Nextor 2.0 User Manual

By Konamiman, 10/2014

Index

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 Programmers Reference* and *Nextor 2.0 Driver Development Guide*. The reader of this document is assumed to have experience with MSX-DOS 2 at least at the user level.

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 4GB 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 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 latter 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, see *3.2. Booting Nextor)* and manually (by using a mapping utility that in turn invokes a new function call, see *3.4.1. MAPDRV: the drive mapping tool*, and *3.6.9. The CALL MAPDRV command*). 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.

Drives can be locked by using the supplied tool LOCK.COM or by invoking the CALL LOCKDRV command from within the BASIC prompt. All drives can be locked, even those belonging to MSX-DOS drivers (including floppy disk drives). See *3.4.5. LOCK: the drive lock and unlock tool*, and *3.6.8. The CALL LOCKDRV command*.

2.5. Reduced and zero allocation information mode

Nextor allows setting 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. See *3.4.6. RALLOC: the reduced/zero allocation information mode* tool.

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.

If an environment item named ZALLOC is created with a value –case insensitive- of ON (command `SET ZALLOC=ON` in the command interpreter prompt), the reduced allocation information mode becomes the *zero allocation information mode*. In this case, ALLOC will return a free space of zero for the drives that have this mode active. This is useful because calculating the free space on a device (at the end of a DIR command, for example) may take a somewhat long time on large devices (about 4 seconds in Z80 mode for a SD card, for example); when the zero allocation information mode is active, this time is reduced to zero.

The zero allocation information mode is available since Nextor 2.0.3.

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. See *3.4.7. Z80MODE: the Z80 access mode tool*.

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. See *3.4.8. FASTOUT: the fast STROUT mode tool*.

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. See the *Nextor 2.0 Programmers Reference* for details.

2.9. Boot keys

The boot time configuration of Nextor can be modified by keeping pressed some special keys while the system is booting. These keys and their behavior are:

- **1**: Force boot in MSX-DOS 1 mode. If the computer is a MSX Turbo-R, switches CPU to Z80 mode.
- **2**: Force boot in MSX-DOS 1 mode. If the computer is a MSX Turbo-R, switches CPU to R800-ROM mode. Note that in MSX-DOS 1 mode, the active CPU is never changed when accessing disk drives; this may cause some storage devices to not work properly, especially those mapped to MSX-DOS drivers such as floppy disk drives.
- **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).
- **4**: (for MSX Turbo-R only) Boot in R800-ROM mode, assign the largest mapper found as the primary mapper (instead of the internal mapper), and free the 64K allocated for the R800-DRAM mode. This is useful for using software that requires a huge amount of mapped RAM and can work only with the primary mapper; note however that there is a big penalty in the system speed.
- **CTRL**: Assign only one drive to each Nextor kernel with a device-based driver regardless of the number of devices controlled by the driver. This overrides the normal behavior, in which Nextor assigns one drive per device found (see *3.2. Booting Nextor*). Note that this key will also affect the floppy disk drive, if any (the second drive emulation will be disabled).
- **SHIFT**: Prevent MSX-DOS kernels from booting, but allow Nextor kernels to boot normally. This is useful to disable the internal floppy disk drive in order to get some extra TPA memory, especially in MSX-DOS 1 mode.

- **slot key**: Prevent the Nextor kernel associated to the specified slot key from booting. This is useful when the kernel ROM must be updated so you need to disable it. The associated keys for each slot are:
    - Q for primary slot 1
    - A for primary slot 2
    - QWER for slots 1-0 to 1-3, respectively
    - ASDF for slots 2-0 to 2-3, respectively
    - ZXCV for slots 3-0 to 3-3, respectively
    - In the rare event that you have a Nextor kernel at slot 0, use the keys UIOP for 0-0 to 0-3, respectively.

  Example: if your Nextor kernel is in primary slot 1, press Q to prevent it from booting. If you have it in slot 2-3, press F.

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 4GB on devices controlled by Nextor device based drivers. See *3.5. The built-in partitioning tool*.

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. See *3.2.1. Booting in DOS 1 mode*.

2.12. Enhanced Disk BASIC

Disk BASIC has been extended with new commands. Also, some of the existing commands have been improved. See *3.6. Extensions to Disk BASIC*.

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.

Note: two variants of the NEXTOR.SYS file exist. See *4.3. Reduced NEXTOR.SYS without Japanese error messages.*

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.

3.1.1. Note for Sunrise IDE/CF users

If you want to burn the Nextor kernel ROM in a Sunrise IDE cartridge or in a Sunrise CF reader cartridge, you can’t use the IDEFLOAD.COM program, because it assumes that the file to be burned has a size of 64K but the Nextor kernel is bigger (128K including the driver in current version). Instead, you should use IDEFL128.COM, a modified version of the program that burns 128K files. This program is available at Konamiman’s web page as well (see Section 5).

Also, note that the Sunrise IDE driver supplied with Nextor 2.0 is an experimental driver that has some limitations, more precisely:

- Only LBA mode block devices are supported, there is no support for CHS or ATAPI devices.
- Devices are reported as fixed by the driver. If you are using a CF card reader and you want to change the card(s), you must switch off your computer; hot swapping of cards is not supported.

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 perform a query to all the available device-based drivers to find out how many devices are being controlled by these drivers (actually, how many logical units, but devices will usually have one single logical unit), and will assign to each driver as many drives as devices are controlled by the driver. If CTRL is pressed at boot time, only one drive is assigned to each driver instead (see *2.9. Boot keys*).

For example, assume that you have two Nextor kernels with a device-based driver attached. The kernel in slot 1 controls one device, while the kernel in slot 2 controls three devices. Then the initial drive assignment would be as follows:

A: for driver on slot 1
B:, C:, D: for driver on slot 2
E:, F: for the internal disk drive

If you boot while pressing CTRL, the assignment will be:

A: for driver on slot 1
B: for driver on slot 2
C: for the internal disk drive (which is affected by CTRL as well)

The internal disk drive would not have any drives attached if you pressed SHIFT while booting (see *2.9. Boot keys*).

After all drives have been assigned to drivers, a device and partition to drive automatic mapping procedure will be run for each of these drives. The procedure is repeated for each driver and 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 other drives assigned to the driver (if any), but skipping the logical units already assigned to a drive.
9. Go to the next device-based driver (if any), and repeat from step 2.

In short: available devices having a FAT12 or FAT16 partition are assigned to the available drives in device and logical unit number order, but the first partition having a NEXTOR.DAT file in the root directory has 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 or the “2” key is kept pressed while booting.

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 has 64K of RAM. Otherwise, only Disk BASIC can be used.

On MSX Turbo-R computers, the CPU mode will be switched to Z80 when booting in MSX-DOS 1 mode, unless the 2 key is pressed during boot (see *2.9. Boot keys*).

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 removable 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.

The device driver may also reply “not sure” when asked for device change status. In that case, the procedure is as follows:

- When Nextor first reads the boot sector of a drive mapped to a device on a device-based driver, it calculates a 16 bit checksum of the boot sector contents and stores it together with the rest of the disk parameters.

- When Nextor asks the driver for device change status and the reply is “not sure”, it re-reads the boot sector of the drive and calculates the checksum again. If it matches the previously stored checksum, Nextor assumes that the device has not been changed. Otherwise, it assumes that the device has changed and it performs the same mapping procedure as when the driver reports a device change.

It is recommended to lock drives mapped to removable devices in order to avoid unnecessary media checks (and unnecessary boot sector reads and checksum calculations).

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. The command line 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 a summary of the parameters accepted by each tool by invoking it without parameters; more detailed help is available as well 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 Programmers Reference* document.

Please note that none of these tools work in MSX-DOS 1 mode. However there are equivalent BASIC CALL commands that provide equivalent functionality for most of the tools.

Some of the tools admit a `<driver slot>` parameter. In all of these, number 0 may be specified instead of a slot number, with the meaning of “the primary controller”.

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>|d|u [<device index>-[<LUN index>]
       [<driver slot>[-<driver subslot>]]]
```

Partition number 1 refers to the first primary partition on the device. Partitions 2 to 4 refer to extended partitions 2-1 to 2-4 if partition 2 of the device is extended, otherwise they refer to primary partitions 2 to 4. Partitions 5 onwards always refer to the extended partition 2-(P-1).

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

There are three 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 device index, but not a slot number. In this case, the partition is assumed to be in the specified device, and the device is assumed to be controlled by the kernel on the same slot of the currently mapped device (this works only if the drive is currently mapped to a device-based driver).

- Supply a device index and a slot number. In this case, the slot corresponds to the Nextor kernel that contains the driver that handles the device.

When a device number is supplied, a logical unit number can be supplied too; default value for the logical unit number 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, except that the NEXTOR.DAT file is not searched) 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.

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. Nextor will never automatically lock a drive.

3.4.6. RALLOC: the reduced/zero 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.

If an environment item named ZALLOC exists whose value –case insensitive- is ON (command `SET ZALLOC=ON` in the command interpreter), then the reduced allocation information becomes the *zero allocation information mode* (available since Nextor 2.0.3): the ALLOC function will return a free space of zero for the drives having this mode active. This makes the function to return immediately, which may be useful on very large or very slow devices.

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.

3.4.10. NSYSVER: the NEXTOR.SYS version changer

Some MSX-DOS command line applications are known to check the version number of MSXDOS2.SYS (NEXTOR.SYS in the case of Nextor) and refuse to work if this number is smaller than a certain value, typically 2.20. This is a problem since the current NEXTOR.SYS version number is 2.0.

As a workaround for this issue, starting at version 2.0 beta 2 the NEXTOR.SYS version number returned by the DOSVER function call is stored in RAM and can be changed easily (see the *Nextor 2.0 Programmers Reference* for more details). A command line tool that allows to easily do this change has been created as well, its name is NSYSVER.COM and can be used as follows:

```
NSYSVER <major version number>.<secondary version number>
```

For example: `NSYSVER 2.20`. Note that this will change only the value of the NEXTOR.SYS version number returned by the DOSVER function call; the VER command will still display the real file version number.

Note: the version number change performed by this tool is temporary and it will cease to have effect (that is, the NEXTOR.SYS version number will revert to its real value) when NEXTOR.SYS is reloaded, either because the BASIC prompt is entered and exited via CALL SYSTEM, or because the computer is rebooted.

Note: do not use this tool with NEXTOR.SYS versions older than 2.0 beta 2.

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. Please note that starting the FDISK tool will delete the current BASIC program from memory.

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 4GB (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, 2G or 4G 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 (4GB devices are usually shipped with a FAT32 partition, so you will need to partition it with FDISK anyway).

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.

Some of the new CALL commands take parameters. These commands can be run without parameters in order to get help on how to use them.

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

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.

This modification does not apply to MSX-DOS 1 mode, in this mode the free cluster count is still returned as a cluster count.

The DSKF command will always return the real free space even if the drive has the reduced allocation information mode active. However, if the drive has the *zero* allocation information mode active, then the value returned will be zero.

3.6.2. The DSKI$ and DSKO$ commands

The DSKI$ function and the DSKO$ command, which allow to read and write one disk sector respectively, now accept 32 bit sector numbers, therefore allowing access to any drive sector, not only the first 65536 sectors.

In order to access sectors with numbers over 32767, the sector number must be specified as a single or double precision constant, expression or variable. If a single precision value is specified and the number is so big that one or more of the least significant digits of the number is lost due to truncation, these commands will fail with an “Overflow” error. This is designed this way to prevent inadvertent access to the wrong sector. For example:

```
10 DEFSNG S
20 S=12345678
30 PRINT S ‘Prints “12345700”
40 PRINT DSKI$(0, S) ‘Throws “Overflow”
```

The previous example will work (provided that the sector exists in the device) if line 10 is changed to `DEFDBL S`. Always use double precision variables if you are going to access arbitrary sector numbers in your BASIC code.

An “Overflow” error will be thrown too if the sector number specified does not fit in 32 bits, that is, if it is greater than 4294967295.

In order to maintain compatibility with the MSX-DOS equivalent command, negative sector numbers are accepted (to which 65536 is added to get the real sector number) but only if the sector number can be evaluated as an integer (16 bit) expression. Therefore the following commands are equivalent and will work if the sector exists in the device:

```
PRINT DSKI$(0, 65535)
PRINT DSKI$(0, &HFFFF)
PRINT DSKI$(0, -1)
DEFINT S: S=-1: PRINT DSKI$(0, S)
```

However, the following will throw a “Disk I/O error”:

```
PRINT DSKI$(0, CDBL(-1))
DEFDBL S: S=-1: PRINT DSKI$(0, S)
```

None of this apply to MSX-DOS 1 mode, in this mode only integer (16 bit) sector numbers are accepted.

3.6.3 The CALL NEXTOR command

This command will simply display a list of the new CALL commands that Nextor provides for the BASIC environment.

3.6.4 The CALL CHDRV command

This command changes the current drive and it exists already in MSX-DOS 2 Disk BASIC. However Nextor expands it in two ways:

- The command is now available in MSX-DOS 1 mode as well.
- The drive number can be specified as a number instead of a drive letter (from 1 being A: to 8 being H:). So for example `_CHDRV(3)` is the same as `_CHDRV("C:")`.

3.6.5 The CALL CURDRV command

This command will simply display the current drive.

3.6.6. The CALL DRIVERS command

This command is equivalent to the DRIVERS.COM tool, which 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”.

3.6.7. The CALL DRVINFO command

This command is equivalent to the DRVINFO.COM utility, which 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”.

3.6.8. The CALL LOCKDRV command

This command 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:"`) or a number, being 1 to 8 for drives A: to H:, or 0 for the current drive.

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

3.6.9. The CALL MAPDRV command

This command that allows changing the drive to device and partition mapping from the BASIC environment. It is equivalent to the MAPDRV.COM tool.

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>|0)`

Maps the specified drive to the specified partition of the specified device, which is controlled by the driver on the specified slot. If 0 is specified instead of a slot number, the slot of the primary controller is used.

- `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 will be performed (the procedure is equal to the one performed at boot time except that the NEXTOR.DAT file will not be searched; see Section 3.2 for more details); 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:") or a number, being 1 to 8 for drives A: to H:, or 0 for the current drive.

- <*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>. As a special case, If 0 is specified as the slot number and no subslot number is specified, the slot of the primary controller is used.

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.

3.6.10. 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.

3.6.11. The CALL USR command

The CALL USR command allows the execution of assembler code from BASIC code. It is equivalent to the standard MSX-BASIC DEF USR command and the USR function, but with an added feature: it allows to specify the input values of the Z80 registers for the code to execute, and to read the output values after the execution.

The syntax of the CALL USR command is as follows:

```
CALL USR(<code address> [,<registers address>])
```

<*code address*> is the address of the assembler code to be executed. Value -1 is treated as a special case: `_USR(-1)` will do nothing but will not throw an error. You can use this feature together with the ON ERROR GOTO command to detect the presence of Nextor from within a BASIC program.

<*registers address*> is the address of a 12 byte buffer for the Z80 registers values. If this parameter is specified, the registers will be loaded with the contents of this area before the code is invoked; after the code execution, the reverse process is performed: the buffer is updated with the values hold by the registers. The order of the registers in the buffer is: F, A, C, B, E, D, L, H, IXl, IXh, IYl, IYh.

Here is a simple BASIC program to test the CALL USR command. Change the registers assignment in lines 40-90 and the address of the code to be invoked in line 100 as appropriate to invoke different code (the MSX BIOS itself is a good source of routines to play around).

```
10 ON ERROR GOTO 20: _USR(-1): ON ERROR GOTO 0: GOTO 30
20 PRINT "Nextor not found!": END
30 DEFINT R: DIM R(12)
40 R(0)=&H2100 'AF
50 R(1)=&H3040 'BC
60 R(2)=&H5060 'DE
70 R(3)=&H7080 'HL
80 R(4)=&H90A0 'IX
90 R(5)=&HB0C0 'IY
100 CALL USR(&H00A2, VARPTR(R(0))) 'Prints a "!" (passed in A as &H21)
110 PRINT "AF=&H";HEX$(R(0))
120 PRINT "BC=&H";HEX$(R(1))
130 PRINT "DE=&H";HEX$(R(2))
140 PRINT "HL=&H";HEX$(R(3))
150 PRINT "IX=&H";HEX$(R(4))
160 PRINT "IY=&H";HEX$(R(5))
```

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. Other improvements

4.1. load” in F7

Nextor will force the computer to boot with the `load″` string assigned to the F7 key, even on MSX1 and MSX2 computers, which have `cload″` assigned by default. Note however that any code that invokes the INIFNK BIOS routine will cause the key to be assigned to `cload″` again (you can try it yourself: `_USR(&H3E)`).

4.2. English error messages in kanji mode

If an environment item named ERRLANG is created with a value –case insensitive- of EN (command `SET ERRLANG=EN` in the command interpreter prompt), error messages in the command interpreter will be displayed in English, instead of Japanese, when the kanji mode is active (`CALL KANJI` in the BASIC interpreter). This feature is available since Nextor 2.0.4.

4.3. Reduced NEXTOR.SYS without Japanese error messages

Two variants of the NEXTOR.SYS file are offered. The full variant contains Japanese equivalents for part of the error messages (such as the “reading/writing” part or the “Abort, Retry, Ignore” string), while the reduced variant contains only the English versions. The advantage of the reduced variant is that it is smaller and using it saves 256 bytes of TPA space compared to the full version.

These two variants are offered since NEXTOR.SYS version 2.01 (released together with kernel version 2.0.4). Note that version 2.00 was already reduced, but had a bug that caused garbage to be displayed instead of the proper error messages in kanji mode.

Note that error messages will be displayed in English regardless of the variant used if the ERRLANG environment item exists with value EN (see *4.2. English error messages in kanji mode*).

5. Change history

This section contains the change history for the different versions of Nextor. Changes that affect application or driver development are not listed here; instead, you should look at the *Nextor 2.0 Programmers Reference* and *Nextor 2.0 Driver Development Guide* documents for a list of changes of that type.

5.1. v2.0.4

- Fixed a bug introduced in v2.0.3 that caused the free disk space to be reported incorrectly by one cluster.
- Fixed a bug that caused “File already in use” errors when copying files within the same drive.
- Fixed a bug that caused the computer to crash when copying files from/to a floppy disk drive in DOS 1 mode.
- Fixed a bug that caused the computer to crash when activating the kanji mode (CALL KANJI from BASIC).
- Introduced the ERRLANG environment item for displaying error messages in English when in kanji mode (see *4.2. English error messages in kanji mode*).
- The system variable KANJTABLE (at &HF30F) now is always filled, regardless of the computer’s character set (thus undoing the feature introduced in v2.0.3 of filling it only on Japanese computers). Users of computers with the corrupted error messages problem should use the ERRLANG environment variable (see *4.2. English error messages in kanji mode*).
- NEXTOR.SYS fixed: it was displaying garbage instead of the proper error messages in kanji mode.
- Two variants of NEXTOR.SYS are offered now: with and without Japanese error messages (see *4.3. Reduced NEXTOR.SYS without Japanese error messages*).

5.2. v2.0.3

- The code that calculates the free space on a FAT16 volume (used by the ALLOC and DSPACE function calls) has been rewritten from scratch for performance. Now calculating the free space on a FAT16 volume takes about 1/10 of the time it took in previous versions.
- The code that decides the FAT type of a volume based on the cluster count has been modified. In previous versions, a volume with a cluster count of 4095 or less was always considered to be holding a FAT12 filesystem. Now, a volume with a cluster count between 4085 and 4095 is considered to hold a FAT12 filesystem unless the boot sector contains the string “FAT16” at position 36.
- The maximum cluster count for partitions generated by FDISK has been corrected from 4085 to 4084 in the case of FAT12 partitions, and from 65526 to 65524 in the case of FAT16 partitions.
- The zero allocation information mode has been added (see *2.5. Reduced and zero allocation information mode* and *3.4.6. RALLOC: the reduced/zero allocation information mode tool*).
- The system variable KANJTABLE (at &HF30F) now is filled from the contents of the BIOS only if the computer has a Japanese character set. This solves the corrupted error messages in some computers upgraded to MSX2+.

5.3. v2.0.2

- One of the changes in the v2.0 final ("Fixed a bug in the kernel code that caused a data area to be overwritten with a pointer") has been rolled back. It was not actually a bug, and the "fix" was preventing Nextor from properly booting in DOS 1 mode.

5.4. v2.0.1

- Corrected a bug in the kernel boot code that rendered Nextor unusable in openMSX and could potentially cause problems in real computers as well. The MKNEXROM tool and the *Nextor 2.0 Driver Development Guide* document have been updated as well.

5.5. v2.0 final

- Fixed a bug in NEXTOR.SYS that caused the "Abort, Retry, Ignore" message not to be displayed properly.
- Fixed a bug in the kernel code that caused a data area to be overwritten with a pointer.

5.6. v2.0 Beta 2

- Fixed a bug that caused the RAM disk to not work properly.
- Fixed a bug that caused the APPEND environment variable to have no effect.
- At boot time, the number of drives assigned to device-based drivers is no longer fixed to two. Instead, one drive per device found is assigned (see *3.2. Booting Nextor*)
- Added the boot keys 2, 4, SHIFT, CTRL, and the slot keys (see *2.9. Boot keys*)
- The command line tools DEVINFO.COM, Z80MODE.COM, MAPDRV.COM, and the CALL MAPDRV command, now admit 0 instead of a slot number, with the meaning of "the primary controller" (see *3.4. The command line tools*, and 3.6.9. The CALL MAPDRV command)
- Added the NSYSVER tool (see *3.4.10. NSYSVER: the NEXTOR.SYS version changer*)

5.7. v2.0 Beta 1

- Fixed a bug that caused device sectors to be read more times than necessary, thus seriously hurting device access speed.
- Fixed a bug that caused the computer crash on the first drive access after the mapped device had changed.
- Fixed a bug that caused the computer to freeze at boot time when using an unpartitioned device.
- Fixed a bug that could prevent booting when the Nextor kernel ROM was built using the ASCII8 mapper.
- Fixed a bug in the _LOCKDRV command, it was changing lock status when it should only check it.
- Fixed a bug that crashed the computer when using the FCB related function calls to access the RAM disk.
- Fixed a bug on the drive number handling in DSKI$ and DSKO$.
- Fixed a bug that caused the internal RAM mapper to be configured as the primary mapper on all MSX models, not only on Turbo-R.
- DSKI$ and DSKO$ now admit 32 bit sector numbers.

- Fixed a bug that caused function call _GDRVR to fail when called via the F37Dh hook.
- Added the _CURDRV command.
- Added the _DRVINFO command.
- Added the _DRIVERS command.
- Added the _NEXTOR command.
- Added the _USR command.
- The _CHDRV command now works in DOS 1 mode.
- The _CHDRV, _MAPDRV and _LOCKDRV commands can now use a number to specify the drive (0=default drive, 1=A:, etc)
- The computer boots with load" assigned to the F7 key even in MSX1 and MSX2.
- The drive from which NEXTOR.SYS is loaded is no more automatically locked.
- All the Nextor new CALL commands show help when ran without parameters.
- The order and of the parameters for the MAPDRV.COM has been changed to be the same as the _MAPDRV command.
- Fixed a bug in the DEVINFO.COM tool: the device manufacturer and name were shown swapped.
- Command line tools now show a short usage information when invoked without parameters.
- FDISK allows now to create partitions up to 4GB.
- FDISK default partition size is now 16M (instead of device capacity) in DOS 1 mode.
- FDISK now shows "warning: only partitions of 16M or less can be used in DOS 1 mode" in DOS 1 mode.
- FDISK now shows "please reset" message after partitioning.
- The DEVINFO.COM tool now shows the device size in MB if the size is under 10GB, or in KB if the size is under 10MB.
- Added the boot sector checksum feature for drivers that report a device change status of “not sure”.
- Devices are now reported as being fixed in the experimental Sunrise IDE driver; hot card swapping is not supported when CF cards.

5.8. v2.0 Alpha 2b

- Previous versions assumed that ROM bank 0 was switched at boot time, and the system crashed if any other bank was switched instead. Now Nextor boots properly regardless of which bank is switched at boot time.
- The MKNEXROM.EXE utility has been modified to accommodate the code change explained in the previous point.
- Corrected a bug in the Sunrise IDE driver that prevented it to work with real hardware.
- Corrected a bug that caused the kernel code to allocate 64K RAM for the DRAM mode even on non-TurboR machines, causing Nextor not work on machines with only 128K of mapped RAM. This has been corrected.
- Added the note for Sunrise IDE/CF users (section 3.1.1) in this manual.

5.9. v2.0 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”.

6. 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