NST's Monitor Extension V7.01

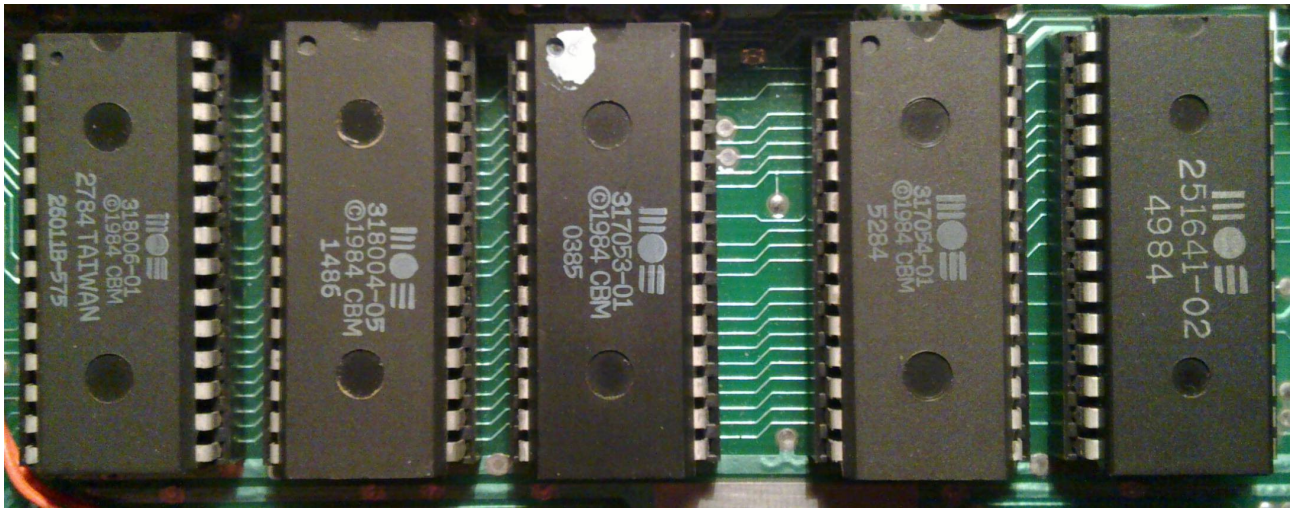
Áttekintés:

- A gépbe gyárilag épített **TEDMON** funkciók bővítése
- „Illegális kódok” kezelése
- Képernyő mindkét irányú „görgetése”
- Lemezes meghajtó memória kezelése
- Lemezes blokk kezelés
- Memóriatörlő funkció
- Stb.

A használathoz szükséges teendők:

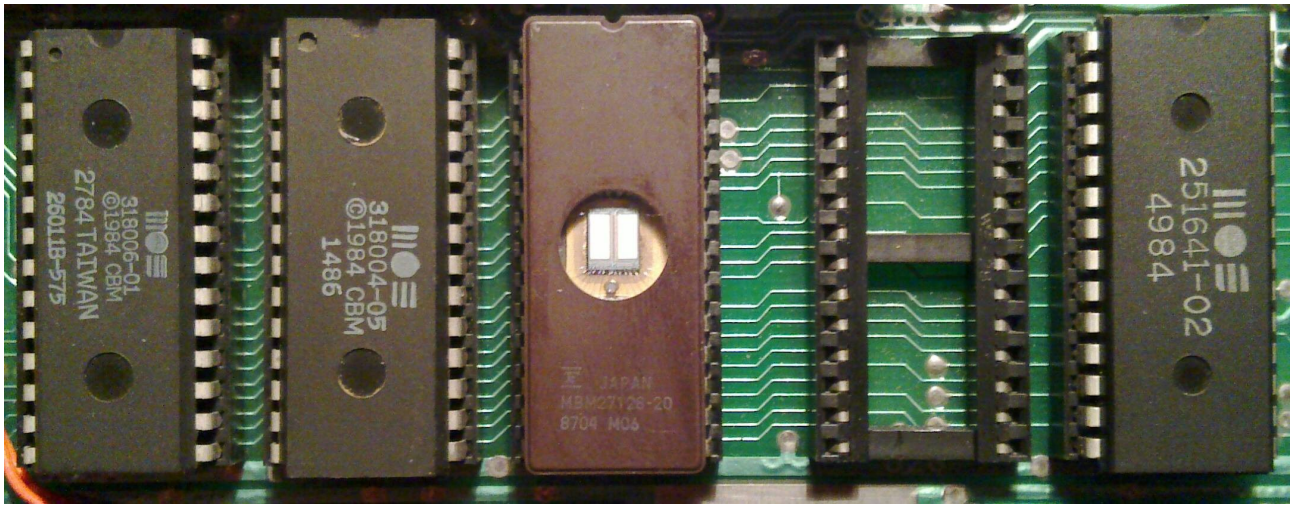
A bővítés (ezentúl **BMX**) a **plus/4** gyárilag (már a beépítés pillanatában is elavult) extra szoftvereinek a helyére kerül. Kipróbálni a legegyszerűbben valamelyik emulátor, például a **YAPE** (http://yape.homeserver.hu/) vagy a **plus4emu** (http://plus4emu.sourceforge.net/) segítségével lehet. Ezeknél a FUNCTION-LO ROM helyére kell a **BMX** ROM tartalmát betölteni, a FUNCTION-HI üresen marad.

Eredeti gépen is egyszerű a megoldás, itt a megfelelően felprogramozott (E)EPROM-ot (pl. 27(C)128) kell a megfelelő helyre berakni. Az eredeti felállás így néz ki:



Ezen IC-k tartalmazzák (balról jobbra) a BASIC, a KERNAL, a FUNCTION-LO és a FUNCTION-HI adatait, ezek ROM-ok. (Az ötödik IC (**251641-02** feliratú) az FPLA, az most nem érdekes.)

Az eredeti ROM-ok közül az **U25** (**317053-01** felirat) és az **U26** (**317054-01** felirat) pozíciójú ROM-ot kell a foglalatából kiemelni, majd az **U25** helyére a megfelelően felprogramozott (E)EPROM-ot beilleszteni:



Az U26 (egyelőre) üresen marad. A gépet összeszerelve ezután már működőképes a **BMX**, semmi más átalakítási teendő nincs.

Kompatibilitás:

A rendszerrel való kompatibilitás fontos szempont, általában a használat nem is okoz gondot. Két probléma szokott felmerülni, az első szinte mindig előkerül annál, aki még nem használt ilyen jellegű bővítést. Ez egy hibás programozói szokásból fakad, amit – valószínűleg – a „**Gépi kódú programozás kezdőknek és haladóknak**” című (amúgy igen informatív és megbecsült) könyv terjesztett el:

> Önálló megszakító rutin írásának egyszerűbb formája az, amikor a $FCB3–$FCC8 és a $CE00–$CE0B ROM-programokat még használjuk. Mindezek feltétlenül működnek, amíg a $FFFE–$FFFF címek tartalma $FCB3. Ennél a módszernél is a $0314-es indirekt címet kell átírni úgy, hogy a saját rutinunkra mutasson. Ennek azonban JMP $FCBE-vel vagy JMP $FCC3-mal kell végződnie. (Mivel ritkán írunk új megszakító rutint a bővítő ROM-ok programjaihoz, gyakorlatilag a JMP $FCC3 is mindig alkalmazható.)
>
> 89

„(Mivel ritkán írunk új megszakító rutint a bővítő ROM-ok programjaihoz, gyakorlatilag a JMP $FCC3 is mindig alkalmazható.)”

Ez a zárójeles rész a probléma forrása. Ha a megszakítási rutin $FCC3 címen fejeződik be, akkor elmarad a megszakítás pillanatában érvényes ROM-BANK visszakapcsolása. Mivel a **BMX** egy másik ROM-BANK-ban található, de a megszakítási rutin nem kapcsolja azt vissza, így a „folytatás” a BASIC ROM-ban próbál futni, ami nem kívánt „borulásokat” eredményez. Ha a ROM megszakítását kell kibővíteni, akkor $FCC3 helyett $FCBE címre ugrással KELL befejezni a rutint.

A másik gond akkor jelentkezik, ha egy program „úgy rak rendet” a memóriában maga után, hogy a KERNAL-os inicializáló rutinok közül meghívja azt is, amelyik a ROM-ok „feltérképezését” végzi. A bővítésnek „hála” ez a rutin nem tér vissza, mert a **BMX** átveszi a RESET végrehajtását. (Normál esetben a ROM-ok inicializálása RESET esetén fut csak le.) Emiatt bizonyos esetekben szükség lehet a **BMX** kikapcsolására.

Memóriahasználat szempontjából a bővítés igyekszik „minimalista” lenni; amilyen részek használva vannak, azok az eredeti **TEDMON** által is használtak többnyire. Ami kivétel: $05F5-től kezdődően kimásolódik pár rutin (RAM-olvasáshoz, lapozásokhoz...), amik szükségesek a futáshoz. A **3-plus-1** indító rutinja eredetileg itt van, a **BMX** rutinjai azonban egy kicsit túlnyúlnak az eredeti lapozó/indító kódon. A **TEDMON** a parancsok beviteléhez használ egy puffert, ez eredetileg $0200..$02FF területen található. A **BMX** a $0200..$0257 területet használja csak (mint a BASIC interpreter), itt 80 karakternyi hely van, ez a „legbonyolultabb” MONITOR parancsoknak is elegendő. Ezt a területet használja még a memória műveletek alatt puffernek, emiatt ezek módosítása / felülírása közvetlen parancsokkal „ellenjavallt”.

Funkciók:

Bekapcsolás illetve a RESET felengedése esetén pár gomb nyomva tartásával „szabályozhatók” bizonyos funkciók:

- A **Control** + **Space** + **Clear/Home** gomb egyidejű nyomásakor egy memóriatörlés történik
- A **2** gomb nyomásakor a **BMX** kikapcsolódik és úgy is marad
- Az **1** gomb nyomásakor a kikapcsolt **BMX** visszakapcsolódik, és úgy is marad
- A **Run/Stop** gomb nyomásakor a MONITOR indul el (mint alapállapotban)

A **BMX** bekapcsolt állapotát a keret illetve háttérszín megváltozása jelzi, a háttérszín egy, a keretszín kettő szinttel alacsonyabb fényerőre áll be. A „kikapcsolt” állapotot tárolja a gép, a $0123 memóriacímre (ez a verem „legteteje”, ami „feljebb” (alacsonyabb címen) van, azt a rendszer használja) $44 („D” mint *Disabled*) kerül ekkor. Ha ezt itt megtalálja, akkor a **BMX** inicializáló rutinja úgy tér vissza, mintha nem csinált volna semmit. Ebben az esetben nem veszi át a RESET végrehajtását. (Kompatibilitási szempontból előnyösebb lenne ehhez valamilyen hardveres segítség, de azt egyszerű ROM-cserével nem lehet megvalósítani. Emulátor esetén a ROM-okat azért könnyű kapcsolgatni, de nem kényelmes.)

A MONITOR parancsai az eredeti **TEDMON** funkcionalitását bővítik ki, ha a **BMX** valamit „nem tud feldolgozni”, átadja az eredeti ROM-nak a feladatot. A leírásban azok a parancsok szerepelnek, amik a **TEDMON**-hoz képest új / kibővített funkciójúak.

Azon parancsok, amik a memória tartalmát listázzák ki a képernyőre valamilyen formában, „görgethető” üzemmódban is működnek. Ha a kurzor a kép alsó sorában (illetve a beállított „ablak” alsó sorában) van, akkor a „LE” kurzormozgató gombbal újabb sorok íródnak ki a képernyőre. Ugyanez igaz felfele is; a kép („ablak”) felső sorában „FEL” gombot nyomva visszafele lehet haladni.

A parancsok nevei – eltérően az eredeti **TEDMON**-tól – nem csak egy karakter hosszúak. A több karakteres nevek a BASIC módján rövidíthetőek. (Tehát a második vagy későbbi karakter SHIFT mellett lenyomva a többi elhagyható.)

Az összes szám paramétert HEXADECIMÁLIS értékként kell megadni. (Ahol nem, ott külön jelezve van.) A paraméterek egymástól való elválasztására a szóköz vagy a vessző használható, ugyanúgy mint a **TEDMON** esetén.

A MONITOR (kibővített) parancsai:

```
HELP

 NST'S MONITOR EXTENSION BY BSZ
 FW VERSION 7.01 - B108 - 131126
 WARNING: ADDICTIVE!

HELP        KILL        GRAPHIC     DIRECTORY
UNIT        TM          TDISK       FDISK
@           *R *W       O           GR GZ G
D           A . ,       M >         I '
E [         F           T           C
H           $ % #       X

HELP [COMMAND] TO DETAILS.
```

HELP:

```
HELP: COMMANDS HELP
  USAGE:
HELP = LIST ALL COMMANDS
HELP <COMMAND> = COMMAND HELP
```

Segítség kérése. Paraméter nélkül megjeleníti az elérhető parancsok listáját, paraméternek valamelyik parancsot megadva a parancshoz tartozó segítséget írja ki. (Ez (a szöveg) elég sok helyet foglal, a későbbiekben esetleges memóriahiány esetén a „magyarázó” funkció kikerülhet.)

KILL:

```
KILL: DISABLE EXTENSIONS
  USAGE:
KILL = DISABLE ALL EXTENSIONS
KILL M = DISABLE MONITOR EXTENSION
```

A **BMX** (különböző funkcióinak) kikapcsolása. Paraméter megadása nélkül MINDEN funkciót kikapcsol, és beállítja a „kikapcsolt” állapotot. (RESET után sem indul el automatikusan, vissza kell a megfelelő kombinációval kapcsolni!) **M** paraméterrel csak a MONITOR bővítményt kapcsolja ki, ez RESET esetén újra visszakapcsolódik.

GRAPHIC:

```
GRAPHIC: SWITCH GRAPHIC MODES
  USAGE:
GRAPHIC = SWITCH BACK TO CHAR MODE
GRAPHIC M = SELECT GRAPHIC MODE M
GRAPHIC M BADR = BITMAP ADDR
GRAPHIC M BADR AADR = BITMAP+ATTR ADDR
```

A TED grafikus üzemmódját lehet vele ki/bekapcsolni. Paraméter nélkül visszakapcsol normál karakteres üzemmódra. Az első paraméter a grafikus üzemmód száma (ugyanaz mint a BASIC esetén, **0**, **1**, **2**, **3**, **4**), a második paraméter a BITMAP címe (Ez **0000**, **2000**, **4000**, **6000**, **8000**, **A000**, **C000**, **E000** illetve ezek „rövid” változata, **00**, **20**, **40**, **60**, **80**, **A0**, **C0**, **E0** lehet) a harmadik az attribútum memória címe („színmemória”, ezek **0000**, **0800**, **1000**, ... címek, illetve ezek „rövid” változata, **00**, **08**, **10**, ...).

DIRECTORY:

```
DIRECTORY: PRINT DISK DIRECTORY
  USAGE:
DIRECTORY = DIR UNIT8 DISK0
DIRECTORY UN = DIR UNITN DISK0
DIRECTORY UN DM = DIR UNITN DISKM
```

A meghajtóban lévő lemez tartalomjegyzékét listázza ki. Paraméter nélkül a beállított (vagy alapesetben 8-as) egységszámú meghajtó 0-ás lemezéről kéri a listát. Másik egységszámhoz meg kell azt adni **Ux** formában. (Például a 9-es egységszámról **U9**, a 10-esről **UA** (!) paraméterrel kérhető lista.) Az adott meghajtóban levő 1-es lemez (természetesen ez csak a két lemezt kezelni tudó meghajtóknál érdekes) listázásához a **D1** paramétert is meg kell adni. (Az eredeti BASIC DIRECTORY parancsa a lemez számát nem kezeli. Szintaktikailag kiértékeli, de a meghajtó felé már nem küldi át, így ott a lemez száma – mint paraméter – nem számít. A **BMX** DIRECTORY parancsa ezt helyesen kezeli.)

UNIT:

```
UNIT: CHANGE DRIVE UNIT
  USAGE:
UNIT = PRINT CURRENT UNIT NO
UNIT N = SET DRIVE UNIT N
```

Az alapértelmezett meghajtó egységszáma állítható be a segítségével. Paraméter nélkül kiírja a jelenlegi egységszámot (alapértelmezett 8). Paraméterként az egységszámot megadva átkapcsolja. Átkapcsolás után az összes parancs, ami a meghajtót valamilyen módon kezeli, ezen egységszámú meghajtót használja. (Az egységszám HEX érték, a 10-es eszköz eléréséhez **A** paramétert kell megadni!)

TM:

```
TM: TEDMON FALLBACK
  USAGE:
TM COMMAND = EXECUTE COMMAND BY TEDMON
```

TEDMON funkció hívása. A „**TM**” parancs után ami a sorban van, azt a ROM-ba épített **TEDMON** értékeli ki és hajtja végre. (Ha szükség lenne átmenetileg a „gyári” működésre, ezzel el lehet érni.)

TDISK:

```
TDISK: WRITE DATAS FROM RAM TO DRIVEMEM
  USAGE:
TDISK SADR EADR DADR = WRITE DATAS (U8)
```

Adatok mozgatása a gép memóriájából a kiválasztott (alapból 8) egység memóriájába. A forrás mindig a **plus/4** RAM memóriája, a cél a kiválasztott meghajtó RAM memóriája (vagy egyéb területei). Az 1541-II esetén a $8000..$FFFF tartomány írása ellenjavallt! Első paraméter a kezdőcím, a második a végcím, ez a terület másolódik. A harmadik paraméter a meghajtó memóriájának a kezdőcíme, ide történik az adatmozgatás.

FDISK:

```
FDISK: READ DATAS FROM DRIVEMEM TO RAM
  USAGE:
FDISK SADR EADR DADR = READ DATAS (U8)
```

Az előző parancs ellentéte, adatok mozgatása a kiválasztott meghajtó memóriájából a **plus/4** memóriájába. Az első paraméter a kezdő, a második a vége cím, ezek a meghajtó memóriából másolandó területet jelentik. A harmadik paraméter a **plus/4** memóriájának a kezdőcíme, innentől kezdve íródnak be azok az adatok, amik a meghajtóból kimásolódnak.

@:

```
@: GET DRIVE STATUS / SEND COMMAND
  USAGE:
@ = GET DRIVE STATUS (U8)
@ COMMAND = SEND COMMAND TO DRIVE (U8)
```

A kiválasztott meghajtó (alapból 8) állapot (STATUS) lekérdezése, illetve parancs küldése. Paraméter nélkül állapotlekérdezés történik, ha van valami paraméter, akkor azt a meghajtó megkapja a parancs-csatornájára. (Például egy „**@N:LEMEZ,XY**” parancs elindít egy lemezformázást, a lemez neve „**LEMEZ**” lesz, az ID-je pedig „**XY**”. A „**@I**” végrehajt egy lemez inicializálást. Stb...)

*R:

```
*R: READ SECTOR FROM DISK (U8) TO RAM
  USAGE:
*R TRACK SECTOR DADR = READ TR:SEC
```

A kiválasztott meghajtóban (alapból 8) lévő lemezről ezzel a paranccsal egy szektort lehet olvastatni a **plus/4** memóriájába. Első paraméterként a TRACK számát, másodikként a SECTOR-ét, harmadikként pedig a gép memóriájának a kezdőcímét kell megadni. A memória címe mindig $00-ra végződik, tehát **1234** nem adható meg, $1200..$12FF tartományba olvassa a szektort ilyen paraméter esetén. (A memória címe „rövid” módon is megadható, **12** megadásakor is $1200..$12FF lesz a célterület.) Az összes paraméter természetesen HEX érték! (Például a lemez BAM szektorát a „***R 12 00 1200**” parancs segítségével lehet beolvasni, $1200..$12FF területre. Ez a 18-as sáv 0-ás szektora.)

*W:

```
*W: WRITE SECTOR FROM RAM TO DISK (U8)
  USAGE:
*W = WRITE PREVIOUSLY READED SECTOR
*W TRACK SECTOR SADR = WRITE TR:SEC
```

A kiválasztott meghajtóban (alapból 8) lévő lemezre egy szektort lehet felírni ezzel a paranccsal. Paraméter nélkül az előzőleg „***R”** paranccsal beolvasott szektort írja vissza ugyanoda, ahonnan a kiolvasás megtörtént. Ha van paraméter, akkor azt a TRACK, SECTOR, memóriacím módon kell megadni, ugyanúgy mint a „***R”** parancsnál.

O:

```
O: CONFIGURE MONITOR
  USAGE:
O = PRINT CURRENT SETUP
O <N> = SELECT ROM BANK <N>
O R = SELECT RAM
O Z = SELECT DRIVE (U8)
O I = EN/DIS ILLEGAL OPCODES
```

A MONITOR konfigurációja módosítható a parancs segítségével. A **TEDMON** egyetlen ilyen paramétert ismer, a ROM/RAM olvasás kapcsolgatását. Ezt a $07F8 címen levő BYTE 7-es bitje állítja. A **BMX** esetében is ez a BYTE szolgál az üzemmódok tárolására, de itt több bit is használatban van, ezért „nem célszerű” ezt a memóriát kézzel állítgatni. (A B7 itt is a ROM/RAM olvasást kapcsolgatja, tehát a „**>07F8 80**” parancs működik, az történik ami szokott. Azonban a „**>07F8 FF**” már mást is kapcsol, így nem várt működés lesz a végeredmény!”) A parancs paraméter nélkül kiírja a jelenleg beállított konfigurációt. Paraméterként a következőket lehet megadni:

- **0**..**F**: A megadott számú ROM BANK elérését kapcsolja be. Tehát nem csak a BASIC + KERNAL terület olvasható, hanem a bővítő ROM-ok tartalma (beleértve a **BMX** ROM-ot) is! Az $FB-re történő ROM-BANK-szám írogatást (amivel a **TEDMON**-t rá lehetett venni a többi BANK alsó felének az olvasására) nem lehet ezentúl megcsinálni, mivel a **BMX**

aktívan használja a lapozást, ez a cím sokszor felülíródik.

- **R**: A RAM olvasást ezzel a paraméterrel lehet bekapcsolni. A ROM olvasására visszakapcsolni a **0**..**F** paraméterek valamelyikével (célszerűen a **0**-val) lehet.
- **Z**: A lemezmeghajtó elérésének a kiválasztása. Ezentúl az ÖSSZES, memóriára is vonatkozó parancs a MEGHAJTÓ MEMÓRIÁJÁBAN DOLGOZIK! (A felhasználó számára ekkor „úgy tűnik", mintha a meghajtó memóriája lenne a gépé, a meghajtón futna a MONITOR program.) Kivételt jelent ezek alól a „**G"** parancs (van külön a meghajtó memóriájában levő programot indító parancs), illetve az összes olyan, ahol direkt a gép-meghajtó közötti memóriamozgatás a parancs célja. De a többi („**A**", „**D**", „**M**", „**>**", ...) mind a meghajtó memóriájában dolgozik. Az 1541-II esetén a $8000..$FFFF tartományt nem szabad írni! Visszakapcsolni a gép memóriájának az elérését az **R** (RAM), vagy a **0**..**F** (ROM) paraméterrel lehet.
- **I**: Illegális kódok kezelésének a be/kikapcsolása. Ha be van kapcsolva, akkor nem csak a megjelenítés engedélyezett („**D"** parancs), hanem a bevitel is („**A"** parancs)!

G:

```
G: GO
  USAGE:
G = GO PROGRAM IN SELECTED ROM
G SADR = GO PROGRAM IN ROM FROM ADDR
```

Program indítása, megfelel a **TEDMON** „**G**" parancsának. Paraméter nélkül a „mentett" címtől indítja a futást, paraméterként a kezdőcímet megadva meg onnan. A $8000..$FFFF területen a ROM-ot indítja! Indítás előtt a „mentett" regiszterértékeket beállítja. (Ezt a „mentést" lehet az „**R**" **TEDMON** paranccsal megnézni.)

GR:

```
GR: GO RAM
  USAGE:
GR = GO PROGRAM IN RAM
GR SADR = GO PROGRAM IN RAM FROM ADDR
```

Ugyanaz mint a „sima" „**G**", de indítás előtt a megszakítás tiltódik, illetve a RAM lapozódik felülre. A $8000..$FFFF tartományban is a RAM-ban lévő kód indul el. Paraméter nélkül a „mentett" címről indít, paraméterrel meg onnan. A regiszterek itt is beállítódnak indítás előtt!

GZ:

```
GZ: GO DRIVE (U8)
  USAGE:
GZ SADR = GO PROGRAM IN DRIVE FROM ADDR
```

Programindítás a meghajtó memóriájában. (Ezt természetesen a meghajtó processzora fogja futtatni.) Itt kötelező címet megadni, illetve nincsenek a regiszterek „mentésből" beállítva, mivel a DOS ezt nem teszi lehetővé.

D:

```
D: DISASSEMBLE
   USAGE:
D = CONTINUE DISASSEMBLING
D SADR = DISASSEMBLE FROM ADDR
D SADR EADR = DISASS. FROM ADDR TO ADDR
```

A memória „diszasszemblálása”; a memória „dekódolása” és kiírása „mnemonikok” és ezekhez esetleg tartozó paraméterek formájában. Paraméterként semmit, kezdőcímet vagy kezdő és végcímet lehet megadni, működése értelemszerű. Ha az illegális kódok engedélyezve vannak, azokat is megjeleníti. A képernyőre kerülő listát a FEL/LE kurzorvezérlő gombokkal görgetni lehet (ugyanúgy mint a többi listát), a LE irány simán működik. A FEL iránynál „visszafele” kell a memóriában haladni, ez nem mindig egyértelmű. ELŐFORDULHAT TÉVESZTÉS, de az esetek nagy részében jól működik a funkció!

A / . / ,:

```
A: ASSEMBLE
.: ASSEMBLE, SEE [A] COMMAND
,: ASSEMBLE, SEE [A] COMMAND
Á: ASSEMBLE
   USAGE:
A DADR OP OP OP MNE OPER = ASSEMBLE
```

„Asszemblálás”, a beírt „mnemonikot” és paramétereit gépi kódra fordítja és beírja a memóriába. (Mindhárom parancs ugyanazt a funkciót látja el.) Ha az illegális kódok kezelése be van kapcsolva, azokat is be lehet így írni. A **TEDMON**-hoz képest egy változás, hogy a „**D**”-vel „diszasszemblált” lista „opkód” részén is lehet módosítani bizonyos megkötésekkel. (Például egy paraméter címét ott átírva is javítódik a memória.)

M:

```
M: MEMORY DUMP
   USAGE:
M = CONTINUE MEMORY DUMP
M SADR = DUMP FROM ADDR
M SADR EADR = DUMP FROM ADDR TO ADDR
```

A memória tartalmát lehet vele HEX értékekként ill. ASCII/PETSCII karakterekként vizsgálni. Ugyanaz mint a **TEDMON** „**M**” parancsa annyi különbséggel, hogy az ASCII/PETSCII részen a karakterkódok 7-es bitjét a **BMX** nem vágja le. Paraméterként semmit, kezdőcímet vagy kezdő és végcímet lehet megadni, működése értelemszerű.

>:

```
>: EDIT MEMORY
  USAGE:
> DADR OP, ... = WRITE BYTE(S) TO ADDR
```

A memória tartalmát lehet vele HEX értékekként szerkeszteni. Paraméterként a cím kötelező, utána jöhet 0..8 darab BYTE értéke, ezek bekerülnek a memóriába. Majd a „kész” sort (8 BYTE) kiírja úgy, ahogy az „**M**” parancs teszi. Ha csak cím van megadva, akkor nem történik memóriaírás (mivel nincs mit), de a nyolc adat a képernyőn megjelenik.

I:

```
I: CHARCODE DUMP
  USAGE:
I = CONTINUE CHARCODE DUMP
I SADR = DUMP FROM ADDR
I SADR EADR = DUMP FROM ADDR TO ADDR
```

A memória tartalmát lehet vele KARAKTERKÓDOK formájában megjeleníteni. (Tehát ha például a memóriában $01 található, akkor a képernyőn a sorban egy „A” betű fog megjelenni. Nem ASCII/PETSCII kód, hanem képernyőkód!) Egy sorban 32 karakter jelenik meg. Paraméterként semmit, kezdőcímet vagy kezdő és végcímet lehet megadni, működése értelemszerű.

':

```
': EDIT MEMORY IN CHARCODE MODE
  USAGE:
' DADR CHARCODES(MAX32) = EDIT MEM
```

Az „**I**” paranccsal a képernyőre kiírt memóriatartalmat lehet szerkeszteni. A cím után maximum 32 karaktert lehet beírni, amit KARAKTERKÓDOK formájában fog a **BMX** a memóriába eltárolni. Paraméterként cím kell, utána 0..32 darab karaktert lehet megadni. A memóriába beírás (ha volt mit) után kiírja a 32 karaktert.

E:

```
E: BINARY DUMP
  USAGE:
E = CONTINUE BINARY DUMP
E SADR = DUMP FROM ADDR
E SADR EADR = DUMP FROM ADDR TO ADDR
```

„Bináris” módon lehet a memóriát megjeleníteni. (Például az „**E D000**” parancs a ROM-ból megjeleníti a karakterkészletet.) A 0-ás bitek „**.**” az 1-esek meg „*****” formában jelennek meg. Egy sorban egy BYTE 8 bitje jelenik meg. Paraméterként semmit, kezdőcímet vagy kezdő és végcímet lehet megadni, működése értelemszerű.

[:

```
[: EDIT MEMORY IN BINARY MODE
  USAGE:
[ DADR 8BIT[.*] = EDIT MEM
```

Az „**E**” paranccsal a képernyőre kiírt bináris memóriatartalmat lehet szerkeszteni. Paraméterként ha csak cím van megadva, akkor annak a tartalmát megjeleníti. Módosítani a tartalmat mind a 8 bit beírásával lehet csak. A bitek lehetnek „**.**” és „*****”, de „**0**” illetve „**1**” értékeket is elfogad. Kiírni a „**.**” / „*****” formában fogja a végeredményt.

F:

```
F: FILL MEMORY
  USAGE:
F SADR EADR OP,... = WRITE PATTERN
F SADR EADR 'STRING = WRITE STRING
F SADR EADR "STRING" = WRITE STRING
```

Memória feltöltése adatokkal. Paraméterként kezdő és végcímet vár, valamint az adatokat, amivel a memóriát fel kell tölteni. A **TEDMON**-nal ellentétben nem csak egy BYTE adható meg, hanem több is (maximum 32), illetve „sztring” is megfelel.

T:

```
T: MEMORY BLOCK TRANSFER
  USAGE:
T SADR EADR DADR = TRANSFER BYTES
```

Memóriatartalom mozgatása. Három paramétert vár: a kezdő és végcímet, illetve az új címet, ahova a memóriatartalmat át kell mozgatni. A **TEDMON**-nal ellentétben mindig a megfelelő irányba másol, ezért egymást átfedő területek esetén is jól működik.

C:

```
C: MEMORY BLOCK COMPARE
  USAGE:
C SADR EADR DADR = COMPARE BYTES
```

Memóriatartalom összehasonlítása. Három paramétert vár: kezdő és végcímet, illetve egy új kezdőcímet. A kezdő és végcím közötti területet hasonlítja az új kezdőcímen levő területtel, a különbségek címeit kiírja. A **Run/Stop** lenyomásával leállítható az összehasonlítás.

H:

```
H: HUNT BYTE(S) OR STRING
   USAGE:
H SADR EADR OP, ... = HUNT BYTES
H SADR EADR 'STRING = HUNT STRING
H SADR EADR "STRING" = HUNT STRING
```

BYTE(-ok) keresése a memóriában. Három (vagy több) paramétert vár: kezdő és végcímet, amik között „vadászik” a tartalomra, illetve minimum egy, maximum 32 BYTE-nyi adatot, amit keres. Keresési tartalomnak megadható „sztring” is, de a **TEDMON**-nal ellentétben idézőjelek között megadva is működik. A **Run/Stop** lenyomásával leállítható a keresés.

$:

```
$: HEX CONVERTER OR DIRECTORY
   USAGE:
$ = SEE DIRECTORY COMMAND
$ WORD = CONVERT HEX TO OTHER
```

HEX szám konvertálása „minden mássá”. Paraméterként egy maximum 4 számjegyű HEX számot vár, amit megjelenít egyéb formákban is. Paraméter nélkül a kiválasztott lemezmeghajtó tartalmát listázza, **Ux** paramétert megadva az **x** egységszámúét.

%:

```
%: BIN CONVERTER
   USAGE:
% BINWORD = CONVERT BIN TO OTHER
```

BIN szám konvertálása „minden mássá”. Paraméterként egy maximum 16 számjegyű BIN értéket vár, amit megjelenít egyéb formákban is.

#:

```
#: DEC CONVERTER
   USAGE:
# DECWORD = CONVERT DEC TO OTHER
```

Decimális szám konvertálása „minden mássá”. Paraméterként egy **0**..**65535** közötti DEC számot vár, amit megjelenít egyéb formákba is.

X:

```
X: EXIT TO BASIC
  USAGE:
X = EXIT
```

Kilépés a monitorból a BASIC értelmezőbe. A funkciója ugyanaz, mint a **TEDMON** „**X**” parancsának.

FIGYELEM: A HASZNÁLAT CSAK SAJÁT FELELŐSSÉGRE! A MŰKÖDÉSRE SEMMIFÉLE GARANCIA NINCS! (Cserébe bármilyen hibajelentést szívesen fogadok.)

FIGYELEM: HUZAMOSABB HASZNÁLATA FÜGGŐSÉGET OKOZ!

2013.11.26. Dokumentáció első publikus változata