

Nextor 2.1 プログラマーズリファレンス

By Konamiman, 2023/11/6 2.0 版翻訳: Mille (2017/4/30) 2.1 改版翻訳: @Lithelia(2025/8/8)

※訳注: 本書の翻訳ベースは「Nextor 2.0 Programmers Reference_JP.pdf」(以後「N2.0PRJ」と略します)及び、以下のリポジトリファイルです。(上記記載の作者名と作成日付は、前者ベース PDF のプロパティより)

<https://github.com/Konamiman/Nextor/blob/v2.1.3/docs/Nextor%202.1%20Programmers%20Reference.md>

目次

1.	はじめに.....	3
2.	既存ファンクションコールの変更	4
2.1.	_STROUT (09h).....	4
2.2.	_ALLOC (1Bh).....	4
2.3.	_RDABS (2Fh) / _WRABS (30h)	5
2.4.	_DPARM (31h).....	6
2.5.	_DEFER (64h).....	8
2.6.	_EXPLAIN (66h).....	9
2.7.	_FORMAT (67h).....	9
2.8.	_DOSVER (6Fh)	11
2.8.1.	Nextor の MSX-DOS 1 モードの検出	13
2.8.2.	各モードでの _DOSVER の戻り値の例	14
3.	新規のファンクションコール.....	15
3.1.	高速 STROUT モードの取得/設定 (_FOUT, 71h).....	15
3.2.	ゼロ終了文字列の表示(_ZSTROUT, 72h)	16
3.3.	ドライブからの物理セクタ読み込み (_RDDRIV, 73h).....	16
3.4.	ドライブへの物理セクタ書き込み (_WRDRV, 74h).....	17
3.5.	割当情報縮小モード状態の取得/設定 (_RALLOC, 75h).....	17
3.6.	ドライブ容量の取得 (_DSPACE, 76h)	18
3.7.	ドライブのロック/ロック解除、ロック状態の取得 (_LOCK, 77h).....	18
3.8.	デバイスドライバ情報の取得 (_GDRV, 78h).....	19
3.9.	ドライブレター情報の取得 (_GDLI, 79h).....	21
3.10.	デバイスパーティション情報の取得 (_GPART, 7Ah).....	22
3.11.	デバイスドライバルーチンの呼び出し (_CDRV, 7Bh).....	25
3.12.	ドライバとデバイスへのドライブレター割り当て (_MAPDRV, 7Ch).....	26
3.13.	ドライバの Z80 アクセスモードの有効/無効 (_Z80MODE, 7Dh).....	28
3.14.	FATドライブ上のクラスタ情報を取得 (_GETCLUS, 7Eh)	29
4.	新しいエラーコード	31
5.	拡張マッパーサポートルーチン	32
5.1.	CALL_MAP: マッパーRAM セグメント内ルーチンの呼び出し (+30h)	32
5.2.	RD_MAP: RAM セグメントから 1 バイト読み取る (+33h).....	33
5.3.	CALL_MAPI: インラインルーチンを使用して、マッパーRAM セグメント内のルーチンを呼び出す (+36h)	33
5.4.	WR_MAP: RAM セグメントに 1 バイト書き込む (+39h).....	34
5.5.	UNAPI RAM ヘルパーの検出手順.....	34
5.6.	重大な変更のお知らせ	35
6.	その他の特徴	36
6.1.	_STROUT (09h) におけるエスケープシーケンス “ESC-Y” の不具合修正	36

6.2.	Nextor のバージョン変更	36
7.	Nextor の内部.....	38
7.1.	ワンタイムブートキー	38
7.2.	ディスクエミュレーションモード	38
7.2.1.	ディスクエミュレーションデータファイル形式	38
7.2.2.	ディスクエミュレーションモードの実行	39
8.	変更履歴	41
8.1.	v2.1.3.....	41
8.2.	v2.1.2.....	41
8.3.	v2.1.1.....	41
8.4.	v2.1.1 beta 2	41
8.5.	v2.1.1 beta 1	41
8.6.	v2.1.0.....	41
8.7.	v2.1.0 RC 1	42
8.8.	v2.1.0 beta 2	42
8.9.	v2.1.0 beta 1	42
9.	問い合わせ	43

1. はじめに

Nextor は MSX 用ディスクオペレーションシステムである MSX-DOS 2 の強化バージョンです。MSX-DOS 2.31 をベースにしており、100%の互換性があります。

本ドキュメントでは開発者視点で、Nextor が MSX-DOS 2 に追加した新機能の解説を行います。基本的には新しいファンクションコールについて解説しますが、その他の有用な情報も含んでいます。Nextor のデバイスドライバー開発については本ドキュメントに含まれていません。デバイスドライバー開発については、"[Newordxtor 2.1 ドライバー開発ガイド](#)"を参照してください。

このドキュメントの読者には一般的な MSX、特に MSX-DOS 2 のアプリケーションを開発した経験がある方を想定しています。特に "MSX2 テクニカル・ハンドブック" の「第 3 部 MSX-DOS」、"[MSX-DOS 2 プログラムインターフェース仕様](#)" と "[MSX-DOS 2 ファンクションコード仕様](#)"、または "MSX-Datapack Volume 3 turboR 版" の「第 2 部 MSX-DOS 2」、に記載されている内容を理解されている方を想定しています。また、事前に "[Nextor 2.1 ユーザーマニュアル](#)" を読んで Nextor についての理解を深めることをお奨めします。

2. 既存ファンクションコールの変更

この章では既存の MSX-DOS 2 のファンクションコールに対する Nextor の変更点を解説します。すべての変更は

_DPARM (31h) の 16bit セクタ番号に関する軽微な問題を除き、互換性を保っています。

Nextor で変更されたファンクションコールについてすべての解説はせず、変更部分についてのみ解説します。変更部分以外の解説については、[“MSX-DOS 2 ファンクションコード仕様”](#)を参照してください。

※訳注:「MSX-DOS 2 ファンクションコード仕様」は、「N2.0PRJ」では“MSX-Datapack Volume 3 turboR 版”の「17 章ファンクションコール」を参照となっていました。

2.1. _STROUT (09h)

高速 STROUT モードが有効になっているとき、表示可能な文字列が 大で 511byte となります。文字列は '\$' (24h) で終了しますが、文字列が 511byte より長いときは、最初の 511byte 分のみ表示されます。高速 STROUT モードはデフォルトでは無効になっており、追加されたファンクションコール_FOUT (71h) で有効にします。

_STROUT	Nextor	MSX-DOS 2
コール手順	DE : 文字列のアドレス	
戻り値	なし	
解説	高速 STROUT モード 無効 : '\$' (24h) で終了 有効 : '\$' または 511byte の短い方で終了	'\$' (ASCII 24h) で終了

2.2. _ALLOC (1Bh)

対象ドライブに対する割当情報縮小 (Reduced Allocation Information) モードが有効になっているとき、クラスタ総数と未使用クラスタ数について、1 クラスタあたりの論理セクタ数を掛け合わせて得られる容量が 32MB を超えるときに、32MB 未満になるように偽の値を返します。割当情報縮小モードはデフォルトではすべてのドライブに対して無効になっており、追加されたファンクションコール_RALLOC (75h) で指定したいドライブに対して有効にします。

さらに Nextor 2.0.3 以降では環境変数 "ZALLOC" に "ON" を設定することにより、割当情報縮小モードは割当情報ゼロモード (Zero Allocation Information) になります。このとき、_ALLOC は割当情報縮小モードに指定されているドライブについて、未使用クラスタ数 0 を返します。

※訳注:「N2.0PRJ」では以下の追加記載がありました。

32MB を超えるドライブにおいて、COMMAND2.COM 2.20 や 2.30 等の FAT16 に対応していないプログラムがドライブの総容量や空き容量を間違えて計算してしまうため、約 32MB と計算するようにします。

_ALLOC	Nextor	MSX-DOS 2
コール手順	E : 論理ドライブ番号 (00h = カレント, 01h = A:など)	
戻り値	A : 1 クラスタあたりのセクタ数 BC : セクタサイズ (常に512=0200h) ※訳注:CD-ROM や MO など、2048 となる媒体もある DE : ディスク上のクラスタの総数 HL : ディスク上の未使用クラスタ数 IX : DPB へのポインタ IY : 最初の FAT セクタへのポインタ	
解説	割当情報縮小モード 無効:通常の戻り値 有効:DE×A、HL×A それぞれについて 値が FFFFh 以下のときは通常の戻り値 FFFFh より大きいときは FFFFh 以下になるまで DE や HL を小さくする 環境変数 ZALLOC = ON のときは HL = 0000h	

2.3. _RDABS (2Fh) / _WRABS (30h)

これらのファンクションコールは FAT12 のドライブに対してのみ有効となります。FAT16 や不明なファイルシステムのドライブに対しては “Not a DOS disk” エラーを返します。

従来の MSX-DOS では FAT12 のみサポートしており、厳密に言うのであれば仕様は変わっていません。しかしながら、MSX-DOS では FAT16 のドライブをアクセスしたときでも FAT12 と認識して動作し、エラーを返しません。Nextor では、これらの関数で FAT16 のドライブは扱えないようにします。これは、CHKDSK や IMPROVE のような、Low Level でのディスクアクセスするプログラムが実行されたときに FAT12 と誤認してデータ破壊を引き起こすことを防ぐためです。

Nextor 対応アプリケーションはこれらのファンクションコールの代わりに、追加されたファンクションコール _RDDRIV (73h) と _WRDRV (74h) を使って下さい。追加されたファンクションコールでは 32bit のセクタ番号を使用し、ファイルシステムのタイプに関係なくディスクのアクセスが可能になります。

_RDABS	Nextor	MSX-DOS 2
コール手順	DE : セクタ番号 L : 物理ドライブ番号 (00h = A:, 01h = B: など) H : 読み出すセクタ数	
戻り値	A : エラーコード (00h = エラーなし)	
解説	ファンクションコールの使い方に変更はないが、 FAT16 ドライブのときはエラーが返る	FAT16 ドライブのときは FAT12 としてアクセスしてしまう

_WRABS	Nextor	MSX-DOS 2
コール手順	DE : セクタ番号 L : 物理ドライブ番号 (00h = A:, 01h = B: など) H : 書き込むセクタ数	
戻り値	A : エラーコード (00h = エラーなし)	
解説	ファンクションコールの使い方に変更はないが、 FAT16 ドライブのときはエラーが返る	FAT16 ドライブのときは FAT12 としてアクセスしてしまう

2.4. _DPARM (31h)

このファンクションではディスクパラメータのオフセット 18h ~ 1Bh に論理セクタの総数を 32bit で保存します。さらに論理セクタの総数が FFFFh を超えるとき、オフセット 09h ~ 0Ah は 0000h となります。

また、オフセット 1Ch にファイルシステムの種類を保存します。

00h : FAT12

01h : FAT16

FFh : その他

注意: 現在、Nextor で扱えるファイルシステムは FAT12 と FAT16 のみです。

_DARM	Nextor	MSX-DOS 2
コール手順	DE : ディスクパラメータ (32byte のバッファ) へのポインタ L : 論理ドライブ番号 (00h = カレントドライブ, 01h = A: など)	
戻り値	A : エラーコード (00h = エラーなし) DE : 保存される	

解説	オフセット 09 ~ 0Ah : 論理セクタの総数 (16bit) ただし、FFFFh を超えるときは 0000h オフセット 18 ~ 1Bh : 論理セクタの総数 (32bit) オフセット 1Ch : ファイルシステムの種類 00h : FAT12, 01h : FAT16, FFh : その他	オフセット 18 ~ 1Fh はシステム予約(常に 00h)
----	---	--------------------------------

ディスクパラメータブロックフォーマット (オフセットは DE+xxh を示す)

オフセット	Nextor	MSX-DOS 2
01 ~ 02h	セクタのサイズ (バイト単位)	
03h	クラスタのサイズ (セクタ単位、2 のn乗で指定)	
04 ~ 05h	予約セクタ数 (ブートセクタで使用するセクタ数)	
06h	FAT の数	
07 ~ 08h	ルートディレクトリエントリの数 (作成可能なエントリの数)	
09 ~ 0Ah	総セクタ数 (16bit、FFFFh を超えるときは 0000h)	総セクタ数 (16bit)
0Bh	メディア ID	
0Ch	FAT サイズ (セクタ単位)	
0D ~ 0Eh	ルートディレクトリの先頭セクタ番号	
0F ~ 10h	最初のデータのセクタ番号	
11 ~ 12h	最大クラスタ番号	
13h	ダーティディスクフラグ(※) (00h 以外で UNDEL 可能)	
14 ~ 17h	ボリューム ID (00 ~ 7Fh で構成、FFFF FFFFh のとき、ボリューム ID なし)	
18 ~ 1Bh	総セクタ数 (32bit)	システム予約 (現在は常に 00h)
1Ch	ファイルシステムの種類 00h = FAT12, 01h = FAT16, FFh = その他	
1Dh ~ 1Fh	システム予約	

※ ダーティディスクフラグは、ディスク中に UNDEL コマンドで復活できるファイルがあることを示すフラグです。したがって、ファイルあるいはディレクトリにクラスタが割り当てられるとこのフラグはリセットされます。

2.5. _DEFER (64h)

ディスクエラーが発生したときにコールされるユーザーのルーチンに渡すパラメータを 32bit セクタ番号対応にします。

MSX-DOS2 でのパラメータ仕様

C : bit 3 - セクタ番号が有効のときにセット

DE : セクタ番号 (C の bit 3 がセットされているとき)

Nextor でのパラメータ仕様

C : bit 3 - セクタ番号が有効で FFFFh以下のときにセット (bit 4 もセットされ、HL=0 となる)

C : bit 4 - セクタ番号が有効のときにセット

HL:DE : セクタ番号 (C の bit 4 がセットされているとき)

_DEFER	Nextor	MSX-DOS 2
コール手順	DE : ディスクエラールーチンのアドレス、0000h で定義解除	
戻り値	A : 00h (エラーは発生しない)	
解説	<p>ファンクションコールの使い方に変更は無く、ユーザールーチンに渡すパラメータが変更されるパラメータ</p> <p>A: エラーの原因となったエラーコード</p> <p>B: 物理ドライブ番号 (00h = A:, 01h = B: など)</p> <p>C: bit 0 - 書き込みでセット bit 1 - 無視の処理が望ましくないときセット bit 2 - オートアボートを指示するときセット bit 3 - セクタ番号が有効で FFFFh 以下のときセット (bit 4 もセット、HL は 0000h となる) bit 4 - セクタ番号が有効のときセット</p> <p>HL:DE : セクタ番号 (32bit) C の b4 がセットされているとき</p> <p>結果</p> <p>A : 00h = システムエラールーチンのコール 01h = アボート 02h = 再試行 03h = 無視</p>	<p>A: エラーの原因となったエラーコード</p> <p>B: 物理ドライブ番号 (00h = A:, 01h = B: など)</p> <p>C: bit 0 - 書き込みでセット bit 1 - 無視の処理が望ましくないときセット bit 2 - オートアボートを指示するときセット bit 3 - セクタ番号が有効のときセット</p> <p>DE : セクタ番号 (16bit) C の b3 がセットされているとき</p> <p>結果</p> <p>A : 00h = システムエラールーチンのコール 01h = アボート 02h = 再試行 03h = 無視</p>

2.6. _EXPLAIN (66h)

環境変数 “ERRLANG” が “EN” に設定されているとき、漢字モードであっても英語のエラーメッセージを返します。この機能は、Nextor 2.0.4 以降で有効です。

_EPLAIN	Nextor	MSX-DOS 2
コール手順	B : 説明すべきエラーコード DE : 64 バイトの文字列バッファへのポインタ	
戻り値	A : 00h B : 00h あるいは変更なし DE : エラーメッセージが入る	
解説	ファンクションコールの使い方に変更はなく、環境変数 ERRLANG = EN のときは漢字モードであっても英語でエラーメッセージを表示	エラーメッセージで表示される言語はエラー発生時の言語モードによる

2.7. _FORMAT (67h)

MSX-DOS 2 ではパラメータ FEh、FFh によって実際はフォーマットせずにブートセクタのみ MSX-DOS 2 用に更新するオプションがあります。この機能は FIXDISK で MSX-DOS 1 用ディスクを MSX-DOS 2 用ディスクに更新するために使われます。

Nextor ではさらに 3 つのパラメータを追加します。

- FDh : FAT12 もしくは FAT16 でブートセクタに有効なディスクパラメータが設定されている場合、ブートセクタ情報を Nextor の標準ブートセクタ情報に更新します。

それ以外のディスクでは “Not a DOS disk” エラーとなります。

- ① ブートセクタのオフセット 03h ~ 0Ah (OEM Name) に “NEXTOR20” をセット
- ② ブートセクタのオフセット 27h にボリューム ID (4byte) 、2Bh にボリューム名 (11byte) 、36h にファイルシステムタイプ (8byte の文字列 “FAT12” または “FAT16”、残りは 20h) をセットし、上記以外のディスクパラメータは変更しません。(※訳注: 原文は “byte 29h” となっているが、27h が正しい。ファイルシステムタイプの長さは「N2.0PRJ」では 32byte だが、8byte が正しい)

Nextor 標準の 1C ~ 3Dh は一般的な拡張ブロックと同じ仕様となっており、既に拡張ブロックが存在しているときは、OEM Name のセットのみを行い、既存のボリューム名 (訳注: 及びボリューム ID) は維持されます。このオプションは他のシステムでフォーマットされたディスクを Nextor の標準ブートセクタ情報に修正したいときに使います。

- FCh : FAT12 のディスクに対しては、「VOL_ID」文字列を含む MSX-DOS 2 の標準ブートセクタ情報に更新します。

FAT16 のディスクに対しては FDh と同じです。ディスクを MSX-DOS 2 で利用する場合に有用です。

- FBh : FAT とルートディレクトリの情報のみ消去するクイックフォーマットを行います。

FAT12 もしくは FAT16 のディスクでないときは “Not a DOS disk” エラーとなります。

オプション 01～09h でフォーマットしたときは MSX-DOS 2 の標準ブートセクタ情報となります。

_FORMAT	Nextor	MSX-DOS 2
コール手順	B：論理ドライブ番号 (00h=カレント, 01h=A:) A：00h 選択文字列を返す 01～09h この選択でフォーマットする 0A～FAh 不正 FBh クイックフォーマット 以下はブートセクタの更新のみ実施 FCh FAT12：MSX-DOS 2 標準で更新 FAT16：Nextor 標準で更新 FDh Nextor 標準で更新 FEh MSX-DOS 標準で更新 オフセット 1Eh 以降は更新しない FFh MSX-DOS 2 標準で更新 HL：バッファへのポインタ (A=01～09h のとき) DE：バッファのサイズ (A=01～09h のとき)	B：論理ドライブ番号 (00h=カレント, 01h=A:) A：00h 選択文字列を返す 01～09h この選択でフォーマットする 0A～FDh 不正 以下はブートセクタの更新のみ実施 FEh MSX-DOS 標準で更新 オフセット 1Eh 以降は更新しない FFh MSX-DOS 2 標準で更新 HL：バッファへのポインタ (A=01～09h のとき) DE：バッファのサイズ (A=01～09h のとき)
戻り値	A：エラーコード (00h = エラーなし) B：選択文字列のスロット (エントリで A = 00h のとき) HL：選択文字列のアドレス (エントリで A = 00h のとき)	

ブートセクタの構造

オフセット	Nextor	MSX-DOS 2
00～02h	8086 のジャンプ命令 (MSX では無視される) (訳注: 基本的には"EB FE 90"となる)	
03～0Ah	文字列 "NEXTOR20"	未使用
0B～0Ch	セクタのサイズ (バイト単位、通常 0200h)	
0Dh	クラスタのサイズ (セクタ単位、2 のn乗で指定)	
0E～0Fh	予約セクタ数 (ブートセクタで使用するセクタ数、通常 0001h)	
10h	FAT の数 (通常 02h)	
11～12h	ルートディレクトリエントリの数 (作成可能なエントリの数)	
13～14h	総セクタ数 (16bit) ただし、FFFFh を超えるときは 0000h	総セクタ数 (16bit)
15h	メディア ID	
16～17h	FAT サイズ (セクタ単位)	
18～19h	トラックあたりのセクタ数	
1A～1Bh	ディスクの面数 (片面／両面)	

1C ~ 1Fh	隠されたセクタ数 (32bit) Nextor ではこの値は変更しません	1C ~ 1Dh	隠されたセクタ数 (0000h)
		1E ~ 1Fh	ブートプログラムの先頭 (30h)へのジャンプ命令 (※訳注: "18 10h")
20 ~ 23h	総セクタ数 (32bit) ただし、0000 FFFFh 以下の時は 0000 0000h	20 ~ 25h	文字列 "VOL_ID"
24h	物理ドライブ番号 (00h)		
25h	ダーティディスクフラグ (00h 以外で UNDEL 可能)		
26h	拡張ブートレコードシグネチャ (29h のときボリューム名、ファイルシステムタイプが有効)	26h	ダーティディスクフラグ (00h 以外で UNDEL 可能)
27 ~ 2Ah	ボリューム ID (フォーマット時に乱数で決定される)		
2B ~ 35h	ボリューム名	2B ~ 2Fh	将来のための予約 (00h で埋める)
36 ~ 3Dh	ファイルシステムタイプ 文字列 "FAT12 " または "FAT16 "	30h ~	ブートプログラム

- 0B ~ 1Dh は FAT(MS-DOS) の BPB (BIOS Parameter Block) 仕様通りです。
MSX-DOS 1 では、1Eh からブートプログラムが始まります。
MSX-DOS 2 の 1C ~ 26h は MSX-DOS 2 独自仕様で、30h からブートプログラムが始まります。
Nextor 標準の 1C ~ 3Dh は 25h を除き FAT の拡張 BPB の仕様通りです。ブートプログラムは規定されていません。● オフセット 26h が 28h または 29h のとき、Nextor はオフセット 25h をダーティディスクフラグとして参照します。
- 総セクタ数が 65535 (0000 FFFFh) 以下のとき、オフセット 20 ~ 23h の総セクタ数 (32bit) が 0000 0000h となるのは FAT の仕様通りです。_DPARM (31h) のオフセット 18 ~ 1Bh の総セクタ数 (32bit) には 32bit の正しい値が入ります。これは Nextor の仕様です。

2.8. _DOSVER (6Fh)

MSX-DOS 2 または Nextor (通常モードもしくは MSX-DOS 1 mode) が動作しているかをアプリケーションから判別するために使用します。

Nextor のバージョンを判別するには下記のパラメータ (ここでは Magic Number と呼びます) をセットしてコールする必要があります。

B = 5Ah

HL = 1234h

DE = ABCDh

IX = 0000h

MSX-DOS 2 上でこの Magic Number をセットしてコールしたとき、IX = 0000h が返ります。MSX-DOS 1 のファンクションコールでは裏レジスタやインデックスレジスタを含むすべてのレジスタの値が破壊されますが、MSX-DOS 2 では裏レジスタとインデックスレジスタについては、そのレジスタが値を返すときを除いて保存されるためです。Nextor 上で Magic Number をセットせずにコールしたとき、IX, IY の値は変更されません。MSX-DOS 2 との互換性を保つためです。

Nextor 上で Magic Number をセットせずにコールしたときも IX, IY の値は変更されません。これは MSX-DOS 2 との互換性を保つためです。

Nextor 上で Magic Number をセットしてコールしたとき、下記の情報が返ります。

B = MSX-DOS カーネルのメジャーバージョン番号のエミュレート (常に 02h)

C = MSX-DOS カーネルのマイナーバージョン番号のエミュレート (常に 31h)

D = NEXTOR.SYS のメジャーバージョン番号 (BCD フォーマット)

E = NEXTOR.SYS のマイナーバージョン番号 (BCD フォーマット)

HL = カーネル内にある OS を示す文字列のアドレス

IXH = 01h

IXL = Nextor カーネルのメジャーバージョン番号 (02 ~ 0Fh)

IYH = Nextor カーネルのセカンダリバージョン番号 (00h ~ 0Fh)

IYL = Nextor カーネルのリビジョン番号 (00 ~ FFh)

アプリケーションからオペレーティングシステムを判別するには下記の手順で行ってください。

1. Magic Number をセットして _DOSVER をコールする
2. A = 00h でないとき (エラーのとき)、MSX-DOS でも Nextor でもない
3. B < 02h なら MSX-DOS 1
4. IX = 00h なら MSX-DOS 2 なので B、C を参照する
5. IXH = 01h なら Nextor なので、IXL, IYH, IYL を参照する
6. IXH が 00h でも 01h でもないとき、MSX-DOS でも Nextor でもない

HL は 00h で終わる OS の名前を示す文字列のアドレスです。例えば “Nextor kernel version 2.0” というような文字列が格納されています。この文字列はマスターとなっているカーネルのロット (ロット番号は F348h に格納) にあり、

_RDSLTL で読むことができます。

_DOSVER	Nextor	MSX-DOS 2
コール手順	B : 5Ah HL : 1234h DE : ABCDh IX : 00h	なし

戻り値	A : エラーコード (00h = エラーなし) B : 02h C : 31h D : NEXTOR.SYS のメジャーバージョン E : NEXTOR.SYS のマイナーバージョン (D, E はいずれも BCD フォーマット) HL : カーネル内の OS を示す文字列のアドレス IXH : 01h IXL : Nextor カーネルのメジャーバージョン IYH : Nextor カーネルのセカンダリバージョン IYL : Nextor カーネルのリビジョン	A : エラーコード (00h = エラーなし) B : MSX-DOS カーネルのメジャーバージョン C : MSX-DOS カーネルのマイナーバージョン D : MSXDOS2.SYS のメジャーバージョン E : MSXDOS2.SYS のマイナーバージョン (B ~ E はいずれも BCD フォーマット)
-----	--	--

2.8.1. Nextor の MSX-DOS 1 モードの検出

_DOSVER は Nextor の MSX-DOS 1 モードでも使用可能で、Nextor を判別することができます。

Nextor の MSX-DOS 1 モードで _DOSVER をコールしたときは、BC = 0100h となり、HL の値は意味を持ちません。MSX-DOS 1 では _DOSVER は F37Dh のフックからしかコールできず (3 章を参照)、IX 及び IY の内容は破壊されます。その為、IX で Nextor かどうかの判定ができません。その代わりに、_DOSVER は A = 01h を返します。

もし MSX-DOS 1 モードで動くアプリケーションで、Nextor カーネルか MSX-DOS 1 カーネルのどちらで動作しているかを知りたいときは、下記の手順となります。

1. Magic Number をセットして _DOSVER (F37Dh) をコールする
2. A = 01h, B = 01h, IXH = 01h なら Nextor の MSX-DOS 1 モードなので、IXL, IYH, IYL を参照する
3. A = 00h でないとき (エラーのとき)、MSX-DOS でも Nextor でもない
4. B < 02h なら MSX-DOS 1
5. IX = 00h なら MSX-DOS 2 なので B, C を参照する
6. IXH = 01h なら Nextor なので、IXL, IYH, IYL を参照する
7. IXH が 00h でも 01h でもないとき、MSX-DOS でも Nextor でもない

2.8.2. 各モードでの_DOSVER の戻り値の例

Kernel	Magic Number をセットしたとき	Magic Number をセットしなかったとき
Nextor MSX-DOS 1 mode	<u>A = 01h</u> <u>B = 01h</u> , C = 00h (MSX-DOS kernel 1.00) HL = カーネル内の OS を示す文字列のアドレス <u>IXH = 01h</u> , IXL = 02h (Nextor kernel 2.0.1) IYH = 00h, IYL = 01h	A = 00h B < 02h
Other OS	<u>A <> 00h</u> (Nextor の MSX-DOS1 kernel のときの A = 01h を先に判別する必要がある)	
MSX-DOS 1	A = 00h <u>B < 02h</u>	
MSX-DOS 2	A = 00h B = 02h, C = 31h (MSX-DOS2 kernel 2.31) D = 02h, E = 30h (MSX-DOS2.SYS 2.30) HL = カーネル内の OS を示す文字列のアドレス <u>IX = 00h</u>	A = 00h B = 02h, C = 31h (MSX-DOS2 kernel 2.31) D = 02h, E = 30h (MSX-DOS2.SYS 2.30) HL = カーネル内の OS を示す文字列のアドレス
Nextor	A = 00h B = 02h, C = 31h (MSX-DOS2 kernel 2.31) D = 02h, E = 10h (NEXTOR.SYS 2.10) HL = カーネル内の OS を示す文字列のアドレス <u>IXH = 01h</u> , IXL = 02h (Nextor kernel 2.0.1) IYH = 00h, IYL = 01h	A = 00h B = 02h, C = 31h (MSX-DOS2 kernel 2.31) D = 02h, E = 10h (NEXTOR.SYS 2.10) HL = カーネル内の OS を示す文字列のアドレス

下線部が判別に用いる戻り値です。

通常は、0005h または F37Dh を使い、下記の順番でチェックします。3 1

“Other OS” → “MSX-DOS 1” → “MSX-DOS2” → “Nextor” (→ “Other OS”)

Nextor の MSX-DOS 1 モードの判別が必要なときは F37Dh を使い、下記の順番でチェックします。

“Nextor MSX-DOS 1 mode” → “Other OS” → “MSX-DOS 1” → “MSX-DOS2” → “Nextor” (→ “Other OS”)

3. 新規のファンクションコール

この章では Nextor で追加された新しいファンクションコールの解説をします。使い方は既存の MSX-DOS のファンクションコールと同様、C レジスタにファンクション番号をセットして 0005h または F37Dh をコールします。ソースコードではファンクションのショートネーム (例えば “_FOUT”) を使用することを推奨します。また、それらの名前はこのマニュアルの機能相互参照表でも使われています。

いくつかの新しいファンクションコールは MSX-DOS 1 モードでも使用できます。現時点では、_GDRVR, _GPART, _CDRVR 及び _GDLI のみです。新しいファンクションコールを MSX-DOS 1 モードで使用する時、以下の制限があります。

- F37Dhをコールしなければなりません。0005h は使用できません。MSX-DOS.SYS が Nextor 用に作られていないためです。
- RAM をバッファとして使うとき、バッファはページ 1 (4000~7FFFh) にはおけません。ファンクション実行中は kernel がページ 1 にあるためです。一部の関数には追加の制限事項が適用されます。詳細は関数の説明を参照してください。

3.1. 高速 STROUT モードの取得/設定 (_FOUT, 71h)

高速 STROUT モードの有効／無効を切り替えます。

_FOUT	C : 71h
コール手順	A : 動作モード 00h = 高速 STROUT モードの状態取得 01h = 高速 STROUT モードをセット B : 有効／無効の切り替え 00h = 無効にする (A = 01h のときのみ) FFh = 有効にする (A = 01h のときのみ)
戻り値	A : エラーコード (00h = エラーなし) B : 高速 STROUT モードの状態 00h = 無効 FFh = 有効

有効にしたとき、_STROUT (09h) と_ZSTROUT (72h) の動作が速くなります。ただし、表示可能な文字列が最大で 511byte となります。文字列は 24h (_STROUT のとき) または 00h (_ZSTROUT のとき) で終了しますが、文字列が

511byte より長いときは先頭からの 511byte分のみ表示します。

3.2. ゼロ終了文字列の表示(_ZSTROUT, 72h)

DE レジスタで指される文字列の文字が通常の「コンソール出力」ファンクションを使用して出力されます。文字列は 00h で終了します。

_ZSTROUT	C : 72h
コール手順	DE : 文字列のアドレス
戻り値	A : 00h (エラーは返さない)

_STROUT (09h) とほぼ同じファンクションですが、文字列の終了は '\$' (24h) ではなく 00h で指定します。Fast STROUT モードが有効になっているとき、表示可能な文字列が 大で 511byte となります。文字列が 511byte より長いときは、最初の 511byte 分のみ表示されます。高速 STROUT モードはデフォルトでは無効になっており、追加されたファンクションコール_FOUT (71h) で有効にします。

3.3. ドライブからの物理セクタ読み込み (_RDDR, 73h)

セクタをファイルとして解釈することなく、ディスクからセクタを直接読み出します。セクタ番号をディスク上の物理的な位置に変換するため、ディスクは有効な DOS のディスクでなければなりません。セクタは現在のディスク転送アドレスに読み出されます。ディスクエラーは通常の方法で知らされます。

_RDDR	C : 73h
コール手順	A : 物理ドライブ番号 (00h = A:, 01h=B: など) B : 読み出すセクタ数 HL:DE : セクタ番号 (32bit)
戻り値	A : エラーコード (00h = エラーなし)

_RDABS (2Fh) とほぼ同じファンクションですが、FAT12 だけでなく FAT16 やその他のファイルシステム、ファイルシステムが存在しない状態であっても読み出し可能です。

3.4. ドライブへの物理セクタ書き込み (_WRDRV, 74h)

セクタをファイルとして解釈することなく、ディスクへセクタを直接書き込みます。セクタ番号をディスク上の物理的な位置に変換するため、ディスクは有効な DOS のディスクでなければなりません。セクタは現在のディスク転送アドレスから書き込まれます。ディスクエラーは通常の方法で知らされます。

_WRDRV	C : 74h
コール手順	A : 物理ドライブ番号 (00h = A:, 01h = B: など) B : 書き込むセクタ数 HL:DE : セクタ番号 (32bit)
戻り値	A : エラーコード (00h = エラーなし)

_WRABS (30h) とほぼ同じファンクションですが、FAT12 だけでなく FAT16 やその他のファイルシステム、ファイルシステムが存在しない状態であっても書き込み可能です。

3.5. 割当情報縮小モード状態の取得/設定 (_RALLOC, 75h)

割当情報縮小モードの有効/無効の取得、または切り替えを行います。各 bit はそれぞれのドライブに割り当てられます。L レジスタの bit 0 が A:ドライブ、bit 1 が B:ドライブ...となります。bit が 1 のときは割当情報縮小モードが有効となり、_ALLOC (1Bh) コールはクラスタ総数と未使用クラスタ数が 32MB を超えるときに 32MB 未満になるように偽の値を返すようになります。各 bit のデフォルトは 0b です。

_RALLOC	C : 75h
コール手順	A : 動作モード 00h =割当情報縮小モードの状態取得 01h =割当情報縮小モードをセット HL : 有効／無効の切り替え 0b = 無効にする (A = 01h のときのみ) 1b = 有効にする (A = 01h のときのみ) (bit 0 = A:, bit 1 = B: など)
戻り値	A : 00h (エラーは返さない) HL : 物理ドライブの割当情報縮小モードの状態 (bit 0 = A:, bit 1 = B: など)

さらに Nextor 起動時に環境変数 "ZALLOC" に "ON" を設定することにより、割当情報縮小モードは割当情報ゼロモードになります。このとき、_ALLOC (1Bh) は割当情報縮小モードに指定されているドライブについて、未使用クラスタ数 0 を返します。

3.6. ドライブ容量の取得 (_DSPACE, 76h)

ドライブの総容量または空き容量を返します。容量情報はドライブのファイルシステムやクラスタサイズにかかわらず kB 単位で返します。kB 未満の余り容量も別のレジスタで返します。

_DSPACE	C : 76h
コール手順	E : 論理ドライブ番号 (00h = カレント, 01h = A: など) A : 動作モード 00h = ドライブの空き容量情報取得 01h = ドライブの総容量情報取得
戻り値	A : エラーコード (00h = エラーなし) HL:DE : 容量 (kB 単位) BC : 余り容量 (byte 単位)

余り容量はドライブの最小割り当て単位が kB 未満のときに 0 以外の値となります。FAT ドライブでは、1 クラスタあたりのセクタ数が 1 で、クラスタ数が奇数のときに 0 以外の値 (特に 512) となります。例えば、あるドライブにおいて 1 クラスタあたりのセクタ数が 0 で未使用クラスタ数が 15 (=7.5kB) のとき、HL = 0000h, DE = 0007h (=7kB) , BC=0100h (512byte) となります。

この容量情報は割当情報縮小モードの影響は受けず、常に正しい値となります。

3.7. ドライブのロック/ロック解除、ロック状態の取得 (_LOCK, 77h)

ドライブのロック及びロックの解除、またはロック状態の確認を行います。ドライブがロックされているとき、ドライブのメディアは交換されないと判断し、メディア交換のステータスを確認しません。それによりメディアへのアクセス速度を改善します。マルチメディアカード等のリムーバブルメディアをメインストレージとして使用するときに有用です。

_LOCK	C : 77h
コール手順	E : 物理ドライブ番号 (00h = A:, 01h = B: など) A : 動作モード 00h = ロック状態の取得 01h = ロック状態の設定 B : ロック／ロック解除の切り替え (A = 01h のときのみ) 00h = ロックを解除する FFh = ロックする

戻り値	A : エラーコード (00h = エラーなし) B : ロック状態 00h = ロックされていない FFh = ロックされている
-----	--

ロックするためには、有効なファイルシステムがドライブとして割り当てられていなければなりません。つまり、ドライブがアクセス可能でなければなりません。

ドライブがロックされた状態でも、アボート(中止)となるディスクエラーが発生したときには、ロックが解除されます。

ロック及びロックの解除を行ったとき、バッファ内のデータはドライブに書き込まれ、無効となります。また、ディスクパラメータブロックの情報もクリアされ、次のアクセス時に再度読み込みます。

デバイスドライバ上で非リムーバブルとして扱っているデバイスに対してもロックを設定することは可能ですが、Nextor はそれらのデバイスに対してメディアの交換を確認することはないので意味がありません。

ロック機能は注意して使う必要があります。ロック状態のデバイスに対してロック解除をせずにメディア交換をする
と交換前のメディアと交換後のメディアの両方でデータを破壊する恐れがあります。

3.8. デバイスドライバ情報の取得 (_GDRVR, 78h)

デバイスドライバの情報を返します。このファンクションは MSX-DOS 1 モードでも動作します。

_GDRVR	C : 78h
コール手順	A : インデックス番号 (01h ~) もしくは 00h (スロット番号とセグメント番号のペアで指定するとき) D : スロット番号 (A = 00h のときのみ) E : セグメント番号もしくは FFh (Nextor のカーネル内のとき) (A = 00h のときのみ) HL : 64byte のバッファのアドレス
戻り値	A : エラーコード (00h = エラーなし) HL で指定したアドレスにある 64byte のバッファにドライバ情報が入る

デバイスドライバはインデックス番号もしくはスロット番号＋セグメント番号のペアで指定します。デバイスドライバをインデックス番号で指定するときは、A レジスタにインデックス番号をセットします。インデックス番号は 1 から始まります。

デバイスのあるスロット番号とセグメント番号はその他のデバイスドライバ情報と一緒にデータバッファに入れられて返ってきます。これはどのデバイスドライバが使われているかを確認するのに有用です。

もし確認したいデバイスのスロット番号とセグメント番号を既に知っているときは、それらの番号を D レジスタと E レジスタにセットし、A レジスタには 00h をセットします。このときもスロット番号とセグメント番号はその他のデバイスドライバ情報と一緒にデータバッファに入れられて返ってきます。

指定したインデックス番号にデバイスドライバーが割り当てられていなかったとき、指定したスロット番号とセグメント番号のペアにデバイスドライバーがないときには、IDRVR エラー (0B6h) が返ります。事前にいくつかのデバイスドライバーがシステムにあるかを知ることはできません。そのため、すべてのデバイスドライバーを見つけるためにはこのファンクションをインデックス番号 1 から数を増やしながらか IDRVR エラー (0B6h) が返るまで繰り返し実行する必要があります。

デバイスドライバー情報フォーマット (オフセットは HL+xxh を示す)

オフセット	内容
00h	デバイスドライバーのスロット番号
01h	デバイスドライバーのセグメント番号 (Nextor や MSX-DOS カーネル内のデバイスドライバーは FFh)
02h	起動時にドライバーに割り当てられた論理ドライブ数
03h	起動時にドライバーに割り当てられたドライブの一番若い番号 (A: = 00h, B: = 01h など) 割り当てられていないときは使われません。(訳注: オフセット 02h=00h の時)
04h	bit 7 : 1b = Nextor 用ドライバー 0b = MSX-DOS 用デバイスドライバー (MSX-DOS カーネル内のとき) bit 6 - 3 : 未使用、常に 0b bit 2 : 1b = ドライバーが DRV_CONFIG ルーチンを実装している場合 bit 1 : 未使用、常に 0b bit 0 : 1b = デバイスベースドライバー 0b = ドライブベースドライバー
05h	デバイスドライバーのメジャーバージョン番号
06h	デバイスドライバーのセカンダリバージョン番号
07h	デバイスドライバーのリビジョン番号
08h ~ 27h	デバイスドライバー名、左詰で残りは 20h で埋める
28h ~ 3Fh	予約 (現在は 00h)

MSX-DOS ドライバーの場合、ドライバフラグバイトは常にゼロとなり、ドライバーのバージョン番号やドライバー名に関する情報は返されません。Nextor はバージョン 2.0.5 以降で DRV_CONFIG ルーチンを使用します。詳細については、[“Nextor 2.1 ドライバー開発ガイド”](#)を参照してください。

ドライブベースドライバー / MSX-DOS 用デバイスドライバー

MSX-DOS のドライバー仕様に沿っており、1 つのドライブレターに対して 1 つのデバイスドライバーが対応しています。

デバイスドライバーがドライブレターの割り当てやパーティション管理を行います。

デバイスベースドライバー

デバイスベースドライバーはドライブ単位ではなく、デバイス単位で動きます。デバイスドライバーの中には “Read sector X of unit N” というようなルーチンはなく、“Return information of device X” や “Read raw data from device X” というようなデバイス単位で動くルーチンがあります。デバイスベースドライバーは最大 7 個のデバイスを管理でき、それぞれのデバイスは 1～7 個の論理ユニットを持つことができます。

3.9. ドライブレター情報の取得 (_GDLI, 79h)

ドライブレターの情報を返します。このファンクションは MSX-DOS 1 モードでも動作します。

_GDLI	C : 79h
コール手順	A : 物理ドライブ番号 (00h = A:, 01h = B: など) HL : 64byte のバッファのアドレス
戻り値	A : エラーコード (00h = エラーなし) HL で指定したアドレスにある 64byte のバッファに、ドライブレター情報が入る

もし物理ドライブ番号 (= ドライブレター数) がシステムでサポートしている数を超えているときは、.IDRV エラー (0DBh) が返ります。ドライブレターが Nextor で指定されていて、かつデバイスドライバーが割り当てられていないときはエラーとなりませんが、ドライブレターの情報は空白となります。ドライブステータスをチェックする必要があります。

“先頭デバイスセクタ番号” は、ドライブの最初の論理セクタとして扱われる物理セクタ番号です。通常はパーティションの先頭セクタで、パーティションが区切られていないときは物理セクタ番号 0 となります。この値が 0 かどうかで デバイスベースドライバーにブロックデバイスがドライブに割り当てられているかどうかを判断してはいけません。その目的では「ドライブステータス」フィールドを使用します。

ドライブレター情報フォーマット (オフセットは HL+xxh を示す)

オフセット	内容
00h	ドライブ状態 00h = 未割り当て 01h = Nextor または MSX-DOS ドライバーのストレージデバイスとして割り当て 02h = 未使用 03h = ファイルがドライブにマウントされている 04h = RAM ディスクに割り当て (他のフィールドはすべて 00h)
01h	デバイスドライバーのスロット番号
02h	デバイスドライバーのセグメント番号 (Nextor や MSX-DOS カーネル内のデバイスドライバーは FFh)
03h	このデバイスドライバーで扱うドライブ数 (ドライブベースドライバーのみ、デバイスベースドライバーでは FFh)

04h	デバイス番号 (デバイスベースドライバーのみ、ドライブベースドライバーもしくは MSX-DOS 用のデバイスドライバーでは 00h)
05h	論理ユニット番号 (デバイスベースドライバーのみ、ドライブベースドライバーもしくは MSX-DOS 用のデバイスドライバーでは 00h)
06h ~ 09h	先頭セクタ番号 (デバイスベースドライバーのみ、ドライブベースドライバーもしくは MSX-DOS 用のデバイスドライバーでは 00h)
0Ah ~ 3Fh	予約 (現在は 00h)

ファイルがドライブにマウントされている場合、データバッファに返される情報は次のようになります。

オフセット	内容
01h	マウントされたファイルが配置されているドライブ (0 = A: など)
02h	フラグ bit 0: マウントモード、0 = 読み書き可能、1 = 読み取り専用
03h	00h に固定
04h ~ 10h	表示可能な形式のファイル名 (最大 12 文字、末尾のゼロを含む)
11h ~ 12h	ファイルの開始クラスタ、2 バイト (利用できない場合は 0)
13h ~ 16h	ファイルの開始セクタ、4 バイト (利用できない場合は 0)

マウントされたファイルの“開始クラスタ”フィールドと“開始セクタ”フィールドは、Nextor 2.1.1 で導入されました。現在、これらのフィールドは常に意味のある情報を含んでいますが、Nextor の将来のバージョンでは(「クラスタ」の概念を持たない非 FAT ファイルシステムのサポートなどにより)、この限りではありません。その場合、これらのフィールドの値はゼロになります。これらのフィールドは Nextor 2.1.1 より前のバージョンでもゼロとして返されるため、この関数呼び出しを使用するアプリケーションプログラムは、使用前にこれらのフィールドの値がゼロでないことを常に確認する必要があります。

3.10. デバイスパーティション情報の取得 (_GPART, 7Ah)

パーティションの情報を返します。このファンクションは MSX-DOS 1 モードでも動作します。

_GPART	C: 7Ah
コール手順	A: デバイスドライバーのスロット番号 B: デバイスドライバーのセグメント番号もしくは FFh (MSX-DOS カーネル内のとき) D: デバイス番号 E: 論理ユニット番号 H: プライマリパーティション番号 (01 ~ 04h) H: bit 7: 0b = パーティション情報の取得 1b = パーティションテーブルエントリを保持するデバイスのセクタ番号を取得 L: 拡張パーティション番号 (プライマリパーティションでは 00h)

戻り値	A : エラーコード (00h = エラーなし) パーティション情報の取得のとき B : パーティションタイプ (指定されたパーティションが存在しないときは 00h) C : パーティションのステータス情報 HL:DE : パーティションの先頭セクタ番号 IX:IY : セクタ単位のパーティションサイズ パーティションテーブルエントリのセクタ番号を取得のとき HL:DE : パーティションテーブルエントリを保持するデバイスセクタ番号
-----	---

このファンクションはデバイスベースドライバに対してのみ使用できます。存在しないデバイスドライバ、ドライバベースドライバもしくは MSX-DOS 用のデバイスドライバが A レジスタと B レジスタで指定されたときは、IDRVR エラー (0B6h) が返ります。もし指定したデバイスと論理ユニット番号の両方もしくは一方がデバイスドライバに存在しないときは、IDEVL エラー (0B5h) が返ります。

ストレージデバイスは通常、パーティションに分割され、それぞれの独立したパーティション内では連続したセクタを使用しています。このファンクションは各パーティションの先頭セクタ番号を見つけることができます。_MAPDRV (7Ch) でそれを使ってドライブレターを割り当て、ファイルシステムにアクセスできるようにします。

パーティションタイプはそのパーティションで使われているファイルシステムの情報を示します。パーティションタイプは以下のいずれかになります。

- 00h : None (パーティションタイプが存在しない)
- 01h : FAT12
- 04h : FAT16 (32MB 以下のとき、廃止)
- 05h : 拡張 (CHS) (下記参照)
- 06h : FAT16 (CHS)
- 0Eh : FAT16 (LBA)
- 0Fh : 拡張 (LBA)

注: バージョン 2.1.2 より前のバージョンでは、Nextor はパーティションタイプコード 05h (拡張 CHS) を持つパーティションのみを拡張パーティションとして認識し、FDISK は拡張パーティションの作成時にこのコードを使用していました。

Nextor 2.1.2 以降では、FDISK は拡張パーティションの作成時にパーティションタイプコード 0Fh (拡張 LBA) を使用するようになり、_GPART を使用して既存のパーティションをスキャンする際には、05h と 0Fh の両方が有効な拡張パーティションタイプコードとして認識されます。以下の説明で「拡張」は「拡張 CHS または拡張 LBA」のいずれかを意味します。

実際にはもっと多くのパーティションタイプが存在していますが、Nextor では扱えないためリストから除外しています。

1 つのデバイスは 1~4 の 4 個までのプライマリパーティションを持つことができます。4 個より多くのパーティションを持つためには、プライマリパーティション 2 のパーティションタイプを「拡張」にします。パーティションタイプ「拡張」のパーティションは中に多くの拡張パーティションを持つことができ、その数は無制限です。プライマリパーティション 2 のパーティションタイプが「拡張」のとき、プライマリパーティション 3 及び 4 は存在しません。

デバイス内に存在するすべてのパーティションを列挙するためには、下記の手順が必要となります。

1. パーティション 1-0 をチェック (プライマリパーティション 1 の拡張パーティション番号 0)
2. パーティション 2-0 をチェックして存在し、パーティションタイプが「拡張」のとき、パーティション 2-1, 2-2, 2-3, ...
とパーティションタイプ 00h が返ってくるまで続けます。
3. パーティション 2-0 が存在しないか、パーティションタイプが「拡張」ではなかったとき、パーティション 3-0 及び 4-0 をチェックします。

デバイス内にパーティションが存在しないこともあります。そのときでも有効なファイルシステムを持ち、セクタ 0 にドライ

ブレターを割り当てることが可能なものがあります。フロッピーディスクや小容量のデバイスがそれにあたります。

パーティションがドライブレターに割り当てられたとき、パーティションの先頭セクタから実際のファイルシステムを確認します。Nextor がファイルシステムを確認するのにパーティションタイプ情報は使いません。

Nextor がパーティションを確認するときにデバイスを読みます。エラー (例えば “Not Ready” 等) が発生したときは、エラーコードを返します。標準のエラー処理ルーチン (DISKVE (F323h)) やユーザー定義 (_DEFER (64h)) のディスクエラー処理ルーチンは呼び出されません。

指定したパーティションがデバイスに存在しないとき (例えばプライマリパーティション番号に 5 以上を指定したり、拡張パーティションが存在しないパーティションで 1 以上の拡張パーティション番号を指定したりしたとき)、B = 00h, A = .IPART (B4h) が返ります。

Nextor 2.1.0 beta 2 以降では、パーティションに関する情報を要求する代わりに、パーティションテーブルが格納されているデバイスのセクタ番号を要求できるようになりました。これは、パーティションテーブルエントリを変更するアプリケーションで有用です。返されたセクタ内のパーティションテーブルエントリの位置は、以下のとおりです。

- ・ プライマリパーティション (拡張パーティション番号 0) が要求された場合、返されるセクタ番号は常に 0 となり、そのセクタ内のパーティションテーブルエントリのオフセットは、プライマリパーティション番号 1、2、3、4 に対してそれぞれ 1BEh、1CEh、1DEh、1EEh となります。
- ・ 拡張パーティション (プライマリパーティション番号 2、拡張パーティション番号 0 以外) が要求された場合、パーティションテーブルエントリは、返されたセクタ内の最初のエントリ (オフセット 1BEh) となります。

パーティションテーブル構造の詳細については、[Wikipedia のマスターブートレコード](#)を参照してください。

3.11. デバイスドライバルーチンの呼び出し (_CDRVR, 7Bh)

デバイスドライバー内のルーチンを直接呼び出します。このファンクションは MSX-DOS 1 モードでも動作します。

_CDRVR	C : 7Bh
コール手順	A : デバイスドライバーのスロット番号 B : デバイスドライバーのセグメント番号もしくは FFh (MSX-DOS カーネル内のとき) DE : ルーチンのアドレス HL : ルーチンに渡す引数 (8byte) のあるバッファのアドレス
戻り値	A : エラーコード (00h = エラーなし) BC, DE, HL : ルーチンの戻り値 IX : ルーチンの戻り値 AF

デバイスベース、ドライブベース、MSX-DOS 用かに関わらず、すべてのデバイスドライバーのルーチンを呼び出すことができますが、デバイスベースドライバーがデバイスを数えたり (DEV_INFO (4163h) や LUN_INFO (4169h))、デバイスのセクタに直接アクセスしたり (DEV_RW (4160h)) することを意図しています。例えば、デバイスのパーティションツールを作るときに使います。呼び出し可能なルーチンの一覧やその詳細については、[“NNextor 2.1 ドライバー開発ガイド”](#)を参照してください。

ルーチンへの引数として AF, BC, DE, HL レジスタの値を 8byte のバッファに用意し、HL レジスタにバッファのアドレスを入れます。バッファには、F, A, C, B, E, D, L, H の順で並べます。ルーチンの戻り値としてのレジスタの値は、AF レジスタの値が IX レジスタに入る以外は、各々のレジスタに直接入れられます。

一部のルーチンではユーザーが用意したバッファに対して読み書きすることができます。ただし、この方法では二つの制限があります。

- ① バッファはプライマリマップスロットにならない
- ② バッファの一部またはすべてがページ 1 (4000h - 7FFFh) にあってはならない

これらの制限はルーチンに渡す引数をおくバッファには適用されません。ただし、F37Dh を用いて呼び出すときにはいかなるパラメータもページ 1 にはおけません。また、引数をおくバッファはルーチンを実行する前にしか使いませんので、ルーチンからの戻り値をおくためのバッファとして使ってもかまいません。

A レジスタ及び B レジスタで指定されたデバイスドライバーが存在しないときは、IDRVR エラー (0B6h) が返ります。

Nextor で追加されたファンクションコール _GDRVR (78h) を使って、存在しているドライバーを確認してください。

3.12. ドライバーとデバイスへのドライブレター割り当て (_MAPDRV, 7Ch)

デバイスベースドライバーのデバイス番号、論理ユニット番号及びその先頭セクタの組み合わせ (mapping data) で指定するパーティションに対してドライブレターを割り当てます。また、起動時の状態に戻したり、すべてのドライブの割り当てを解除したりすることも出来ます。このファンクションは MSX-DOS 1 モードでも動作しますが、後述するいくつかの制限があります。

_MAPDRV	C : 7Ch
コール手順	A : 物理ドライブ番号 (00h = A:, 01h = B: など) B : 動作モード 00h = ドライブレターの割り当てを解除 01h = ドライブレターの割り当てを起動時の状態に戻す 02h = ドライブレターを mapping data で指定するパーティションに割り当てる 03h = ファイルをドライブに割り当てる HL : mapping data (8byte) のあるバッファのアドレス (B = 02h のときのみ)
戻り値	A : エラーコード (00h = エラーなし)

B = 00h ではドライブの割り当てを解除します。その時点からドライブへのアクセスが不可能になり、すべてのアクセス

は “Invalid drive” エラーとなります。既に割り当てが解除されているときは何も起きず、エラーにもなりません。

B = 01h では指定したドライブレターの割り当ては起動時の状態に戻ります。起動時に割り当てられていなかったり、ドライブベースドライバーで割り当てられていたり、MSX-DOS 用のドライバーで割り当てられていたときは、その状態に戻ります。起動時にデバイスベースドライバーに割り当てられていたときは、自動割り当てルールに従って再割り当てされます。自動割り当て機能については、“[Nextor 2.1 ユーザーマニュアル](#)” に記載されています。再割り当ての際に起動時と同じ割り当て状態に戻るかは、同じデバイスドライバーで管理するリムーバブルデバイスの状態や、他のドライブの状態によります。再割り当てが失敗したとき (例えばデバイスやパーティションがなかったとき)、実行前の状態にかかわらず、ドライブレターの割り当ては解除され、.IDEVL エラー (0B5h) が返ります。

B = 02h では HL レジスタで指定したアドレスにあるバッファ内の mapping data にしたがって、ドライブレターが割り当てられます。この方法ではデバイスベースドライバーに対してどのドライブレターでも自由に割り当てられます。起動時にドライブレターが割り当てられていなかったり、異なるドライブレターに割り当てられていたりしたときでも可能です。

バッファ内の mapping data の構造は以下の通りです。

オフセット	内容
00h	デバイスドライバーのスロット番号
01h	デバイスドライバーのセグメント番号 (Nextor カーネル内のデバイスドライバーは FFh)

02h	デバイス番号
03h	論理ユニット番号
04 ~ 07h	先頭セクタ番号

指定したデバイスドライバーが存在しない、またはデバイスベースドライバーでないときは、.IDRVR エラー (0B6h) が返ります。デバイスドライバー内にデバイスや論理ユニットが存在しないときは IDEVL エラー (0B5h) が返ります。このとき、実行前の状態から割り当て状態は変更されません。

H:ドライブを指定したときに RAM ディスクが存在しているときは、.RAMDX エラー (0BCh) が返ります。

先頭セクタ番号はデバイスにおける物理セクタで、ドライブの論理セクタ 0 として使われます。通常、パーティションの先頭セクタで、_GPART (7Ah) で確認できます。ただし、指定したセクタにファイルシステムがあつて、かつ Nextor で認識できるかどうかはチェックしません。有効なファイルシステムが存在しなくても _MAPDRV は成功しますが、次にアクセスしたときに “Not a DOS disk” エラーとなります。

また、メディアが入っていない状態のリームーバブルデバイスにドライブレターを割り当てることも可能です。この場合でも _MAPDRV は成功しますが、次にアクセスしたときに “Disk offline” エラーとなります。

同じ組み合わせ (ドライバー、デバイス、論理ユニット、先頭セクタの番号) に対して二つのドライブレターを割り当てることはできません。各ドライブのセクタバッファが同期せず、データの不整合が起きることを防ぐためです。このとき、.IDEVL エラー (0B5h) が返ります。すでに割り当てられているドライブレターを変更したいときは、先に古いドライブレターの割り当てを解除する必要があります。

また、ドライブベースドライバーや MSX-DOS 用のデバイスドライバーに対して、ドライブレターを指定して割り当てることは出来ません。

ドライブレターの割り当てを変更する前に、オープンされているすべてのファイルハンドルはこのファンクションによって閉じられます。これは _CLOSE (45h) と同じです。そのため、バッファに問題があつて、デバイスへの書き込み時にエラーが発生したときは、ディスクエラーとなります。

MSX-DOS 1 モードで使った時、下記の制限が発生します。

- ① 指定するドライブレターは起動時にデバイスベースドライバーに割り当てられていなければなりません。
起動時に割り当てられていない、またはドライブベースドライバーや MSX-DOS 用のデバイスドライバーに割り当てられているドライブレターは変更できません。
- ② 新しいドライブレターの割り当てはパーティションとデバイスの両方もしくは片方が異なるものが指定できますが、スロットは起動時に割り当てられたドライブと同じでなければなりません。Nextor の kernel が 1 個だけ存在しているときには問題になりません。

これらの制限は Nextor のアーキテクチャによるものです。

B = 03h では、HL で渡されたファイル名または FIB のファイルがドライブにマウントされます。ファイルのマウントは Nextor 2.1.0 以降で利用可能で、Nextor 2.1.1 以降では、マウントするにはホストファイルシステム内の連続したセクタに格納されている必要があります。ファイルが小さすぎる(512 バイト未満)または大きすぎる(32MB を超える)場合は、.BFSZ エラー (0B1h) が返されます。ファイルが連続したセクタに格納されていない場合は、.ICLUS エラー (0B0h) が返されます。

3.13. ドライバーの Z80 アクセスモードの有効/無効 (_Z80MODE, 7Dh)

このファンクションは MSX-DOS 用デバイスドライバー (MSX-DOS カーネル内のデバイスドライバー) に対して Z80 アクセスモードの有効/無効を設定します。このファンクションは MSX turboR でのみ有効です。MSX, MSX2, MSX2+ では.IDRVR エラー (0B6h) となります。

_Z80MODE	C : 7Dh
コール手順	A : デバイスドライバーのスロット番号 B : 動作モード 00h = 現在の Z80 アクセスモードの状態を取得 01h = Z80 アクセスモードを設定する D : 有効/無効の切り替え 00h = Z80 アクセスモードを無効にする (B = 01h のときのみ) FFh = Z80 アクセスモードを設定する (B = 01h のときのみ)
戻り値	A : エラーコード (00h = エラーなし) D : 現在の Z80 アクセスモードの状態 00h = Z80 アクセスモードが無効 FFh = Z80 アクセスモードが有効

Z80 アクセスモードが対象のデバイスドライバーに対して有効になっているとき、そのデバイスドライバーで管理するドライブにアクセスする前に Nextor が CPU を Z80 に切り替えます。無効のときは CPU の切り替えを行いませんので、

R800 のままドライブにアクセスします。

起動時にはすべての MSX-DOS 用デバイスドライバーに対して Z80 アクセスモードが有効になっています。いくつかの古いデバイスは R800 モードではアクセスできず、それらのデバイスから起動出来ないためです。起動後にはこのファンクションを使って Z80 アクセスモードを無効にしたり、後で再度有効にしたりすることができます。

Z80 アクセスモードは設定するデバイスドライバーで管理しているすべてのドライブに対して有効/無効を切り替えます。

個々のドライブに対して、Z80 アクセスモードの有効/無効を設定することはできません。

Z80 アクセスモードは MSX-DOS 用デバイスドライバーに対してのみ有効です。Nextor はドライブベースドライバーやデバイスベースドライバーに対しては CPU を切り替えません。

3.14. FAT ドライブ上のクラスタ情報を取得 (_GETCLUS, 7Eh)

このファンクションは、指定されたドライブの指定されたクラスタ番号に関する情報をユーザーバッファに格納します。ただし、ドライブが FAT12 または FAT16 ファイルシステムにマッピングされている場合に限りです。このファンクションは、_RDDRIV および _WRDRV と組み合わせて使用することで、デフラグツールなどの低レベルのディスク操作を実行するツールに有用です。

_GETCLUS	C : 7Eh
コール手順	A : 論理ドライブ番号 (00h = カレント, 01h = A: など) DE : クラスタ番号 HL : クラスタ番号情報 (16byte) のあるバッファのアドレス
戻り値	A : エラーコード (00h = エラーなし)

バッファ内のクラスタ番号情報の構造は以下の通りです。

オフセット	内容
00h ~ 01h	クラスタのエントリを含む FAT セクタ番号 (2 バイト)
02h ~ 03h	クラスタのエントリが配置されている FAT セクタ内のオフセット (0~511) (2 バイト)
04h ~ 07h	クラスタが参照する最初のデータセクタ番号 (4 バイト)
08h ~ 09h	クラスタの FAT エントリの値 (2 バイト)
0Ah	ドライブのクラスタサイズ (セクタ数) (1 バイト)
0Bh	フラグ (1 バイト) bit 0 : FAT12 のときに 1b bit 1 : FAT16 のときに 1b bit 2 : クラスタの FAT エントリが奇数の場合に 1b (FAT12 のみ) bit 3 : クラスタがファイルの最後のクラスタの場合に 1b bit 4 : クラスタが空いている場合に 1b bit 5-7 : 未使用、常にゼロ
0Ch ~ 0Fh	未使用、常にゼロ

ドライブが FAT12 でも FAT16 でもない場合、この関数は .NDOS エラー (0F6h) を返します。つまり、関数がエラーを返さない場合は、常に FAT12 フラグまたは FAT16 フラグのいずれかがセットされます。ただし、Nextor の将来のバージョンでは、他の FAT バリエーションのサポートが追加された場合、このエラーは発生しない可能性があります。そのため、常に両方のフラグを確認し、片方のフラグがクリアされているからといって、もう片方のフラグがセットされているとは考えないようにしてください。

指定されたクラスタ番号が指定されたドライブに存在しない場合は、.ICLUS エラー (0B0h) が返されます。0 と 1 は常に無効なクラスタ番号であることに注意してください (最初の FAT セクタにあるこれらのエントリのバイトは未使用です)。

FAT エントリのクラスタの値は、FAT ファイルシステムにおける通常の意味を持ちます。

- ① 0 はクラスタが空いていることを意味します。
- ② FAT 12 の場合は 0FF8h～0FFFh、FAT 16 の場合は FFF8h～FFFFh は、クラスタがファイルの最後のクラスタであることを意味します。
- ③ その他の値は、ファイルのデータが続く次のクラスタの番号です。

便宜上、「クラスタが空いている」フラグと「クラスタがファイルの最後のクラスタである」フラグが用意されています。これにより、最初の 2 つのケースは、FAT エントリ自体の値を確認しなくても簡単に検出できます。

FAT12 の FAT エントリは、「エントリが奇数」フラグによって偶数または奇数のいずれかになります。それぞれのケースで値は異なります。「X」が「FAT セクタ内のオフセット」の値で、「peek(X)」が X に格納されているバイト値である場合、値は以下ようになります。

- ① 偶数エントリの場合： $\text{peek}(X) + ((\text{peek}(X+1) \text{ and } \&\text{HF}) * 256)$
- ② 奇数エントリの場合： $((\text{peek}(X) \setminus 16) \text{ and } \&\text{HF}) + (\text{peek}(X+1) * 16)$

便宜上、FAT12 の偶数エントリと奇数エントリの例を示す図(この図のベースは MSX2 テクニカル・ハンドブックの 3 部 3 章に記載の「図 3.12 FAT の実例」)を以下に示します。

先頭アドレス→	4 ビット		
	4 ビット	4 ビット	
	F	0	Media ID
+1	F	F	ダミーエントリ
+2	F	F	ダミーエントリ
+3	1	2	クラスタエントリ 2、オフセット 3、値は 412h(偶数エントリ)
+4	3	4	クラスタエントリ 3、オフセット 4、値は 563h(奇数エントリ)
+5	5	6	
+6	7	8	クラスタエントリ 4、オフセット 6、値は 978h(偶数エントリ)
+7		9	

FAT セクタのオフセットが 511 の場合、エントリは「エントリを含む FAT セクタ番号」の最後のバイトと次のセクタの最初のバイトに分割されることに注意してください。これは FAT12 でのみ発生する可能性があります。

4. 新しいエラーコード

Nextor で追加された新しい機能を扱うときのエラーをハンドルするため、新しいエラーコードを定義しました。

MSX-DOS 1 モードにおいても Nextor の新しい機能がサポートされている場合は、同じエラーコードが返ります。

新しいエラーコードの一覧と詳細は下記の通りです。

※訳注: 以下の日本語のエラーメッセージは、source¥kernel¥bank1¥msg.mac ファイルに記載のものです。

- Invalid device driver / 無効なデバイスドライバです (.IDRVR, 0B6h)

デバイスドライバに対する操作がリクエストされたときに、指定されたデバイスドライバが存在しないか、タイプが異なる(例えば、MSX-DOS または ドライブベースドライバに対してデバイスベースドライバ用の操作がリクエストされたような場合)。

- Invalid device or LUN / 無効なデバイスまたは論理ユニットです (.IDEVL, 0B5h)

デバイスベースドライバを使うデバイスに対する操作がリクエストされたときに、指定されたデバイスがデバイスドライバに存在しない、または指定された論理ユニットが指定されたデバイスに存在しない。

- Invalid partition number / 無効なパーティション番号 (.IPART, 0B4h)

デバイスのパーティション情報が要求されたが、指定されたパーティションがデバイスに存在しない。

- Partition is already in use / パーティションが使用中です (.PUSED, 0B3h)

デバイスドライバ、デバイス、論理ユニット、先頭セクタの組み合わせに対してドライブレターの割り当てを試みたが、同じ組み合わせに対して別のドライブレターが既に割り当てられているとき。

- File is mounted / ファイルがマウントされています (.FMNT, 0B2h)

マウントされたファイルを開こうとしたり、変更しようとしたり、あるいはマウントされたファイルに関連するその他の許可されていない操作を実行しようとしたとき。

- Bad file size / 無効なファイルサイズ (.BFSZ, 0B1h)

512 バイト未満または 32 MB を超えるファイルをマウントしようとしたとき。

- Invalid cluster number or sequence / 無効なクラスタ番号またはシーケンス (.ICLUS, 0B0h)

_GETCLUS 関数に指定されたクラスタ番号がドライブ内に存在しないか、マウントするファイルが _MAPDRV に指定されたが、そのファイルはホスト ファイル システム内の連続したセクタに格納されていない。

5. 拡張マッパーサポートルーチン

オリジナルの MSX-DOS 2 のマッパーサポートルーチン(“[MSX-DOS 2 プログラムインターフェース仕様](#)”の「5.2 マッパーサポートルーチン」または、“[MSX-Datapack Volume 3 turboR 版](#)”の「第 2 部 15 章 マッパーサポートルーチン」を参照)に、データの読み取り、データの書き込み、および別の RAM セグメントに配置されたルーチンの呼び出しを可能にする 4 つの新しいルーチンを拡張しました。これらのルーチンは、既存のルーチン RD_SEG、WR_SEG、CAL_SEG、CALLS と同様に動作しますが、パラメータとして RAM セグメント番号だけでなく、スロット番号と RAM セグメント番号の組み合わせでも指定できます。これらのルーチンは、拡張 BIOS ベースの検出メカニズムを含む“[UNAPI RAM ヘルパー仕様](#)”と互換性があります。つまり、UNAPI RAM ヘルパーの存在に依存するアプリケーションプログラムは、最初にスタンドアロンヘルパーをインストールすることなく、Nextor ですぐに使用できます。

※訳注:「N2.0PRJ」には、以下の記載がありました。ルーチンの内容が変更となっているため、記載内容の大部分(取り消し線部分)は現在のルーチンとは異なっています。

~~セグメントより小さい容量 (1~16,378byte) のメモリブロックを TPA セグメント (起動時に割り当てられたセグメント) もしくは新しく割り当てたセグメントに確保するルーチンを追加しました。これらのルーチンは、マッパーサポートルーチンのジャンプテーブルを拡張して使えるようにしています。ジャンプテーブルのアドレスはマッパーサポートの拡張 BIOS コールを使用して取得します。新しいルーチンの名前とアドレスは以下の通りです。~~

~~+30h: BKL_ALLOG~~

~~+33h: BLK_FREE~~

~~どちらのルーチンもその時点でページ 2 に割り当てられた RAM に対して実行されます。TPA セグメントやメモリマッパーで割り当てたセグメント、あるいはメモリマッパーでない RAM であっても、読み書きできるメモリがページ 2 にあれば使用可能です。しかしながら、ここでは説明の都合上、ページ 2 に割り当てたセグメントと仮定しています。~~

新しいルーチンの名前とアドレスは以下の通りです。

+30h : CALL_MAP

+33h : RD_MAP

+36h : CALL_MAPI

+39h : WR_MAP

5.1. CALL_MAP : マッパーRAM セグメント内ルーチンの呼び出し (+30h)

CALL_MAP	ジャンプテーブルの+30h
コール手順	IYH : スロット番号 IYL : セグメント番号 IX : 対象ルーチンのアドレス AF, BC, DE, HL : 対象ルーチンのパラメータ
戻り値	AF, BC, DE, HL, IX, IY : 対象ルーチンの戻り値

ルーチンはページ 1 内の指定されたスロットとセグメントを切り替えることによって呼び出されるため、ルーチン アドレスもページ 1 内になければなりません。

5.2. RD_MAP : RAM セグメントから 1 バイト読み取る (+33h)

RD_MAP	ジャンプテーブルの+33h
コール手順	A : スロット番号 B : セグメント番号 HL : 対象アドレス(上位 2 ビットは無視されます)
戻り値	A : 指定されたアドレスから読み込んだデータ F, BC, DE, HL, IX, IY 保存

5.3. CALL_MAPI : インラインルーチンを使用して、マップパーRAM セグメント内のルーチンを呼び出す (+36h)

CALL_MAPI	ジャンプテーブルの+36h
コール手順	AF, BC, DE, HL : 対象ルーチンのパラメータ
戻り値	AF, BC, DE, HL, IX, IY : 対象ルーチンの戻り値

このルーチンは以下のように呼び出します。

```
CALL CALLSEG

CALLSEG:
    CALL <CALL_MAPI のアドレス>
    DB mmeeeeeeeb
    DB <セグメント番号>
    ;no RET is needed here (ここに RET は不要)
```

mm はマップパーズロットで、標準マップパーサポートルーチンによって提供されるマップパー変数テーブル内のインデックス(0~3)です(“[MSX-DOS 2 プログラムインターフェース仕様](#)”の「5.2 マップパーサポートルーチン」または、“[MSX-Datapack Volume 3 turboR 版](#)”の「第 2 部 15 章 マップパーサポートルーチン」を参照)。

eeeeee は呼び出されるルーチンで、セグメントのアドレス 4000h から始まるジャンプテーブルのインデックス(0~63)です。つまり、0 は 4000h、1 は 4003h、2 は 4006h などを意味します。

マップパーズロットとセグメント番号の指定方法は奇妙ですが、呼び出し全体を 5 バイトにまとめることができます。これにより、BIOS ルーチン CALLF で通常行われるのと同じように、このルーチンをフックと共に使用できます。

5.4. WR_MAP : RAM セグメントに 1 バイト書き込む (+39h)

WR_MAP	ジャンプテーブルの+39h
コール手順	A : スロット番号 B : セグメント番号 E : 書き込む値 HL : 対象アドレス (上位 2 ビットは無視されます)
戻り値	A : 指定されたアドレスから読み込んだデータ F, BC, DE, HL, IX, IY 保存

5.5. UNAPI RAM ヘルパーの検出手順

Nextor は、これらの新しいマッパーサポートルーチンを既存の [UNAPI RAM ヘルパー仕様](#) と互換性を持たせるために、UNAPI RAM ヘルパー検出手順を実装しています。参考までに、検出手順を以下に再掲します。

RAM ヘルパーの存在を確認し、そのルーチンのアドレスを取得するには、EXTBIO (0FFCAh) を DE=2222h、HL=0、A=FFh で呼び出す必要があります。RAM ヘルパーがインストールされていない場合は、出力時に HL=0 が返されます。RAM ヘルパーがインストールされている場合は、以下のレジスタ値が返されます。

HL = ページ 3 のジャンプテーブルのアドレス
BC = ページ 3 の縮小マッパーテーブルのアドレス (指定されていない場合は 0)
A = ジャンプテーブルのエントリ数

Nextor の場合、以下のようになります。

- ① HL は CALL_MAP の位置 (マッパーサポートルーチンのジャンプテーブルの先頭からのオフセット +30h) を指します。
- ② BC は常にゼロとして返されます (縮小されたマッパーテーブルは、マッパーサポートルーチンが存在しない場合にのみ、UNAPI RAM ヘルパーに必須です)。
- ③ A は常に 4 になります (MSX UNAPI 仕様では WR_MAP ルーチンが定義されていないため 3 になります。これは互換性に影響のない変更です)。

5.6. 重大な変更のお知らせ

Nextor 2.1.0 (RC1) より前のバージョン (2.1.0 のアルファ版およびベータ版を含む) では、RD_MAP、WR_MAP、CALL_MAP、CALL_MAPI エントリポイントで使用されているマッパースポートルーチンのジャンプテーブル領域は、現在は利用できない 2 つのルーチン (BLK_ALLOC と BLK_FREE) で使用されていました。これは互換性を破る変更であり、これらのルーチンを使用している既存のアプリケーションでは変更が必要になります。

これらの削除されたルーチンは Nextor では使用されなくなりましたが、ソースコードは Nextor コードベースの一部として保持されています ([source/kernel/bank4/bkalloc.mac](https://github.com/nextor/kernel/blob/master/bank4/bkalloc.mac))。そのため、これらのルーチンを使用するアプリケーションがある場合は、そのファイルのコードをアプリケーションに組み込むだけで、ルーチンを直接呼び出すことができます。

削除されたルーチンに関するドキュメントは、[Nextor 2.0 プログラマーズリファレンス](#)でご覧いただけます。

6. その他の特徴

この章では Nextor がサポートするその他の特長について解説します。

6.1. _STROUT (09h) におけるエスケープシーケンス “ESC-Y” の不具合修正

_STROUT (09h) は “\$” (24h) で終わる文字列を出力します。エスケープシーケンスの中には、画面上の任意の位置にカーソルを移動させるものがあります。例えば、カーソルを (x, y) に移動したいときは ESC(1Bh), “Y” (59h), x+32, y+32 を指定します。例 (10, 5) であれば、ESC, “Y”, 42(10+32), 37 (5+32)、つまり 1Bh,59h,2Ah,25h となります。x は横方向、y は縦方向のカーソル位置で、オフセットとして 20h を加算した値を使用します。

バグはこのエスケープシーケンスで x または y に 4 が指定されたときに発生します。x または y が 4 の場合、オフセットの 20h(32) が加算された値が 24h となり、_STROUT (09h) における文字列出力の終了条件に一致してしまいます。そのため、MSX-DOS が文字列の終了マークと誤認識して出力を打ち切ります。

このバグは Nextor で修正され、ESC-Y は問題なく使用できます。

6.2. Nextor のバージョン変更

いくつかの MSX-DOS アプリケーションは MSX-DOS2.SYS のバージョンをチェックしていることはよく知られています。

(Nextor では NEXTOR.SYS が MSX-DOS2.SYS にあたります。)そして、もし MSX-DOS2.SYS のバージョンがアプリケーションが必要とするバージョンより小さいとき、特に 2.20 より小さいときに動作を中断することがあります。このようなアプリケーションは、Nextor でも NEXTOR.SYS のバージョンが 2.1 のため問題となります。

この問題の回避方法として、NEXTOR.SYS の 2.0 beta 2 からは、_DOSVER (6Fh) の戻り値を RAM に保管しておき、簡単に書き換えられるようにしています。NSYSVER.COM を使用することによって簡単に書き換えられます。NSYSVER.COM の詳細については [“Nextor 2.1 ユーザーマニュアル”](#) を参照してください。もし、プログラムで書き換えたいときは下記の手順となります。

1. MSX-DOS 上で、ページ 0 が TPA RAM セグメントのときに、0001h に保存してあるアドレス (ジャンプテーブルへのアドレス) を読み込みます。
2. 1. で得られた値に 32h を加算します。(ジャンプテーブルのサイズが 32h なので、ジャンプテーブルの次のアドレスになります。)
3. 2. で得られたアドレスが 16bit の数字を保管する場所です。_DOSVER (6Fh) 実行時に DE レジスタに入れて返す値が保管されています。

例えば、NEXTOR.SYS のバージョンを 2.31 に書き換えるには下記のコードとなります。

LD	IX, (0001h)
LD	BC, 32h
ADD	IX, BC
LD	(IX), 31h
LD	(IX+1), 02h

MSX-DOS2.SYS や NEXTOR.SYS のバージョンは BCD コードになっていることに注意してください。

もちろん、この変更は一時的なものです。書き換え後であっても、BASIC に行って“CALL SYSTEM”等で Nextor に戻るか、リセットなどの再起動によって NEXTOR.SYS が再度読み込まれると元の値に戻ってしまいます。

この機能は NEXTOR.SYS の 2.0 beta 2 以降においてのみ有効です。

7. Nextor の内部

7.1. ワンタイムブートキー

ワンタイムブートキーのメカニズムは、起動時にアドレス A100h にゼロで終了する署名文字列"NEXTOR_BOOT_KEYS"が見つかった場合に起動します。この場合、英数字キーの状態はキーボードから読み取られるのではなく、署名に続くバイトから取得されます(下表参照)。ビットが1にセットされている場合、キーが押されたとみなされます。

アドレス	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
A111h	7	6	5	4	3	2	1	0
A112h	F	E	D	C	B	A	9	8
A113h	N	M	L	K	J	I	H	G
A114h	V	U	T	S	R	Q	P	O
A115h			CTRL	SHIFT	Z	Y	X	W

現在、Nextor がブート時に実際に使用するキーはすべてではないことに注意してください(例: 6 から 9 の番号)。ただし、Nextor の将来のバージョンでテーブル内の現在使用されていないキーのいずれかが使用される場合、ワンタイムブートキーメカニズムを使用するときに、キーの状態はこのテーブルで定義された位置にあることが予想されます。

7.2. ディスクエミュレーションモード

このセクションでは、ディスクエミュレーションモードに関する詳細について説明します。

7.2.1. ディスクエミュレーションデータファイル形式

Nextor がエミュレーションセッションで使用するディスクエミュレーションデータファイルは、ヘッダーと各ディスクイメージファイルに関する情報を含むテーブルで構成されています。ヘッダーの内容は以下のとおりです。

オフセット	内容
00h ~ 0Fh	署名文字列 "NEXTOR_EMU_DATA",0
10h	ディスクイメージファイルテーブル内のエントリ数
11h	起動時にマウントするファイルの 1 から始まるインデックス
12h ~ 13h	エミュレーションセッション中に作業領域として使用するアドレス、またはこの領域を割り当てる必要がある場合は 0 (2 バイト、リトルエンディアン)
14h ~ 17h	未使用、常にゼロ (4 バイト)

ディスクイメージ ファイルテーブルの各エントリは以下のとおりです。

オフセット	内容
00h	ファイルを含むデバイスの番号 (0 = エミュレーション データ ファイルと同じ)
01h	ファイルを含む論理ユニットの番号 (デバイス番号が 0 でない場合)
02h ~ 05h	ファイルが始まる物理デバイスセクタ番号 (4 バイト、リトルエンディアン)
06h	ファイルのサイズ (セクタ単位、2 バイト、リトルエンディアン)

ディスクイメージファイルテーブルエントリ内のデバイス番号が 0 の場合、Nextor はディスクイメージファイルがエミュレーションデータファイルと同じデバイスおよび論理ユニットにあると想定します (この場合、テーブルエントリ内の論理ユニット番号は無視されます)。EMUFILE.COM ツールは、すべてのディスクイメージファイルが、作成中のエミュレーションデータファイルと同じデバイスおよび論理ユニットにある場合、すべてのエントリのデバイス番号を 0 に設定します。

ディスクイメージファイルテーブルの最後のエントリ以降のエミュレーションデータファイルの内容は、Nextor カーネルによって無視されます。EMUFILE.COM ツールは、ここにディスクイメージファイル名の表示可能なリストを配置します。このリストは、コマンドプロンプトで TYPE /B datafile を実行することで表示できます。

7.2.2. ディスクエミュレーションモードの実行

起動時に、Nextor はディスクエミュレーションデータファイルへのポインタを見つけると、ディスクエミュレーションモードに入ります。このポインタは、デバイス番号と論理ユニット番号、そして物理デバイスセクタ番号で構成されます。

このドキュメントとユーザーマニュアルでは「エミュレーションデータファイル」という用語を使用していますが、実際のファイルは必須ではありません。Nextor カーネルは、エミュレーションデータが配置されているセクタ番号のみを認識し、そのセクタがファイルの一部であるかどうかは考慮しません。通常、この情報を保存するにはファイルを使用するのが最も便利ですが、例えば FAT 直前の予約済みセクタをこの目的で使用するツールを開発することも可能です。

Nextor は、起動時に RAM 内で以下の情報を見つけると、ワンタイムディスクエミュレーションモードに入ります。

アドレス	内容
A000h	署名文字列 "NEXTOR_EMU_DATA",0
A010h	エミュレーションデータを含むデバイスの番号
A011h	エミュレーションデータを含む論理ユニットの番号
A012h	エミュレーションデータを含む物理デバイスセクタ番号 (4 バイト、リトルエンディアン)

上記の情報が見つからない場合、Nextor はプライマリ Nextor コントローラ内の使用可能なデバイスの最初のパーティションテーブルエントリで以下の情報を見つけると、永続ディスクエミュレーション モードに入ります。

セクタ オフセット	パーティションテーブル エントリオフセット	パーティション テーブルの内容	Nextor での内容
1BEh	+0	ステータスバイト	ビット 0 = 1b : ディスクエミュレーションモードに入る

1BFh	+1	スタート CHS	エミュレーションデータを含むデバイスの番号
1C0h	+2	スタート CHS	エミュレーションデータを含む論理ユニットの番号
1C1h	+3	スタート CHS	エミュレーションデータを含む物理デバイスセクタ番号の MSB(最上位バイト)
1C2h	+4	パーティションタイプ	ゼロ以外の指定が必要
1C3h	+5	エンド CHS	エミュレーションデータを含む物理デバイスセクタ番号の 3 番目のバイト
1C4h	+6	エンド CHS	エミュレーションデータを含む物理デバイスセクタ番号の 2 番目のバイト
1C5h	+7	エンド CHS	エミュレーションデータを含む物理デバイスセクタ番号の LSB(最下位バイト)

例えば、エミュレーションデータがデバイス 1、論理ユニット 2、セクタ 33445566h に配置され、パーティションが FAT16(パーティションタイプ 0Eh)で、ステータスバイトに「アクティブ」フラグが設定されている場合、永続エミュレーション用に設定されたパーティションテーブルエントリの開始位置は次のようになります(16 進数)。

81 01 02 33 0E 44 55 66

Nextor がエミュレーションモードに入る条件は、ステータスバイトのビット 0 が設定され、パーティションタイプコードが 0 以外(FAT に限らず他の任意の値も可能)であることです。また、デバイスと論理ユニットの情報はエミュレーションデータファイルのみを参照することに注意してください。ディスクイメージファイル自体は別のデバイスに配置されていても構いません。

また、注意すべき点として、Nextor はエミュレーション データが NEXTOR_EMU_DATA 署名で始まっているかどうかを確認します。署名がない場合はディスク エミュレーション モードに入らずに通常どおり起動します。

8. 変更履歴

この章では、Nextor のバージョン毎の変更履歴について解説しますが、アプリケーション開発の観点から重要な変更点のみを記載しています。一般的な仕様の変更点については [“Nextor 2.1 ユーザーマニュアル”](#) を参照してください。また、デバイスドライバ開発に関する変更点については [“Nextor 2.1 ドライバ開発ガイド”](#) を参照してください。

※訳注: v2.1.2 以降の変更については、GitHub の変更履歴(<https://github.com/Konamiman/Nextor/tags>)を参考にしました。

8.1. v2.1.3

- `_GETCLUS` が 16 バイトではなく 64 バイトのデータをコピーする問題を修正しました。

8.2. v2.1.2

- `_DOSVER` 関数は、MSX-DOS 1 モードでレジスタ C を変更しないようになりました。

8.3. v2.1.1

- EXTBIO 処理中のスロット管理に関するバグを修正しました。このバグにより、特定のスロット/サブスロットの組み合わせで 2 つの Nextor カーネルとマッパ RAM が存在する場合に、COMMAND2.COM がクラッシュしていました(#103)。
 - この修正は@uniabis によるものです。ありがとうございます！

8.4. v2.1.1 beta 2

- `_GDLI` は、ファイルがマウントされるときに、「開始クラスタ」と「開始セクタ」という 2 つの新しいフィールドを返します。

8.5. v2.1.1 beta 1

- 修正: 拡張 BIOS 処理コードによって DE レジスタペアが破損していましたが、ドライバー自身の EXTBIO フック処理コードによって破損した場合でも、常に保持されるようになりました。(#88)

8.6. v2.1.0

- FAT12 または FAT16 ボリューム内のクラスタに関する情報を取得するための新しい関数呼び出しを追加しました (#35)。

- セグメント番号とスロット番号の両方を指定して、マッパー RAM にアクセスするための、UNAPI RAM ヘルパー互換のマッパーサポートルーチンを追加しました (#34)。
- LUN_INFO によって返される新しいフラグにより、Nextor がデバイスとドライブの自動割り当てのためのデバイス検索時にデバイスを無視するように指示できます (#54)。
- デフォルトの DPB はバンク 0 と 3 の固定アドレス 7BAAh を取得するため、確実にカスタマイズできます。
- PROMPT ルーチンがアドレス 41E8h でドライバーから利用可能になりました (#42)。
- C で記述されたすべてのツールが再びコンパイルされるようになりました (#36)。
- **重大な変更** (アプリケーション開発者向け): ALL_BK および FRE_BK ルーチンは利用できなくなりました。

8.7. v2.1.0 RC 1

- _GETCLUS ファンクションコールが導入されました。
- UNAPI RAAM ヘルパー互換ルーチンが追加されました。
- 拡張マッパーサポートルーチン ALL_BK と FRE_BK が削除されました。

8.8. v2.1.0 beta 2

- _GPART はパーティションのステータスバイトを返すようになり、パーティションに関する情報ではなく、パーティションテーブルエントリを保持するデバイスセクタ番号を取得できるようになりました。

8.9. v2.1.0 beta 1

- _GDRVR は、ドライバーが DRV_CONFIG ルーチンを実装しているかどうかを示す追加フラグを返すようになりました。
- _MAPDRV は、ファイルのマウントを許可します。
- _GDLI は、ファイルがマウントされているドライブに関する新しい情報セットを返します。
- エラーコード「ファイルがマウントされています」(.FMNT, 0B2h) と、「無効なファイルサイズ」(.BFSZ, 0B1h) が導入されました。

9. 問い合わせ

Nextor 及び関連ツールの最新版は、Konamiman's MSX page からダウンロード出来ます。

<http://www.konamiman.com>

バグ報告や提案については、下記宛先までご連絡ください。

konamiman@konamiman.com

日本語翻訳に関する誤訳の報告や文章の改善については、下記宛先までご連絡ください。

aeg@mvp.biglobe.ne.jp / @Lithelia (Twitter/X)