will be FAT12, partitions from 33MB to 2GB (the maximum supported partition size) will be FAT16.

- Partitions of 16MB or less will have three sectors per FAT or less, therefore they can be used in MSX-DOS 1 mode.

- Partitions up to 32MB will have a MSX-DOS 2 boot sector, partitions of 33MB and more will have a standard boot sector.

- If four partitions or less are defined, they will be created as primary partitions. If five partitions or more are defined, then the first one will be primary and the others will be extended partitions contained within the second primary partition. Remember that Nextor only scans primary partitions during the automatic drive to device and partition mapping process.

- To get an optimum cluster size, it is recommended to define the partition sizes as powers of two (that is: 1M, 2M, 4M, 8M, 16M or 32M for FAT12 partitions; 64M, 128M, 256M, 512M, 1G or 2G for FAT16 partitions). If this is not possible, it is better to select the partition size as slightly smaller than the closest power of two than slightly higher (that is, for example 31M is better than 33M).

Remember that Nextor can handle devices with FAT16 partitions and standard boot sectors; if you use a factory-partitioned device of 2GB or less you probably don’t need to partition it, unless you want to create MSX-DOS 1 compatible partitions.

The partitioning tool works in MSX-DOS 1 mode too. Note however that the tool will always allow you to create partitions larger than 16M, which are not compatible with MSX-DOS 1.

3.6. Extensions to Disk BASIC

Nextor adds some new commands to Disk BASIC, mainly to ease the management of devices and partitions from this environment. Also, some of the commands that already existed in MSX-DOS have been extended or improved.

Unless otherwise stated, the Nextor modifications of Disk BASIC are not available in MSX-DOS 1 mode.

3.6.1. The DSKF command

The DSKF command, which tells the free space available on a drive, returns a free cluster count in MSX-DOS. In Nextor the behavior of this command has been changed: now returns a free KB count.

This behavior represents a breaking change relative to MSX-DOS. However, most of the existing programs that use this command do not actually calculate the free space count in KB, displaying the raw cluster count to the user instead. Also, for many years the most popular storage media for MSX computers has been the 2DD floppy disk, in which the cluster size is 1K, so many users were incorrectly assuming that the DSKF command was returning a KB count anyway.

3.6.2. The CALL MAPDRV command

A new command has been added that allows changing the drive to device and partition mapping from the BASIC environment: CALL MAPDRV. This command is available in MSX-DOS 1 mode too.

The CALL MAPDRV syntax is explained below. Some of the parameters are optional, therefore all the possible variations are explained, starting with the most complete (using all parameters) one. Details about the possible values for each parameter are explained later.

- `CALL MAPDRV(<drive>, <partition>, <device>, <slot>)`

Maps the specified drive to the specified partition of the specified device, which is controlled by the driver on the specified slot.

- `CALL MAPDRV(<drive>, <partition>, <device>)`

Maps the specified drive to the specified partition of the specified device. The driver slot is assumed to be the same of the device which contains the partition already mapped to the drive; if the drive is not currently mapped to a device-based driver, an “Invalid device driver” error will be thrown.

- `CALL MAPDRV(<drive>, <partition>)`

Maps the specified drive to the specified partition. The device is assumed to be the same one that contains the partition already mapped to the drive; if the drive is not currently mapped to a device-based driver, an “Invalid device driver” error will be thrown.

- `CALL MAPDRV(<drive>, -1)`

Leaves the specified drive unmapped. Further attempts to access the drive will throw a “Bad drive name” error (“Disk I/O error” in MSX-DOS 1 mode).

- `CALL MAPDRV(<drive>, -2)`
- `CALL MAPDRV(<drive>)`

Maps the specified drive to its default value. If at boot time the drive was unmapped or was mapped to a MSX-DOS driver or to a Nextor drive-based driver, then the drive will be reverted to its original mapping state. Otherwise, and automatic mapping procedure (as explained in Section 3.2) will be performed; this may result or not on the drive having the same mapping it had at boot time, depending on which devices are available and how the other drives are mapped.

The command parameters syntax is as follows:

- <*drive*> is a string with the drive letter followed by a colon. For example `"A:"`.

- <*partition*> is a number in the range 0-255, interpreted as follows:
  - 0: Assumes that the device has no partitions. The drive will be mapped to the absolute sector 0 of the device.
  - 1: First primary partition of the device.
  - 2, 3 or 4: If device partition 2 is extended, the number is interpreted as the first, second or third extended partition, respectively. Otherwise, the number is interpreted as the second, third or fourth primary partition of the device, respectively.
  - 5 or greater: The number is interpreted as the (n-1)th extended partition of the device.

- <*device*> is a device index in the range 1-7. If the device has multiple logical units, use the formula `<device>+16*<logical unit>`. The possible values for the logical unit are 1-7 too (0 is accepted as well and interpreted as 1).

- <*slot*> is a slot number in the range 0-3. If the slot is expanded, use the formula `<main slot>+4*<subslot>`.

In MSX-DOS 1 mode there are some additional restrictions imposed by the Nextor architecture:

- The specified drive must have been mapped to a device-base driver at boot time. It is not possible to change the mapping of a drive that was unmapped or mapped to a MSX-DOS driver or a Nextor drive-based driver at boot time.

- The new mapping information may specify a different partition and/or device, but the driver slot must be the same that was assigned to the drive at boot time. This is not an issue if there is only one Nextor kernel in the system.

Also, please note that in MSX-DOS 1 mode, if you map a drive to an unsupported partition type (a FAT16 partition or a FAT12 partition having more than 3 sectors per FAT) you will always get a “Disk I/O error” when accessing that drive. This does not mean that the device is actually faulty, only that Nextor refuses to access it.

3.6.3. The CALL MAPDRVL command

The CALL MAPDRVL command is identical to the CALL MAPDRV command, except that it will perform a drive lock (see Sections 2.4 and 3.4.5) immediately after changing the drive mapping.

Note that this command is NOT available in MSX-DOS 1 mode, in which the concept of “drive lock” does not exist.

3.6.4. The CALL LOCKDRV command

A new command has been added that allows to lock and unlock drives (see Sections 2.4 and 3.4.5). It is used as follows:

- `CALL LOCKDRV(<drive>)`

Displays the current lock status of the drive.

- `CALL LOCKDRV(<drive>, 0)`

Unlocks the drive.

- `CALL LOCKDRV(<drive>, <any non-0 number>)`

Locks the drive.

<*drive*> is a string with the drive letter followed by a colon. For example `"A:"`.

3.7. New BASIC error codes

The following new BASIC error codes are defined to handle the possible errors of the new BASIC commands. These errors are available in MSX-DOS 1 mode as well for the commands that work in this environment. The numbers in parenthesis are the error codes.

- `Invalid device driver (76)`

This error will be thrown by the CALL MAPDRV command in any of these events:

  - The specified slot number does not contain a Nextor device-based driver.
  - No slot number is specified, but the drive is not currently mapped to a Nextor device-based driver.
  - In MSX-DOS 1 mode, the drive was not originally mapped to a Nextor device-based driver, or was mapped to a different driver.

- `Invalid device or LUN (77)`

This error will be thrown by the CALL MAPDRV command in any of these events:

  - The device and/or LUN with the specified index is not available on the specified or implicit driver.
  - The device and/or LUN with the specified index exists on the specified or implicit driver, but it is not a block device.

- `Invalid partition number (78)`

This error will be thrown by the CALL MAPDRV command if the specified partition does not exist on the specified or implicit device.

- `Partition already in use (79)`

This error will be thrown by the CALL MAPDRV command if you try to map a combination of partition, device and driver that is already mapped on another drive. You can however map the same combination to the same drive again.

4. Change history

This section contains the change history for the different versions of Nextor. Only the changes that are meaningful from the end user point of view are listed; for information on internal changes, please look at the *Nextor 2.0 Alpha 2 Programmers Reference* and *Nextor 2.0 Alpha 2 Driver Development Guide* documents.

4.1. Alpha 2

- Corrected some serious bugs related to partition management. Especially one that prevented the system from booting when a completely blank device (not containing any partition) was attached to a device-based driver.
- The built-in partitioning tool (CALL FDISK) now works in MSX-DOS 1 mode.
- The DSKF command now returns the drive free space in KB.
- Added the CALL MAPDRV, MAPDRVL and LOCKDRV commands.
- The media change behavior in MSX-DOS 1 mode is now defined.
- Trying to map a partition to the same drive again no longer throws a “Partition already in use” error.
- Trying to access an unsupported partition type in MSX-DOS 1 mode now throws a “Disk I/O error”.

5. Contact

You can get the latest version of Nextor and the associated tools at Konamiman’s MSX page:

http://www.konamiman.com

For bug reports or suggestions, please write at:

konamiman@konamiman.com