Utilization:

If we need to write some program, which is working directly with harddisk, we don't have to write our own routines, but we can use routines from MDOS3.

Services calling:

You can call services after paging into divIDE memory. Because MDOS is acting like D80, we can use paging routines, which are already done:

```
SHADE       ld      a,$4F
            ld      de,TAB-26
            call    $25AB
            ld      hl,0
            ld      (TAB),hl
            ld      hl,$3EF7
            ld      (TAB+2),hl
            rst     0
            ret
TAB         dw      0
            dw      $3EF7
```

And for paging back into ZX ROM:

```
ZXROM       ld      a,32
            ld      (16119),a
            jp      $1700
```

All services are called with call $24 (36 dec), number of service is in A, parameters in other registers. No service changes **IY** register.

Example:

```
            ld      a,($3E6B)
            ld      e,a
            ld      a,4
            call    $24
```

Reading data from virtual floppy is done the same way as reading from physical drive. Changed are routines DREAD and DWRITE, which will be used by programmer. Working on lower level is useless. Catching errors works also the same way as by real media, that's because virtual floppy tries to emulate real floppy the best way possible.

For successful using of all services is necessary, that after booting MDOS3 user pressed NMI button. After then data about partitions on all connected devices are readed. Because after booting there is no floppy selected, we can suppose that first steps will be into the NMI menu. For all cases GET_VER services returns status of initialisation.

Services description:

0 **GET_VER**

out: L = day, H = month, DE = year, A = init MDOS3

changes: BC

desc: No input parameters, in registers HL,DE MDOS3 version is returned. Versions are not serially numbered, instead of that date is used. The higher is DEHL the higher is MDOS3 version.
If A = 0 (and z flag is set) MDOS3 was not initialised and it is not possible to use properly services marked with asterisk and tables are empty. Initialisation means, that user MUST press NMI at first.

1 **GET_MAPTAB**

out: HL = MAPTAB table adress
desc: No input parameters, in HL is returned adress of MAPTAB table.

2 GET_GEOM
out: HL = GEOM table adress
desc: No input parameters, in HL is returned adress of GEOM table (disks geometry).

3 GET_AKTPAR
out: HL = AKTPAR table adress
desc: No input parameters, in HL is returned adress of AKTPAR table. AKTPAR is table in which are details about actual partition ('W' key in NMI menu).

4 GET_PART_TABLE
out: HL = PART_TABLE table adress
desc: No input parameters, in HL is returned adress of PART_TABLE table. This table contains data about detected partitions on connected drives.

5* DRIVEPART
in: E = drive number (0-3)
out: IX = adress of partition details, which belong to floppy in selected drive, A = partition number, flag Z = error (drive is empty)
changes: BC,HL,DE
desc: Input parameter is number of drive vith virtual floppy (need not to be mapped on HDD), in IX will be adress of details about partition, which contains that floppy. In A register will be partition number. If drive is empty, flag Zero will be set.

6 PREPOCET
in: IX = drive system variables adress (ie: $3E00)
out: IX = MAPTAB adress for appropriate drive
desc: Will change adress of system variables to MAPTAB adress, for example if you want to know where is appropriate drive mapped. Service is not resistant against bad input.

7 CMMAPTAB
in: E = drive number (0-3)
out: IX = MAPTAB adress for appropriate drive
desc: Alternate way how to get MAPTAP for selected drive. Service is not resistant against bad input.

8 DRVCMPS
in: E = physical drive number (fd0, fd1)
out: IX = adress of drive (fd0 or fd1) parameters
changes: BC
desc: For physical reset of drives, values are stored on different place. After mapping change are parameters taken from there. These parameters are set after mapping on fd0 or fd1. Until then are not usable.

9 DIVIDE0
in: DEHL = dividend, C = divisor
out: DE = result, A = remainder
changes: B
desc: 32-bit division. DE = higher part of "four-register", lower part of "four-register"

10 INC32
in: 32-bit number
out: 32bit number, flag Z = BCDE = 0
desc: adds 1 to "four-register" BCDE (in BCDE is ussualy stored LBA adress of sector).

11 DEC32
in: BCDE = 32-bit number

out: BCDE = 32bit number, flag Z = BCDE = 0, flag NC = BCDE=-1
desc: substracts 1 from "four-register" BCDE (in BCDE is ussualy stored LBA adress of sector).

12 ADD32HL
in: BCDE = 32-bit number, HL = 16-bit number
out: 32-bit number, flag C = overflow
desc: adds HL to "four-register" BCDE.

13 DEC32HL
in: BCDE = 32-bit number, HL = 16-bit number
out: BCDE = 32-bit number, flag C = overflow
desc: substracts HL from "four-register" BCDE.

14 ADD1693
in: BCDE = 32-bit number
out: BCDE = 32-bit number, does not set flags
desc: adds 1693 to "four-register

15 DEC1693
in: BCDE = 32-bit number
out: BCDE = 32-bit number, does not set flags
desc: substracts 1693 from "four-register" BCDE

16* NEXTDISK
in: BCDE = 32-bit number as pointer to virtual disk, IX = pointer to partition table appropriate to drive
out: BCDE = 32-bit number as pointer to next virtual disk, flag C = out of partition
changes: HL
desc: Sets BCDE to next disk, if we get out of partition, C flag is set. IX can be returned by service 4. Service does not test, if there is valid virtual disk on specified sector.

17* PREVDISK
in: BCDE = 32-bit number as pointer to virtual disk, IX = pointer to partition table appropriate to drive
out: BCDE = 32-bit number as pointer to previous virtual disk, flag NC = out of partition
changes: HL
desc: Sets BCDE to previous disk, if we get out of partition, NC flag is set. IX can be returned by service 4. Service does not test, if there is valid virtual disk on specified sector.

18 NASDRNMI
changes: HL,BC,DE,IX
desc: Service set system into correct state. If application changed mapping in MAPTAB table, calling this service is needed, to do changes in drives parameters. It will also perform return of physical drives' heads to the beginning of the disk and HW reset of physical drives after first fd0 or fd1 select.

19 BACKSCR
desc: Service will store VRAM1 into divIDE memory. It can be restored with service 20.

20 RESTSCR
desc: Service will restore VRAM1 from divIDE memory, stored by service 19.

21 SET_BOOTDISK
in: E = drive number (0-3)
changes: HL, BC, DE, IX
desc: Disk from which boot was performed is set into drive. Service will perform MAPTAB table change, but does not set right mapping to HDD. Original disk is memorized.

22 RES_BOOTDISK

in: E = drive number (0-3)
changes: HL, BC, DE, IX
desc: Sets back, what did service 20. If service is called without previous usage of service 20, nonsencial valuse are set into MAPTAB table.

23 CLSMAPTA
changes: HL,BC,DE,IX
desc: Restores MAPTAB and AKTPAR tables as they were after booting.

24 DRVSELS
in: E = physical drive number (0-1)
desc: Service will rev up physical drive (fd0,fd1).

25* READHDD
in: BCDE = LBA sector, HL=data, 4th bit of B register = master/slave device
out: BC = error number
changes: HL, BC, DE, IX
desc: Reads sector. It is converted to CHS using disk geometry. Master/slave is set using 4th bit of B register. Error number is the same as in output of DREAD routine. For data reading it is recomended to set disk into drive and use floppy sector reading routines (DREAD, BREAD). If operation is successful, then HL=HL+512.

26* WRITEHDD
in: BCDE = LBA sector, HL=data, 4th bit of B register = master/slave device
out: BC = error number
changes: HL, BC, DE, IX
desc: Writes sector. It is converted to CHS using disk geometry. Master/slave is set using 4th bit of B register. Error number is the same as in output of DWRITE routine. Service does not check WRITE PROTECT of drive. For data writing it is recomended to set disk into drive and use floppy sector writing routines (DWRITE, BWRITE). If operation is successful, then HL=HL+512.

27 TST_BSY
in: 4th bit of B register = master/slave
out: A = 0 + flag Z: ready, A=255 + flag NZ: busy
desc: Tests or waits (cca 2 secs) untill BUSY bit in status register is not set. Then is possible to send into register parameters and commands. All HDD read/write commands use this command and it is not necessary to use it before.

28 READIDATA
in: HL = data, 4th bit of B register = master/slave
out: BC = error number
changes: HL, BC, DE
desc: Reads ID sector from ATA device to adress in HL. That means, that this service is sending ECh command. Error output is the same as in service 24 (and internal routines for drive handling, DREAD).

29 READIDATAPI
in: HL = data, 4th bit of B register = master/slave
out: BC = error number
changes: HL, BC, DE
desc: Reads ID sector from ATAPI device to adress in HL. That means, that this service is sending A1h command. Error output is the same as in service 24 (and internal routines for drive handling, DREAD).

Tables:

In text many tables are mentioned. In following text structures of these tables will be explained. The most

basic table is MAPTAB, on which whole MDOS3 is standing.

```
MAPTAB      db      2,0,0,0,0    ;drive A:
            db      3,0,0,0,0    ;drive B:
            db      4,0,0,0,0    ;drive C:
            db      5,0,0,0,0    ;drive D:
            db      3            ;emulations TAPE
            db      0            ;Rewrite mode
```

Drives mapping table. It defines whether drive is mapped to fd0, fd1 or HDD. If drive is mapped to HDD, table contains number of LBA sector, where floppy is stored. For every drive are defined 5 bytes, for tape emulator is defined one byte.

offset 0: mapping; Number 0=fd0, 1=fd0, 2-5 = HDD, for drive A is 2, for B is 3, for C is 4 and for D is 5 and for tape emulator 6. If we are not emulating, zero is here.

offset 1-4: virtual disk. There is stored number of LBA sector, where virtual disk is stored. Thsi sector points to infosector. Number is stored from lower to higher byte. In last (4) byte can be 4th bit set to 0 in case of MASTER device, to 1 in case of SLAVE device.

TAPE emulatiom:

Rewrite mode: 0 = When saving file with name that already exists, standard question „Rewrite old file“ will be displayed.

1 = More files with the same name can be saved.

2 = Will rewrite old file without further questioning.

```
GEOM        db      $D2,$03,$08,$20    ;master
            db      $EA,$01,$02,$20    ;slave
```

Geometry of MASTER and SLAVE.

For every device (MASTER, SLAVE) there are defined 4 bytes, in which geometry is stored.

offset 0-1: Cylinders

offset 2: Heads

offset 3: Sectors

```
AKTPAR      db      1            ;number of partition
            db      $20,0,0,0    ;begin partition
            db      $E0,D1,3,0   ;length partition
```

In this table are stored informations about actual partition in NMI menu ('W' key). Partition number can be one of these values:

offset 0: 1-4: primary partition on MASTER device

5-8: primary partition on SLAVE device

254: the whole MASTER device

255: the whole SLAVE device

Note: Because in NMI menu the whole disk (ie hda) can be selected, is it maked this way. In that case beginning of partition is set to place, where first virtual disk was found and length of partition is re-counted to the end of disk.

offset 1-4: Partition number: LBA sector, where partition begins

offset 5-9: Partition length: LBA adress of partition length. Last LBA sector can be counted this way: beginning+length + 1.

```
PART_TABLE  db      1            ;partition number
            db      $20,0,0,0    ;beginning of partition
            db      $E0,D1,3,0   ;length of partition
            db      6            ;partition number
            db      $80,$3E,0,0  ;beginning of partition
            db      0,$3C,0,0    ;length of partition
            db      0            ;end character
```

In this table are data about partitions found during initialisation (first entering of NMI menu). There are only MDOS3 partitions or the whole disk (if there is no MDOS3 partition). For each partitions is assigned 9 bytes. These 9 bytes have the same structure as AKTPAR table:

offset 0: 1-4: primary partition on MASTER device
5-8: primary partition on SLAVE device
255: the whole SLAVE disk, first floppy image found, data recalculated
254: the whole MASTER disk, first floppy image found, data recalculated
253: the whole SLAVE disk, first floppy image has not been searched yet
252: the whole MASTER disk, first floppy image has not been searched yet
Table ends with zero.

Note:
It is marked this way, because the whole disk (ie. hda) can be selected in NMI menu. In that case we recognize two cases:

1) Selection of disk was made, NMI found first floppy image and data are recalculated.
2) Selection of disk was not made yet, therefore we do not know where first floppy image begins.

If first floppy image has not been found yet, beginning of partition is set to sector 0 and length is set to lenght of the whole disk in sectors. If we already have first image, is partition begining set to this image and length is recalculated in order to fit to fit to the last sector of disk.