Komentovaný výpis MDOSu

verze 1.0 a jeho opravy

1. Jako obvykle začneme úvodem

Je to již několik let, co se mezi nás dostala disketová jednotka, pojmenovaná D40 nebo D80. Za tu dobu se velice rychle stala jednou z nejpoužívanějších periférií, připojených k našemu starému ZX Spectru nebo novějšímu Didaktiku ze Skalice, kde se také uvedená disketová jednotka vyrábí (spíše vyráběla, protože v současné době už to je v útlumu). Za tu dobu na ní vzniklo i mnoho software, ať lepší nebo horší kvality, který uměl s danou disketovou jednotkou pracovat, a tím i zvyšovat její použitelnost. Tím nechci říci, že se s ní pracuje špatně. Pracuje se s ním velice dobře, ale posuďte sami, že když kopírujete soubory pomocí DISKCOPY a pomocí MOVE, tak je v tom přece jen nějaký rozdíl. Jsou to hlavně toolsy, disk doktory, vznikl tabulkový procesor, slovník, textový editor. A hlavně dochází k úpravám her, převážně dílových. Odpadá tak věčné nahrávání každého dílu z kazety. To, co vzniklo za tyto roky, je obrovský kus práce. Jenomže pořád tady něco chybí. Ano, je to **komentovaný výpis obsahu ROM** v disketové jednotce. Každý autor programu pro D40 si musel udělat sám nějaký takový výpis, aby vůbec zjistil, jak D40 pracuje. A takových výpisů jistě existuje velké množství. Někdo přišel na to, někdo na ono. Samotný autor nedal výpis nikomu k dispozici. Na mých několik dotazů mi bylo vždy odpovězeno, že obsah podléhá autorským právům a že mi ho tedy nemohou poskytnout (prodávají výrobek a neposkytnou o něm informace, aby se mohl vytvořit software), a po rozmluvě s jedním ze spolupracovníků autora mi bylo dokonce řečeno, že ho nevlastní ani autor (on ho sice má, ale komentáře jsou jenom asi 2 řádky) – takže „pomoz si sám“. Samozřejmě, že došlo k publikaci různých adres, kde jsou uloženy některé podprogramy (v ZX Magazínu, kniha Rutiny ROM D40), ale to jsou jenom takové zlomky (je to asi stejné, jako publikace několika adres v ZX ROM) a ještě jsou některé věci nepřesné (to není výtka autorům, snažili se). Právě to mě vedlo k tomu, že jsem se pokusil o vytvoření celkového popisu disketové jednotky a okomentování celého MDOSu (tak se jmenuje operační systém, který ovládá D40). Chtěl bych touto cestou poděkovat Standovi Skapovi, který mi dal assemblerový výpis (já už jsem si ho musel okomentovat), a Jardovi Krejčímu, který mi vnuknul tento nápad – „Bylo by dobré, kdyby to někdo udělal“. No a nejvíce sám sobě, že jsem u toho vydržel a neshodil to ze stolu (aspoň bych si tam uklidil). Doufám, že Vám tato pomůcka pomůže k tvorbě dalších kvalitních programů na ZX Spectrum nebo Didaktik s touto disketovou jednotkou.

KVAKSOFT

PS: Tento komentovaný výpis jsem však musel psát v Microsoft Wordu, protože do Spectra se mi nevešel (a to mám ve Speccy 272 KB).

1.1. Disketová jednotka D40/80

Co to vůbec je disketová jednotka? Je to zařízení, které slouží k uložení dat na vnější paměti – diskety. Nahrazuje nám tak kazetový magnetofon. Rychlost přenosu dat je o mnoho rychlejší než u kazetového magnetofonu a cena médií se v přepočtu na 1 byte liší jen o směšnou částku. Roste i spolehlivost ukládaných dat (jak se to vezme, protože i diskety mají své mouchy).

1.2. Operační systém MDOS

Operační systém MDOS (M Disk Operating System) nám umožňuje používat příkazy jazyka BASIC pro komunikaci s disketovou jednotkou. Je umístěn v paměti ROM disketové jednotky. Je trochu odlišný od jiných operačních systémů (MICRODRIVE, DISCIPLE, BETADISK atd.) a nespolupracuje s programy, které mají svůj syntaktický analyzátor – např. SKYLINE BASIC (tam si ale jde dodefinovat), MEGA BASIC, BETA BASIC atd. Popis příkazů operačního systému MDOS si uvedeme přímo v komentovaném výpisu.

2. Technické údaje o disketové jednotce

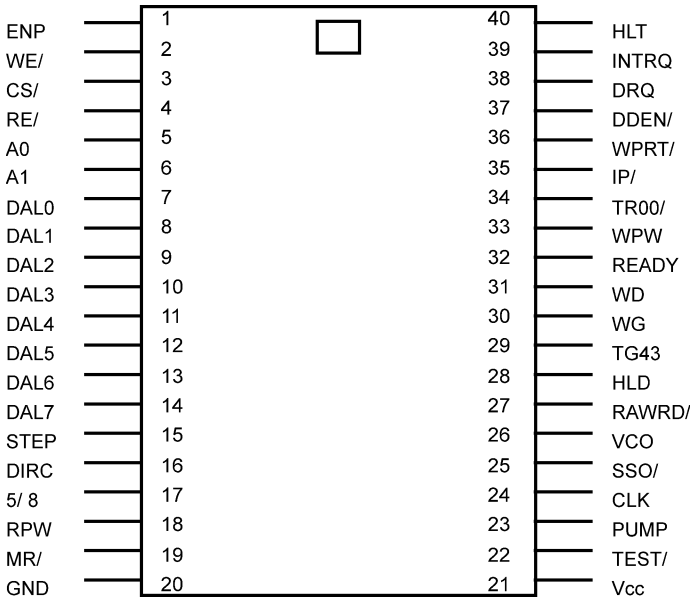
2.1. Řídící jednotka

Základ tvoří řadič WD2797, paměť EPROM 16 KB, paměť SRAM 2 KB a obvod 8255. Řadič zabezpečuje komunikaci mezi počítačem a mechanikami. Řídící jednotka umožňuje připojení dvou mechanik, přestože řadič WD2797 umí ovládat až 4 mechaniky. Paměť EPROM obsahuje operační systém MDOS a je od adresy 0 do #37FF. Paměť SRAM je od #3800 do #3FFF (překrývá EPROM) a slouží k uložení pomocných dat disketové jednotky. Obvod 8255 slouží jako interface pro připojení ostatních periférií.

2.2. Řadič WD2797

WD2797 můžeme považovat za sloučení obvodů FD1771 a FD179x, je kompatibilní s formátem IBM 3740 v režimu jednoduché hustoty a s formátem IBM 34 v režimu dvojité hustoty. Je zabudován v disketových jednotkách D40 a D80 s verzí MDOSu 1.0 a jeho opravách.

Rozhraní procesoru se skládá z 8-mi bitové oboustrané sběrnice pro přenos dat, stavu a řídícího slova.



Obrázek č. 0 – rozmístění vývodů obvodu WD2797

2.2.1. Popis vývodů

Tabulku s popisem jenotlivých vývodů naleznete na další straně

Pin	Signál	Označení	Popis
1	ENABLE PRECOMP	ENP	‚0' na tomto vstupu umožní zápis prekompenzace pro použití režimu dvojité hustoty při zápisu dat
19	MASTER RESET/	MR/	‚0' (T>50µs) na tomto vstupu provede reset zařízení a nastaví 3 do registru příkazu. Během aktivního MR/ je nulován bit „Not ready". Když MR/ signál přejde do ‚1' a je proveden příkaz RESTORE nezávisle na stavu signálu READY od drivu a do sektoru registru je nastavena hodnota ‚1'.
20	GROUND	GND	Zem napájení.
21	POWER	Vcc	Kladný pól napájení (+5V)
		Rozhraní počítače	
2	WRITE ENABLE/	WE/	‚0' na tomto vstupu oznamuje platná data na DAL vodičích pokud CS=‚0'.
3	CHIP SELECT/	CS/	‚0' na tomto vstupu způsobí připojení obvodu a umožní komunikaci s procesorem
4	READ ENABLE/	RE/	‚0' na tomto vstupu řídí sejmutí dat z vybraného registru dat na vodiče DAL při CS=‚0'.
5, 6	REGISTR SELECT LINES	A0, A1	Výběr registru pro I/O
7–14	DATA ACCES LINES	DAL0–7	8-bitová obousměrná sběrnice pro přenos dat, příkazu a stavu.
24	CLOCK	CLK	Tento vstup vyžaduje hodinový signál pro vnitřní časování (2 MHz pro 8" a 1 MHz pro 5 1/4").
38	DATA REQUEST	DRQ	Tento výstup informuje, že v data registru jsou připravena data (čtení) nebo že data registr je volný (zápis). Signál je nulován, pokud procesor data převezme nebo dodá.
39	INTER. REQUEST	INTRQ	Tento výstup je nastaven při dokončení výkonu příkazu a je nulován po přečtení status registru nebo zápisem příkazu.
		Rozhraní disku	
15	STEP	STEP	Výstup kroku. Vytváří puls pro každý krok mezi stopami.
16	DIRECTION	DIRC	Výstup směru kroku. Pro krok ke kraji je ‚0'.
17	5 1/4", 8" SELECT	5/8	Vstup pro výběr kmitočtu VCO pro použití 5 1/4" nebo 8" drivu.
18	READ PULSE WIDTH	RPW	Vstup pro vnější potenciometr pro nastavení fázového komparátoru pro datový separátor.
22	TEST/	TEST/	‚0' na tomto vstupu umožňuje nastavení trimru VCO, RPW, WPW.
23	PUMP	PUMP	Vysokoimpedanční výstupní signál pro regulaci VCO.
25	SIDE SELECT OUTPUT	SSO	Logická úroveň výstupu výběru strany je přímo řízena od příznaku ‚S' v příkazech I a II, pokud U=log. 1 SSO je nastaven také na log. 1.
27	RAW READ/	RAWRD/	Vstup signálu dat přímo z drivu.
28	HEAD LOAD	HLD	Výstup řídící nastavení hlavy k médiu při R/W operacích.
29	TRACK GATHER	TG43	Výstup informuje drive, že hlava je umístěna mezi stopou 44 a vnitřními stopami. Je nastaven pouze při provádění R/W operací.
30	WRITE GATE	WG	Výstup oznamuje drivu, že data na WD jsou platná.
31	WRITE DATA	WD	Výstup dat do drivu.
32	READY	READY	Tento vstup indikuje připravenost drivu. Před provedením operace R/W musí být ‚1', jinak nejsou operace provedeny a je generováno přerušení. Tento vstup je v invertovaném tvaru obsazen ve status registru.
33	WRITE PRECOMP	WPW	Vstup pro připojení potenciometru pro řízení prekompenzace zápisu.
34	TRACK 00/	TR00/	Vstup, kterým drive oznamuje, že hlava se nachází nad stopou 0.
35	INDEX PULS/	IP/	Signál z diskety. Nulový puls znamená indexovou díru.
36	WRITE PROTECT/	WP/	Vstup je vyhodnocován při operaci zápisu. Pokud je zde ‚0', operace nebude provedena a je nastaven „Write prt" bit ve status registru.
37	DOUBLE DENSITY/	DDEN/	Tímto vstupem se vybírá dvojitá ‚0' nebo jednoduchá ‚1' hustota.
40	HEAD LOAD TIMING	HLT	Pokud je zde ‚1', považuje řadič hlavu za přiloženou.

2.2.2. Organizace řadiče

Posuvný registr dat – 8-bitový registr, je použit jako registr, který během R/W operací převádí data sériová na paralelní a naopak.

Registr dat – 8-bitový registr spolupracující s posuvným registrem. Tento registr je naplněn z DAL pod řízením procesoru.

Registr stopy – 8-bitový registr, obsahuje aktuální číslo stopy. To je zvyšováno pokaždé, když je hlava posunuta ke středu diskety a snižováno, když hlava krokuje směrem k okraji. Obsah registru je porovnáván se zaznamenaným číslem stopy v ID poli během R/W a verify operací. Registr stopy může být naplněn z DAL. Registr nemůže být naplněn pokud zařízení není připraveno.

Registr sektoru – 8-bitový registr, obsahuje adresu pozice ve stopě (číslo sektoru). Obsah registru je porovnáván se zaznamenaným číslem sektoru v ID poli během R/W operací. Registr sektoru nemůže být naplněn, pokud zařízení není připraveno.

Registr příkazu – 8-bitový registr, obsahuje příkaz, který má být zanedlouho proveden. Tento registr nelze nastavit, pokud není zařízení připraveno. Výjimkou je příkaz násilného přerušení. Zápis je možný přes DAL, ale není možné příkaz číst.

Stavový registr – 8-bitový registr, obsahuje informace o stavu zařízení. Stav registru je funkcí typu právě provedeného příkazu. Registr se čte pomocí DAL, ale nelze ho zapsat.

CRC logika – tato logika je použita pro zjištění chyby a generování 16-ti bitového kontrolního součtu (CRC). CRC zahrnuje všechny informace počínaje ID polem až do CRC znaku.

Dále je zde *aritmeticko logická jednotka*, *časování a řízení*, *datový separátor*.

2.2.3. Rozhraní procesoru

Je zjednodušeno na 8-bitovou sběrnici DAL a odpovídajících řídících signálů. DAL vodiče jsou použity pro přenos dat, stavu a řídícího slova mezi procesoren a řadičem. DAL je třístavová a je vyhodnocena jako výstupní při CS=‚0‘ a RE=‚0‘ a jako vstupní při CS=‚0‘ a WE=‚0‘. Při přenášení dat mezi CPU a řadičem vybíráme registry pomocí vodičů A0, A1 v kombinaci RE a WE.

A1	A0	čtení (RE=‚0‘)	zápis (WE=‚0‘)
0	0	stav	příkaz
0	1	stopa	stopa
1	0	sektor	sektor
1	1	data	data

Jsou dosažitelné délky sektoru 128, 256, 512, 1024 bytů. Délka sektoru je nastavena v ID poli v době formátování.

Označení délky sektoru	Počet bytů dat v sektoru
00	128
01	256
02	512
03	1024

2.2.4. Popis příkazů řadiče

WD2797 akceptuje 11 příkazů. Příkaz může být nastaven do registru příkazů, když je bit „BUSY“ ve stavovém slově ‚0‘ s výjimkou příkazu násilného přerušení. Vždy, když je příkaz prováděn, je „BUSY“ bit stavového slova nastaven na ‚1‘. Po dokončení příkazu je generováno přerušení a bit je nulován. Stavový registr vždy indikuje kompletní dokončení operace nebo chybu. Pro snadnější rozlišení jsou příkazy rozděleny do čtyř typů (viz tabulka na další straně).

Typ	Příkaz	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r1	r0
I	Seek	0	0	0	1	h	V	r1	r0
I	Step	0	0	1	T	h	V	r1	r0
I	Step in	0	1	0	T	h	V	r1	r0
I	Step out	0	1	1	T	h	V	r1	r0
II	Read sector	1	0	0	m	L	E	U	0
II	Write sector	1	0	1	m	L	E	U	a0
III	Read address	1	1	0	0	0	E	U	0
III	Read track	1	1	1	0	0	E	U	0
III	Write track	1	1	1	1	0	E	U	0
IV	Force interrupt	1	1	0	1	i3	i2	i1	i0

Vysvětlivky

r0, r1	rychlost krokování		
V	ověřování čísla stopy	V=0	bez ověření
		V=1	ověření čtením čísla stopy
h	přiložení hlavy	h=0	hlava nepřiložena
		h=1	hlava přiložena
T	definice stopy	T=0	není
		T=1	definována registrem stopy
a0	značka adresy dat	a0=0	#FB (DAM)
		a0=1	#F8 (vypuštěná DAM)
U	definice strany	U=0	strana 0
		U=1	strana 1
E	zařazení prodlevy	E=0	není prodleva
		E=1	je vložena prodleva 15 ms
L	délka sektoru	ID	00 01 01 11
		L=0	256 512 1024 128
		L=1	128 256 512 1024
m	vícenásobný záznam	m=0	jednoduchý záznam
		m=1	vícenásobný záznam
i0, i1, i2, i3	podmínky pro přerušení	i0=1	přechod do připravenosti
		i1=1	přechod do nepřipravenosti
		i2=1	index puls
		i3=1	přímé přerušení
		i0–i3=0	zastaví bez přerušení

2.2.4.1. Příkazy typu I

RESTORE **– Příkaz, který nastaví hlavu na stopu 0. Je vykonáván po tu dobu, kdy je TR00=,0'. Jestliže je TR00=,1', je hlava nad stopou 0, do registru stopy je zapsána 0 a je generováno přerušení. Pokud TR00=,0' i po 255 krokovacích pulsech, WD2797 zastaví činnost a konec je hlášen bitem „Seek error" ve stavovém slově. Pokud V=,1', je provedeno ověření stopy čtením. Bit „h" řídícího slova umožňuje přitáhnout hlavu hned na začátku příkazu.**

SEEK **– Příkaz porovná obsah registru stopy a registru dat obsahující číslo požadované stopy. WD2797 definuje registr stopy a vydává krokovací pulsy v potřebném směru, dokud obsah registru stopy není roven obsahu datového registru. V případě V=,1' je provedeno ověření stopy. Pokud h=,1', je hned na začátku příkazu provedeno přiložení hlavy. Po úplném dokončení příkazu je generováno přerušení. Při řízení více drivů jedním řadičem musí být číslo stopy vždy definováno před výkonem příkazu SEEK, protože řadič používá jeden registr dat pro všechny mechaniky (nemá pro každou mechaniku jeden registr – je důležité si to uvědomit).**

STEP – Provádí posun hlavy o jeden krok. Při provádění tohoto příkazu vydá WD2797 krokovací puls do drivu. Směr krokování je stejný jako v předchozím příkazu krokování. Pokud V=,1', potom po době definované polem r1, r0 je provedeno ověření. Pokud T=,1', je registr stopy definován. Bit „h" umožňuje přiložení hlavy na začátku příkazu. Po kompletním ukončení příkazu je generováno přerušení.

STEP IN – Provádí posun hlavy o jeden krok směrem od stopy 0. Při provádění tohoto příkazu pošle WD2797 krokovací puls a předtím nastaví směr krokování od stopy 0. Pokud T=,1', je registr stopy zvýšen o 1. Po době, definované r0, r1, je při V=,1' provedeno ověření stopy. Bit „h" umožňuje přiložení hlavy na začátku příkazu. Po kompletním ukončení příkazu je generováno přerušení.

STEP OUT – Provádí posun hlavy o jeden krok směrem k stopě 0. Při provádění tohoto příkazu pošle WD2797 krokovací puls a předtím nastaví směr krokování k stopě 0. Pokud T=,1', je registr stopy snížen o 1. Po době, definované r0, r1, je při V=,1' provedeno ověření stopy. Bit „h" umožňuje přiložení hlavy na začátku příkazu. Po kompletním ukončení příkazu je generováno přerušení.

2.2.4.2. Příkazy typu II

Sem patří příkazy READ SECTOR a WRITE SECTOR. První, co řadič udělá po převzetí těchto příkazů je, že naplní sektor registr číslem požadovaného sektoru. Dokud je příkaz typu II prováděn, je nastaven „BUSY" bit ve status registru. S ID polem na disku se porovnává registr stopy. Pokud srovnání nevyšlo, je provedeno načtení dalšího ID pole a znovu porovnáno. Pokud je nyní vpořádku, je porovnán regist sektoru s ID, a pokud nevyšlo, je čteno další ID a provedeno nové porovnání. Pokud ID pole i CRC je odpovídající, následuje datové pole pro zápis nebo čtení podle druhu příkazu. Řadič musí najít ID pole během 5 otáček diskety. V opačném případě je nastaven bit „RNF", příkaz je zastaven a je generováno přerušení.

READ SECTOR – Při provádění příkazu je přiložena hlava, nastaven „BUSY" bit stavového slova a po nalezení odpovídajícího ID pole jsou data z disku předávána procesoru. Značka adresy dat datového pole musí být nalezena uvnitř 30 bytů pro SD nebo 43 bytů pro DD v posledním ID poli. Pokud se tak nestane, je vyhledávání opakováno. Když je první byte z datového pole nahrán do posuvného registru, je přenesen do datového registru a je aktivováno DRQ. Když je další byte načten do posuvného registru, je přenesen do datového registru a je znovu aktivováno DRQ. Pokud procesor nepřečetl do té doby obsah datového registru, jsou data ztracena a je nastaven bit „Lost data". Tento postup je opakován až do kompletního přečtení datového pole. Jestliže je na konci nalezen chybný CRC součet, je nastaven bit CRC error ve stavovém registru a příkaz je zastaven. Na konci čtení je ve stavovém registru v 5. bitu zaznamenán typ značky dat: ‚1' = vypuštěna datová značka, ‚0' = datová značka.

WRITE SECTOR – Při vykonávání příkazu je přiložena hlava a nastaven „BUSY" bit ve stavovém slově. Když je nalezeno odpovídající ID pole, je generován signál DRQ. Řadič odpočítá 11 (SD) nebo 22 (DD) bytů a aktivuje vstup WG, pokud bylo DRQ obslouženo (procesorem byla dodána data). Pokud se tak nestalo, je nastaven „Lost data" bit (data ztracena) a je generováno přerušení. Jestli bylo DRQ obslouženo, je WG aktivován a 6 nulových bytů (SD) nebo 12 bytů (DD) je zapsáno na disk. V té době je zapsána značka adresy dat tak, jak je určeno a0 bitem příkazu. Pro a0=,1' je vypuštěna značka dat, a0=,0' je značka dat. WD2797 potom zapíše datové pole a geneguje DRQ pro procesor. Pokud nejsou v potřebné době dodána nová data, je nastaven bit „Lost data" a na disk jsou zapsány nuly. Příkaz není zastaven. Po zapsání posledních dat na disk jsou řadičem zapsány dva byty CRC následované bytem #FE. WG výstup je pasivován.

2.2.4.3. Příkazy typu III

READ ADDRESS – Při vykonávání příkazu je přiložena hlava a nastaven „BUSY" bit. Z disku je načteno následující ID pole a 6 datových bytů ID pole je přeneseno do datového registru a je generováno DRQ pro každý tento byte. Pořadí je na následující: číslo stopy, číslo sektoru, číslo strany, délka sektoru, CRC LSB, CRC MSB.

Všechny CRC znaky jsou přeneseny do počítače, WD2797 je kontroluje a při nesouhlasu je nastaven „CRC error" bit. Číslo stopy je přeneseno do registru. Na konci činnosti je generováno přerušení a je nulován „BUSY" bit.

READ TRACK – Při výkonu tohoto příkazu je přiložena hlava a nastaven „BUSY“ bit. Čtení začíná úvodním synchronizačním pulsem za prvním index pulsem a pokračuje až do následujícího index pulsu. Všechny mezery a hlavičky jsou přeneseny do datového registru a DRQ je generováno pro každý byte. Po kompletním provedení příkazu je generováno přerušení. Tento příkaz nelze použít pro kopírování celých stop vzhledem k synchronizačním problémům.

WRITE TRACK – Příkaz k formátování stopy. Při provádění příkazu je přiložena hlava a nastaven „BUSY“ bit. Zápis začíná zavedením synchronizace po prvním index pulsu a pokračuje až do následujícího index pulsu, kdy je generováno přerušení. DRQ je aktivován ihned po zadání příkazu, ale zápis nezačné dříve, než je dodán první byte do datového registru. Pokud datový registr není naplněn (v době tří bytů), je činnost zastavena, řadič se ohlásí „Not busy“, je nastaven „Lost data“ bit a generováno přerušení. Jestliže není v potřebné době registr naplněn, jsou dosazeny nuly.

2.2.4.4. Příkaz typu IV

FORCE INTERRUPT – Příkaz násilného přerušení je zejména použit pro zastavení vícenásobného čtení/zápisu sektoru nebo pro zabezpečení typu I stavu ve stavovém registru. Tento příkaz může být nahran do registru příkazů kdykoliv. Jestli je nějaký příkaz vykonáván („BUSY“ bit je ‚1‘), bude tento příkaz zastaven a „BUSY“ bude nulován. Nižší 4 bity příkazu určují podmínky přerušení.

2.2.5. Stavový registr

Při vykonávání jakéhokoliv příkazu, mimo násilného přerušení, je „BUSY“ bit nastaven na ‚1‘ a stavové bity jsou definovány pro nový příkaz. Jestliže je přijato násilné přerušení, je „BUSY“ nulován a stavové bity nejsou změněny. Pokud je příkaz přerušení přijat, když není žádný příkaz vykonáván, je „BUSY“ nulován a stavové bity jsou definovány. V tom případě stav odpovídá vykonání příkazu typu I. Uživatel má možnost číst stavový registr pomocí programového řízení nebo použití DRQ s DMA nebo přerušovací metodou. Když je čten datový registr, je DRQ ve stavovém slově i DRQ výstup automaticky nulován. Zápis do datového registru působí stejně. „Busy“ bit stavového slova může být sledován uživatelským programem pro určení, zda je příkaz kompletní, v případě, že nepoužijeme INTRQ. Při použití INTRQ není doporučeno zjišťovat stav „BUSY“ bitu, protože testování „BUSY“ by znamenalo nulování INTRQ výstupu. Pro správné programové řízení je třeba znát časové prodlevy, proto si teď uvedeme tabulku pro 2 MHz.

činnost	následující činnost	prodleva FM	prodleva MFM
zápis příkazu	čtení BUSY bitu 0	12 µs	6 µs
zápis příkazu	čtení stavu 1–7	28 µs	14 µs
zápis do registru	čtení z jiného registru	0 µs	0 µs

Nyní si uvedeme tabulku stavového registru

Bit	Typ I	Read address	Read sector	Read track	Write sector	Write track
7	Not ready	Not ready	Not ready	Not ready	Not ready	Not ready
6	Write protect	0	0	0	Write protect	Write protect
5	Head load	0	Rec type	0	0	0
4	Seek error	RNF	RNF	0	RNF	0
3	CRC error	CRC error	CRC error	0	CRC error	0
2	Track 0	Lost data	Lost data	Lost data	Lost data	Lost data
1	INDEX	DRQ	DRQ	DRQ	DRQ	DRQ
0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

Vysvětlivky

Not ready	Tento bit indikuje, že drive není připraven. Pokud je ‚0', je drive připraven. Jeho stav je určen negací READY vstupu a logickým součtem s MR.
Write protect	Pokud je bit nastaven, je disketa chráněná proti zápisu. Stav je určen negací vstupu WRPT.
Head load	Pokud je nastaven, je hlava přiložena k médiu. Bit je dán logickým součinem stavů HLT a HLD.
Seek error	Pokud je nastaven, nebylo úspěšné požadované ověření nastavení stopy.
CRC error	CRC načtené v ID poli není v pořádku.
TRACK 0	Bit indikuje, že se hlava nachází nad stopu 0.
INDEX	Bit informuje, že byla z drivu načtena indexová značka. Bit je určen negací vstupu ID.
BUSY	Nastaven, pokud řadič právě vykonává přikaz.
Rec type	Při čtení záznamu vrací značku dat: ‚0' = datová značka, ‚1' = bez datové značky.
RNF	Nastavený bit hlásí, že sektor nebyl nalezen.
Lost data	Při nastavení informuje, že nebyla dodržena spolupráce řadiče a procesoru (DRQ a DAL) a byla ztracena data.
DRQ	Bit kopíruje stav DRQ výstupu. Nastavení znamená, že datový registr je volný pro zápis nebo připraven pro čtení.

2.3. WD2797 v D40

Řadič pracuje v režimu dvojité hustoty (MFM) a dvoustranné diskety. Signál „READY" není zapojen, MDOS si připravenost diskety zjišťuje programově. Pokud se podíváte do schématu zapojení disketové jednotky a dáte si tu práci, že si zjistíte ovládací porty, dojdete k následujícím výsledkům:

Výběr mechanik je na portu #89 (137).

DS0 – 0. bit
DS1 – 1. bit
MO0 – 2. bit
MO1 – 3. bit
NMI – 6. bit

(DS=drive select; MO=motor on)

Ke čtení status registru a zadání příkazu řadiči je port #81 (129).

Zaslání čísla stopy je na portu #83 (131).

Zaslání čísla sektoru je na portu #85 (133).

Jistě Vás zarazilo, že na 6. bitu výběru mechanik je NMI. Nespletl jsem se. Tento bit je velice důležitý při vykonávání čtení a zápisu na disketu v MDOSu. Tyto operace totiž fungují v MDOSu přes NMI! Pro každý byte, který je načten nebo zapsán, je vyvoláno NMI. Funguje to následovně: při požadavku čtení nebo zápisu se při vybírání a roztáčení mechaniky nastaví 6. bit. Nyní se pošle řadiči příkaz pro čtení nebo zápis na disketu. Jak bylo výše uvedeno v popisu těchto operací, jestliže signál DRQ je ve stavu 1, je řadič připraven přijmout byte pro zapsání na disketu nebo má v datovém registru načten další byte z diskety. Jakmile je DRQ roven 1 a byl nastaven 6. bit při výběru mechaniky, je vyvoláno NMI, které skáče na adresu #0066. Zde je v MDOSu skok na adresu v registru IX, kde se buď provede zápis bytu do datového registru řadiče nebo načtení byte z datového registru řadiče. Po provedení operace se řízení vrátí zpět do místa, kde došlo k vyvolání přerušení. Toto se opakuje 512 krát. Po provedení operace se 6. bit zase vynuluje – je to důležité, protože např. při operaci SEEK, RESTORE a dalších typu I je taky nastaven DRQ a NMI by nám stejně bylo k ničemu (spíše by nadělalo více problémů než užitku).

Zdá se to být sice trochu komplikované, ale zato geniální řešení a asi ten nejrychlejší a nejspolehlivější způsob provádění R/W operací (tento způsob trvá nejméně taktů).

2.4. Paměť EPROM a SRAM

Paměť EPROM obsahuje operační systém MDOS, který nám rozšiřuje standartní BASIC o novou třídu příkazů, které pracují s disketovou jednotkou. Začíná na adrese #0000 a končí na #37FF. Jak vůbec dochází k tomu, že lze používat tuto paměť?

Řídící jednotka obsahuje obvod, který je ovládaný dekodérem instrukcí a který pracuje takto: při čtení instrukce z adres #0000 nebo #0008 se přepne na výběr vnější paměti – EPROM a při čtení instrukce z adresy #1700 se přepne na výběr vnitřní paměti – ZX ROM. Adresy #0000 a #0008 jsou vstupní brány do EPROM ze ZX ROM a adresa #1700 je vstupní brána do ZX ROM z EPROM. Nemusí se tedy provádět žádný OUT, ale přepínání je zajištěno hardwarem. Jak se zpracovávají příkazy MDOSu? Pokud jste alespoň trochu seznámeni se ZX ROM a syntaktickou analýzou BASIC příkazů v ní, tak to pro Vás nebude nic složitého. Při analýze příkazu MDOSu ZX ROM dojde k tomu, že takovou syntaxi příkazu nezná a skáče na adresu #0008 pro výpis chyby. Jenomže tato adresa slouží jako vstupní brána do EPROM, kde je umístěn MDOS. Takže při skoku na adresu #0008 dojde k přestránkování a místo ZX ROM je MDOS v ROM D40. A ten začne zpracovávat příkaz od místa, kde analyzátor BASICu v ZX ROM přestal. Pokud daný příkaz zná, zpracuje ho a vykoná nebo vypíše chybu a vrátí řízení ZX ROM.

Spolu s EPROM je v řídící jednotce paměť SRAM, která je od adresy #3800 do adresy #3FFF. Přepíná se společně s pamětí EPROM. MDOS si do ní ukládá potřebné informace, aby nezabíraly zbytečně místo v paměti počítače.

2.5. Tlačítko SNAP

Celkem zajímavé zařízení, které slouží k záznamu celého obsahu paměti včetně registrů a přerušení na disk do souboru. Abych to řekl správně, tak tlačítko vyvolá nemaskované přerušení NMI, kdy je vygenerován skok na adresu #0066. Tady je v ZX ROM uložena rutina, která obsluhuje nemaskované přerušení NMI. Jak jistě víte, tak tato rutina v ZX Spectru obsahuje chybu, ale v Didaktiku Gama už je opravena. Ale i tak by nám to bylo na nic – maximálně pro skok do BASICu. Jenomže nám se uloží obsah paměti na disketu. Jak? K tomu právě slouží druhá část obvodu, který nám stránkuje EPROM. Při stisknutí tlačítka se sice provede skok na adresu #0066, ale daný obvod vnutí na tuto adresu instrukci RST 0 (skok na adresu #0000) místo skutečné instrukce, která je na této adrese uložena (a tam je PUSH AF). Takže po skoku na adresu #0066 se místo instrukce PUSH AF provede RST 0 a tak se skočí na adresu #0000, která slouží jako vstupní brána do EPROM. Je to podobné jako u RST 8. A tady už si MDOS zjistí, že došlo k stisknutí tlačítka SNAP (podle návratové adresy na vrcholu zásobníku, kde je #0067) a provede uložení obsahu paměti na disketu. Potom se řízení vrátí zpět na to místo, kde došlo k přerušení tlačítkem SNAP a program pokračuje dále. Takto to ale funguje jen tehdy, pokud je od adresy #0000 ZX ROM. Pokud je od #0000 ROM D40, funguje NMI normálně. Někdy se však může stát, že se program po návratu ze snapu zablokuje. Důvod je ten, že v okamžiku, kdy došlo k stisknutí tlačítka SNAP, program právě prováděl nějakou specifickou operaci se zásobníkem (například nějakou modifikaci programu pomocí zásobníku). Protože při vyvolání NMI se na zásobník ukládá návratová adresa a další registry, mohlo dojít k přepsání dat, která jsou pro program důležitá, a tak se po návratu z NMI mohl zhroutit. Takže i data, která jsme si pomocí NMI uložili na disketu jsou taky neplatná (když si totiž nahrajete snap zpět do paměti, dostanete se do úplně stejné situace). Proto se doporučuje používat tlačítko SNAP v té době, kdy program čeká např. na stisk klávesy. Tlačítko SNAP lze v jednom programu používat vícekrát. Paměť se ukládá do souborů se jménem SNAPSHOTXX.S, kde XX je pořadové číslo snapu od resetu počítače. Pokud již takový snap soubor na disketě existuje, je přepsán bez dotazu na přepsání.

2.6. Obvod 8255

Jak už jsem uvedl, tak slouží k připojení ostatních periférií. Tento obvod tvoří základ mnoha interfaců (UR 4, M/P atd). Aby nedošlo ke kolizi s jiným obvodem 8255 (např. u Didaktik GAMA, kde je obvod 8255 zabudován uvnitř), je možné tento obvod v D40 zablokovat. Příkaz na odblokování je **OUT 153, 16** a na zablokování je **OUT 153, 0**. Konektor INTERFACE je shodný s konektorem interface M/P. Porty obvodu jsou umístěny na těchto adresách:

- adresa 31 (#1F) – port A
- adresa 63 (#3F) – port B
- adresa 95 (#5F) – port C
- adresa 127 (#7F) – řídící slovo

Mimo samotného zablokování a odblokování obvodu 8255 lze ještě povolit a zakázat odblokování obvodu. Příkaz **OUT 145, 0** zakáže odblokování obvodu 8255 a **OUT 145, 32** povolí odblokování obvodu. Bližší popis tohoto obvodu můžete najít v Amatérském rádiu nebo ZX Magazínu.

2.7. Mechaniky

Vcelku se dá říci, že nemůžete připojit libovolnou mechaniku. Sami výrobci totiž museli pečlivě vybírat z velkého počtu mechanik. Problém se Vám neprojeví, pokud budete připojovat pouze jednu mechaniku. Jestli však budete připojovat dvě mechaniky (a z nich bude alespoň jedna 5 1/4" DD mechanika), můžete (ale nemusíte) se dostat do problémů. Když zapojíte dvě mechaniky, můžou pracovat okamžitě a bez problémů nebo nebudou pracovat vůbec – při resetu se rozběhnou, ale nepřihlásí se (přijdete o to nádherné hrkání s hlavou). Jedná se právě o 5 1/4" DD mechaniky (360 KB). Jak by se dal tento problém vyřešit? Jako první Vám doporučím vyhledat odborný servis, to pokud jste slabší povahy. Jestli jste však trochu odvážnější, můžete se pokusit o vlastní opravu. Když se podíváte na mechaniku (ještě jednou připomínám, že se jedná pouze o 5 1/4", mechaniky, u 3,5" mechanik to není), najdete na místě, kde se připojuje kabel malý obvod s 8-mi vývody. Je to „hybridní“ rezistor a jestli už ho neodstranil výrobce, musíte ho odstranit vy (pokud se vám ho nechce odpájet, ulomte ho neustálým vikláním. Jestli máte 5 1/4" DD mechaniku přímo od výrobce, tak se podívejte, jestli ho tam máte (já ho u originální mechaniky neměl). Pokud ani po této drastické operaci nebudou mechaniky spolupracovat, musíte vyhledat servisní středisko – já jsem to zkoušel a fungovalo to (myslím to odstranění obvodu). Jak jsem již předtím uvedl, dají se připojit pouze dvě mechaniky, což je dáno obvodem pro výběr mechanik (celkem lze k WD2797 připojit 4 mechaniky, k D40 by se daly připojit 3 mechaniky, protože místo pro 3. mechaniku na portu 137 je – 4. a 5. bit). Záznam na mechaniky se provádí v režimu MFM, mechaniky pracují s formátem DD (double density). Můžete tedy připojit DD nebo HD mechaniku 5 1/4", nebo 3,5" mechaniku. Pokud připojíte HD mechaniku, bude pracovat v režmu DD (takže žádných 1,44 MB, ale jen 720 KB). Standardně se dodává pouze jedna mechanika. Druhou si lze připojit přes konektor EXTENDED. Při připojování druhé mechaniky musí být její piny zapojeny úplně stejně jako u první mechaniky (např. pokud je pin č. 5 první mechaniky připojen na nějaký vývod řídící jednotky, je zde připojen i pin č. 5 druhé mechaniky). Jediná výjimka je u pinů DS0, DS1, MO0, MO1. Piny MO0 a MO1 u druhé mechaniky musíte vždy zaměnit (tam, kde je u první mechaniky zapojen pin MO0, je u druhé mechaniky zapojen pin MO1). Tyto signály slouží k roztočení motoru mechaniky. Pokud byste je nezaměnili, rozeběhly by se Vám obě mechaniky najednou nebo by se nerozeběhla ani jedna. U pinů DS0 a DS1 máme dvě možnosti. Buď je připojíme stejně jako u první mechaniky a potom, když budeme nastavovat pořadí mechanik, tak první nastavíme jako A a druhou jako B, nebo piny DS0 a DS1 u druhé mechaniky zaměníme a obě mechaniky nastavíme jako A. Signál DSx nám totiž vybírá mechaniku, se kterou se bude pracovat. Všechny ostatní piny u druhé mechaniky mají stejný význam jako u první, takže musí být zapojeny stejně.

Toto byl tedy základní popis technické stránky disketové jednotky D40. Samozřejmě nebyl úplně vyčerpávající, ale to není úkolem této knihy. Nyní se budeme spíše věnovat popisu operačního systému MDOS a prostředků, které používá.

3. Disketa

Disketa je magnetický kotouč, který je buď jednostranný nebo dvoustranný, je rozdělen na soustředné kružnice, kterým se říká stopy. Každá stopa je rozdělena na bloky – sektory. Formát MDOSu je 40 (80) stop na jednu stranu, 9 sektorů na stopu, 512 bytů na sektor. Není to však dáno, protože si můžete naformátovat disketu, jak chcete (42 stop, 8 sektorů nebo jinak). Pouze počet bytů na sektor je dán pevně – ani to však být nemusí (např. 256 bytů), ale MDOS takovou disketu neumí přečíst (WD2797 však ano, a když si napíšete vlastní rutinu na čtení, můžete přečíst libovolný formát diskety). Stopy jsou číslovány od 0, sektory od 1. Nyní si ukážeme, jak to na disketě může vypadat:

obrázek naleznete na další straně

obr.1

1
2
3
4
5
6
7
8
9

obr. 2

0
1
2
3
4
5
6
7
8

obr. 3

1
2
3
4
5
6
7
8
9
10

obr. 4

1
5
9
4
8
3
7
2
6

obr. 5

1
3
5
7
9
2
4
6
8

obr. 6

0	2	4	6
1	3	5	7

obr. 7

0	1	2	3

Na obrázku 1. je ukázáno fyzické číslování sektorů na stopě při formátu 9 sektorů na stopu – formát MDOSu. Na obrázku č. 2 je ukázáno, jak čísluje sektory MDOS. Nečísluje je od 1, ale od 0. Na obrázku č. 3 je ukázáno fyzické číslování sektorů při formátu 10 sektorů na stopu. Dále budeme rozlišovat dva druhy číslování sektorů: fyzické a logické. Fyzické je dáno číslem stopy (0–79/159) a sektoru ve stopě (1–9), logické je dáno pouze číslem sektoru (0–719/1439). Na obrázku č. 4 a č. 5 je ukázán formát 9 sektorů na stopu. Jistě jste si však všimli, že pořadí sektorů je promícháno. Tomu se říká INTERLEAVE (sektor prokládání). Je to počet otáček, na kolik se přečte najednou celá stopa. Klasický MDOS používá 1 : 1 (na jednu otáčku se přečte jedna stopa). Na obrázku č. 4 je 1 : 4 a na obrázku č. 5 je 1 : 2. Řeknete si, proč používat 1 : 4, když můžu používat 1 : 1 a stopa se mi načte rychleji? Ono to není zase tak jednoduché. Představte si, že máme sektor prokládání 1 : 1 a že budete číst sektor po sektoru z jedné stopy. Když načtete jeden sektor, tak se hlava nachází nad začátkem dalšího sektoru. Nyní vám vznikne krátka pauza (např. nějaký výpočet) a potom budete chtít číst další sektor, který se nachází hned za předchozím, který jsme načetli předtím. Jenomže než se provedl daný výpočet, ujela nám hlava, takže se nyní nenachází nad začátkem tohoto sektoru, ale třeba někde uprostřed sektoru. Takže nyní musíme čekat celou jednu otáčku, než se hlava dostane zpět na začátek požadovaného sektoru, a až nyní ho můžeme načíst. Kdyby jste třeba používali sektor prokládání 1 : 2, tak nemusíte čekat celou jednu otáčku, ale pouze čekáte, až hlava dojede na konec sektoru a vesele teď čtete požadovaný sektor. Teď si ale zase řeknete, proč používá MDOS sektor prokládání 1 : 1, když to vlastně může zdržovat práci? Autoři na to mysleli a zkrátili pauzu mezi čtením sektorů na minimum, takže se to všechno stihne. Pokud by jste třeba chtěli používat systém CP/M, tak by bylo vhodné použít sektor prokládání 1 : 2, protože tam výpočet mezi čtením jednotlivých sektorů trvá celkem dost dlouho (DATAPUTER má 1 : 2). Tento sektor prokládání nemá vliv na programovou podporu (že by se to muselo nějak hlídat). Je to jenom fyzická záležitost uspořádání sektorů na disketě. Vytváří se při formátování diskety. Na obrázku č. 6 je ukázáno číslování stop při dvoustranném formátu. Vidíte, že na jedné straně jsou sudé stopy a na strané druhé jsou liché stopy. Na obrázku č. 7 je číslování stop při jednostranném formátu.

Jak už jsem uvedl výše, je formát MDOSu 40 (80) stop na jedné straně, 9 sektorů na stopu, 512 bytů na sektor. Můžete si však formát změnit. Pokud budete zvětšovat počet sektorů, tak se jich na jednu stopu vejde maxmimálně 10 (více to už skutečně nejde) a nejméně 1 – to však nezkoušejte, protože s takovou disketou Vám MDOS nebude určitě pracovat, protože předpokládá, že na jedné stopě je alespoň 6 sektorů (z důvodů uložení FAT tabulky). Pokud budete měnit počet stop, tak nejvíce jich je asi 43 (83) na jedné straně. Pokud budete chtít více, tak už nelze ručit za kvalitu uložených dat nehledě na to, že si můžete zničit mechaniku (jako jistý MB&DG, který utrhl hlavičku). Pokud chcete zmenšit počet stop, tak minimálně to je jedna stopa (to však bude taky dělat problémy MDOSu). Myslím, že nejlepší je dodržovat klasický formát a tím se vyhnete všem problémům.

4. Struktura diskety MDOSu

Disketa se dělí na čtyři oblasti:

- BOOT
- FAT tabulka
- adresář
- datová oblast

4.1. BOOT

1. sektor na 0. stopě (podle MDOSu 0. sektor na 0. stopě), kde jsou uloženy informace o disketě – jméno, formát, parametry diskety a systém, pod kterým byla naformátována. Proč je na tomto místě? Protože na každé disketě je tento sektor na stejném fyzickém a logickém místě. Jeho struktura je následující:

+128	48 bytů	informace o všech mechanikách, které byly připojeny v době formátování
+176	12 bytů	informace o disketě a mechanice
+192	10 bytů	jméno diskety
+202	2 byty	dva náhodné byty
+204	4 byty	text „SDOS“

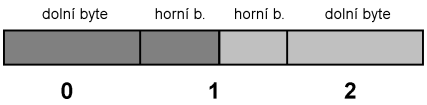
Informace o všech připojených mechanikách v době formátování není důležitá (nikde se v MDOSu nevyužívá). Jediné využití vidím v tom, že pokud dojde k nějaké chybě v sektoru a informace o disketě budou poškozená, dalo by se to obnovit z těchto informací. Ale vážně pochybuji o tom, že to půjde, protože může dojít k poškození i těchto dat a potom jsou na nic. Nevím, z jakého důvodu tam autoři tyto informace vložili. Informace o disketě a mechanice obsahují formát diskety a další informace. Strukturu těchto informací si popíšeme později. Dále je jméno diskety, které má deset znaků. A za ním následují dva náhodně vygenerované byty (no náhodně, vezmou se prostě z registru R), takže pokud budete mít dvě diskety se stejným jménem, tak ještě půjdou rozlišit podle těchto dvou bytů (pravděpodobnost je 1 : 65535). No a poslední informací je značka, pod kterým systémem byla disketa naformátována. MDOS zde má text „SDOS“. Všechny ostatní byty v BOOTu jsou vyplněny hodnotou 0. Můžete si sem uložit některé svoje informace, ale musíte dát pozor, abyste si něco nepřepsali.

4.2. FAT

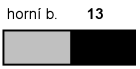
Je uložena v sektorech 2, 3, 4, 5, 6 (podle MDOSu 1, 2, 3, 4, 5) nulté stopy. FAT (File Alocation Table) je tabulka umístění souborů na disketě. Zabírá pět sektorů a obsahuje 1705 položek. Tabulka je 12-ti bitová, takže každá položka zabírá 1,5 byte, což je trochu neobvyklé. Každá položka je přiřazena jednomu sektoru. Jak to vůbec funguje? Soubor je umístěn v sektorech na disketě. Sektory jdou v nějaké řadě. Této řadě budeme říkat stezka. Jak projít tuto stezku? Nejdříve vyzvedneme číslo prvního sektoru uložení souboru. Z této hodnoty se vypočte adresa odpovídající položky ve FAT. A na této adrese je uloženo číslo následujícího sektoru souboru. Vyzvedneme číslo tohoto sektoru a opakujeme celý cyklus, dokud nenarazíme na značku konce souboru. Nyní si ukážeme, jak jsou uloženy položky ve FAT.

Na obrázku č. 8 vidíte způsob uložení položek ve FAT. Do 3 bytů jsou uloženy dvě položky. Do jednoho sektoru se tedy vejde 341 položek FAT a ještě nám zůstane půl bytu. Tento je vyplněn hodnotou 13. Tomuto půlbytu se říká zalomení FATky (když to tak pojmenoval GEORGE K., tak to ponechme). Druhá polovina je normálně využita. Zalomení FATky se nachází na lichých adresách. Také si všimněte, že sudé položky začínají na sudých adresách a liché na lichých adresách. Jak se ukládá hodnota do položky FAT tabulky? Řekněme, že chceme uložit do 1. položky nějakou hodnotu. Hodnotu vydělíme 256, výsledek si uložíme do B, zbytek do A. Na adresu 0 uložíme A, do A si vyzvedneme obsah adresy 1, ponecháme jen dolní 4 bity, registr B vynásobíme 16, přičteme k A a uložíme na adresu 1. Pokud chceme uložit do 2. položky nějakou hodnotu, tak hodnotu vydělíme 256, výsledek si dáme do B, zbytek do A, na adresu 2 uložíme A, do A vyzvedneme obsah adresy 1, ponecháme horní 4 bity, přičteme B a uložíme A na adresu 1. Největší číslo, které můžeme uložit do jedné položky FAT, je 4095 (to nás nijak neomezuje, protože máme pouze 1075 položek FAT).

obr. 8



obr. 9



Hodnoty, které jsou použity systémem a které mají nějaký význam:

Vyšší půlbyte		Nižší byte		Celková hodnota
%1101	(13)	%11011101	(221)	3549

Tato hodnota značí, že daný sektor je obsazen systémem (je v nich uložen BOOT, FAT, adresář) a sektory, které už nejsou dostupné (např. při formátu 40 stop, 9 sektorů to jsou sektory 720–1704)

%1101	(13)	%11111111	(255)	3583

Tato hodnota značí, že daný sektor je vadný a nelze ho nijak použít. Takto se označuje při formátování diskety.

%1100	(12)	%00000000	(0)	3072

Tato hodnota značí, že délka dat v tomto sektoru je nulová a sektor patří souboru s nulovou délkou nebo souboru, při jehož ukládání na disk došlo k chybě (musel se nějak ukončit).

%0000	(0)	%00000000	(0)	0

Tato hodnota značí, že sektor je nepoužit, takže do něj mohou být zapsána data.

Teď si ještě řekneme, jak se vytváří značka konce souboru. Vezmeme délku dat v posledním sektoru (je to vlastně zbytek po dělení délky souboru hodnotou 512). K vyššímu půlbytu přičteme %00001110 (14) – to neplatí pro prázdné soubory – a dostaneme tak značku konce souboru. Tuto hodnotu uložíme do poslední položky souboru ve FAT.

4.3. Adresář

Je uložen v sektorech 7, 8, 9, 10, 11, 12, 13, 14 (podle MDOSu 6, 7, 8, 9, 10, 11, 12, 13). Jsou zde uloženy všechny hlavičky souborů. Délka jedné hlavičky je 32 bytů, do jednoho sektoru se jich vejde 16, takže můžete mít na disketě až 128 souborů.

Stuktura hlavičky:

- **0** přípona souboru (P, C, N, B, S, Q) nebo hodnota 229, pokud je položka prázdná
- **1–10** jméno souboru, je-li kratší než 10 znaků, je doplněno nulami
- **11–12** první dva byty délky souboru (0 – 65535)
- **13–14** počáteční adresa souboru, u BASIC programu to je startovní řádek
- **15–16** u BASIC programu délka programu bez proměnných
- **17–18** číslo prvního sektoru souboru
- **19** nula
- **20** atributy souboru
- **21** třetí byte délky souboru pokud je délka větší než 65535
- **22–31** bez využití, zaplněno hodnotou 229

Pokud srovnáte s páskovou hlavičkou, je prvních 17 bytů stejných (až na první byte, kde je jiné kódování přípony). ‚P' odpovídá programu v BASICu, ‚C' odpovídá znakovému poli, ‚N' odpovídá číselnému poli, ‚B' odpovídá bloku bytů, ‚S' je snapshot a ‚Q' je sekvenční soubor. Třetí byte délky souboru je použit z důvodu zavedení sekvencí, protože mohou být delší než 65535 bytů. Pokud je uložena v příponě hodnota 229 (#E5), je položka adresáře označena jako prázdná.

Dále systém MDOS rozlišuje 8 atributů (je 8 bitů):

7. bit H–Hidden	– příkaz CAT nevypisuje	3. bit R–Readable	– soubor je možné číst
6. bit S–System	– není nikde použito	2. bit W–Writeable	– do souboru je možno zapisovat
5. bit P–Protected	– není nikde použito	1. bit E–Executeable	– soubor je možné spustit
4. bit A–Archive	– není nikde použito	0. bit D–Deleteable	– soubor je možné smazat

4.4. Datová oblast

Začíná od sektoru 15 (podle MDOSu 14) a zde jsou již uložena data. Její délka je: **počet_stop_na_stranu × počet_sektorů_na_stopu × počet_stran – počet_system_sektorů + – počet_vadných_sektorů** sektorů.

5. Informace o disketě a mechanice

Každá mechanika má v paměti SRAM vyhrazeno 12 bytů pro uložení technických informací o ní a o disketě, která je v ní vložena.

Pro tyto informace je v paměti SRAM vyhrazeno 48 bytů (12 bytů pro 4 mechaniky). Začínají na adrese #3E00 (DRPARZN). Za těmito informacemi je vyhrazeno pro každou mechniku místo pro jméno diskety v mechanice (12 bytů – jméno drivu).

<table>
<tr><th>Byte</th><th>Bit</th><th colspan="5">Informace</th><th></th></tr>
<tr><td rowspan="11">1.</td><td>0.</td><td colspan="4">informace, jestli je mechanika připojena</td><td>1 – mechanika je připojena
0 – mechanika není připojena</td><td rowspan="12">Parametry diskety</td></tr>
<tr><td>1.</td><td colspan="4">informace, jestli při poslední operaci s mechanikou došlo k chybě nebo byly zastaveny mechaniky</td><td>1 – nedošlo k chybě a mechaniky běží
0 – došlo k chybě nebo byly zastaveny mechaniky</td></tr>
<tr><td>2.</td><td colspan="4">informace, jaká je to mechanika (A nebo B)</td><td>0 – mechanika A
1 – mechanika B</td></tr>
<tr><td>3.</td><td colspan="4">informace, kolika stopá je mechanika</td><td>0 – D80
1 – D40</td></tr>
<tr><td>4.</td><td colspan="4">informace, kolika stranná disketa je v mechanice</td><td>0 – jednostranná
1 – oboustranná</td></tr>
<tr><td>5.</td><td colspan="4">jaká je disketa v mechanice</td><td>0 – 40/80-ti stopá disketa v 40/80-ti stopé mechanice
1 – 40-ti stopá disketa v 80-ti stopé mechanice</td></tr>
<tr><td rowspan="6">6., 7.</td><td colspan="5">rychlost krokování mechaniky</td></tr>
<tr><td>7. bit</td><td>6. bit</td><td>2 MHz</td><td>1 MHz</td></tr>
<tr><td>0</td><td>0</td><td>3 ms</td><td>6 ms</td></tr>
<tr><td>0</td><td>1</td><td>6 ms</td><td>12 ms</td></tr>
<tr><td>1</td><td>0</td><td>10 ms</td><td>20 ms</td></tr>
<tr><td>1</td><td>1</td><td>15 ms</td><td>30 ms</td></tr>
<tr><td>3.</td><td colspan="6">Zde je uložen počet stop na straně diskety (pokud je zde 0, disk není definován).</td></tr>
<tr><td>4.</td><td colspan="6">Zde je uložen počet sektorů na stopě diskety.</td></tr>
<tr><td>5.</td><td colspan="6">Zde se ukládá číslo stopy, kam byla naposledy vystavena hlava mechaniky.</td><td rowspan="4">Par. mechanik</td></tr>
<tr><td>6.</td><td colspan="6">Jsou zde uloženy stejné parametry jako v 2. bytu mimo 5. bitu. Ten se týká pouze mechaniky.</td></tr>
<tr><td>7.</td><td colspan="6">Zde je uložen počet stop mechaniky.</td></tr>
<tr><td>8.</td><td colspan="6">Zde je uložen počet sektorů na stopě mechaniky.</td></tr>
<tr><td>9.–11.</td><td colspan="6">Vyplněno nulami.</td></tr>
</table>

6. Opravy a úpravy MDOSu

Protože žádný učený z nebe nespadl, má systém MDOS i několik malých chyb, na které bylo už upozorňováno v ZX Magazínu (ale jsou i další). První verze MDOSu, který nese datum 17. května 1991, obsahuje všechny publikované chyby (i ty nepublikované). Časem, když se na ně přišlo (na publikované, o nepublikovaných nikdo neví), tak byly některé opraveny (MDOS nese datum 1. září 1992). Samotné opravy však nezpůsobují nekompatibilitu MDOSů. Opravy MDOSu jsou v komentovaném výpisu zvýrazněny.

7. Komentovaný výpis MDOSu

RST #00

Základní vstupní bod do ROM D40. Při čtení instrukce z adresy #0000 je místo ZX ROM přistránkována ROM D40. Zde se vstupuje při RESETu, tisku chybového hlášení, čtení/zápisu znaku z/do sekvenčního souboru, SNAPu a návratu z rutin ZX ROM zpět do ROM D40.

```
0000 START      NOP                          ;taková nic neříkající instrukce
0001            JR    #0068, START1          ;skoč na testování, jaká operace se provádí

0003            DB    #FF, #FF, #FF, #FF, #FF ;nevyužito
```

RST #08

Druhý vstupní bod do ROM D40. Sem se skáče při interpretaci příkazů pro práci s disketovou jednotkou. ZX ROM rozpozná, že nezná danou syntaxi příkazu a zavolá RST #08. Při čtení instrukce z adresy #0008 dojde k přestránkování a začne se kontrolovat syntaxe příkazu v ROM D40. Pokud ani ROM D40 nezná daný příkaz, řízení se vrací zpět do ZX ROM a je generováno chybové hlášení.

```
0008 SYNTAX     LD    HL, (#5C5D) CH_ADD     ;do HL dej adresu znaku pro dekódování
000B            JP    #0215, SYNTAX1         ;skoč na výběr a kontrolu příkazů

000E            DB    #FF, #FF               ;nevyužito
```

RST #10

Tisk znaku v registru A. Má stejný význam jako RST #10 v ZX ROM. Přes tento podprogram ale nelze vysílat znaky do sekvenčních souborů.

IN: A kód znaku, který se bude tisknout
OUT: vytiskne znak na otevřený kanál

```
0010 PRINTA     RST   #28                    ;volej podprogram pro volání rutiny ZX ROM
0011            DW    #0010                  ;podprogram PRINT-A-1 tisk znaku v A
0013            RET                          ;vrať se

0014            DB    #FF, #FF, #FF, #FF     ;nevyužito
```

RST #18

Načte do registru A obsah místa adresovaného systémovou proměnnou CH_ADD. Návrat, jestli se jedná o znak použitelný k tisku, jinak je obsah CH_ADD zvětšen a vše se opakuje. Má stejný význam jako RST #18 v ZX ROM.

IN: –
OUT: A obsah místa adresovaného sytémovou proměnnou CH_ADD

```
0018 SET-CHAR   RST   #28                    ;volej podprogram pro volání rutiny ZX ROM
0019            DW    #0018                  ;podprogram SET-CHAR načtení znaku
001B            RET                          ;vrať se

001C            DB    #FF, #FF, #FF, #FF     ;nevyužito
```

RST #20

Načte do A další znak při interpretaci BASIC řádku. Má stejný význam jako RST #20 v ZX ROM.

IN: –
OUT: A znak pro dekódování

```
0020 NEXT-CHAR  RST  #28                ;volej podprogram pro volání rutiny ZX ROM
0021            DW   #0020              ;podprogram NEXT-CHAR načtení dalšího znaku
0023            RET                     ;vrať se

0024            DB   #FF, #FF, #FF, #FF ;nevyužito
```

RST #28

Velice důležitá rutina. Slouží k volání rutiny v ZX ROM přímo z ROM D40. Za příkazem RST #28 následuje adresa rutiny, který se bude volat v ZX ROM. Po návratu z rutiny se zpět přestránkuje do ROM D40 a program pokračuje za adresou rutiny v ZX ROM. Vypadá to takto:

```
        RST  #28
        DW   adresa rutiny v ZX ROM
        …    (tady pokračuje program po návratu z rutiny ZX ROM a je přistránkována ROM D40)

0028 CALLZXROM  JP   #003B, CALLZX1     ;skoč na podprogram volající rutiny ze ZX ROM

002B            DB   #FF, #FF, #FF, #FF, #FF ;nevyužito
```

RST #30

Slouží k testování, jestli se provádí test syntaxe příkazu nebo se provádí příkaz, podle 7. bitu proměnné FLAGS.

IN: –
OUT: NZ vykonává se příkaz
Z provádí se kontrola syntaxe příkazu

```
0030 TESTSYN    BIT  7, (IY+#01) FLAGS ;testuj, jestli je kontrola syntaxe nebo provedení příkazu
0034            RET                     ;vrať se

0035            DB   #FF, #FF, #FF      ;nevyužito
```

RST #38

Podprogram pro obsluhu přerušení od ULA v případě, že je přestránkováno do ROM D40 a přerušení je povoleno. Platí pouze pro IM 1. Tam se totiž volá každou padesátinu sekundy RST #38. Aby nedošlo k zamrznutí počítače, je zde krátký prográmek, který to ošetřuje. Pokud tedy budete používat rutiny ROM D40 a přerušení IM 1, nemusíte po přestránkování zakazovat přerušení.

```
0038 MASK-INT   JP   #25CE, INTERRUPT   ;skoč na obsluhu přerušení
```

Zde se provádí volání rutin ZX ROM (adresa rutiny se nachází za voláním podprogramu).

```
003B CALLZX1    PUSH AF                 ;schovej si A a příznaky
003C            LD   A, (#3EEE) SNAPINF ;vyzvedni informaci o snapu
003F            AND  A                  ;a otestuj, jestli se provádí snap
```

Pokud se provádí snap, nelze volat rutiny ze ZX ROM.

```
0040            JP   NZ, #034A, SNPRET  ;ano → skoč na návrat přes snap
0043            POP  AF                 ;obnov AF
```

Nyní vyzvedneme adresu volané rutiny.

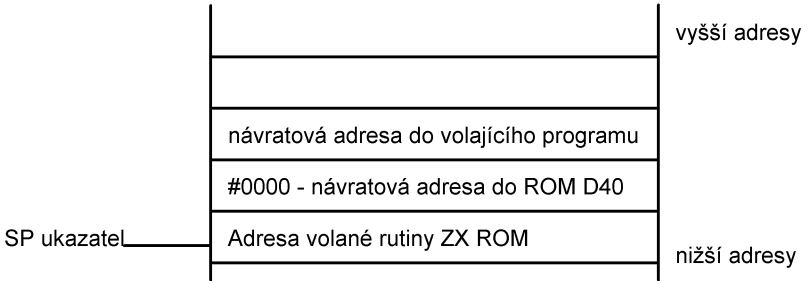
```
0044            EX   (SP), HL           ;nastav HL na uložení adresy rutiny, která se bude volat
0045            LD   (#3E66), DE SAVEDE ;schovej si na chvíli DE
0049            LD   E, (HL)            ;vezmi do DE adresu požadované rutiny
004A            INC  HL
004B            LD   D, (HL)
```

```
004C            INC   HL                ;v HL je nyní návratová adresa do volajícího programu

   Nastavíme si adresy na zásobník.
004D            EX    (SP), HL          ;ulož si návratovou adresu do volajícího programu zpět
                                        ;na zásobník
004E            PUSH  HL                ;ulož si HL
004F            LD    HL, #3EF7 SYSFLAG ;do HL adresa uložení kódu činnosti
0052            LD    (HL), #4F         ;nastav činnost „volání rutiny ZX ROM“
0054            LD    HL, #0000 START   ;do HL dej návratovou adresu z rutiny ZX ROM do
                                        ;ROM D40 (po návratu se bude skákat na adresu #0000,
                                        ;kde se podle obsahu SYSFLAG provede návrat do
                                        ;volajícího programu)
0057            EX    (SP), HL          ;ulož ji na zásobník a obnov HL
0058            PUSH  DE                ;ulož adresu volané rutiny na zásobník
0059            LD    DE, (#3E66) SAVEDE ;obnov si DE

   A provedeme rutinu s návratem do ROM D40.
005D            JP    #1700, STANDROM   ;skoč na přestránkování do ZX ROM
```

Pořadí položek na zásobníku




```
0060            DB    #FF, #FF, #FF, #FF, #FF, #FF    ;nevyužito
```

NMI

Tento podprogram se používá při přenosu dat z/do mechaniky. Je vyvolán při požadavku DRQ a INTRQ řadiče, jsou-li povoleny (6. bit na portu 137). Při přenosu dat je 512-krát (pro každý byte) vyvoláváno hardwarové přerušení NMI, které skáče na tuto adresu. V registru IX je adresa, kam se bude skákat.

IN: IX adresa podprogramu, kam se bude skákat
OUT: načtení/zápis byte z/na disketu do/z paměti

```
0066 NMI        JP    (IX)              ;skoč na adresu v IX
```

Zde se testuje, jaká operace se provádí (RESET, návrat z volání rutiny ZX ROM, SNAP, čtení/zápis z/do sekvenčního souboru, tisk chybového hlášení). Danou operaci určuje buď kód v SYSFLAG nebo návratová adresa do volajícího programu.

```
0068 START1     EX    (SP), HL          ;vezmi návratovou adresu do programu, ulož ji do HL
                                        ;a ulož si HL na zásobník
0069            PUSH  BC                ;ulož si BC
006A            PUSH  AF                ;a AF
```

Uložíme si stav přerušení a vektor přerušení.

```
006B            LD    A, I                ;vezmi horní byte vektoru přerušeni a stav IFF (stav
                                          ;přerušení, jestli bylo povoleno nebo zakázáno)
006D            DI                        ;zakaž přerušení
006E            PUSH  AF                  ;ulož horní byte vektoru přerušení a stav přerušení
006F            POP   BC                  ;a vyzveni vše do BC
0070            LD    (#3EEC), BC IREG2   ;a ulož do paměti SRAM
0074            PUSH  HL                  ;ulož si návratovou adresu do programu na zásobník
```

Nyní zkontrolujeme, jestli jsou v pořádku kontrolní byty. Pokud nejsou v pořádku, inicializuje se MDOS.

```
0075            LD    HL, #3EEF SYSMRK    ;do HL adresa začátku tabulky kontrolních bytů
0078            LD    B, #08              ;celkem tvoří tabulku 8 bytů
007A CHCOLD     LD    A, H                ;vytvoř obraz bytu
007B            XOR   L
007C            CP    (HL)                ;souhlasí s obsahem tabulky?
007D            JR    NZ, #00B4, COLD     ;ne → skoč na inicializaci MDOSu
007F            INC   HL                  ;posuň se na další byte v tabulce
0080            DJNZ  #007A, CHCOLD       ;opakuj B-krát
```

Tabulka je v pořádku, otestujeme nyní, jaká činnost se bude provádět podle obsahu SYSFLAG.

```
0082            LD    A, (HL)             ;vezmi kód činnosti ze SYSFLAG
0083            LD    (HL), #20           ;nastav kód činnosti „žádná činnost“
0085            CP    #4F                 ;je kód činnosti „návrat z rutiny ZX ROM“?
0087            JP    Z, #013A, ROMRET    ;ano → skoč na návrat do volajícího programu
008A            CP    #45                 ;je kód činnosti „výpis chybového hlášení“?
008C            JP    Z, #0140, ERRCOD    ;ano → skoč na výpis chybového hlášení MDOSu nebo
                                          ;ZX ROM
008F            POP   HL                  ;obnov do HL návratovou adresu do programu
```

Nyní je zde něco, čemu jsem nepřišel na kloub. Vypadá to jako pozůstatek ladící rutiny. Pokud za skokem na adresu #0000 následují znaky „*“ a „=“, je změněn border podle 3. bytu, vypsán obsah registru HL (v době volání RST #00) do otevřeného kanálu a řízení se vrací zpět do volajícího programu za 3. byte.

```
0090            LD    A, (HL)             ;vezmi byte za volání adresy #0000
0091            CP    „*“                 ;je to hvězdička?
0093            JP    Z, #01B0, SPECCOMM  ;ano → skoč na interpretaci speciálního příkazu
```

Nyní se provede kontrola návratové adresy, odkud byl skok na adresu #0000, podle tabulky, ve které jsou uloženy povolené návratové adresy. Pokud některá z těchto adres odpovídá návratové adrese do volajícího programu, neprovádí se RESET, ale určitý typ operace. Jsou to čtení/zápis z/do sekvenčního souboru a SNAP.

```
0096            PUSH  DE                  ;ulož si DE
0097            LD    B, H                ;ulož si návratovou adresu do BC pro porovnání
0098            LD    C, L
0099            LD    HL, #019A IORTAB    ;do HL začátek tabulky povolených návratových adres
009C TESTROUT   LD    E, (HL)             ;vezmi návratovou adresu z tabulky do DE
009D            INC   HL
009E            LD    D, (HL)
009F            INC   HL
00A0            EX    DE, HL              ;předej do HL a polohu v tabulce dej do DE
00A1            LD    A, H                ;je konec tabulky?
00A2            OR    L
00A3            JR    Z, #00B4, COLD      ;ano → skoč na inicializaci MDOSu
```

```
00A5            SBC   HL, BC                ;porovnej HL s návratovou adresou do programu
00A7            EX    DE, HL                ;ukazatel v tabulce dej zpět do HL
00A8            LD    E, (HL)               ;vezmi do DE adresu rutiny z tabulky, která se bude
                                            ;volat, pokud je návratová adresa povolena
00A9            INC   HL
00AA            LD    D, (HL)
00AB            INC   HL
00AC            JR    NZ, #009C, TESTROUT   ;adresy nesouhlasí → skoč na otestování další položky
                                            ;tabulky
```

Pokud byla návratová adresa nalezena v tabulce, obnovíme všechny registry do původního stavu a provedeme skok na vybranou rutinu.

```
00AE            EX    DE, HL                ;adresu rutiny do HL
```

Obnovíme registry do původního stavu.

```
00AF            POP   DE                    ;obnov DE
00B0            POP   AF                    ;obnov AF
00B1            POP   BC                    ;obnov BC
00B2            EX    (SP), HL              ;adresu rutiny ulož na zásobník a obnov HL
00B3            RET                         ;skoč do rutiny
```

Inicializační rutina. Provádí se při resetu. Testuje RAM, inicializuje kontrolní byty, testuje připojené mechaniky a nastavuje jejich parametry, nastavuje systémové proměnné MDOSu a inicializuje připojený obvod 8255.

Nejdříve se otestuje SRAM, jestli není vadná.

```
00B4 COLD       LD    HL, #0000             ;začátek MDOSu do HL
00B7            LD    DE, #3800             ;začátek SRAM do DE
00BA            LD    BC, #0800             ;délka SRAM je 2 KB
00BD            LDIR                        ;presuň do paměti SRAM
```

Porovnáme obsah MDOS ROM a SRAM.

```
00BF            LD    HL, #0000             ;začátek MDOSu do HL
00C2            LD    DE, #3800             ;začátek SRAM do DE
00C5            LD    BC, #0800             ;délka SRAM je 2 KB
00C8 RAMTEST    LD    A, (DE)               ;vezmi obsah buňky SRAM do A
00C9            INC   DE                    ;a posuň se na adresu další buňky
00CA            CPI                         ;porovnej s obsahem MDOS ROM
00CC            JR    NZ, #012F, RAMERR     ;nesouhlasí → skoč na chybu RAM
00CE            JP    PE, #00C8, RAMTEST    ;opakuj BC krát
```

Nyní provedeme vymazání paměti SRAM (dosadíme nuly do všech buněk).

```
00D1            LD    HL, #3800             ;začátek SRAM do HL
00D4            LD    D, H                  ;okopíruj do DE
00D5            LD    E, L
00D6            INC   DE                    ;a posuň se na další adresu
00D7            LD    BC, #0800             ;délka je 2 KB (maže se o 1 byte více, ale tady to nevadí)
00DA            LD    (HL), #00             ;plnící hodnota je nula
00DC            LDIR                        ;vyplň všechny buňky nulou
```

Vytvoříme tabulku kontrolních bytů.

```
00DE            LD    HL, #3EEF SYSMRK      ;adresa tabulky kontrolních bytů do HL
00E1            LD    B, #08                ;celkem 8 bytů
00E3 SETMRK     LD    A, H                  ;vytvoř hodnotu
```

```
00E4            XOR  L
00E5            LD   (HL), A            ;a ulož ji do tabulky
00E6            INC  HL                 ;posuň se na další buňku
00E7            DJNZ #00E3, SETMRK      ;opakuj B-krát
```

Nastavíme systémové proměnné MDOSu.

```
00E9            LD   (HL), #20          ;nastav na SYSFLAG kód činnosti „žádná činnost“
```

Nyní se zde nachází celkem zbytečná část, kterou autoři zapoměli při odladění vymazat. Pokud při resetu budete držet klávesy SPACE–M–B, uloží se do systémové proměnné DEBUG nenulová hodnota a budou se Vám při operacích vypisovat kontrolní tisky některých důležitých proměnných. Tato část programu testuje stisknutou trojici kláves.

```
00EB            LD   A, #7F             ;do A dej horní byte portu
00ED            IN   A, (#FE)           ;přečti hodnotu z portu #7FFE
00EF            RRA                     ;bylo stisknuto SPACE?
00F0            JR   C, #0101, NODEB    ;ne → přeskoč testy ostatních kláves
00F2            RRA                     ;byl stisknut SYMBOL SHIFT?
00F3            JR   NC, #0101, NODEB   ;ano → přeskoč ostatní testy
00F5            RRA                     ;bylo stisknuto M?
00F6            JR   C, #0101, NODEB    ;ne → přeskoč ostatní testy
00F8            RRA                     ;bylo stisknuto N?
00F9            JR   NC, #0101, NODEB   ;ano → přeskoč poslední test
00FB            RRA                     ;bylo stisknuto B?
00FC            JR   C, #0101, NODEB    ;ne → nebyl trojhmat, skoč
00FE            LD   (#3E60), A DEBUG   ;ulož informaci „byl trojhmat“
```

Nyní nastavíme výchozí parametry disků A: a B:.

```
0101 NODEB      LD   DE, #3E00 DRPARZN  ;do DE začátek tabulek disků v SRAM
0104            LD   HL, #0EF8 ROMDRPAR ;adresa standardních parametrů v ROM D40 do HL
0107            LD   BC, #18            ;24 bytů (2 × 12 bytů) pro dvě mechaniky
010A            LDIR                    ;přesuň do SRAM
010C            LD   A, „A“             ;jméno aktuálního drivu bude „A“
010E            LD   (#3EAA), A ACDRIVE ;ulož do jména aktuálního drivu
0111            LD   HL, #5800          ;do HL adresu začátku atributů obrazovky
0114            LD   DE, #5801          ;do DE adresu začátku atributů obrazovky plus 1
0117            LD   BC, #0300          ;délka je 768 bytů (zase se vyplňuje o 1 byte více)
011A            LD   (HL), #12          ;PAPER červený, INK červený
011C            LDIR                    ;a teď nám zčervená obrazovka
011E            LD   A, #02             ;BORDER bude taky červený
0120            OUT  (#FE), A           ;už je
0122            LD   SP, #4000          ;zásobník nastav až na konec SRAM
0125            CALL #2216, HWINIT      ;testuj mechaniky a port 8255
```

Test disketové jednotky proběhl, takže se vrátíme do ZX ROM, aby se mohlo pokračovat v inicializaci počítače. Návrat je proveden zajímavým způsobem. Zásobník je nastaven na adresu #1019, kde je uložena hodnota #0001 v ZX ROM. Tato hodnota je tam vždy, takže při návratu se skáče na adresu #0001, kde se pokračuje v inicializaci. Nemůžeme se vrátit na adresu #0000, protože by došlo k zacyklení.

```
0128            DI                      ;zakaž přerušení pro inicializaci ZX ROM
0129            LD   SP, #1019          ;návratová adresa do ZX ROM je #0001 (abychom se
                                        ;nezacyklili) a ta je na adrese #1019 v ZX ROM
```

```
012C        JP    #1700, STANDROM    ;skoč na přestránkování do ZX ROM a pokračování
                                     ;v inicializaci počítače
```

RAMERR

Zde se skáče, jestli při testu SRAM dojde k chybě (vadná SRAM, špatně zapojený kabel, atd.). Mění se border a počítač nějakou dobu vrčí. Potom se znovu zkusí reset.

```
012F RAMERR XOR   A                  ;na začátku je v A nula
0130 CYCLE  DEC   A                  ;sniž A
0131        OUT   (#FE), A           ;nastav BORDER a repro
0133        EX    (SP), HL           ;časová prodleva
0134        EX    (SP), HL           ;trvá 38 taktů
0135        JR    NZ, #0130, CYCLE   ;opakuj A-krát
0137        JP    #0000, START       ;zkus znovu reset
```

Zde dochází k návratu z rutin ROM D40 a z volání rutin ZX ROM. Důležité je, že zůstane přistránkována ROM D40.

```
013A ROMRET POP   HL                 ;vezmi návratovou adresu do HL
013B        POP   AF                 ;obnov registry
013C        POP   BC
013D        EX    (SP), HL           ;ulož návratovou adresu na zásobník a obnov HL
013E        EI                       ;povol přerušení
013F        RET                      ;vrať se zpět do volajícího programu
```

Sem se skáče, pokud dojde k návratu z chybového hlášení. Pokud při nějaké operaci dojde k chybě, volá se chybové hlášení na #0204 (obdoba RST #08 v BASICu) s kódem chyby v A. Tam se přepíše obsah adresy v ERR_SP nulou, proměnná SYSFLAG je nastavena na činnost „tisk chybového hlášení“ a provede se návrat do ZX ROM na adresu #000B. Dojde k vyzvednutí kódu chyby, vyčištění zásobníku a dalších proměnných a řízení se vrací na adresu, na kterou ukazuje proměnná ERR_SP. Protože sem MDOS uložil 0, provede skok na adresu #0000, kde dojde k přestránkování znovu do ROM D40, kde se podle SYSFLAG provede skok na ERRCOD, kde je opsán kus programu ze ZX ROM s malými změnami (v BASICu se tam vrací řízení po nalezení chyby), který vytiskne buď chybové hlášení MDOSu a provede návrat do ZX ROM za tisk chybového hlášení, nebo provede skok do ZX ROM před tisk chybového hlášení, pokud je to hlášení BASICu. Potom již pokračuje program v ZX ROM. To vše se děje za předpokladu, že ERR_SP ukazuje na adresu, na níž je uložena hodnota #1303. Pokud tam taková hodnota není, znamená to, že je v paměti program na ošetření chybových hlášení. Není tedy přepsán obsah adresy, na kterou ukazuje ERR_SP, ale provede se pouze návrat na adresu #000B, kde dojde k vyčištění zásobníku, proměnných BASICu, vyzvednutí kódu chyby a provede se skok do do rutiny ošetřující chybové hlášení (je vynechán skok na adresu #0000 a tisk hlášení).

```
0140 ERRCOD POP   HL                 ;vezmi návratovou adresu do HL
0141        POP   AF                 ;obnov registry
0142        POP   BC
0143        EX    (SP), HL           ;ulož návratovou adresu na zásobník a obnov HL
0144        EI                       ;povol přerušení
```

Nyní je opsán kus programu ze ZX ROM s malými změnami.

```
0145        HALT                     ;čekej na přerušení
0146        RES   5, (IY+#01) FLAGS  ;nastav signál „možno další klávesu“
014A        BIT   1, (IY+#30) FLAGS2 ;byl použit buffer tiskárny?
014E        JR    Z, #0153, NOCOPYBUF ;ne → přeskoč čištění bufferu
0150        RST   #28                ;volej podprogram pro volání rutiny ZX ROM
0151        DW    #0ECD              ;podprogram COPY-BUFF vyčištění bufferu tiskárny
```

```
0153 NOCOPYBUF  LD    A, (#5C3A) ERR_NR   ;vezmi číslo chyby do A
0156            INC   A                   ;zvyš o 1
0157            PUSH  AF                  ;a ulož si tuto hodnotu
0158            LD    HL, #0000           ;nuluj systémové proměnné
015B            LD    (IY+#37), H FLAGX   ;FLAGX, X_PTRhi, DEFADD
015E            LD    (IY+#26), H X_PTRhi
0161            LD    (#5C0B), HL DEFADD
0164            INC   HL                  ;zajisti, aby proud 0
0165            LD    (#5C16), HL STRMS6  ;ukazoval na kanál „K“
0168            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0169            DW    #16B0               ;podprogram SET-MIN vyčištění všech pracovních
                                          ;oblastí a kalkulátorový zásobník
016B            RES   5, (IY+#37) FLAGX   ;nastav režim EDIT
016F            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0170            DW    #0D6E               ;podprogram CLS-LOWER smazání dolní obrazovky
0172            SET   5, (IY+#02) TVFLAG  ;nastav signál „vyčistit dolní část obrazovky“
0176            CALL  #217B, ERAVAR       ;vymaž pomocné proměnné MDOSu
0179            POP   AF                  ;vyzvedni hodnotu hlášení
017A            CP    #1C                 ;byl kód chyby < #1C (hlášení ZX ROM)?
017C            JR    NC, #0185, ERRMDOS  ;ne → skoč na výpis chyby MDOSu
017E            LD    HL, #1335           ;do HL návratová adresa do ZX ROM, bude se tisknout
                                          ;chybové hlášení ZX ROM
```

Provádí se návrat do ZX ROM na adresu v HL.

```
0181 HLROMRET   PUSH  HL                  ;ulož návratovou adresu do ZX ROM na zásobník
0182            JP    #1700, STANDROM     ;skoč na přestránkování do ZX ROM
```

Vytiskneme chybové hlášení MDOSu.

```
0185 ERRMDOS    LD    B, A                ;kód chyby ulož do B
0186            ADD   A, #07              ;uprav na znak
0188            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0189            DW    #15EF               ;podprogram OUT-CODE vyslání znaku v A do kanálu
018B            LD    A, „ “              ;mezera
018D            RST   #10                 ;tiskni mezeru
018E            LD    A, B                ;vezmi pořadí požadovaného hlášení
018F            LD    DE, #03AF SYSMSG    ;do DE začátek systémových hlášení
0192            CALL  #01C8, PRTMES       ;piš požadované hlášení
0195            LD    HL, #1349           ;návratová adresa do ZX ROM, budeme pokračovat už
                                          ;v ZX ROM za výpisem chybového hlášení
0198            JR    #0181, HLROMRET     ;ulož ji na zásobník a přestránkuj zpět do ZX ROM
```

Tady se nachází tabulka povolených návratových adres, odkud se může volat RST #00 (nedojde při tom k inicializaci D40 a resetu) a odpovídajících rutin. V komentovaném výpisu ZX ROM sice na těchto adresách nenajdete přímo příkaz RST #00, ale věřte, že tam je. První slovo je návratová adresa do ZX ROM a druhé slovo je adresa rutiny v ROM D40.

```
019A IORTAB     DW    #22C3               ;návratová adr. do ZX ROM při čtení znaku z kanálu
019C            DW    #0DD9               ;adresa rutiny v ROM D40 pro čtení znaku z kanálu
019E            DW    #25AC               ;návratová adr. do ZX ROM pro zápis znaku do kanálu
01A0            DW    #0E23               ;adresa rutiny v ROM D40 pro zápis znaku do kanálu
01A2            DW    #27A6               ;návratová adr. do ZX ROM pro čtení znaku z kanálu
01A4            DW    #0DD9               ;adresa rutiny v ROM D40 pro čtení znaku z kanálu
```

```
01A6                DW   #27AD          ;návratová adr. do ZX ROM pro zápis znaku do kanálu
01A8                DW   #0E1E          ;adresa rutiny ROM D40 pro zápis znaku do kanálu
01AA                DW   #0067          ;návratová adr. do ZX ROM při NMI (snapu)
01AC                DW   #02E7          ;adresa rutiny ROM D40 pro uložení snapu
01AE                DW   #0000          ;konec tabulky
```

Nyní následuje tělo programu, které je pokračováním onoho záhadného pozůstatku z ladění, kdy se testuje 2. znak „=“, pokud není, provede se reset, pokud je, provede se výpis obsah registru HL v době volání RST #00, změní se border a program se regulérně vrátí zpět

```
01B0 SPECCOMM  INC  HL              ;posuň se na 2. byte za RST #00
01B1           LD   A, (HL)         ;vyzvedni obsah adresy do A
01B2           CP   „=“             ;je tam rovnítko?
01B4           JP   NZ, #00B4, COLD ;ne → skoč na inicializaci MDOSu
01B7           INC  HL              ;posuň se na 3. byte za RST #00
01B8           LD   A, (HL)         ;vyzvedni obsah adresy do A
01B9           INC  HL              ;posuň se na další byte a tak vytvoř návratovou adresu
01BA           POP  BC              ;vyzvedni dvě hodnoty ze zásobníku (AF a BC)
01BB           POP  BC
01BC           EX   (SP), HL        ;ulož návratovou adresu na zásobník a obnov HL
01BD           EI                   ;povol přerušení
01BE           OUT  (#FE), A        ;změň border
01C0           LD   C, L            ;dej návratovou adresu z HL do BC
01C1           LD   B, H
01C2           CALL #0FA6, BCPRT    ;vypiš obsah BC na právě otevřený kanál
01C5           JP   #1700, STANDROM ;skoč na přestránkování do ZX ROM
```

PRTMES

Podprogram pro tisk položky z tabulky textů. Každá položka končí znakem, jehož 7. bit je nastaven na 1. Samotná tabulka začíná invertovaným znakem.

IN: A číslo položky (od nuly)
DE začátek tabulky, která začíná invertovaným znakem
OUT: výpis položky na obrazovku

```
01C8 PRTMES    EX   DE, HL            ;dej začátek tabulky do HL
01C9           INC  A                 ;zvyš A o 2
01CA           INC  A
```

Budeme hledat požadovanou položku.

```
01CB SETMESS   DEC  A                 ;sniž číslo položky
01CC           JR   Z, #01D5, PMESSAGE ;položka nalezena → skoč na výpis položky
01CE SETNMESS  BIT  7, (HL)           ;konec položky?
01D0           INC  HL                ;posuň se na další znak
01D1           JR   Z, #01CE, SETNMESS ;není konec → hledej konec položky
01D3           JR   #01CB, SETMESS    ;zkontroluj další položku textu
```

Nyní vytiskneme text položky na obrazovku.

```
01D5 PMESSAGE  LD   A, (HL)           ;vezmi znak z textu do A
01D6           PUSH HL                ;ulož si adresu znaku
01D7           RES  7, A              ;zruš 7. bit
```

Otestujeme, jestli se v textu nebude vypisovat jméno disku nebo souboru. Pokud se v textu vyskytne hodnota

#23, tiskne se jméno disku uložené v DNZONE1, při hodnotě #40 se tiskne jméno souboru v FNZONE1.

```
01D9            CP   #23                ;má se vložit jméno disku?
01DB            LD   HL, #3E80 DNZONE1  ;do HL adresa 1. jména disku pro I/O
01DE            JR   Z, #01F1, PRNAME   ;ano → skoč na tisk jména disku
01E0            CP   #40                ;má se vložit jméno souboru?
01E2            LD   HL, #3E8A FNZONE1  ;do HL adresa 1. jména souboru pro I/O
01E5            JR   Z, #01F1, PRNAME   ;ano → skoč na tisk jména souboru
01E7            POP  HL                 ;vyzvedni adresu uložení znaku
01E8            RST  #28                ;volej podprogram pro volání rutiny ZX ROM
01E9            DW   #0C3B              ;podprogram PO-SAVE tisk znaku v A
01EB            BIT  7, (HL)            ;je konec položky?
01ED            RET  NZ                 ;ano → vrať se z výpisu
01EE PNEXTCH    INC  HL                 ;posuň se na další znak v položce
01EF            JR   #01D5, PMESSAGE    ;pokračuj ve výpisu položky
```

Pokud vznikne požadavek vytisknout v textu jméno disku nebo jméno souboru, je proveden skok sem, odkud se po vytištění vrátíme zpět do tisku položky. V HL je adresa uložení buď jména souboru nebo disku. Je tištěno buď 10 znaků nebo dokud se nenarazí na 0 (znak „konec jména“).

```
01F1 PRNAME     PUSH BC                 ;schovej si BC
01F2            LD   B, #0A             ;délka jména je až 10 bytů
01F4 PRNAMEL    LD   A, (HL)            ;vezmi znak
01F5            AND  A                  ;je konec jména?
01F6            JR   Z, #0200, STOPPRNM ;ano → skoč na konec výpisu jména
01F8            INC  HL                 ;posuň se na další znak ve jméně
01F9            PUSH BC                 ;schovej si počet znaků k tisku
01FA            RST  #28                ;volej podprogram pro volání rutiny ZX ROM
01FB            DW   #0C3B              ;podprogram PO-SAVE tisk znaku v A
01FD            POP  BC                 ;obnov počet znaků k tisku
01FE            DJNZ #01F4, PRNAMEL     ;opakuj B-krát
```

Bylo vytištěno celé jméno.

```
0200 STOPPRNM   POP  BC                 ;obnov BC a HL
0201            POP  HL
0202            JR   #01EE, PNEXTCH     ;skoč na pokračování tisku položky
```

ERRR

Zde je vstupní bod, když dojde k nějaké chybě. Je to vlastně obdoba RST #08. Lze volat jak chyby MDOSu, tak i chyby ZX BASICu.

IN: A číslo chyby
OUT: je vytištěno chybové hlášení a řízení se vrací zpět do ZX ROM

```
0204 ERRR       PUSH AF                 ;ulož si kód chyby
```

Nejdříve otestujeme, jestli nedošlo k chybě při snapu, protože potom se nevypisuje žádné chybové hlášení, ale vrací se zpět přes návrat ze snapu.

```
0205            LD   A, (#3EEE) SNAPINF ;vyzvedni informaci, jestli není snap
0208            AND  A                  ;je snap?
0209            JP   NZ, #034A, SNPRET  ;ano → skoč na návrat do ZX ROM přes snap
020C            POP  AF                 ;vyzvedni zpět kód chyby
020D            LD   HL, #5C3A ERR_NR   ;adresa uložení kódu chyby do HL
0210            LD   (HL), A            ;ulož kód chyby do ERR_NR
0211            PUSH HL                 ;ulož si adresu ERR_NR
```

```
0212                LD   HL, (#5C5D) CH_ADD  ;do HL adresa znaku pro dekódování
```

SYNTAX1

Toto je vstupní bod pro dekódování příkazu MDOSu a skok na podprogramy, které vykonávají daný příkaz MDOSu.

```
0215 SYNTAX1        POP  BC                  ;vyzvedni návratovou adresu do ZX ROM do BC
```

Pokud probíhá snap, nekontroluje se syntaxe, ale vrací se zpět přes návrat ze snapu.

```
0216                LD   A, (#3EEE) SNAPINF  ;vyzvedni informace, jestli je snap
0219                AND  A                   ;je snap?
021A                JP   NZ, #034A, SNPRET   ;ano → skoč na návrat do ZX ROM přes snap
021D                LD   A, (#3E60) DEBUG    ;byl při resetu trojhmat?
0220                AND  A
0221                JR   Z, #0266, COMMAND   ;ne → přeskoč výpis ladících tisků
```

Zase zapomenutá rutina ladících tisků. Je vytištěna návratová adresa do ZX ROM při volání RST #08, adresa tvaru příkazu v tabulce syntaxe a výška zásobníku při volání RST #08.

```
0223                PUSH BC                  ;ulož si návratovou adresu
0224                LD   HL, (#5C65) STKEND  ;vezmi adresu vrcholu zásobníku kalkulátoru do HL
0227                LD   DE, #000A           ;posuň o 10 bytů
022A                ADD  HL, DE              ;vrchol zásobníku kalkulátoru
022B                LD   (#5C65), HL STKEND  ;ulož nový vrchol
022E                LD   A, #02              ;kanál 2 (horní část obrazovky)
0230                RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0231                DW   #1601               ;podprogram CHAN-OPEN otevření kanálu
0233                POP  BC                  ;vyzvedni návratovou adresu
0234                PUSH BC                  ;a znovu ji ulož
0235                CALL #0FA6, BCPRT        ;piš obsah BC na obrazovku
0238                LD   A, „ “              ;mezera do A
023A                RST  #10                 ;vytiskni mezeru
023B                LD   BC, (#5C74) T_ADDR  ;adresa další položky v tabulce syntaxe do BC
023F                CALL #0FA6, BCPRT        ;piš obsah BC na obrazovku
0242                LD   A, „ “              ;mezera
0244                RST  #10                 ;vytiskni mezeru
0245                LD   HL, #0000           ;vezmi hodnotu SP do HL
0248                ADD  HL, SP
0249                LD   DE, (#5C3D) ERR_SP  ;adresa položky na zásobníku při chybě do DE
024D                DEC  DE                  ;sniž o dvě
024E                DEC  DE
024F                EX   DE, HL              ;dej do HL
0250                AND  A                   ;nuluj CY před odčítáním
0251                SBC  HL, DE              ;odečti, v HL je teď výška zásobníku i pro daný příkaz je
                                             ;to vlastně počet návratových adres při kontrole syntaxe
                                             ;daného příkazu
0253                LD   B, H                ;dej ji do BC
0254                LD   C, L
0255                CALL #0FA6, BCPRT        ;piš obsah BC na obrazovku
0258                LD   A, #0D              ;nový řádek
025A                RST  #10                 ;posuň se na nový řádek
025B                LD   HL, (#5C65) STKEND  ;vezmi adresu vrcholu zásobníku kalkulátoru
025E                LD   BC, #FFF6           ;do BC dej –10
```

```
0261            ADD  HL, BC                ;sniž na původní hodnotu
0262            LD   (#5C65), HL STKEND    ;ulož nový vrchol zásobníku kalkulátoru
0265            POP  BC                    ;vyzvedni návratovou adresu
```

Nyní určíme podle návratové adresy, adresy v tabulce syntaxe a výšky zásobníku při RST #08, kde došlo k chybě a jaký příkaz má být vykonán. Postupně budeme procházet tabulku, ve které jsou všechny návratové adresy, kam by se vrátilo řízení po RST #08 při chybě v syntaktické analýze daného příkazu.

```
0266 COMMAND    LD   IX, #05FF SYNTAB      ;adresa tabulky chybových adres do IX
026A SETCOMM    LD   L, (IX+#00)           ;vezmi adresu příkazu v tabulce syntaxe z tabulky do
026D            LD   H, (IX+#01)           ;HL
0270            LD   A, H                  ;je konec tabulky?
0271            OR   L
0272            JR   Z, #02A3, NOCOM       ;ano → skoč na výpis chyby
0274            LD   DE, (#5C74) T_ADDR    ;adresa položky v tabulce syntaxe do DE, kde došlo
                                           ;k nalezení chyby
0278            SBC  HL, DE                ;porovnáme, jestli jsou shodné
027A            JR   NZ, #029C, NEXTCOM    ;ne → skoč na posun na další položku v tabulce
027C            LD   L, (IX+#02)           ;vyzvedni návratovou adresu z tabulky
027F            LD   H, (IX+#03)
0282            SBC  HL, BC                ;porovnáme, jestli jsou stejné návratové adresy
0284            JR   NZ, #029C, NEXTCOM    ;ne → skoč na posun na další položku v tabulce
0286            LD   HL, #0000             ;vezmi hodnotu SP do HL
0289            ADD  HL, SP
028A            LD   DE, (#5C3D) ERR_SP    ;adresa položky na zásobníku při chybě do DE
028E            EX   DE, HL                ;dej ji do HL
028F            AND  A                     ;nuluj CY a odečti –
0290            SBC  HL, DE                ;dostaneme počet vnoření, než bylo zavoláno RST #08
0292            LD   E, (IX+#04)           ;vyzvedni výšku zásobníku z tabulky do DE
0295            LD   D, (IX+#05)
0298            SBC  HL, DE                ;porovnej je
029A            JR   Z, #02CB, DOCOM       ;výška zásobníku stejná → skoč na provedení příkazu
```

Posuneme se na další položku v tabulce.

```
029C NEXTCOM    LD   DE, #0008             ;každý příkaz zabírá 8 bytů v tabulce
029F            ADD  IX, DE                ;posuň se na další příkaz
02A1            JR   #026A, SETCOMM        ;pokračuj v hledání příkazu v tabulce
```

Sem se jde, pokud nebyl daný příkaz nalezen. Musí se vytisknout chybové hlášení a řízení se vrací zpět do ZX ROM. Nejdříve se provedou základní operace v ZX ROM, potom se řízení vrátí do ROM D40, kde se tiskne požadované hlášení MDOSu a řízení se vrátí zpět do ZX ROM. Pokud se jedná o chybu ZX ROM, vrací se řízení zpět do ZX ROM o něco dříve pro tisk chyby ZX ROM.

```
02A3 NOCOM      PUSH BC                    ;ulož si adresu uložení chyby (při chybě syntaxe to je
                                           ;návratová adresa do ZX ROM, při chybě v MDOSu to
                                           ;je ERR_NR)
02A4            LD   HL, #000B             ;vracet se budeme na adresu #000B, abychom se
02A7            PUSH HL                    ;nezacyklili a uložíme ji na zásobník
```

Nyní otestujeme, jestli se při chybě vrací řízení přímo do BASICu na adresu #1303 nebo byla tato návratová adresa přepsána (obsah adresy adresované proměnnou ERR_SP). Pokud je adresa #1303 a je chyba MDOSu, tiskne se hlášení na obrazovku a řízení se vrací zpět do BASICu. Pokud byla tato adresa přepsána a došlo k chybě MDOSu

nebo ZX ROM, vrací se řízení na tuto adresu bez toho, že by se tisklo jakékoliv hlášení. Využívá se pro ON ERROR GOTO. Je třeba dát pozor, že nedošlo k nulování některých důležitých systémových proměnných MDOSu, které mohou způsobit dost závažné problémy (dokonce i chybný zápis na disketu).

```
02A8        LD   HL, (#5C3D) ERR_SP   ;adresa položky na zásobníku při chybě do HL
02AB        LD   E, (HL)              ;do DE adresa, kam se skáče při chybovém hlášení
02AC        INC  HL
02AD        LD   D, (HL)
02AE        EX   DE, HL               ;dej ji do HL
```

Nejdříve otestujeme, jestli někdo nepřepsal návratovou adresu do BASICu, protože používá ošetření chybových hlášení.

```
02AF        LD   BC, #1303            ;do BC adresu MAIN-4 v ZX ROM
02B2        AND  A                    ;nuluj CY
02B3        SBC  HL, BC               ;jsou adresy shodné?
02B5        JR   NZ, #02C2, ERROR     ;ne → skoč na návrat do ZX ROM bez tisku ch. hlášení
02B7        EX   DE, HL               ;do HL dej adresu položky na zásobníku při chybě
```

Není přepsána návratová adresa do BASICu, bude se tedy tisknout chybové hlášení na obrazovku.

```
02B8        LD   (HL), #00            ;nuluj obsah adresy
02BA        DEC  HL
02BB        LD   (HL), #00            ;po obsloužení chyby v ZX ROM se řízení vrací na
                                      ;adresu #0000 pro obsloužení chyby v MDOSu
02BD        LD   HL, #3EF7 SYSFLAG    ;do HL adresa kódu činnosti
02C0        LD   (HL), #45            ;ulož kód činnosti „výpis chybového hlášení“
02C2 ERROR  CALL #2536, DSKSTP        ;zastav mechaniky
02C5        LD   HL, (#5C5D) CH_ADD   ;do HL adresa znaku pro dekódování – nutné pro návrat
                                      ;do ZX ROM
02C8        JP   #1700, STANDROM      ;skoč na přestránkování zpět do ZX ROM
```

Zde se pokračuje po nalezení příkazu v tabulce. Je vyzvednuta adresa rutiny, která daný příkaz provádí a je proveden skok na daný podprogram.

```
02CB DOCOM  LD   SP, (#5C3D) ERR_SP   ;adresa položky na zásobníku při chybě do SP –
                                      ;vyčistíme zásobník
02CF        LD   HL, #1B76 STMT-RET   ;do HL návratová adresa do ZX ROM při ukončení
                                      ;příkazu MDOSu
02D2        PUSH HL                   ;ulož ji na zásobník
02D3        LD   HL, #02E1 RETURN     ;návratová adresa z rutiny vykonávající příkaz MDOSu
02D6        PUSH HL                   ;ulož ji také na zásobník
02D7        LD   (#5C74), HL T_ADDR   ;a změň adresu další položky v tabulce syntaxe
02DA        LD   L, (IX+#06)          ;vyzvedni adresu podprogramu vykonávajícího
02DD        LD   H, (IX+#07)          ;příkaz do HL
02E0        JP   (HL)                 ;skoč na daný podprogram
```

RETURN

Tento podprogram zastaví mechaniky a přestránkuje do ZX ROM. Vrací se zde řízení po vykonání příkazu MDOSu.

```
02E1 RETURN CALL #2536, DSKSTP        ;vypni mechaniky
02E4        JP   #1700, STANDROM      ;přestránkuj zpět do ZX ROM
```

SNAPR

Tento program slouží k uložení SNAP souboru na disk. Při stisku tlačítka SNAP je hardwarově vnucena na adresu #0066 instrukce RST #00. Při procházení tabulky návratových adres se zjistí, že došlo k SNAPu (návratová adresa je #67). Snapování je uložení obsahu paměti na disk do souboru s příponou „S“. Velikost souboru je #C080 (49280) bytů. Počáteční adresa je #3F80 (16256). Obsah paměti se nahrává i s obsahy všech registrů. Velice jednoduše si můžete vyrobit svůj vlastní snap programově:

```
LD    HL, návratová adresa, kam se program vrátí po provedení snapu
PUSH HL
LD    HL, #0067
PUSH HL
JP    #0000
```

Tento krátký prográmek vám vytvoří na disketě snap soubor. Ale nyní již snap v ROM D40.

```
02E7 SNAPR    LD    (#3FFE), SP SAVESP    ;ulož hodnotu SP do SRAM
02EB          LD    SP, #3FFE SAVESP      ;nastav SP do oblasti, kam se postupně uloží obsahy
02EE          PUSH AF                     ;všech registrů
02EF          PUSH BC
02F0          PUSH DE
02F1          PUSH HL
02F2          EXX
02F3          EX    AF, AF'               ;včetně druhé banky
02F4          PUSH AF
02F5          PUSH BC
02F6          PUSH DE
02F7          PUSH HL
02F8          PUSH IX
02FA          PUSH IY
02FC          LD    BC, (#3EEC) IREG2     ;vyzvedni vektor přerušení a stav přerušení do BC
0300          PUSH BC                     ;a ulož ho na zásobník
0301          IM    1                     ;nastav režim přerušení IM 1
0303          LD    A, #FF                ;ulož informaci o tom, že se provadí SNAP
0305          LD    (#3EEE), A SNAPINF    ;na SNAPINF tedy uložme nenulovou hodnotu
```

Nastavíme jméno disku, na který se bude ukládat snap a jméno souboru.

```
0308          LD    HL, #3EAA ACDRIVE     ;do HL adresa jména drivu, který je aktuální
030B          LD    DE, #3E80 DNZONE1     ;do DE adresa 1. jména disku pro I/O
030E          LD    BC, #000A             ;má délku 10 znaků
0311          LDIR                        ;přenes jako jméno aktuálního disku
0313          LD    HL, #03A4 SNAPNM      ;adresa textu „SNAPSHOT00S“ v ROM D40 do HL
0316          LD    DE, #3E8A FNZONE1     ;do DE adresa 1. jména souboru pro I/O
0319          LD    BC, #000B             ;délka je 11 bytů
031C          LDIR                        ;přenes jako jméno pracovního souboru
031E          LD    A, (#3E61) SNPCOUNT   ;vyzvedni počítadlo snapshotů do A
0321          INC   A                     ;zvyš o 1
0322          LD    (#3E61), A SNPCOUNT   ;ulož novou hodnotu do počítadla
```

Upravíme číslo snapu na ASCII vyjádření pro uložení do jména souboru

```
0325          DEC   A                     ;uprav zpět na aktualní číslo
0326          LD    B, #00                ;do B dáme počítadlo desítek
0328 DECLOP   SUB   #0A                   ;odečti od čísla snapu 10
032A          JR    C, #032F, DECOK       ;je zbytek < 10? ano → skoč
```

```
032C            INC   B                    ;zvyš desítky
032D            JR    #0328, DECLOP        ;pokračuj v převodu
032F DECOK      ADD   A, #3A               ;přičti zpět 10 a uprav na ASCII
0331            LD    (#3E93), A SNONMB2   ;ulož do názvu souboru jako nižší číslici
0334            LD    A, B                 ;do A dej desítky
0335            ADD   A, #30               ;a uprav je na ASCII
0337            LD    (#3E92), A SNONMB1   ;ulož do názvu souboru jako vyšší číslici
033A            EI                         ;povol přerušení
033B            LD    HL, #3F80            ;počáteční adresa dat ukládaných na disk do HL
033E            LD    (#3E74), HL STARTADR ;a do hlavičky
```

Provedeme zápis souboru.

```
0341            LD    DE, #C080            ;délka ukládaných dat je 49280 bytů
0344            CALL  #1A00, SAVRUN        ;ulož soubor na disk
0347            CALL  #2536, DSKSTP        ;vypni mechaniky
```

Provedeme návrat ze snapu. Není celkem v pořádku, protože pokud jste používali přerušení IM 1 a změnili jste hodnotu registru I (tzn. není #3F), je špatně nastaveno přerušení (místo IM 1 je nastaveno IM 2) a počítač se ve většině případů zhroutí. Další nedostatek je, že se neuchovává obsah registru R, takže pokud je v programu test na jeho obsah, při nahrávání snapu do paměti Vám nebude program fungovat. Celkové řešení návratu ze snapu také není zrovna to pravé, protože k vůli jedné instrukce mít kus programu dvakrát je dost velký přepych, když je k dispozici pamět SRAM. Jistě by se to dalo řešit elegantněji.

```
034A SNPRET     CALL  #2536, DSKSTP        ;vypni mechaniky
034D            DI                         ;zakaž přerušení
034E            LD    SP, #3FE8 SVREG      ;nastav zásobník pro obnovení registrů
0351            XOR   A                    ;informace o ukončení snapu
0352            LD    (#3EEE), A SNAPINF   ;ulož na SNAPINF
0355            POP   AF                   ;čti stav a mód přerušení
0356            JP    PE, #0376, SNPRT1    ;bylo povolené přerušení? ano → skoč
```

Při snapu bylo zakázané přerušení.

```
0359            LD    I, A                 ;obnov I registr
035B            CP    #3F                  ;byl nastaven na #3F (přerušení IM 1)?
035D            JR    Z, #0361, NOIM2      ;ano → přeskoč změnu režimu přerušení
035F            IM    2                    ;nastav režim IM 2
0361 NOIM2      POP   IY                   ;obnov registry procesoru z druhé banky
0363            POP   IX
0365            POP   HL
0366            POP   DE
0367            POP   BC
0368            POP   AF
0369            EX    AF, AF'              ;a teď z první banky
036A            EXX
036B            POP   HL
036C            POP   DE
036D            POP   BC
036E            POP   AF
036F            LD    SP, (#3FFE) SAVESP   ;obnov nastavení zásobníku
0373            JP    #1700, STANDROM      ;skoč na přestránkování zpět do ZX ROM a návrat do
                                           ;přerušeného programu
```

Při snapu bylo povolené přerušení.

```
0376 SNPTR1     LD   I, A                  ;obnov I registr
0378            CP   #3F                   ;byl nastaven na #3F (IM 1)?
037A            JR   Z, #037E, NOIM21      ;ano → přeskoč změnu režimu přerušení
037C            IM   2                     ;nastav mód IM 2
037E NOIM21     POP  IY                    ;obnov registry procesoru z druhé banky
0380            POP  IX
0382            POP  HL
0383            POP  DE
0384            POP  BC
0385            POP  AF
0386            EX   AF, AF'               ;a teď z první banky
0387            EXX
0388            POP  HL
0389            POP  DE
038A            POP  BC
038B            POP  AF
038C            LD   SP, (#3FFE) SAVESP    ;obnov nastavení zásobníku
0390            EI                         ;povol přerušení
0391            JP   #1700, STANDROM       ;skoč na přestránkování zpět do ZX ROM a návrat do
                                           ;přerušeného programu
```

SNPLOA

Tento program nám natáhne snap do paměti. Jméno snapu je ve FNZONE1.

IN: jméno snapu ve FNZONE1
OUT: natažení a spuštění snapu

```
0394 SNPLOA     LD   SP, #3F80             ;nastav zasobnik pod první byte snapu
0397            LD   IX, #3F80             ;do IX počáteční adresa dat
039B            LD   DE, #C080             ;délka dat je 49280 bytu
039E            CALL #19AE, LOABLK         ;načti soubor
03A1            JP   #034A, SNPRET         ;skoč na spuštění snapu přes návrat ze snapu
```

SNAPNM

Text „SNAPSHOT00S“.

```
03A4 SNAPNM     53 4E 41 50 53 48 4F 54 30 30 53          ;SNAPSHOT00S
```

SYSMSG

Tabulka textů chybových hlášení pro MDOS. Začíná invertovaným znakem (#AA) a každé hlášení končí invertovaným znakem. Hlášení 0-R jsou hlášení ZX ROM, byty #AA slouží na doplnění pro výpočet hlášení MDOSu.

```
03AF SYSMSG     DB   #AA                   ;invertovaný znak
03B0            AA AA AA AA AA AA AA AA
03B8            AA AA AA AA AA AA AA AA
03C0            AA AA AA AA AA AA AA AA
03C8            AA AA AA AA

03CC REP-S      46 69 6C 65 20 6E 6F 74 20 66 6F 75 6E E4 ;File not found
03DA REP-T      46 69 6C 65 20 65 78 69 73 74 F3          ;File exists
03E5 REP-U      44 69 73 6B 20 66 75 6C EC                ;Disk full
```

```
03EE REP-V      44 69 72 65 63 74 6F 72 79 20 66 75 6C EC   ;Directory full
03FC REP-W      41 64 76 61 6E 63 65 64 20 66 65 61 74 75 72 E5   ;Advanced feature
040C REP-X      42 61 64 20 64 65 76 69 63 65 20 74 79 70 E5   ;Bad device type
041B REP-Y      44 65 76 69 63 65 20 69 64 65 6E 74 20 6D 69 73   ;Device ident mis
                73 69 6E E7                                 ;sing
042F REP-Z      44 65 76 69 63 65 20 75 6E 61 76 61 69 6C 61 62   ;Device unavailab
                6C E5                                       ;le
```

Hodnoty #A0 vyplňují mezeru mezi „Z“ a „a“ v kódu ASCII.

```
0441            A0 A0 A0 A0 A0 A0
0447 REP-a      44 65 76 69 63 65 20 49 2F 4F 20 65 72 72 6F F2   ;Device I/O error
0457 REP-b      42 61 64 20 76 6F 6C 75 6D 65 20 6E 61 6D E5   ;Bad volume name
0466 REP-c      42 61 64 20 66 69 6C 65 20 74 79 70  E5      ;Bad file type
0473 REP-d      56 6F 6C 75 6D 65 20 6E 6F 74 20 66 6F 75 6E E4   ;Volume not found
0483 REP-e      46 69 6C 65 20 69 73 20 72 65 61 64 20 70 72 6F 74   ;File is read prot
                65 63 74 65 E4                              ;ected
0499 REP-f      46 69 6C 65 20 69 73 20 77 72 69 74 65 20 70 72 6F   ;File is write pro
                74 65 63 74 65 E4                           ;tected
04B0 REP-g      46 69 6C 65 20 69 73 20 6E 6F 74 20 65 78 65 63 75   ;File is not execu
                74 61 62 6C E5                              ;table
04B6 REP-h      46 69 6C 65 20 69 73 20 64 65 6C 65 74 65 20 70 72   ;File is delete pr
                6F 74 65 63 74 65 E4                        ;otected
04DE REP-i      42 61 64 20 72 65 63 6F 72 64 20 6E 75 6D 62 65 F2   ;Bad record number
04EF REP-j      49 6D 70 6F 73 73 69 62 6C 65 20 74 6F 20 52 45 4E   ;Impossible to REN
                41 4D C5                                    ;AME
0503 REP-k      49 6D 70 6F 73 73 69 62 6C 65 20 74 6F 20 43 4F 50   ;Impossible to COP
                D9                                          ;Y
0515 REP-l      43 6F 72 72 75 70 74 65 64 20 46 41 54 20 73 74 72   ;Corrupted FAT stru
                75 63 74 75 72 E5                           ;cture
052C REP-m      53 74 72 65 61 6D 20 61 6C 72 65 61 64 79 20 6F 70   ;Stream already op
                65 EE                                       ;en
053F REP-n      44 72 69 76 65 20 69 73 20 6E 6F 74 20 72 65 61 64   ;Drive is not read
                F9                                          ;y
0551 REP-o      53 65 65 6B 20 65 72 72 6F F2               ;Seek error
055B REP-p      53 65 63 74 6F 72 20 6E 6F 74 20 66 6F 75 6E E4   ;Sector not found
056B REP-q      43 52 43 20 65 72 72 6F F2                  ;CRC error
0574 REP-r      44 69 73 6B 20 69 73 20 77 72 69 74 65 20 70 72 6F   ;Disk is write pro
                74 65 63 74 65 E4                           ;tected
058B REP-s      49 6E 74 65 72 6E 61 6C 20 65 72 72 6F F2   ;Internal error
0599 TEXT1      50 6C 65 61 73 65 20 69 6E 73 65 72 74 20 76 6F 6C   ;Please insert vol
                75 6D 65 20 23 8D                           ;ume /zde se vloží jméno diskety/
05B0 TEXT2      45 72 61 73 65 20 61 6C 6C 20 66 69 6C 65 73 20 BF   ;Erase all files?
05B1 TEXT3      52 65 77 72 69 74 65 20 6F 6C 64 20 66 69 6C 65 20   ;Rewrite old file?
                BF
05D3 TEXT4      41 6C 6C 20 64 61 74 61 20 77 69 6C 6C 20 62 65 20   ;All data will be
                64 69 73 63 61 72 64 65 64 20 21 20 20 A0   ;discarded!
05F2 REP-x      46 69 6C 65 20 74 6F 6F 20 6C 6F 6E E7      ;File too long
```

SYNTAB

Nyní následuje tabulka, podle které se rozlišuje, jaký příkaz má být vykonán. Má následující strukturu:

DW adresa příkazu v tabulce syntaxe v ZX ROM
DW návratová adresa, odkud byl volán RST #08

```
                DW   výška zásobníku při volání RST #08
                DW   adresa programu vykonávající příkaz

   Příkaz CAT
05FF            DW   #1B15, #1726          ;REPORT O „Invalid stream“
0603            DW   #0002, #11DF
   Příkaz CAT
0607            DW   #1B15, #1C8B          ;REPORT C „Nonsense in BASIC“
060B            DW   #0000, #11DF
   Příkaz ERASE
060F            DW   #1B12, #1726          ;REPORT O „Invalid stream“
0613            DW   #0002, #12D3
   Příkaz POKE
0617            DW   #1AB2, #1C8B          ;REPORT C „Nonsense in BASIC“
061B            DW   #000A, #06C1
   Příkaz MOVE
061F            DW   #1B0E, #1726          ;REPORT O „Invalid stream“
0623            DW   #0002, #1A54
   Příkaz MOVE
0627            DW   #1B0C, #1C8B          ;REPORT C „Nonsense in BASIC“
062B            DW   #0004, #1306
   Příkaz FORMAT
062F            DW   #1B08, #1726          ;REPORT O „Invalid stream“
0633            DW   #0002, #1320
   Příkaz SAVE
0637            DW   #1A00, #1C8B          ;REPORT C „Nonsense in BASIC“
063B            DW   #0008, #1701
   Příkaz LOAD
063F            DW   #1A01, #1C8B          ;REPORT C „Nonsense in BASIC“
0643            DW   #0008, #1704
   Příkaz MERGE
0647            DW   #1A03, #1C8B          ;REPORT C „Nonsense in BASIC“
064B            DW   #0008, #1707
   Příkaz LET
064F            DW   #1A7B, #1C8B          ;REPORT C „Nonsense in BASIC“
0653            DW   #0006, #06F0
   Příkaz LIST
0657            DW   #1AAF, #1C8B          ;REPORT C „Nonsense in BASIC“
065B            DW   #0008, #086F
   Příkaz LLIST
065F            DW   #1ADD, #1C8B          ;REPORT C „Nonsense in BASIC“
0663            DW   #0008, #086F
   Příkaz READ
0667            DW   #1ACA, #1C8B          ;REPORT C „Nonsense in BASIC“
066B            DW   #0006, #0A50
   Příkaz RESTORE
066F            DW   #1AD0, #1C8B          ;REPORT C „Nonsense in BASIC“
0673            DW   #000A, #0A4B
   Příkaz OPEN #
0677            DW   #1B00, #1766          ;REPORT F „Invalid file name“
067B            DW   #0006, #0AC9
```

```
   Příkaz OPEN #
067F               DW   #1B00, #1766          ;REPORT F „Invalid file name“
0683               DW   #0008, #0AC9
   Příkaz OPEN #
0687               DW   #1B00, #1C8B          ;REPORT C „Nonsense in BASIC“
068B               DW   #0000, #0BDB
   Příkaz OPEN #
068F               DW   #1AFF, #1C8B          ;REPORT C „Nonsense in BASIC“
0693               DW   #0008, #0B7B
   Příkaz CLOSE #
0697               DW   #1B04, #1766          ;REPORT F „Invalid file name“
069B               DW   #0006, #0C14
   Příkaz CLOSE #
069F               DW   #1B03, #1C8B          ;REPORT C „Nonsense in BASIC“
06A3               DW   #0008, #0C2D
   Příkaz RUN
06A7               DW   #1AAC, #1BB1          ;REPORT 0 „OK“
06AB               DW   #0000, #1139
   Příkaz PRINT
06AF               DW   #1A9D, #1C8B          ;REPORT C „Nonsense in BASIC“
06B3               DW   #000A, #07E5
   Příkaz LPRINT
06B7               DW   #1ADA, #1C8B          ;REPORT C „Nonsense in BASIC“
06BB               DW   #000A, #07E5
06BF               DW   #0000                 ;konec tabulky
```

Nyní jsou podprogramy provádějící příkazy MDOSu.

POKE

Příkaz pro zápis dat do SRAM. Může se zapisovat do 512-ti bytů od adresy #3E00.

Syntaxe: POKE # adresa, byte

adresa je v rozsahu 0–511, je to relativní hodnota

byte je v rozsahu 0–255 a je to hodnota, která se na danou adresu zapíše.

Slouží k zápisu hodnoty do paměti SRAM. Příkaz POKE používejte jen v případech, kdy změněná data nezpůsobí zhroucení MDOSu. Následně může dojít i k narušení dat na disketě.

Nejdříve zkontrolujeme znak „#“, jestli se nejedná o chybu v klasickém příkazu POKE BASICu ZX ROM.

```
06C1 POKE          RST  #18                   ;načti aktuální znak
06C2               CP   „#“                   ;je to znak „#“?
06C4               JR   Z, #06CB, HASHOK      ;ano → v pořádku, skoč na pokračování
```

Není to příkaz POKE #, tiskne se hlášení a provede se návrat do ZX ROM.

```
06C6 REPORTC       LD   A, #0B                ;REPORT C „Nonsense in BASIC“
06C8               JP   #0204, ERRR           ;piš hlášení a skoč do ZX ROM
```

Budeme analyzovat parametry příkazu POKE #.

```
06CB HASHOKRST #20                            ;načti další znak
06CC               RST  #28                   ;volej podprogram pro volání rutiny ZX ROM
06CD               DW   #1C82                 ;podprogram EXPT-NUM1 ohodnocení číselného
                                              ;výrazu – v zásobníku kalkulátoru je teď uložena adresa,
                                              ;kam se bude ukládat
06CF               RST  #18                   ;načti aktuální znak
```

```
06D0            CP    „ , “                ;je to čárka?
06D2            JR    NZ, #06C6, REPORTC   ;ne → skoč na REPORT C „Nonsense in BASIC“
06D4            RST   #20                  ;načti další znak
06D5            RST   #28                  ;volej podprogram pro volání rutiny ZX ROM
06D6            DW    #1C82                ;podprogram EXPT-NUM1 ohodnocení číselného
                                           ;výrazu – v zásobníku kalkulátoru je teď uložena
                                           ;hodnota, která se bude ukládat do SRAM
06D8            CALL #1057, ISSYNCONTR     ;otestuj, jestli není kontrola syntaxe
```

Pokud se neprovádí kontrola syntaxe, provedeme příkaz. Nejdřive zkontrolujeme adresu, musí být menší než 512. Můžeme totiž zapisovat jen do 512-ti bytů.

```
06DB            RST   #28                  ;volej podprogram pro volání rutiny ZX ROM
06DC            DW    #1E85                ;podprogram TWO-PARAM vyzvednutí dvou hodnot
                                           ;ze zásobníku kalkulátoru, BC=adresa, A=data
06DE            PUSH  AF                   ;schovej si data
06DF            LD    A, B                 ;dej B do A
06E0            CP    #02                  ;je adresa < 512?
06E2            JR    C, #06E9, NOOUT      ;ano → v pořádku, skoč
```

Adresa je mimo povolený rozsah.

```
06E4            LD    A, #0A               ;REPORT B „Integer out of range“
06E6            JP    #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Vypočteme adresu a uložíme data do paměti.

```
06E9 NOOUT      POP   AF                   ;obnov data
06EA            LD    HL, #3E00 DRPARZN    ;počáteční adresa pro ukládaní dat je #3E00 (15872)
06ED            ADD   HL, BC               ;přičti relativní posun (adresu) k HL
06EE            LD    (HL), A              ;ulož data do SRAM
06EF            RET                        ;vrať se přes RETURN do ZX ROM
```

LET FN

Příkaz na změnu jména souboru. Změní původní jméno souboru na nové. Nemění příponu. Pokud existuje na disketě soubor, který má jméno stejné jako je nové jméno souboru, příkaz se neprovede a je hlášena chyba.

Syntaxe: LET FN ("StaréJménoSouboru")="NovéJménoSouboru"

LET ATTR

Příkaz na změnu atributů u vybrané skupiny souborů. Změní u všech souborů, jejichž jméno a přípona odpovídá vložené masce, atributy.

Syntaxe: LET ATTR("MaskaSouboru")="Atributy"

Oba tyto příkazy mají stejný vstupní bod do programu. Část programu pro kontrolu syntaxe mají společnou. Podle FN/ATTR se potom určí, co se bude provádět.

```
06F0 LETFNATTR  RST   #18                  ;načti aktuální znak
06F1            CP    #AB                  ;je požadováno ATTR?
06F3            LD    HL, #0723 LETATTR    ;do HL adresa obslužné rutiny pro LET ATTR
06F6            JR    Z, #0702, SELLET     ;ano → skoč
06F8            CP    #A8                  ;je požadováno FN?
06FA            LD    HL, #0778 LETFN      ;do HL adresa obslužné rutiny pro LET FN
06FD            JR    Z, #0702, SELLET     ;ano → skoč
```

Není to žádný z nich.

```
06FF GOREPC     JP    #06C6, REPORTC       ;skoč na REPORT C „Nonese in BASIC“
```

Teď je společná část programu pro vyzvednutí parametrů obou příkazů.

```
0702 SELLLET    LD    (#3E78), HL VALSYX    ;ulož si adresu rutiny obsluhy příkazu
0705            RST   #20                   ;načti další znak
0706            CP    „(“                   ;je to levá závorka?
0708            JR    NZ, #06FF, GOREPC     ;ne → skoč na REPORT C „Nonsense in BASIC“
070A            RST   #20                   ;načti další znak
070B            RST   #28                   ;volej podprogram pro volání rutiny ZX ROM
070C            DW    #1C8C                 ;podprogram EXPT-EXP vyhodnocení řetězce
                                            ;je vyzvednuto 1. jméno souboru
070E            RST   #18                   ;načti aktuální znak
070F            CP    „)“                   ;je to pravá závorka?
0711            JR    NZ, #06FF, GOREPC     ;ne → skoč na REPORT C „Nonsense in BASIC“
0713            RST   #20                   ;načti další znak
0714            CP    „=“                   ;je to rovnítko?
0716            JR    NZ, #06FF, GOREPC     ;ne → skoč na REPORT C „Nonsense in BASIC“
0718            RST   #20                   ;načti další znak
0719            RST   #28                   ;volej podprogram pro volání rutiny ZX ROM
071A            DW    #1C8C                 ;podprogram EXPT-EXP vyhodnocení řetězce
                                            ;je vyzvednuto buď jméno souboru nebo atributy
071C            CALL #1057, ISSYNCONTR      ;otestuj, jestli není kontrola syntaxe
```

Pokud se neprovádí kontrola syntaxe, provedem příkaz.

```
071F            LD    HL, (#3E78) VALSYX    ;adresa obslužné rutiny příkazu do HL
0722            JP    (HL)                  ;skoč na danou obslužnou rutinu
```

LET ATTR

Zde se již zajišťuje provedení příkazu LET ATTR. V zásobníku kalkulátoru jsou na vrcholu parametry uložení řetězce atributů a pod nimi jsou uloženy parametry řetězce jména souboru.

```
0723 LETATTR    RST   #28                   ;volej podprogram pro volání rutiny ZX ROM
0724            DW    #2BF1                 ;podprogram STK-FETCH vyzvednutí 5 bytů ze
                                            ;zásobníku, které tvoří parametry řetězce atributů
0726            LD    A, B                  ;dej do A horní byte délky atributů
0727            AND   A                     ;je řetězec atributů dlouhý?
0728            JR    NZ, #0743, REPORTA    ;ano → skoč na REPORT A „Invalid argument“
072A            LD    B, C                  ;do B dej nižší byte délky řetězce atributů
072B            LD    C, #00                ;na začátku nebude žádný atribut
072D            LD    A, B                  ;délka řetězce atributů do A
072E            AND   A                     ;je nulová?
072F            JR    Z, #074E, ATRNAME     ;ano → skoč na zpracování jména souboru s tím,
                                            ;že nejsou žádné atributy (v A je 0)
```

Protože byly zadány atributy, zjistíme, které byly vloženy a které z nich jsou definovány v tabulce atributů. Na začátku je v C nula, což znamená, že žádný atribut není vložen.

```
0731 ANALATR    LD    HL, #127B DEFATTR     ;do HL adresa tabulky atributů souboru v ROM D40
0734            LD    A, (DE)               ;vyzvedni první atribut z řetězce
0735            AND   #DF                   ;převeď na velká písmena
0737            INC   DE                    ;posuň ukazatel na další atribut v řetězci
0738            PUSH DE                     ;ulož ukazatel
0739            LD    E, #80                ;do E dáme %10000000
```

Prohlédneme tabulku definovaných atributů a rotujeme obsahem E, kde bude uložen v případě nalezení bitové umístění daného atributu.

```
073B RFINDATR   CP   (HL)                 ;je definován takový atribut v tabulce?
073C            INC  HL                   ;posuň se na další v tabulce
073D            JR   Z, #0748, SETATR     ;je definován → skoč na zařazení
073F            RRC  E                    ;posuň se na další bit
0741            JR   NC, #073B, RFINDATR  ;není poslední → opakuj pro další
```

Atribut není definován v tabulce.

```
0743 REPORTA    LD   A, #09               ;REPORT A „Invalid argument“
0745            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Když ho najdeme, přidáme ho do C jako bit v E.

```
0748 SETATR     LD   A, C                 ;do A dej prozatímní atributy
0749            OR   E                    ;přidej nalezený atribut
074A            LD   C, A                 ;uschovej prozatímní atributy do C
074B            POP  DE                   ;obnov ukazatel na řetězec
074C            DJNZ #0731, ANALATR       ;opakuj B-krát
```

Nyní provedeme analýzu řetězce v zásobníku. V A jsou uloženy nové atributy, které bude mít daný soubor.

```
074E ATRNAME    EX   AF, AF'              ;ulož si nové atributy do druhé banky
074F            CALL #0FCF, DIVSTRING     ;rozděl řetězec na zásobníku na jméno disku a souboru
0752            CALL #0F9E, TESTNM        ;otestuj, jestli bylo zadáno jméno souboru
0755            CALL #107C, ARRANGNM      ;uprav jméno souboru ve FNZONE1 na masku
```

Najdeme a rozběhneme mechaniku se jménem v DNZONE1.

```
0758            CALL #1043, SETWDNM       ;nastav jméno disku v DNZONE1 pro I/O
075B            CALL #1C8F, SETACT        ;roztoč mechaniku, která má stejné jméno jako
                                          ;v DNZONE1
075E            CALL #212B, FIRSTMASK     ;načti 1. položku adresáře vyhovující masce v FNZONE1
0761            JP   NZ, #1FB1, REPORTS   ;nenalezena → skoč na REPORT S „File not found“
0764 WATTR      EX   AF, AF'              ;vrať zpět atributy do A a ulož si číslo položky
0765            PUSH HL                   ;ulož si ukazatel na položku v buferu adresáře
0766            EX   (SP), IX             ;dej ukazatel do IX
0768            LD   (IX+#14), A          ;zapiš atribut na místo pro atributy v hlavičce
076B            EX   (SP), IX             ;vrať zpět ukazatel
076D            POP  HL                   ;do HL
076E            CALL #1E65, WSCADR        ;zapiš sektor adresáře v buferu
0771            EX   AF, AF'              ;ulož si znovu atribut a do A číslo položky, od které se
                                          ;bude pokračovat v hledání dalšího souboru
0772            CALL #212D, NEXTMASK      ;načti další položku adresáře vyhovující masce
                                          ;v FNZONE1
0775            RET  NZ                   ;nenalezena → vrať se přes RETURN do ZX ROM
0776            JR   #0764, WATTR         ;skoč na zápis atributu a hledání další položky
```

LET FN

Zde se zajišťuje provedení příkazu LET FN. Na vrcholu zásobníku kalkulátoru jsou parametry 2. jména souboru a pod nimi jsou parametry 1. jména souboru.

```
0778 LETFN      CALL #07C9, ANSTRING      ;analyzuj 2. řetězec a nastav jméno disku a souboru do
                                          ;FNZONE1 a DNZONE1
077B            CALL #1C8F, SETACT        ;roztoč mechaniku, která má stejné jméno jako
                                          ;v DNZONE1
```

Nejdříve zjistíme, jestli již na disketě neexistuje soubor se stejným jménem, jako je nové jméno souboru.

```
077E            CALL #212B, FIRSTMASK     ;načti 1. položku adresáře vyhovující masce v FNZONE1
0781            JR   NZ, #0788, NOEXIST   ;nenašel → skoč
```

Takové jméno souboru, jako je nové jméno souboru, už existuje.

```
0783            LD   A, #1C               ;REPORT T „File exists“
0785            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Schováme si jméno disku a nové jméno souboru.

```
0788 NOEXIST    LD   HL, #3E80 DNZONE1    ;do HL adresa 1. jména disku pro I/O
078B            LD   DE, #3E95 DNZONE2    ;do DE adresa 2. jména disku pro I/O
078E            LD   BC, #15              ;délka 21 bytů
0791            LDIR                      ;uschovej si nové jméno souboru a jméno disku
0793            CALL #07C9, ANSTRING      ;analyzuj 1. řetězec a nastav jméno disku a souboru do
                                          ;FNZONE1 a DNZONE1
```

Porovnáme přípony, protože musí zůstat stejné.

```
0796            LD   HL, #3E94 EXTE1      ;do HL adresa přípony starého jména souboru
0799            LD   A, (#3EA9) EXTE2     ;vyzvedni příponu nového jména souboru
079C            CP   (HL)                 ;porovnej je se starým
079D            JR   Z, #07A4, EXTISSOME  ;jsou stejné → skoč
```

Jsou rozdílné přípony.

```
079F REPORTj    LD   A, #32               ;REPORT j „Impossible to RENAME“
07A1            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Porovnáme, jestli jsou shodné jména disků. Musí být stejná, protože příkaz LET FN slouží pouze k přejmenování souboru, který se nachází na jedné disketě.

```
07A4 EXTISSOME  LD   HL, #3E80 DNZONE1    ;do HL adresa jména prvního disku
07A7            LD   DE, #3E95 DNZONE2    ;do DE adresa jména druhého disku
07AA            LD   BC, #000A            ;10 bytů
07AD            CALL #1F0E, VERIFY        ;teď je porovnej
07B0            JR   NZ, #079F, REPORTj   ;rozdílné → skoč na REPORT j „Impossible to
                                          ;RENAME“
07B2            CALL #212B, FIRSTMASK     ;načti 1. položku adresáře vyhovující masce v FNZONE1
07B5            JP   NZ, #1FB1, REPORTS   ;neexistuje → skoč na REPORT S „File not found“
07B8            INC  HL                   ;posuň se na jméno souboru v hlaviče
07B9            LD   DE, #3E9F FNZONE2    ;do DE adresa nového jména souboru
07BC            LD   BC, #000A            ;10 bytů
07BF            EX   DE, HL               ;přehoď ukazatele
07C0            LDIR                      ;přesuň nové jméno do hlavičky
07C2            CALL #1E65, WSCADR        ;zapiš sektor adresáře v buferu
07C5            CALL #217B, ERAVAR        ;vymaž pomocné proměnné MDOSu
07C8            RET                       ;vrať se přes RETURN do ZX ROM
```

ANSTRING

Rozdělí a analyzuje řetězec na zásobníku, nastaví disk se stejným jménem a upraví jméno souboru na masku. Potom provede kontrolu přípony a jestli nebyla vložena, hlásí chybu. Chyba se také hlásí, pokud bylo použito wildchars.

IN: parametry řetězce na vrcholu zásobníku kalkulátoru
OUT: nastavení disku a masky souboru do DNZONE1 a FNZONE1 a kontrola přípony

```
07C9 ANSTRING   CALL #0FCF, DIVSTRING   ;rozděl řetězec na zásobníku na jméno disku a souboru
07CC            CALL #1043, SETWDNM     ;nastav jméno disku v DNZONE1 pro I/O
07CF            CALL #10E2, ANALWDNM    ;analyzuj jméno disku v DNZONE1
07D2            INC  A                  ;bylo chybně vloženo jméno disku?
07D3            JP   Z, #2337, REPORTX  ;ano → skoč na REPORT X „Bad device type“
07D6            CALL #107C, ARRANGNM    ;uprav jméno souboru v FNZONE1 na masku
07D9            JP   C, #1019, REPORTF  ;bylo použito wildchars → skoč na REPORT F „Invalid
                                        ;file name“
07DC            LD   A, (#3E94) EXTE1   ;vyzvedni příponu masky
07DF            CP   „?“                ;je to „?“?
07E1            JP   Z, #1019, REPORTF  ;ano → skoč na REPORT F „Invalid file name“, protože
                                        ;přípona musí být vložena.
07E4            RET                     ;vrať se
```

PRINT, LPRINT

Příkazy pro výpis obsahu sekvenčního souboru na obrazovku nebo na tiskárnu.
Syntaxe: PRINT *"JménoSouboru"
LPRINT *"JménoSouboru"
Umožňuje vypisovat pouze soubory s příponou „Q“. Vstupní bod obou příkazů je stejný, liší se otevřený kanál.

```
07E5 L-PRINT    RST  #18                ;načti aktuální znak
07E6            CP   „*“                ;je to hvězdička?
07E8            JP   NZ, #06C6, REPORTC ;ne → skoč na REPORT C „Nonsense in BASIC“
```

Vyzvedneme a analyzujeme parametr příkazu.

```
07EB            RST  #20                ;načti další znak
07EC            RST  #28                ;volej podprogram pro volání rutiny ZX ROM
07ED            DW   #1C8C              ;podprogram EXPT-EXP vyhodnocení řetězce –
                                        ;vyhodnotí vložené jméno disku a souboru
07EF            CALL #1057, ISSYNCONTR  ;otestuj, jestli není kontrola syntaxe
```

Pokud se neprovádí kontrola syntaxe, provedeme příkaz. Jako první nastavíme jméno disku a souboru.

```
07F2            CALL #0FCF, DIVSTRING   ;rozděl řetězec na zásobníku na jméno disku a souboru
07F5            CALL #1043, SETWDNM     ;nastav jméno disku v DNZONE1 pro I/O
07F8            CALL #10E2, ANALWDNM    ;analyzuj jméno disku v DNZONE1
07FB            INC  A                  ;bylo chybně vloženo jméno disku?
07FC            JP   Z, #2337, REPORTX  ;ano → skoč na REPORT X „Bad device type“
07FF            CALL #107C, ARRANGNM    ;uprav jméno souboru v FNZONE1 na masku
0802            JP   C, #1019, REPORTF  ;bylo použito wildchars → skoč na REPORT F „Invalid
                                        ;file name“
0805            JR   Z, #080F , LPSETEXT ;nebyla vložena přípona → skoč
```

Provedeme test přípony na „Q“.

```
0807            LD   A, (#3E94) EXTE1   ;vyzvedni příponu souboru do A
080A            CP   „Q“                ;je to „Q“?
080C            JP   NZ, #1019, REPORTF ;ne → skoč na REPORT F „Invalid file name“
080F LPSETEXT   LD   A, “Q“             ;přípona bude „Q“
0811            LD   (#3E94), A EXTE1   ;ulož příponu souboru do masky
0814            CALL #1C8F, SETACT      ;roztoč mechaniku, která má stejné jméno jako
                                        ;v DNZONE1
0817            CALL #212B, FIRSTMASK   ;načti 1. položku adresáře vyhovující masce v FNZONE1
081A            JP   NZ, #1FB1, REPORTS ;neexistuje → skoč na REPORT S „File not found“
```

Zkontrolujeme, jestli soubor není chráněn proti čtení.

```
081D          CALL #1283, GETATR          ;vyzvedni atributy souboru
0820          BIT  3, A                   ;je READ PROTECTED?
0822          JP   Z, #1FBD, REPORTe      ;ano → skoč na REPORT e „File is read protected“
0825          LD   A, #11                 ;do A relativní adresa uložení prvního sektoru
                                          ;v hlavičce souboru
0827          CALL #0FAD, ADDHLA          ;přičti k HL
082A          LD   E, (HL)                ;vyzvedni číslo prvního sektoru, ve kterém je uložen
082B          INC  HL                     ;soubor, do DE
082C          LD   D, (HL)
082D          EX   DE, HL                 ;přesuň ho do HL
```

Nyní budeme načítat sektor po sektoru a vypisovat jeho obsah. Toto je hlavní vykonávací procedura. V registru HL je číslo prvního sektoru souboru.

```
082E PRINTSEC CALL #1CF1, GETWTEST        ;načti obsah položky ve FAT a otestuj chybu ve FAT
0831          LD   A, D                   ;vyšší byte obsahu do A
0832          CP   #0C                    ;nulová délka souboru?
0834          RET  Z                      ;ano → vrať se
0835          LD   (#3E78), DE VALSYX     ;ulož si obsah DE (číslo dalšího sektoru)
0839          CALL #1DF9, LOGFYZ          ;převeď logický sektor na fyzický sektor a stopu
083C          LD   DE, #0101              ;1 sektor, žádné opakování
083F          LD   HL, #3A00 AUXBUF       ;do HL adresa bufferu
0842          CALL #22A2, BREADA          ;načti sektor
0845          LD   DE, (#3E78) VALSYX     ;vyzvedni zpět obsah položky do DE
0849          LD   A, D                   ;vyšší byte do A
084A          CP   #0E                    ;poslední sektor?
084C          JR   C, #0854, PRINTALL     ;ne → skoč
084E          AND  #01                    ;ponech bit pro zjištění počtu bytů v sektoru
0850          LD   D, A                   ;dej zpět do D
0851          OR   E                      ;je menší než 512 bytů?
0852          JR   NZ, #0857, PRINTBUFF   ;ano → skoč
0854 PRINTALL LD   DE, #0200              ;tisknu celý sektor – délka 512 bytů
0857 PRINTBUFF LD  HL, #3A00 AUXBUF       ;počáteční adresa bufferu do HL
```

Vypíšeme ze sektoru DE bytů.

```
085A PRNTLOOP PUSH DE                     ;ulož si počet tisknutých bytů
085B          PUSH HL                     ;ulož si ukazatel v bufferu
085C          LD   A, (HL)                ;vyzvedni znak
085D          RST  #10                    ;vytiskni ho
085E          POP  HL                     ;obnov ukazatele do bufferu
085F          POP  DE                     ;a čítač
0860          INC  HL                     ;posuň se na další znak
0861          DEC  DE                     ;sniž čítač o jedničku
0862          LD   A, D                   ;už jsi vypsal celý buffer?
0863          OR   E
0864          JR   NZ, #085A, PRNTLOOP    ;ne → opakuj
0866          LD   HL, (#3E78) VALSYX     ;vyzvedni obsah posledně čtené položky do HL
0869          LD   A, H                   ;dej vyšší byte do A
086A          CP   #0E                    ;byl to poslední sektor?
086C          RET  NC                     ;ano → vrať se přes RETURN do ZX ROM
086D          JR   #082E, PRINTSEC        ;skoč na tisk dalšího sektoru
```

LIST, LLIST

Příkazy pro výpis informací o počítačové sestavě (verzi operačního systému MDOS, kolik je připojeno mechanik, jména drivů, který drive je aktuální, kolik je volné paměti, délka proměnných, délka BASIC programu hodnota RAMPTOP)

Syntaxe: LIST *
LLIST *

Vstup těchto příkazů je stejný, liší se pouze otevřený kanál.

```
086F L-LIST     RST  #18                  ;vezmi aktuální znak
0870            CP   „*“                  ;je to hvězdička?
0872            JP   NZ, #06C6, REPORTC   ;ne → skoč na REPORT C „Nonsense in BASIC“
0875            RST  #20                  ;vezmi další znak
0876            CALL #1057, ISSYNCONTR    ;otestuj, jestli není kontrola syntaxe
```

Pokud není kontrola syntaxe, provedeme příkaz. Toto je hlavní prováděcí procedura.

```
0879            LD   A, #0D               ;nový řádek
087B            RST  #10                  ;tiskni
087C            XOR  A                    ;položka 0 – „MDOS Release: 1.0 (17-May-91) ©
                                          ;Didaktik Skalica 1991“
087D            LD   DE, #0971 INFMES     ;do DE adresa tabulky textů příkazů LIST, LLIST
0880            CALL #01C8, PRTMES        ;vypiš položku textu
0883            LD   A, #1                ;položka 1 – „Drives defined:“
0885            LD   DE, #0971 INFMES     ;do DE adresa tabulky textů příkazů LIST, LLIST
0888            CALL #01C8, PRTMES        ;vypiš položku textu
088B            LD   A, „A“               ;začíná se mechanikou A
088D            LD   IX, #3E00 DRPARZN    ;do IX začátek tabulek parametrů mechanik
```

Vytiskneme všechny mechaniky, které lze připojit.

```
0891 RINFO2     PUSH IX                   ;ulož si ukazatel na tabulku parametrů
0893            PUSH AF                   ;ulož si mechaniku
0894            LD   A, (IX+#02)          ;je definována mechanika?
0897            AND  A
0898            JR   Z, #08A5, RINFO1     ;ne → skoč
089A            POP  AF                   ;obnov ASCII kód drivu
089B            PUSH AF                   ;a znovu ho ulož
089C            RST  #10                  ;piš označení mechaniky
089D            LD   A, #02               ;položka 2 – „ :, “
089F            LD   DE, #0971 INFMES     ;do DE adresa tabulky textů příkazů LIST, LLIST
08A2            CALL #01C8, PRTMES        ;vypiš položku textu
08A5 RINFO1     POP  AF                   ;obnov si ASCII kód drivu
08A6            POP  IX                   ;obnov si ukazatel na tabulky parametrů mechanik
08A8            LD   DE, #000C            ;posuň se na parametry další mechaniky
08AB            ADD  IX, DE
08AD            INC  A                    ;posuň se na další mechaniku
08AE            CP   „E“                  ;už byly zkontrolovány všechny čtyři mechaniky?
08B0            JR   C, #0891, RINFO2     ;ne → opakuj
08B2            LD   A, #03               ;položka 3 – „ Drives instaled:“
0894            LD   DE, #0971 INFMES     ;do DE adresa tabulky textů příkazů LIST, LLIST
0897            CALL #01C8, PRTMES        ;vypiš položku textu
089A            LD   A, „A“               ;začíná se mechanikou A
089C            LD   IX, #3E00 DRPARZN    ;do IX začátek tabulek parametrů mechanik
```

Vytiskneme všechny připojené mechaniky.

```
08C0 RINFO4   PUSH IX                 ;ulož si ukazatel na tabulky parametrů
08C2          PUSH AF                 ;a ASCII kód drivu
08C3          BIT  0, (IX+#00)        ;je drive připojen?
08C7          JR   Z, #08D4, RINFO3   ;není → skoč
08C9          POP  AF                 ;obnov ASCII kód drivu
08CA          PUSH AF                 ;a zase ho ulož
08CB          RST  #10                ;tiskni označení mechaniky
08CC          LD   A, #02             ;položka 2 – „ :, “
08CE          LD   DE, #0971 INFMES   ;do DE adresa tabulky textů příkazů LIST, LLIST
08D1          CALL #01C8, PRTMES      ;vypiš položku textu
08D4 RINFO3   POP  AF                 ;obnov si ASCII kód drivu
08D5          POP  IX                 ;a ukazatel na tabulky parametrů mechanik
08D7          LD   DE, #000C          ;posuň se na parametry další mechaniky
08DA          ADD  IX, DE
08DC          INC  A                  ;a posuň se na další mechaniku
08DD          CP   „E“                ;už byly zkontrolovány všechny čtyři mechaniky?
08DF          JR   C, #08C0, RINFO4   ;ne → skoč
08E1          LD   A, #04             ;položka 4 – „Current Device:“
08E3          LD   DE, #0971 INFMES   ;do DE adresa tabulky textů příkazů LIST, LLIST
08E6          CALL #01C8, PRTMES      ;vypiš položku textu
```

Vytiskneme jméno drivu, který je nastaven jako aktuální.

```
08E9          LD   HL, #3EAA ACDRIVE  ;do HL adresa jména drivu, který je aktuální
08EC          CALL #128D, PRTSTR      ;vytiskni jméno
08EF          LD   A, #05             ;položka 5 – „Volumes available:“
08F1          LD   DE, #0971 INFMES   ;do DE adresa tabulky textů příkazů LIST, LLIST
08F4          CALL #01C8, PRTMES      ;vypiš položku textu
```

Vytiskneme jména všech připojených drivů.

```
08F7          CALL #1F49, INITALLDR   ;načti a nastav parametry všech připojených drivů
08FA          LD   B, #04             ;maximálně 4 disky
08FC          LD   HL, #3E30 DRNAMES  ;do HL adresa uložení jmen drivů
08FF RINFO5   PUSH BC                 ;ulož si počítadlo
0900          PUSH HL                 ;a ukazatel na jména disků
0901          LD   A, (HL)            ;vyzvedni první znak jména
0902          AND  A                  ;je jméno?
0903          JR   Z, #0913, RINFO6   ;není → skoč
0905          LD   A, #20             ;mezera
0907          RST  #10                ;tiskni
0908          LD   A, #20             ;mezera
090A          RST  #10                ;tiskni
090B          POP  HL                 ;obnov ukazatel na jména drivů
090C          PUSH HL                 ;a zase ho ulož
090D          CALL #128D, PRTSTR      ;vytiskni jméno drivu
0910          LD   A, #0D             ;nový řádek
0912          RST  #10                ;tiskni
0913 RINFO6   POP  HL                 ;obnov ukazatel na jména drivů
0914          POP  BC                 ;a obnov počítadlo mechanik
0915          LD   A, #0C             ;posuň ukazatel na další jméno drivu
0917          CALL #0FAD, ADDHLA
091A          DJNZ #08FF, RINFO5      ;opakuj B-krát
```

Vytiskneme systémové proměnné BASICu.

```
091C    LD   A, #06              ;položka 6 – „Length of Program:“
091E    LD   DE, #0971 INFMES    ;do DE adresa tabulky textů příkazů LIST, LLIST
0921    CALL #01C8, PRTMES       ;vypiš text položky
0924    LD   HL, (#5C4B) VARS    ;do HL začátek proměnných
0927    PUSH HL                  ;ulož na zásobník
0928    LD   DE, (#5C53) PROG    ;do DE začátek programu v BASICu
092C    AND  A                   ;nuluj CY
092D    SBC  HL, DE              ;v HL je délka BASIC programu (VARS-PROG)
092F    LD   C, L                ;dej ji do BC
0930    LD   B, H
0931    CALL #0FA6, BCPRT        ;vypiš délku BASIC programu
0934    LD   A, #07              ;položka 7 – „Length of Variables:“
0936    LD   DE, #0971 INFMES    ;do DE adresa tabulky textů příkazů LIST, LLIST
0939    CALL #01C8, PRTMES       ;vypiš text položky
093C    LD   HL, (#5C59) E_LINE  ;do HL adresa konce proměnných
093F    POP  DE                  ;obnov začátek proměnných
0940    AND  A                   ;nuluj CY
0941    SBC  HL, DE              ;do HL délka proměnných (E_LINE-VARS)
0943    LD   C, L                ;dej ji do BC
0944    LD   B, H
0945    CALL #0FA6, BCPRT        ;vypiš délku proměnných
0948    LD   A, #08              ;položka 8 – „Top of RAM:“
094A    LD   DE, #0971 INFMES    ;do DE adresa tabulky textů příkazů LIST, LLIST
094D    CALL #01C8, PRTMES       ;vypiš text položky
0950    LD   BC, (#5CB2) RAMTOP  ;do BC adresa posledního byte pro BASIC
0954    CALL #0FA6, BCPRT        ;vypiš hodnotu RAMTOPu
0957    LD   A, #09              ;položka 9 – „Free memory:“
0959    LD   DE, #0971 INFMES    ;do DE adresa tabulky textů příkazů LIST, LLIST
095C    CALL #01C8, PRTMES       ;vypiš text položky
095F    RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0960    DW   #1F1A               ;podprogram FREE-MEM zjištění obsazené paměti
0962    LD   HL, #FFFF           ;do HL 65535
0965    AND  A                   ;nuluj CY
0966    SBC  HL, BC              ;odečti BC, v HL je velikost volné paměti
0968    LD   C, L                ;dej ji do BC
0969    LD   B, H
096A    CALL #0FA6, BCPRT        ;piš volnou pamět
096D    LD   A, #0D              ;nový řádek
096F    RST  #10                 ;tiskni
0970    RET                      ;vrať se přes RETURN do ZX ROM
```

INFMES

Tabulka textů pro příkazy LIST * a LLIST *. Začíná invertovaným znakem.

```
0971 INFMES   DB   #80                              ;invertovaný znak
  Položka 0
0972          4D 44 4F 53 20 52 65 6C 65 61 73 65   ;MDOS Release
              3A 20 31 2E 30 20 28 31 37 2D 4D      ;:1.0 (17-M
              61 79 2D 39 31 29 0D 28 43 29 20 44   ;ay-91) (C) D
              69 64 61 6B 74 69 6B 20 53 6B 61 6C   ;idaktik Skal
              69 63 61 20 31 39 39 31 0D 8D         ;ica 1991
```

V opravené verzi MDOSu 1.0 je změněno hlášení na *MDOS Release: 1.0 (01-Sep-92) (C) Didaktik Skalica 1992*. Potom to vypadá takto:

```
0972    4D 44 4F 53 20 52 65 6C 65 61 73 65   ;MDOS Release
        3A 20 31 2E 30 20 28 30 31 2D 53      ;:1.0 (01-S
        65 70 2D 39 32 29 0D 28 43 29 20 44   ;ep-92) (C) D
        69 64 61 6B 74 69 6B 20 53 6B 61 6C   ;idaktik Skal
        69 63 61 20 31 39 39 32 0D 8D         ;ica 1992
```

```
  Položka 1
09AB    44 72 69 76 65 73 20 44 65 66 69 6E   ;Drives defin
        65 64 20 20 3A A0                     ;ed:
  Položka 2
09BD    3A 2C A0                              ;:,
  Položka 3
09C0    08 08 20 0D 44 72 69 76 65 73 20 49   ;Drives I
        6E 73 74 61 6C 6C 65 64 3A A0         ;nstalled:
  Položka 4
09D6    08 08 20 0D 43 75 72 72 65 6E 74 20   ;Current
        44 65 76 69 63 65 20 20 3A A0         ;Device:
  Položka 5
09EC    3A 0D 0D 56 6F 6C 75 6D 65 73 20      ;: Volumes
        41 76 61 69 6C 61 62 6C 65 3A 8D      ;available:
  Položka 6
0A02    4C 65 6E 67 74 68 20 6F 66 20 50 72   ;Lenght of Pr
        6F 67 72 61 6D 20 20 3A A0            ;ogram:
  Položka 7
0A17    0D 4C 65 6E 67 74 68 20 6F 66 20 56   ;Lenght of V
        61 72 69 61 62 6C 65 73 3A A0         ;ariables:
  Položka 8
0A2D    0D 0D 54 6F 70 20 6F 66 20 52 41      ;Top of RA
        4D 20 3A A0                           ;M:
  Položka 9
0A3C    0D 46 72 65 65 20 6D 65 6D 6F 72 79   ;Free memory
        3A A0 A0                              ;:
```

RESTORE

Příkaz pro zápis obsahu paměti do specifikovaného sektoru.

Syntaxe: RESTORE *"Zařízení nebo JménoSouboru", sektor, adresa

Zapíše obsah paměti od adresy do logického sektoru. Pokud se použije „Zařízení“, začíná logický sektor od 1. logického sektoru na disketě a končí posledním přístupným sektorem na disketě (1. logický sektor je 1. fyzický sektor na 0. stopě, 2. logický sektor je 2. fyzický sektor na 0. stopě atd.). Lze tak tedy zapsat na jakékoliv místo diskety (pokud je přístupné). Pokud se použije jméno souboru, začíná logický sektor od prvního sektoru uložení souboru na disketě a končí posledním sektorem uložení souboru. Je to stezka souboru (1. logický sektor je první sektor uložení souboru, 2. logický sektor je druhý sektor uložení souboru atd.).

```
0A4B RESTORE    LD   HL, #2296 BWRITE   ;adresa rutiny příkazu WRITESECTOR
0A4E            JR   #0A53, RRPRG      ;jdi na společnou část
```

READ

Příkaz pro načtení specifikovaného sektoru do paměti.

Syntaxe: READ * "Zařízení nebo JménoSouboru", sektor, adresa

Načte logický sektor z diskety do paměti na adresu. Číslovaní sektorů je stejné jako u příkazu RESTORE.

```
0A50 READ       LD   HL, #22A5 BREAD     ;adresa rutiny příkazu READSECTOR
```

Protože mají příkazy RESTORE a READ stejnou syntaxi a liší se pouze v tom, že příkaz RESTORE zapisuje a příkaz READ čte, je jejich vykonávací program společný. Liší se pouze ve skoku na prováděnou operaci čtení/zápis sektoru.

```
0A53 RRPRG      LD   (#3E78), HL VALSYX  ;ulož si adresu rutiny na dobu kontroly syntaxe
0A56            RST  #18                 ;načti aktuální znak
0A57            CP   „*“                 ;je to hvězdička?
0A59 JMPREPC    JP   NZ, #06C6, REPORTC  ;ne → skoč na REPORT C „Nonsense in BASIC“
```

Provedeme syntaktickou analýzu parametrů příkazu. Jako první to je jméno zařízení nebo souboru.

```
0A5C            RST  #20                 ;načti další znak
0A5D            RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0A5E            DW   #1C8C               ;podprogram EXPT-EXP vyhodnocení řetězce –
                                         ;je vyzvednuto buď jméno souboru nebo zařízení
0A60            LD   B, #02              ;budeme analyzovat dva parametry
```

Vyzvedneme dva parametry příkazu. První je číslo logického sektoru a druhé je adresa uložení sektoru v paměti.

```
0A62 RRPRGPAR   PUSH BC                  ;ulož si čítač parametrů
0A63            RST  #18                 ;načti aktuální znak
0A64            CP   „ , “               ;je to čárka?
0A66            JR   NZ, #0A59, JMPREPC  ;ne → skoč na REPORT C „Nonsense in BASIC“
0A68            RST  #20                 ;načti další znak
0A69            RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0A6A            DW   #1C82               ;podprogram EXPT-EXP vyhodnocení číselného výrazu
0A6C            POP  BC                  ;obnov čítač parametrů
0A6D            DJNZ #0A62, RRPRGPAR     ;opakuj B-krát
0A6F            CALL #1057, ISSYNCONTR   ;otestuj, jestli není kontrola syntaxe
```

Není kontrola syntaxe, provedeme proto příkaz.

```
0A72            RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0A73            DW   #1E99               ;podprogram FIND-INT2 vyzvednutí čísla do BC
                                         ;vyzvedni 2. parametr – adresu uložení
0A75            LD   (#3E7A), BC VALSYY  ;ulož si ji do SRAM
0A79            RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0A7A            DW   #1E99               ;podprogram FIND-INT2 vyzvednutí čísla do BC
                                         ;vyzvedni 1. parametr – číslo logického sektoru
0A7C            PUSH BC                  ;ulož si ho na zásobník
0A7D            CALL #0FCF, DIVSTRING    ;rozděl řetězec na zásobníku na jméno disku a souboru
0A80            CALL #1043, SETWDNM      ;nastav jméno disku v DNZONE1 pro I/O
0A83            CALL #1C8F, SETACT       ;roztoč mechaniku, která má stejné jméno jako
                                         ;v DNZONE1
0A86            LD   A, (#3E8A) FNZONE1  ;přečti první znak jména souboru v FNZONE1
0A89            AND  A                   ;bylo vloženo jméno?
0A8A            JR   Z, #0AB4, RRINDISK  ;ne → skoč
```

Budeme hledat logický sektor souboru.

```
0A8C            CALL #107C, ARRANGNM     ;uprav jméno souboru v FNZONE1 na masku
0A8F            JP   C, #1019, REPORTF   ;bylo použito wildchars → skoč na REPORT F „Invalid
                                         ;file name“
```

```
0A92            CALL #212B, FIRSTMASK     ;načti 1. položku adresáře vyhovující masce v FNZONE1
0A95            JP   NZ, #1FB1, REPORTS   ;neexistuje → skoč na REPORT S „File not found“
0A98            LD   A, #11               ;posuň se na adresu uložení 1. sektoru
0A9A            CALL #0FAD, ADDHLA
0A9D            LD   A, (HL)              ;vyzvedni do HL první sektor uložení souboru
0A9E            INC  HL
0A9F            LD   H, (HL)
0AA0            LD   L, A
0AA1            POP  BC                   ;číslo hledaného logického sektoru do BC
```

Nyní najdeme požadované číslo logického sektoru ve stezce souboru.

```
0AA2 RRFINDSC   LD   A, B                 ;už byl nalezen?
0AA3            OR   C
0AA4            JR   Z, #0AB5, RRFINDOK   ;ano → skoč
0AA6            DEC  BC                   ;sniž počítadlo o jedničku
0AA7            CALL #1CF1, GETWTEST      ;načti položku FAT a otestuj chybu ve FAT
0AAA            EX   DE, HL               ;dej obsah do HL
0AAB            BIT  3, H                 ;už jsi na konci souboru?
0AAD            JR   Z, #0AA2, RRFINDSC   ;ne → skoč
```

Soubor není uložen v tolika sektorech. Logický sektor je tedy mimo rozsah.

```
0AAF            LD   A, #31               ;REPORT i „Bad record number“
0AB1            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Je to logický sektor drivu a ne souboru.

```
0AB4 RRINDISK   POP  HL                   ;číslo logického sektoru do HL
```

Nyní načteme nebo zapíšeme daný sektor.

```
0AB5 RRFINDOK   CALL #1DF9, LOGFYZ        ;převeď logický sektor na fyzický sektor a stopu
0AB8            LD   DE, #0100            ;1 sektor, 255 opakování
0ABB            CALL #217B, ERAVAR        ;vymaž pomocné proměnné MDOSu
0ABE            LD   A, (#3E6B) WORKDR    ;do A drive, se kterým se pracuje
0AC1            LD   HL, (#3E78) VALSYX   ;dej adresu rutiny pro čtení nebo zápis do HL
0AC4            PUSH HL                   ;a ulož ji na zásobník
0AC5            LD   HL, (#3E7A) VALSYY   ;do HL dej adresu, kam zapisovat/číst
0AC8            RET                       ;proveď operaci a vrať se přes RETURN do ZX ROM
```

Sekvenční soubory

MDOS nám poskytuje rozšíření práce se sekvenčními soubory. Umožňuje připojit na kanál soubor pro vstup dat z nebo do kanálu. K otevření kanálu slouží příkaz OPEN a k uzavření příkay CLOSE. Pro vstup a výstup do/z kanálu se používají klasické příkazy PRINT a INPUT.

Příkaz OPEN slouží k otevření kanálu.

Syntaxe: OPEN #n,["JménoSouboru1[.PříponaSouboru1]"][,"JménoSouboru2[.PříponaSouboru2]"]

Příkaz otevře kanál n. Soubor JménoSouboru1 je připojen jako vstup do kanálu a soubor JménoSouboru2 jako výstup z kanálu. Takže ze JménoSouboru1 lze číst a do JménoSouboru2 lze zapisovat. Samozřejmě si můžeme otevřít soubor jenom pro vstup nebo pro výstup. Při otevření souboru pro vstup nebo výstup si MDOS vytvoří pro každý soubor bufer, přes který se děje veškerá komunikace. Do něj si vždy buď nahrává sektor, pokud se jedná o vstupní soubor, a čte z něj data. Až dojde na konec buferu, načte si další sektor. Pokud se jedná o výstupní soubor, data do něj zapisuje a až je bufer plný, je zapsán na disk a zapisuje se znovu od začátku buferu. Bufer tedy tvoří jakousi vyrovnávací pamět, která když se zaplní, je zapsána na disk, nebo když je prázdná, je zaplněna z disku (caching). Velikost buferu pro každý soubor je 544 bytů. Skládá se z hlavičky (32 bytů) a z oblasti pro uložení dat (512 bytů).

Struktura hlavičky:

adresa rutiny pro zápis znaku do kanálu	2 byty
adresa rutiny pro čtení znaku z kanálu	2 byty
kód kanálu	1 byte
číslo drivu, kde je uložen soubor připojený na kanál	1 byte
jméno disku, kde je uložen soubor připojený na kanál	12 bytů
číslo fyz. stopy a sektoru, kde je uložena hlavička souboru v adresáři	2 byty
adresa uložení hlavičky souboru v buferu adresáře	2 byty
délka souboru	3 byty
číslo sektoru, který se zrovna používá jako poslední sektor souboru	2 byty
počet znaků ke čtení/zapsaných	2 byty
ukazatel na aktuální pozici v buferu	2 byty
nevyužito	1 byte

Bufery se ukládají za systémové proměnné BASICu před začátek BASIC programu. Pokud budete chtít otevřít soubor se jménem „K“, „P“ nebo „S“ a bude bez přípony, tak to nepůjde (obejít se to dá přidáním přípony, aby nebyl název souboru dlouhý jeden znak, protože jsou tyto kódy rezervovány). Je taky nebezpečné připojovat soubor na kanál 0, protože při volání chybového hlášení (chyby nebo hlášení OK) nebo editoru je adresa kanálu 0 „tvrdě“ přepsána, takže se ztratí ukazatel na bufer a soubor už bude nepřístupný. Dále nejde přesměrovat výstup příkazů MDOSu do kanálu v důsledku struktury MDOSu.

Příkaz CLOSE slouží k uzavření kanálu.

Syntaxe: CLOSE #n

Příkaz uzavře kanál n. Pokud byl na tento kanál připojen soubor pro výstup, je jeho bufer uložen na disk jako poslední sektor souboru.

OPENIN

Vstupní bod, pokud chceme připojit na kanál soubor pro vstup (čtení).

```
0AC9 OPENINPUT  LD   HL, (#5C65) STKEND  ;vezmi adresu konce zásobníku kalkulátoru
0ACC            LD   DE, #000A           ;o 10 bytů
0ACF            ADD  HL, DE              ;posuň konec zásobníku kalkulátoru
0AD0            LD   (#5C65), HL STKEND  ;ulož nový konec zásobníku kalkulátoru
0AD3            RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0AD4            DW   #171E               ;podprogram STR-DATA vyzvedni data pro číslo
                                         ;proudu, které je na zásobníku kalkulátoru
```

Zjistíme, jestli už daný kanál není otevřen.

```
0AD6            LD   (#3E78), HL VALSYX  ;ulož si adresu kanálu v oblasti proudových dat
0AD9            LD   A, B                ;je kanál otevřen?
0ADA            OR   C
0ADB            JR   Z, #0AEB, STRNOOPEN;ne → skoč na otevření
0ADD            LD   A, B                ;do A vyšší byte posunu
0ADE            AND  A                   ;už mu byl přiřazen nějaký soubor?
0ADF            JR   NZ, REPORTm         ;ano → skoč na REPORT m „Stream already open“
0AE1            LD   A, C                ;do A nižší byte posunu
0AE2            CP   #11                 ;byl už mu přiřazen některý z kanálů „K“, „S“, „R“, „P“?
0AE4            JR   C, #0AEB, STRNOOPEN;ano → skoč
```

Na kanál už byl připojen jiný soubor.

```
0AE6 REPORTm    LD   A, #35              ;REPORT m „Stream already open“
0AE8            JP   #0204, ERRR         ;piš hlášení a skoč do ZX ROM
```

Otevře soubor pro čtení

```
0AEB STRNOOPEN CALL #0D14, ANAOPENM    ;rozlož řetězec na zásobníku na jméno souboru a disku
0AEE           PUSH IX                 ;ulož si IX
0AF0           CALL #0DB5, SETSTRNM    ;nastav jméno disku v DNZONE1 a otestuj příponu
                                       ;souboru na „B“ a „Q“ a použití wildchars
0AF3           CALL #1C8F, SETACT      ;roztoč mechaniku, která má stejné jméno jako
                                       ;v DNZONE1
0AF6           CALL #212B, FIRSTMASK   ;načti 1. položku adresáře vyhovující masce v FNZONE1
0AF9           JP   NZ, #1FB1, REPORTS ;nenašel → skoč na REPORT S „File not found“
```

Otestujeme, jestli soubor není chráněn proti čtení.

```
0AFC           CALL #1283, GETATR      ;vyzvedni atributy souboru
0AFF           BIT  3, A               ;je READ PROTECTED?
0B01           JP   Z, #1FBD, REPORTe  ;ano → skoč na REPORT e „File is read protected“
```

Nyní si vytvoříme bufer v RAM a nastavíme kanálové informace otevíraného kanálu.

```
0B04           PUSH HL                 ;ulož si adresu položky v buferu adresáře
0B05           CALL #0D6F, MAKE544B    ;vytvoř si volné místo pro bufer daného kanálu
0B08           PUSH HL                 ;ulož si začátek buferu
0B09           CALL #0D21, SETSTRBUF   ;vypočti offset buferu a ulož ho od tabulky informací
                                       ;o kanálech
0B0C           POP  HL                 ;obnov adresu začátku buferu
0B0D           LD   (#3E7A), HL VALSYY ;ulož si začátek buferu
```

Budeme vytvářet hlavičku buferu. Nejdříve si uložíme adresy rutin pro výstup a vstup a kód kanálu.

```
0B10           LD   DE, #15C4          ;do DE adresa rutiny pro zápis znaku
```

Protože je tento soubor otevřen pouze pro čtení, je při pokusu o zápis vyvoláno chybové hlášení na adrese #15C4 v ZX ROM.

```
0B13           CALL #0D67, LD(HL)DE    ;ulož do hlavičky buferu
0B16           LD   DE, #22C2          ;do DE adresa rutiny pro načtení znaku
```

Na této adrese je v ZX ROM příkaz RST #0, který vyvolá přestránkování do ROM D40, kde se podle návratové adresy určí, že je volána rutina pro čtení znaku z kanálu.

```
0B19           CALL #0D67, LD(HL)DE    ;ulož do hlavičky buferu
0B1C           LD   A, #EB             ;do A kód názvu kanálu – určuje, že je připojen soubor
0B1E           CALL #0D6C, LD(HL)A     ;ulož do hlavičky buferu
```

Nyní nastavíme informace o souboru.

```
0B21 MAKEHBUF  LD   A, (#3E6B) WORKDR  ;do A drive, se kterým se pracuje
0B24           CALL #0D6C, LD(HL)A     ;ulož číslo drive do hlavičky buferu
0B27           EX   DE, HL             ;do DE ukazatel do hlavičky buferu
0B28           PUSH IX                 ;do HL adresu parametrů drivu z IX
0B2A           POP  HL
0B2B           LD   A, #30             ;posuň se na jméno disku, se kterým se pracuje
0B2D           CALL #0FAD, ADDHLA
0B30           LD   BC, #000C          ;12 bytů (jméno + dva náhodné byty)
0B33           LDIR                    ;přesuň do hlavičky buferu
0B35           EX   DE, HL             ;vrať ukazatel do hlavička buferu zpět do HL
0B36           LD   DE, (#3E6F) ADRSCTR ;vyzvedni číslo stopy a sektoru naposledy načteného
                                       ;do buferu adresáře
```

```
0B3A            CALL #0D67, LD(HL)DE    ;a ulož si ho do hlavičky
0B3D            POP  DE                 ;obnov si adresu položky v buferu adresáře
0B3E            PUSH DE                 ;a znovu ji ulož
0B3F            CALL #0D67, LD(HL)DE    ;a ulož ji ještě do hlavičky buferu
0B42            EX   (SP), IX           ;dej adresu položky v buferu adresáře do IX a ulož si IX
0B44            LD   A, (IX+#0B)        ;vyzvedni délku souboru
0B47            CALL #0D6C, LD(HL)A     ;a ulož ji do hlavičky buferu
0B4A            LD   A, (IX+#0C)
0B4D            CALL #0D6C, LD(HL)A
0B50            LD   A, (IX+#15)
0B53            CALL #0D6C, LD(HL)A
0B56            LD   E, (IX+#11)        ;vyzvedni číslo prvního sektoru uložení souboru do DE
0B59            LD   D, (IX+#12)
0B5C            CALL #0D67, LD(HL)DE    ;a ulož do hlavičky buferu
0B5F            POP  IX                 ;obnov IX
0B61            LD   DE, #0000          ;počet znaků ke čtení je na začátku nula
0B64            CALL #0D67, LD(HL)DE    ;ulož do hlavičky buferu
0B67            LD   E, L               ;dej ukazatel do hlavičky buferu do DE
0B68            LD   D, H
0B69            INC  DE                 ;a zvyš ji o tři
0B6A            INC  DE
0B6B            INC  DE                 ;v DE je začátek uložení dat v buferu
0B6C            CALL #0D67, LD(HL)DE    ;a ulož do hlavičky buferu
0B6F            CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
0B72            POP  IX                 ;obnov IX
```

Nyní se vrátíme zpět do ZX ROM na pokračování provádění příkazu OPEN.

```
0B74            LD   HL, #1B00          ;do HL adresa v tabulce parametrů příkazů v ZX ROM
                                        ;pro pokračování příkazu OPEN
0B77            LD   (#5C74), HL T_ADDR ;ulož ji do adresy jako další položku v tabulce syntaxe
0B7A            RET                     ;a vrať se na další zpracování příkazu OPEN
```

OPENOUT

Vstupní bod, pokud se na kanál připojuje pouze soubor pro výstup (OPEN #n,„JménoSouboru“).

```
0B7B ONLYOUT    RST  #30                ;je kontrola syntaxe?
0B7C            JR   Z, #0B93, NOOPEN1  ;ano → skoč
```

Provádí se příkaz, proto otestujeme, jestli už daný kanál není otevřen.

```
0B7E            RST  #28                ;volej podprogram pro volání rutiny ZX ROM
0B7F            DW   #171E              ;podprogram STR-DATA vyzvednutí dat pro daný kanál
0B81            LD   (#3E78), HL VALSYX ;ulož si adresu kanálu v oblasti proudových dat
0B84            LD   A, B               ;už je kanál otevřen?
0B85            OR   C
0B86            JR   Z, #0B93, NOOPEN1  ;ne → skoč
0B88            LD   A, B               ;do A vyšší byte posunu
0B89            AND  A                  ;byl už mu přiřazen nějaký soubor?
0B8A            JP   NZ, #0AE6, REPORTm ;ano → skoč na REPORT m „Stream already open“
0B8D            LD   A, C               ;do A nižší byte posunu
0B8E            CP   #11                ;byl už mu přiřazen některý z kanálů „K“, „S“, „R“, „P“?
0B90            JP   NC, #0AE6, REPORTm ;ne → skoč na REPORT m „Stream already open“
```

Provedeme kontrolu parametrů příkazu

```
0B93 NOOPEN1    RST   #18                    ;vezmi aktuální znak
0B94            CP    „ , “                  ;je to čárka?
0B96            JP    NZ, #06C6, REPORTC     ;ne → skoč na REPORT C „Nonsense in BASIC“
0B99            RST   #20                    ;vezmi další znak
0B9A            RST   #28                    ;volej podprogram pro volání rutiny ZX ROM
0B9B            DW    #1C8C                  ;podprogram EXPT-EXP vyhodnocení řetězce
0B9D            CALL #1057, ISSYNCONTR       ;otestuj, jestli není kontrola syntaxe
```

Neprovádí se kontrola syntaxe, provedeme příkaz.

```
0BA0            CALL #0FCF, DIVSTRING        ;rozděl řetězec na zásobníku na jméno disku a souboru
```

Otevřeme soubor pro zápis se jménem ve FNZONE1 a jménem disku v DNZONE1. Na začátku je nastaven jako soubor s nulovou délkou.

```
0BA3 OPENOUTF   PUSH IX                      ;ulož si IX
0BA5            CALL #0DB5, SETSTRNM         ;nastav jméno disku a souboru, otestuj příponu souboru
0BA8            CALL #1C8F, SETACT           ;roztoč mechaniku, která má stejné jméno jako
                                             ;v DNZONE1
```

Zjistíme, jestli takový soubor již neexistuje.

```
0BAB            CALL #212B, FIRSTMASK        ;načti 1. položku adresáře vyhovující masce v FNZONE1
0BAE            JR    NZ, #0BBB, OPENNULF    ;nenalezena → skoč
0BB0            CALL #1283, GETATR           ;vyzvedni atributy souboru
0BB3            BIT   2, A                   ;je WRITE PROTECTED?
0BB5            JP    Z, #1A10, REPORTf      ;ano → skoč na REPORT f „File is write protected“
```

Vymažeme existující soubor, neptáme se na přepsání souboru.

```
0BB8            CALL #1F88, DFILER           ;proveď smazání vyhledaného souboru
```

Vytvoříme soubor s nulovou délkou.

```
0BBB OPENNULF   CALL #0D31, SETEMPTYF        ;vytvoř soubor se jménem v FNZONE1 jako soubor
                                             ;s nulovou délkou
0BBE            PUSH HL                      ;ulož si ukazatel na položku adresáře v buferu
0BBF            CALL #0D6F, MAKE544B         ;vytvoř si místo pro bufer o velikosti 544 bytů
0BC2            PUSH HL                      ;ulož si adresu začátku buferu
0BC3            CALL #0D21, SETSTRBUF        ;vypočti offset buferu a ulož ho do tabulky informací
                                             ;o kanálech
0BC6            POP   HL                     ;obnov adresu buferu
0BC7            LD    DE, #25AB              ;do DE adresa rutiny pro zápis znaku
0BCA            CALL #0D67, LD(HL)DE         ;ulož do hlavičky buferu
```

Na této adrese v ZX ROM je instrukce RST #0, která vyvolá přestránkování do ROM D40. Tam se podle návratové adresy do volajícího programu určí skok na rutinu pro výstup znaku do sekvenčního souboru.

```
0BCD            LD    DE, #15C4              ;do DE adresa rutiny pro čtení znaku
0BD0            CALL #0D67, LD(HL)DE         ;ulož do hlavičky buferu
```

Protože je tento soubor otevřen pouze pro zápis, je na místo adresy rutiny uložen skok na chybové hlášení v ZX ROM.

```
0BD3            LD    A, #EB                 ;do A kód názvu kanálu – určuje, že je připojen soubor
0BD5            CALL #0D6C, LD(HL)A          ;ulož do hlavičky buferu
0BD8            JP    #0B21, MAKEHBUF        ;vytvoř zbytek hlavičky a vrať se do ZX ROM
```

OPENIO

Vstupní bod, pokud chceme připojit na kanál soubor jako výstup z kanálu, přičemž už je na něj připojen soubor pro vstup. (OPEN #n, "JmenoSouboru1", "JménoSouboru2").

```
0BDB OPENIO     RST  #18                    ;vezmi aktuální znak
0BDC            CP   „ , “                  ;je to čárka?
0BDE            JP   NZ, #06C6, REPORTC     ;ne → skoč na REPORT C „Nonsense in BASIC“
0BE1            RST  #20                    ;vezmi další znak
0BE2            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
0BE3            DW   #1C8C                  ;podprogram EXPT-EXP vyhodnocení řetězce
                                            ;vyzvedneme JménoSouboru2
0BE5            CALL #1057, ISSYNCONTR      ;test, jestli není kontrola syntaxe
```

Není kontola syntaxe, provedeme příkaz.

```
0BE8            CALL #0FCF, DIVSTRING       ;rozděl řetězec na zásobníku na jméno disku a souboru
0BEB            CALL #0BA3, OPENOUTF        ;otevři výstupní soubor se jménem ve FNZONE1
0BEE            LD   HL, (#3E7A) VALSYY     ;do HL adresu buferu souboru, který slouží pro vstup
```

Je třeba přepsat adresy rutin pro vstup a výstup, protože jsou otevřeny soubory jak pro vstup, tak i pro výstup. Tyto adresy se zapisují do hlavičky buferu souboru pro vstup, protože odtud budou vyzvedávány systémem v ZX ROM.

```
0BF1            LD   DE, #27AC              ;do DE adresu rutiny pro zápis znaku
0BF4            CALL #0D67, LD(HL)DE        ;ulož ji do hlavičky buferu
0BF7            LD   DE, #27A5              ;do DE adresu rutiny pro načtení znaku
0BFA            CALL #0D67, LD(HL)DE        ;ulož ji do hlavičky buferu
0BFD            LD   A, #EB                 ;do A kód názvu kanálu – určuje, že je připojen soubor
0BFF            CALL #0D6C, LD(HL)A         ;ulož do hlavičky buferu
```

Protože při otevírání souboru pro výstup došlo k přepsání kanálových informací u souboru, který je připojen na stejný kanál a je otevřen pro vstup, (offset teď ukazuje na bufer souboru, který je otevřen pro zápis), musíme odečíst 544 bytů (délka buferu), aby offset ukazoval zpět na bufer souboru pro vstup. Lze to provést, protože oba bufery jsou vždy za sebou.

```
0C02            LD   HL, (#3E78) VALSYX     ;do HL adresu kanálu v oblasti proudových dat
0C05            CALL #0DB0, LDDE(HL)        ;vyzvedni obsah do DE
0C08            LD   BC, #FDE0              ;do BC dej –544
0C0B            EX   DE, HL                 ;do HL offset buferu
0C0C            ADD  HL, BC                 ;přičti BC (odečti 544)
0C0D            EX   DE, HL                 ;a dej zpět do DE
0C0E            DEC  HL                     ;nastav adresu kanálu do původního stavu
0C0F            DEC  HL
0C10            CALL #0D67, LD(HL)DE        ;a ulož tam nový offset
0C13            RET                         ;vrať se přes RETURN do ZX ROM
```

CLOSESTR

Tento podprogram zavírá proud. V A je číslo kanálu

IN: A číslo kanálu (0–15)
OUT: uzavře daný kanál, vyprázdní bufer souboru na disk

```
0C14 CLOSESTR   LD   HL, (#5C65) STKEND     ;vezmi adresu vrcholu zásobníku kalkulátoru do HL
0C17            LD   DE, #0005              ;posuň o 5 bytů
0C1A            ADD  HL, DE
0C1B            LD   (#5C65), HL STKEND     ;a ulož nový vrchol zásobníku kalkulátoru
```

Vyzvedneme parametr a otestujeme ho.

```
0C1E            RST  #28            ;volej podprogram pro volání rutiny ZX ROM
0C1F            DW   #1E94          ;podprogram FIND-INT1 vyzvednutí čísla do A
                                    ;vyzvedneme číslo proudu
0C21            RST  #28            ;volej podprogram pro volání rutiny ZX ROM
0C22            DW   #1721          ;podprogram pro kontrolu rozsahu čísla proudu
                                    ;zkontroluj, jestli je číslo v rozsahu 0 až 15
```

Uzavřeme soubor otevřený pro zápis s vyprázdněním buferu a uzavřeme kanál.

```
0C24 CLOUTSTRF  PUSH HL             ;ulož si adresu proudových dat pro daný kanál
0C25            CALL #0C4B, CLOPENF ;uzavři soubory připojené na kanál
0C28            POP  HL             ;obnov si adresu
0C29            RST  #28            ;volej podprogram pro volání rutiny ZX ROM
0C2A            DW   #16EB          ;podprogram zavři proud
0C2C            RET                 ;vrať se přes RETURN do ZX ROM
```

CLOSEALLSTR

Uzavře všechny kanály, na které je připojen sekvenční soubor.

```
0C2D CLOSEALL   CALL #1057, ISSYNCONTR  ;otestuj, jestli není kontrola syntaxe
```

Není kontrola syntaxe, provedeme příkaz.

```
0C30            XOR  A              ;začíná se kanálem 0
```

Budeme postupně zavírat kanály, na které je připojen sekvenční soubor.

```
0C31 CLOSEALL1  PUSH AF             ;ulož si číslo zavíraného kanálu
0C32            RST  #28            ;volej podprogram pro volání rutiny ZX ROM
0C33            DW   #1721          ;podprogram vyzvednutí dat pro kanál v A
0C35            PUSH HL             ;ulož si adresu oblasti proudových dat pro daný kanál
0C36            LD   HL, (#5C4F) CHANS  ;do HL adresa, kde jsou uloženy informace o kanálech
0C39            ADD  HL, BC         ;posuň se na data o kanálu
0C3A            INC  HL             ;posuň se na kód kanálu
0C3B            INC  HL
0C3C            INC  HL
0C3D            LD   A, (HL)        ;dej kód kanálu do A
0C3E            POP  HL             ;obnov adresu oblasti proudových dat pro daný kanál
0C3F            CP   #EB            ;je na tento kanál připojen soubor?
0C41            CALL Z, #0C24, CLOUTSTRF ;ano → uzavři soubory připojené na daný kanál
0C44            POP  AF             ;obnov čítač kanálů
0C45            INC  A              ;posuň se na další
0C46            CP   #10            ;uzavřeno všech 16 kanálů?
0C48            JR   C, #0C31, CLOSEALL1 ;ne → opakuj
0C4A            RET                 ;vrať se přes RETURN do ZX ROM
```

CLOPENF

Uzavře soubory připojené na kanál. V BC je offset buferu.

```
0C4B CLOPENF    LD   A, B           ;je už proud zavřen?
0C4C            OR   C
0C4D            RET  Z              ;ano → vrať se
0C4E            LD   HL, (#5C4F) CHANS  ;do HL adresa, kde jsou uloženy informace o kanálech
0C51            ADD  HL, BC         ;posuň se na hlavičku buferu
```

```
0C52            LD   D, (HL)              ;vyzvedni adresu rutiny pro zápis znaku
0C53            DEC  HL                   ;do DE
0C54            LD   E, (HL)
0C55            EX   DE, HL               ;a dej ji do HL
0C56            LD   BC, #25AB            ;do BC adresa rutina pro zápis znaku, když je otevřen
                                          ;pouze soubor pro zápis
0C59            SBC  HL, BC               ;jsou stejné?
0C5B            JR   Z, #0C72, CLOSEOUTF  ;ano → skoč na uzavření souboru otevřeného pro
                                          ;zápis s vyprázdnění buferu
0C5D            LD   BC, #0201            ;do BC 513, v HL je buď #201, pokud byly otevřeny
                                          ;soubory pro zápis i pro čtení nebo #F019, pokud byl
                                          ;otevřen pouze soubor pro čtení
0C60            AND  A                    ;nuluj CY
0C61            SBC  HL, BC               ;odečti BC od HL
0C63            JR   Z, #0C6D, CLOSEOF1   ;byly otevřeny soubory pro zápis i pro čtení → skoč
```

Zrušíme daný prostor. V HL je začátek buferu.

```
0C65 CLOERROOM  EX   DE, HL               ;dej do HL ukazatel na začátek buferu
0C66            LD   BC, #0220            ;544 bytů
0C69            CALL #0D80, DESTRBYTE     ;zruš prostor a uprav adresy kanálů
0C6C            RET                       ;vrať se
```

Zrušíme bufer souboru pro čtení. V HL je adresa buferu.

```
0C6D CLOSEOF1   PUSH DE                   ;ulož ukazatel na začátek buferu
0C6E            CALL #0C65, CLOERROOM     ;vymaž nejdříve bufer, ve kterém byl soubor na čtení
0C71            POP  DE                   ;obnov ukazatel na začátek buferu – teď ukazuje na
                                          ;bufer souboru, který byl otevřen pro zápis
```

CLOSEOUTF

Uzavře soubor, který byl otevřen pro zápis, s uložením buferu na disk.

```
0C72 CLOSEOUTF  PUSH IX                   ;ulož si IX
0C74            EX   DE, HL               ;do HL ukazatel na hlavičku buferu
```

Nejdříve nastavíme parametry diskety a souboru.

```
0C75            INC  HL                   ;posuň se na jméno disku
0C76            INC  HL
0C77            INC  HL
0C78            INC  HL
0C79            INC  HL
0C7A            INC  HL
0C7B            LD   DE, #3E80 DNZONE1    ;do DE adresa 1. jména disku pro I/O
0C7E            LD   BC, #000A            ;10 znaků
0C81            LDIR                      ;přesuň jméno disku, na kterém je uložen soubor, který
                                          ;byl otevřen pro zápis
0C83            INC  HL                   ;posuň se na uložení sektoru a stopy adresáře, ve kterém
0C84            INC  HL                   ;je uložena hlavička souboru
0C85            CALL #1C8F, SETACT        ;roztoč mechaniku, která má stejné jméno jako
                                          ;v DNZONE1
0C88            CALL #0DB0, LDDE(HL)      ;vyzvedni sektor a stopu adresáře do DE
0C8B            LD   C, E                 ;a zkopíruj je do BC
0C8C            LD   B, D
```

```
0C8D    LD   (#3E6F), DE ADRSCTR ;ulož jako číslo sektoru a stopy adresáře, který byl
                                 ;naposledy načten z disku
0C91    LD   A, (#3E6B) WORKDR   ;do A drive, se kterým se pracuje
0C94    LD   (#3E71), A ADRDR    ;ulož jako drive, ze kterého se naposledy četl adresář
0C97    LD   DE, #0101           ;jeden sektor, žádné opakování
0C9A    PUSH HL                  ;ulož si ukazatel na hlavičku buferu
0C9B    LD   HL, #3800 DIRBUF    ;do HL adresa uložení buferu adresáře
0C9E    CALL #22A2, BREADA       ;načti sektor
0CA1    POP  HL                  ;obnov si ukazatel na hlavičku buferu
0CA2    CALL #0DB0, LDDE(HL)     ;vyzvedni adresu položky v buferu adresáře do DE
0CA5    PUSH DE                  ;a ulož si ji
0CA6    POP  IX                  ;do IX
```

Vypočteme délku souboru.

```
0CA8    CALL #0DB0, LDDE(HL)     ;vyzvedni první dva byty délky souboru z buferu do DE
0CAB    PUSH DE                  ;a ulož si je
0CAC    LD   C, (HL)             ;vyzvedni třetí byte délky souboru z buferu do C
0CAD    INC  HL                  ;posuň se na další byte hlavičky buferu
0CAE    CALL #0DB0, LDDE(HL)     ;vyzvedni číslo naposledy zapisovaného sektoru do DE
0CB1    EX   DE, HL              ;a dej ho do HL
0CB2    EX   (SP), HL            ;ulož si ho na zásobník a vyzvedni první dva byty délky
                                 ;souboru ze zásobníku
0CB3    PUSH HL                  ;a znovu je ulož
0CB4    EX   DE, HL              ;do HL vrať zpět ukazatel na hlavičku buferu
0CB5    CALL #0DB0, LDDE(HL)     ;vyzvedni počet zapsaných znaků v buferu do DE
0CB8    LD   (IX+#0B), E         ;ulož do prvního bytu délky souboru nižší byte počtu
                                 ;zapsaných znaků
0CBB    POP  AF                  ;vyzvedni druhý byte délky souboru, první je vždy nula
0CBC    ADD  A, D                ;přičti vyšší byte počtu zapsaných znaků
0CBD    LD   (IX+#0C), A ;a ulož jako druhý byte délky souboru
0CC0    LD   A, C                ;do A dej třetí byte délky souboru
0CC1    ADC  A, #00              ;a přičti příznak přetečení
0CC3    LD   (IX+#15), A         ;ulož jako třetí byte délky souboru
```

Zapíšeme hlavičku souboru na disk.

```
0CC6    CALL #1E65, WSCADR       ;zapiš sektor adresáře
0CC9    CALL #21A1, DRVSYS       ;zjisti adresu parametrů drivu, se kterým se pracuje
```

Nyní provedeme zápis posledního sektoru. Musíme rozlišit dvě možnosti. Jestli už byl do souboru zapsán nějaký sektor (stezka obsahuje koncovou značku) nebo nebyl zapsán ještě žádný sektor (soubor je nastaven jako soubor s nulovou délkou).

```
0CCC    LD   A, D                ;byl zapsán nějaký znak do buferu?
0CCD    OR   E
0CCE    EX   (SP), HL            ;vyzvedni číslo naposledy zapisovaného sektoru buferu
                                 ;a ulož si ukazatel na hlavičku
0CCF    JR   Z, #0D03, CLOSEEND  ;ne → skoč na zrušení prostoru, protože se nic nezapisuje
0CD1    PUSH DE                  ;ulož si počet zapsaných znaků v buferu
0CD2    CALL #1CF1, GETWTEST     ;načti položku FAT a otestuj chybu ve FAT
0CD5    LD   A, D                ;do A vyšší byte obsahu
0CD6    CP   #0C                 ;byl vůbec zapsán nějaký sektor souboru?
0CD8    JR   Z, #0CE7, CLEMPTYF  ;ne → skoč na vytvoření koncové značky
```

Do souboru byl zapsán sektor, má tedy koncovou značku. Najdeme další sektor, který vložíme do stezky souboru.

```
0CDA            PUSH HL                 ;ulož si číslo sektoru
0CDB            CALL #20F6, FIEMPTYFAT  ;najdi první prázdnou položku FAT od HL
0CDE            JP   NZ, #20BC, RETREP  ;nenalezena → skoč na REPORT U „Disk full“
0CE1            POP  DE                 ;obnov číslo naposledy zapisovaného sektoru
0CE2            EX   DE, HL             ;a přehoď sektory
0CE3            CALL #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE
0CE6            EX   DE, HL             ;a znovu je přehoď
```

Vytvoříme koncovou značku. V HL je číslo posledního sektoru, do kterého se zapíše obsah buferu.

```
0CE7 CLEMPTYF   POP  DE                 ;obnov si počet zapsaných znaků v buferu
0CE8            LD   A, D               ;do A vyšší byte
0CE9            OR   #0E                ;vytvoř značku konec souboru
0CEB            LD   D, A               ;dej ji zpět do D
0CEC            CALL #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE
0CEF            CALL #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
0CF2            CALL #1DF9, LOGFYZ      ;převeď logický sektor na fyzický sektor a stopu
0CF5            POP  HL                 ;obnov ukazatel na bufer
0CF6            PUSH HL                 ;a znovu ho ulož
0CF7            INC  HL                 ;posuň se na začátek dat
0CF8            INC  HL
0CF9            INC  HL
```

Uložíme obsah buferu do posledního sektoru souboru.

```
0CFA            LD   DE, #0100          ;jeden sektor, 255 opakování
0CFD            LD   A, (#3E6B) WORKDR  ;do A drive, se kterým se pracuje
0D00            CALL #2296, BWRITE      ;zapiš sektor
```

Zrušíme prostor pro bufer souboru otevřeného pro zápis.

```
0D03 CLOSEEND   POP  HL                 ;obnov ukazatel do hlavičky buferu
0D04            LD   DE, #FFE3          ;do DE –29
0D07            ADD  HL, DE             ;odečti od HL – v HL je adresa začátku hlavičky buferu
0D08            LD   BC, #0220          ;délka je 544 bytů
0D0B            CALL #0D80, DESTRBYTE   ;zruš prostor a uprav adresy kanálů
0D0E            CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
0D11            POP  IX                 ;obnov IX
0D13            RET                     ;vrať se přes RETRUN do ZX ROM
```

ANALOPENNM

Vyzvedne řetězec ze zásobníku, zkontroluje, jestli jeho délka není nulová a rozdělí ho na jméno disku a souboru

IN: na zásobníku kalkulátoru je řetězec
OUT: NZ – v DNZONE1 je jméno disku a v FNZONE1 je jméno souboru

```
0D14 ANAOPENM   RST  #28                 ;volej podprogram pro volání rutiny ZX ROM
0D15            DW   #2BF1              ;podprogram STK-FETCH vyzvednutí parametrů řetězce
0D17            LD   A, B               ;je prázdný?
0D18            OR   C
0D19            JP   Z, #1019, REPORTF  ;ano → skoč na REPORT F „Invalid file name“
0D1C            CALL #, 0FB2, ANALSTE   ;rozděl řetězec na zásobníku na jméno souboru a disku
0D1F            AND  A                  ;nastav NZ
0D20            RET                     ;vrať se
```

SETSTRBUF

Vypočte relativní adresu buferu pro otevíraný kanál a a uloží ho do informací o otevíraném kanále.

IN: HL začátek buferu
na VALSYX je uložena adresa, kde jsou uloženy informace o otevíraném kanálu

OUT: DE relativní adresa buferu
offset je uložen do informacích o kanále

```
0D21            LD   DE, (#5C4F) CHANS   ;do DE adresa, kde jsou uloženy informace o kanálech
0D25            AND  A                   ;nuluj CY
0D26            SBC  HL, DE              ;odečti adresu začátku kanálových dat od adresy
                                         ;začatku prostoru
0D28            INC  HL                  ;zvyš o jedničku
0D29            EX   DE, HL              ;a ulož rozdíl do DE
0D2A            LD   HL, (#3E78) VALSYX  ;do HL adresa kanálu proudových dat pro otevřený kanál
0D2D            LD   (HL), E             ;ulož si sem relativní posun na začátek buferu pro daný
0D2E            INC  HL                  ;kanál
0D2F            LD   (HL), D
0D30            RET                      ;vrať se
```

SETEMPTYFIL

Vytvoří soubor s nulovou délkou

IN: ve FNZONE1 je jméno souboru
IX adresa parametrů drivu

OUT: je vytvořen soubor se jménem ve FNZONE1 s nulovou délkou

```
0D31 SETEMPTYF  CALL #202C, FINDANDFILL  ;najdi první neprázdnou položku adresáře a dej do ní
                                         ;jméno souboru z FNZONE1
```

Vynulujeme informace o délce a počáteční adrese souboru

```
0D34            LD   B, #06              ;6 bytů
0D36 CLSHEADINF LD   (HL), #00           ;nastav délku souboru, počáteční adresu a délku
                                         ;proměnných na nulu
0D38            INC  HL                  ;posuň se na další byte položky adresáře
0D39            DJNZ #0D36, CLSHEADINF   ;opakuj B-krát
```

Najdeme první sektor a označíme ho jako sektor souboru s nulovou délkou.

```
0D3B            PUSH HL                  ;ulož si ukazatel do položky adresáře v buferu
0D3C            LD   HL, #0000           ;budeme prohledávat FAT od začátku
0D3F            CALL #20F6, FIEMPTYFAT   ;najdi první prázdnou položku FAT od HL
0D42            JP   NZ, #20BC, RETREP   ;nenalezena → skoč na REPORT U „Disk full“
0D45            EX   DE, HL              ;dej číslo volného sektoru do DE
0D46            POP  HL                  ;obnov ukazatel do položky adresáře do HL
0D47            LD   (HL), E             ;a ulož nalezený sektor jako číslo prvního sektoru
                                         ;souboru
0D48            INC  HL
0D49            LD   (HL), D
0D4A            INC  HL
0D4B            PUSH HL                  ;ulož si ukazatel do položku adresáře v buferu
0D4C            EX   DE, HL              ;do HL číslo prvního sektoru
0D4D            LD   DE, #0C00           ;do DE příznak délka souboru nulová
```

```
0D50            CALL #1D1E, WRTOFAT     ;zapiš do položky FAT v HL obsah DE
0D53            POP  HL                 ;obnov ukazatel do položky adresáře do HL
0D54            LD   (HL), #00          ;ulož si do hlavičky nulu
0D56            INC  HL                 ;a posuň se na další byte
0D57            LD   (HL), #0F          ;ulož atributy RWED
0D59            INC  HL                 ;posuň se na další byte
0D5A            LD   (HL), #00          ;ulož do informace o délce souboru nulu
0D5C            LD   DE, #FFEB          ;do DE dej –21
0D5F            ADD  HL, DE             ;odečti od HL
0D60            CALL #1E65, WSCADR      ;zapiš sektor adresáře v buferu
0D63            CALL #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
0D66            RET                     ;vrať se
```

LD(HL)DE

Uloží na (HL) registr E a (HL+1) registr D.

IN: DE ukládaná hodnota
HL adresa, kam se bude ukládat
OUT: obsah DE se uloží na (HL) a (HL+1) a obsah HL je zvětšen o 2

```
0D67 LD(HL)DE   LD   (HL), E            ;ulož E
0D68            INC  HL                 ;posuň se na další adresu
0D69            LD   (HL), D            ;ulož D
0D6A            INC  HL                 ;posuň se na další adresu
0D6B            RET                     ;vrať se
```

LD(HL)A

Uloží na (HL) registr A.

IN: A ukládaná hodnota
HL adresa, kam se bude ukládat
OUT: obsah A se uloží na (HL) a obsah HL je zvětšen o 1

```
0D6C LD(HL)A    LD   (HL), A            ;ulož A
0D6D            INC  HL                 ;posuň se na další adresu
0D6E            RET                     ;vrať se
```

MAKE544B

Vytvoří prostor 544 bytů za systémovými proměnnými BASICu před BASIC programem.

IN: –
OUT: vytvořeno 544 bytů volného místa
HL adresa začátku volného místa

```
0D6F MAKE544B   LD   BC, #0220          ;prostor 544 bytů
0D72            JR   #0D77, MAKEROOM    ;skoč na vytvoření prostoru
```

MAKE1088B

Vytvoří prostor 1088 bytů za systémovými proměnnými BASICu před BASIC programem.

IN: –
OUT: vytvořeno 1088 bytů volného místa
HL adresa začátku volného místa

```
0D74 MAKE1044B  LD   BC, #0440          ;prostor 1088 bytů
```

Vytvoříme prostor. V registru BC je počet volných bytů.

```
0D77 MAKEROOM LD   HL, (#5C53) PROG   ;do HL začátek programu v BASICu
0D7A          DEC  HL                 ;sniž o jedničku
0D7B          RST  #28                ;volej podprogram pro volání rutiny ZX ROM
0D7C          DW   #1655              ;podprogram MAKE-ROOM vytvoření prostoru BC bytů
0D7E          INC  HL                 ;posuň se na první byte prostoru
0D7F          RET                     ;vrať se
```

DESTRBYTE

Zruší prostor BC bytů a nastaví kanálové informace. Používá se např. při rušení buferu souboru připojeného na kanál. Pokud je takový bufer rušen, musí se změnit relativní adresy všech bufer, které jsou uloženy za rušeným buferem (relativní adresa se zmenší o 544 bytů). Je to důležité, protože pokud by se nezměnily, ukazatele na tyto bufery v kanálových informacích by byly neplatné (ukazovaly by jinam).

IN: HL adresa začátku rušeného prostoru
BC délka rušeného prostoru
OUT: zruší daný prostor a nastaví kanálové informace

```
0D80 DESTRBYTE PUSH HL                 ;ulož si ukazatel na začátek prostoru
0D81           PUSH BC                 ;ulož si počet bytů
0D82           RST  #28                ;volej podprogram pro volání rutiny ZX ROM
0D83           DW   #19E8              ;podprogram RECLAIM-2 zrušení prostoru BC bytů
0D85           POP  BC                 ;obnov počet bytů
0D86           POP  HL                 ;a ukazatel na začátek prostoru
0D87           LD   DE, (#5C4F) CHANS  ;do DE adresa, kde jsou uloženy informace o kanálech
0D8B           AND  A                  ;nuluj CY
0D8C           SBC  HL, DE             ;odečti – v HL je relativní adresa zrušeného prostoru
0D8E           INC  HL                 ;zvyš o 1
0D8F           PUSH HL                 ;ulož si relativní adresu
0D90           LD   HL, #5C10 STRMS    ;do HL adresy kanálů
0D93           LD   A, #13             ;19 kanálů
```

Posuneme všechny relativní adresy buferů, které byly uloženy za zrušeným prostorem (došlo ke smazání prostoru).

```
0D95 CRECTADR  CALL #0DB0, LDDE(HL)    ;vyzvedni relat. adresu buferu do DE
0D98           EX   (SP), HL           ;do HL relat. adresu začátku prostoru
0D99           SBC  HL, DE             ;odečti délku buferu
0D9B           ADD  HL, DE             ;a znovu ho přičti
0D9C           JR   NC, #0DAA, NOCRECT ;je menší → skoč na další kanál
0D9E           EX   DE, HL             ;dej relat. adresu prostoru do DE a relat. adr. buferu do HL
```

Protože daný kanál ukazuje na bufer, který se nachází za rušeným prostorem, musíme snížit relativní adresu buferu.

```
0D9F           AND  A                  ;nuluj CY
0DA0           SBC  HL, BC             ;odečti délku prostoru
0DA2           EX   DE, HL             ;vrať zpět adresy
0DA3           EX   (SP), HL           ;do HL zpět adresu kanálů a ulož relat. adresu prostoru
0DA4           DEC  HL                 ;dej do původního stavu
0DA5           DEC  HL
0DA6           CALL #0D67, LD(HL)DE    ;a ulož novou relativní adresu buferu
0DA9           EX   (SP), HL           ;do HL dej relativní adresu prostoru a ulož adresu kanálů
0DAA NOCRECT   EX   (SP), HL           ;do HL adresu kanálů
```

```
0DAB            DEC  A                    ;už byly zkontrolovány všechny kanály?
ODAC            JR   NZ, #0D95, CRECTADR ;ne → opakuj
0DAE            POP  HL                   ;obnov si ukazatel na začátek prostoru
0DAF            RET                       ;vrať se
```

LDDE(HL)

Vyzvedne DE z (HL) a (HL+1).

IN: HL adresa, odkud se bude vyzvedávat
OUT: DE v E je obsah z (HL) a v D obsah z (HL+1)
HL obsah HL je zvětšen o 2

```
0DB0 LDDE(HL)   LD   E, (HL)              ;vem obsah adresy do E
0DB1            INC  HL                   ;posuň se na další adresu
0DB2            LD   D, (HL)              ;vem obsah adresy do D
0DB3            INC  HL                   ;posuň se na další adresu
0DB4            RET                       ;vrať se
```

SETSTRNM

Analyzuje jméno souboru a disku pro otevření sekvenčního souboru a otestuje příponu pro sekvenční soubor. Provádí se také test na wildchars a pokud byly použity, hlásí se chyba.

IN: jméno disku v DNZONE1 a jméno souboru ve FNZONE1
OUT: nastavené jméno disku a jméno souboru na masku, kontrola na přípony „B“ a „Q“

```
0DB5 SETSTRNM   CALL #1043, SETWDNM       ;nastav jméno disku v DNZONE1 pro I/O
0DB8            CALL #10E2, ANALWDNM      ;analyzuj jméno disku v DNZONE1
0DBB            INC  A                    ;bylo vloženo chybné jméno disku?
0DBC            JP   Z, #2337, REPORTX    ;ano → skoč na REPORT X „Bad device type“
0DBF            CALL #107C, ARRANGNM      ;uprav jméno souboru v FNZONE1 na masku
0DC2            JP   C, #1019, REPORTF    ;bylo použito wildchars → skoč na REPORT F „Invalid
                                          ;file name“
0DC5            LD   A, (#3E94) EXTE1     ;vyzvedni příponu souboru
0DC8            CP   „B“                  ;je to „B“?
0DCA            RET  Z                    ;ano → vrať se
0DCB            CP   „Q“                  ;je to „Q“?
0DCD            RET  Z                    ;ano → vrať se
0DCE            CP   „?“                  ;je to „?“ (přípona nebyla zadána)?
0DD0            JP   NZ, #1019, REPORTF   ;ne → skoč na REPORT F „Invalid file name“
```

Přípona nebyla určena, nastavíme ji tedy na „Q“.

```
0DD3            LD   A, „Q“               ;přípona bude „Q“
0DD5            LD   (#3E94), A EXTE1     ;ulož příponu souboru
0DD8            RET                       ;vrať se
```

READFROMSTR

Rutina pro načtení znaku ze sekvenčního souboru do A. Je volána z adres #22C2 a #27A5.

IN: DE adresa začátku hlavičky buferu
OUT: A znak ze sekvenčního souboru, který je připojen na aktuální kanál

```
0DD9 RDFROMSTR  EI                        ;povol přerušení
0DDA            RES  3, (IY+#02) TVFLAG   ;zruš režim edit
```

Do registru BC vyzvedneme adresu, kam se vrací program po provedení podprogramu WAIT-KEY (načtení znaku z aktuálního kanálu). Je to proto, aby po načtení znaku z připojeného souboru nedošlo k pípnutí jako po stisknutí klávesy, pokud se WAIT-KEY volá z procedury EDITOR, obsluhující příkaz INPUT.

```
0DDE            LD   HL, (#5C3D) ERR_SP   ;do HL adresa položky na zásobníku při chybě
0DE1            DEC  HL                   ;sniž ji
0DE2            LD   B, (HL)              ;vyzvedni horní byte do B
0DE3            DEC  HL                   ;sniž ji
0DE4            PUSH HL                   ;a ulož na zásobník
0DE5            LD   C, (HL)              ;vyzvedni dolní byte do C – v BC je návratová adresa,
                                          ;kam se vrátí řízení z podprogramu WAIT-KEY
0DE6            LD   HL, #0F3B            ;návratová adresa po načtení znaku do podprogramu
                                          ;EDITOR
0DE9            AND  A                    ;nuluj CY
0DEA            SBC  HL, BC               ;porovnej s adresou na zásobníku
0DEC            POP  HL                   ;obnov ukazatel na adresu v zásobníku
0DED            JR   NZ, #0DF1, NOINPUT   ;nebyly stejné → skoč
0DEF            LD   (HL), #48            ;změň návratovou adresu do podprogramu EDITOR
                                          ;(přeskoč pípnutí klávesnice)
```

Nyní provedeme načtení znaku ze souboru.

```
0DF1 NOINPUT    LD   HL, #0018            ;do HL 24
0DF4            ADD  HL, DE               ;přičti k ukazateli na hlavičku, v HL je teď ukazatel na
                                          ;počet znaků v buferu, kolik jich lze ještě číst
0DF5            LD   E, (HL)              ;vyzvedni do DE počet znaků ke čtení
0DF6            INC  HL
0DF7            LD   D, (HL)
0DF8            INC  HL
0DF9            LD   C, (HL)              ;a do BC aktuální pozici v buferu
0DFA            INC  HL
0DFB            LD   B, (HL)
0DFC            LD   A, D                 ;je ještě nějaký znak ke čtení?
0DFD            OR   E
0DFE            JR   NZ, #0E10, NOEMPTBF  ;ano → skoč
```

Protože byl bufer přečten, musíme načíst další sektor souboru.

```
0E00            PUSH HL                   ;ulož si ukazatel na hlavičku
0E01            CALL #0E42, STRRDNSEC     ;načti další sektor souboru do buferu
0E04            POP  HL                   ;obnov ukazatel na hlavičku
0E05            JR   C, #0E0C, STRNEXT    ;není konec souboru → skoč
0E07            OR   #FF                  ;nastav NC, NZ – REPORT 8 „End of file“
0E09            JP   #1700, STANDROM      ;skoč na přestránkování a návrat do ZX ROM
```

Nastavíme aktuální pozici v buferu na začátek buferu.

```
0E0C STRNEXT    LD   C, L                 ;zkopíruj ukazatel na hlavičku do BC
0E0D            LD   B, H
0E0E            INC  BC                   ;a posuň ho v BC na začátek dat
0E0F            INC  BC
```

Snížíme počet bytů a načteme znak.

```
0E10 NOEMPTBF   DEC  DE                   ;sniž počet znaků ke čtení
0E11            LD   A, (BC)              ;vyzvedni znak z buferu
```

Uložíme ukazatele do hlavičky buferu a provedeme návrat do ZX ROM.

```
0E12 INCPOINTERS INC   BC                  ;posuň ukazatel na další znak
0E13             LD    (HL), B             ;a ulož si ho do hlavičky
0E14             DEC   HL
0E15             LD    (HL), C
0E16             DEC   HL
0E17             LD    (HL), D             ;taky si ulož počet znaků ke čtení
0E18             DEC   HL
0E19             LD    (HL), E
0E1A             SCF                       ;nastav C
0E1B             JP    #1700, STANDROM     ;skoč na přestránkování do ZX ROM
```

WRITETOSTR

Rutina pro zápis znaku do kanálu. Zde se vstupuje, pokud je na kanál připojen soubor pro čtení i pro zápis. Je volána z adresy #27AC.

IN: DE adresa začátku hlavičky buferu
A zapisovaný znak

```
0E1E WRTOSTR2    LD    HL, #023A           ;nastav posun na druhý bufer
0E21             JR    #0E26, WRTOSTR      ;skoč do společné části
```

Stejné jako u WRTOSTR2, ale s tím rozdílem, že na kanál je připojen pouze soubor pro zápis. Je volána z adresy #25AB.

```
0E23 WRTOSTR1    LD    HL, #001A           ;nastav se na první bufer (druhý není)
```

Toto je společná část zápisu znaku do sekvenčního souboru.

```
0E26 WRTOSTR     EI                        ;povol preřušení
0E27             ADD   HL, DE              ;nastav ukazatel do hlavičky patřičného buferu
0E28             LD    E, (HL)             ;vyzvedni počet zapsaných znaků do DE
0E29             INC   HL
0E2A             LD    D, (HL)
0E2B             INC   HL
0E2C             LD    C, (HL)             ;a aktuální pozici v buferu do BC
0E2D             INC   HL
0E2E             LD    B, (HL)
0E2F             LD    (BC), A             ;ulož A do buferu na (BC)
0E30             INC   DE                  ;zvyš počítadlo zapsaných znaků
0E31             BIT   1, D                ;už bylo zapsáno 512 znaků?
0E33             JR    Z, #0E12, NOFULLBUF ;ne → skoč
```

Protože je bufer plný, musíme ho zapsat na disk.

```
0E35             PUSH  HL                  ;ulož si ukazatel do hlavičky buferu
0E36             CALL  #0E9B, WFLSTRSC     ;zapiš vyplněný bufer na disk a najdi další volný sektor
0E39             POP   HL                  ;obnov si ukazatel do hlavičky
0E3A             LD    C, L                ;zkopíruj ho do BC
0E3B             LD    B, H
0E3C             INC   BC                  ;posuň se na adresu, kde je uložen počet zapsaných
znaků
0E3D             LD    DE, #0000           ;do DE ulož nulu (žádný znak zapsán v buferu)
0E40             JR    #0E12, INCPOINTERS  ;skoč na nastavení ukazatelů a jejich uložení
```

STRRDNSEC

Načte další sektor souboru do buferu. Vrací C, pokud ještě takový je.

IN: HL adresa uložení 2. byte aktuální pozice v buferu v hlavičce
OUT: C ještě není konec souboru
DE délka dat v načteném sektoru
do buferu je načten další sektor souboru

```
0E42 STRRDNSEC  PUSH IX                 ;ulož si IX
0E44            INC  HL                 ;posuň se na začátek dat v buferu
0E45            INC  HL
0E46            PUSH HL                 ;ulož si začátek dat v buferu
0E47            CALL #0E86, STRDRNMSC   ;nastav z hlavičky jméno disku a vyzvedni číslo
                                        ;naposledy čteného sektoru
0E4A            PUSH HL                 ;ulož si ukazatel na hlavičku buferu – ukazuje na číslo
                                        ;naposledy čteného sektoru
0E4B            EX   DE, HL             ;dej do HL číslo naposledy čteného sektoru
0E4C            BIT  3, H               ;poslední sektor?
0E4E            JR   NZ, #0E80, ISLAST  ;ano → skoč
```

Načteme další sektor souboru.

```
0E50            CALL #1C8F, SETACT      ;roztoč mechaniku, která má stejné jméno jako
                                        ;v DNZONE1
0E53            CALL #1CF1, GETWTEST    ;načti položku FAT a otestuj chybu
0E56            CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
0E59            CALL #1DF9, LOGFYZ      ;převeď logicky sektor na fyzický sektor a stopu
0E5C            POP  HL                 ;obnov ukazatel na hlavičku buferu
0E5D            CALL #0D67, LD(HL)DE    ;ulož číslo nového načteného sektoru v buferu
0E60            LD   A, D               ;do A vyšší číslo sektoru
0E61            CP   #0C                ;je to sektor souboru s nulovou délkou?
0E63            JR   Z, #0E81, ISEMPTY  ;ano → skoč
0E65            BIT  3, D               ;je to poslední sektor souboru?
0E67            JR   Z, #0E70, IS512B   ;ne → skoč
0E69            LD   A, D               ;do A znovu vyšší číslo sektoru
0E6A            AND  #01                ;ponech 0. bit
0E6C            LD   D, A               ;a vrať zpět do D – v DE je teď délka dat v posledním
                                        ;sektoru
0E6D            OR   E                  ;je 512 bytů?
OE6E            JR   NZ, #0E73, NO512B  ;ne → skoč
```

Délka dat v sektoru je 512 bytů.

```
0E70 IS512B     LD   DE, #0200          ;délka načtených dat v sektoru je 512 bytů
```

Provedeme načtení sektoru do buferu.

```
0E73 NO512B     POP  HL                 ;obnov ukazatel na začátek dat v buferu
0E74            PUSH DE                 ;ulož si délku dat
0E75            LD   DE, #0100          ;jeden sektor, 255 opakování
0E78            CALL #22A2, BREADA      ;načti sektor
0E7B            POP  DE                 ;obnov registry
0E7C            POP  IX
0E7E            SCF                     ;nastav C
0E7F            RET                     ;vrať se
```

Už není žádný sektor ke čtení.

```
0E80 ISLAST     POP  HL                 ;obnov ukazatel do hlavičky buferu
```

Návrat s tím, že byl překročen konec souboru.

```
0E81 ISEMPTY    POP  HL                 ;obnov registry
0E82            POP  IX
0E84            AND  A                  ;nastav NC
0E85            RET                     ;vrať se
```

STRDRNMSC

Nastaví jméno disku, na kterém se nachází soubor připojený ke kanálu a vrací v DE číslo naposledy čteného nebo zapsaného sektoru pro zápis do buferu.

IN: HL adresa počátku uložení dat v buferu
OUT: v DNZONE1 je jméno disku, na kterém je uložen daný soubor
DE číslo naposledy čteného nebo zapsaného sektoru
HL adr. uložení naposledy čteného nebo zapsaného sektoru v hlavičce buferu

```
0E86 STRDRNMSC  LD   DE, #FFE6          ;do DE dej –26
0E89            ADD  HL, DE             ;odečti od HL – v HL je adresa uložení jména disku
0E8A            LD   DE, #3E80 DNZONE1  ;do DE adresa 1.jména disku pro I/O
0E8D            LD   BC, #000A          ;10 znaků
0E90            LDIR                    ;přesuň jméno disku, na kterém je uložen soubor
                                        ;připojený ke kanálu
0E92            LD   DE, #0009          ;do DE 9
0E95            ADD  HL, DE             ;přičti zpět k HL
0E96            LD   E, (HL)            ;do DE vyzvedni číslo sektoru, který byl naposledy
0E97            INC  HL                 ;načten do buferu nebo zapsán na disketu
0E98            LD   D, (HL)
0E99            DEC  HL                 ;v HL je ukazatel na číslo sektoru v hlavičce buferu
0E9A            RET                     ;vrať se
```

WFULLSTRSC

Zapíše vyplněný bufer do sektoru na disk a najde další prázdný sektor.

IN: HL adresa uložení 2. bytu aktuální pozice v buferu
OUT: zapíše bufer na disk do souboru a najde další volný sektor

```
0E9B WFLSTRSC   PUSH IX                 ;ulož si IX
0E9D            INC  HL                 ;posuň se na začátek dat v buferu
0E9E            INC  HL
0E9F            PUSH HL                 ;a ulož si ukazatel na začátek dat
0EA0            CALL #0E86, STRDRNMSC   ;nastav jméno disku a vyzvedni číslo naposledy
                                        ;zapsaného sektoru
0EA3            LD   (#3E78), HL VALSYX ;ulož si ukazatel na hlavičku buferu (je v ní adresa
                                        ;uložení naposledy zapsaného sektoru)
0EA6            PUSH DE                 ;ulož si číslo naposledy zapsaného sektoru
0EA7            CALL #1C8F, SETACT      ;roztoč mechaniku, která má stejné jméno jako
                                        ;v DNZONE1
0EAA            POP  HL                 ;do HL vyzvedni číslo naposledy zapsaného sektoru
```

Nyní otestujeme, jestli už byl zapisován nějaký sektor do souboru nebo je to soubor s nulovou délkou.

```
0EAB            CALL #1CF1, GETWTEST    ;načti položku FAT a otestuj chybu
```

```
0EAE            LD    A, D                  ;do A dej vyšší byte obsahu položky
0EAF            CP    #0C                   ;byl už zapsán nějaký sektor?
0EB1            JR    Z, #0EC6, NULENGTH    ;ne → skoč
```

Najdeme tedy nový volný sektor, který přidáme na konec stezky souboru, zapíšeme do něj obsah buferu a uložíme ho jako poslední zapsaný sektor.

```
0EB3            PUSH  HL                    ;ulož si číslo naposledy zapsaného sektoru
0EB4            CALL  #20F6, FIEMPTYFAT     ;najdi prázdnou položku FAT od HL
0EB7            JP    NZ, #20BC, RETREP     ;nenalezena → skoč na REPORT U „Disk full“
0EBA            POP   DE                    ;do DE číslo naposledy zapsaného sektoru
```

Vložíme nalezený nový sektor do stezky souboru.

```
0EBB            EX    DE, HL                ;do HL dej číslo naposledy zapsaného sektoru a do DE
                                            ;číslo nalezeného sektoru
0EBC            CALL  #1D1E, WRTOFAT        ;zapiš do položky FAT obsah DE
0EBF            LD    HL, (#3E78) VALSYX    ;vyzvedni ukazatel do hlavičky buferu
0EC2            CALL  #0D67, LD(HL)DE       ;ulož si nové číslo naposledy zapsaného sektoru
0EC5            EX    DE, HL                ;dej do HL číslo nového sektoru
```

Vytvoříme koncovou značku souboru.

```
0EC6 NULENGTH   LD    DE, #0E00             ;do DE značka konec souboru
0EC9            CALL  #1D1E, WRTOFAT        ;zapiš do položky FAT obsah DE
0ECC            CALL  #1DF9, LOGFYZ         ;převeď logický sektor na fyzický sektor a stopu
```

Zapíšeme bufer do sektoru a stopy v BC.

```
0ECF            POP   HL                    ;obnov ukazatel na začátek dat
0ED0            PUSH  HL                    ;a znovu si ho ulož
0ED1            LD    DE, #0100             ;jeden sektor, 255 opakování
0ED4            LD    A, (#3E6B) WORKDR     ;do A drive, se kterým se pracuje
0ED7            CALL  #2296, BWRITE         ;zapiš sektor
0EDA            POP   HL                    ;obnov ukazatel na začátek dat
```

Zvětšíme délku souboru o 512 bytů.

```
0EDB            LD    DE, #FFF7             ;do DE dej –9
0EDE            ADD   HL, DE                ;odečti od HL – v HL je adresa uložení 2. byte délky
                                            ;souboru
0EDF            LD    A, (HL)               ;vyzvedni druhý byte
0EE0            ADD   A, #02                ;přičti k němu 2 (512 bytů)
0EE2            LD    (HL), A               ;a ulož ho zpět
0EE3            JR    NC, #0EEB, STRWOK     ;pokud nedošlo k posunu → skoč
0EE5            INC   HL                    ;posuň se na další byte
0EE6            INC   (HL)                  ;a zvyš obsah o 1
0EE7            JR    NZ, #0EEB, STRWOK     ;nedošlo k posunu → skoč
```

A nyní je chyba, která se však nikdy neprojeví. Nedovolí nám to kapacita diskety. Jde tady o zvýšení 4. bytu délky souboru, který však není k dispozici (tam už je uloženo číslo naposledy zapsaného sektoru). To by však šlo pouze za předpokladu, že na disketě je více než 32768 sektorů a pokud by jsme vytvořili soubor, který by zabíral více než oněch 32768 sektorů. To však nejde.

```
0EE9            INC   HL                    ;posuň se na čtvrtý byte
0EEA            INC   (HL)                  ;a zvyš jeho obsah o 1
```

Zapíšeme změny FAT, inicialicujeme proměnné MDOSu a provedeme návrat.

```
0EEB STRWOK     CALL #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
0EEE            CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
0EF1            POP  IX                 ;obnov IX
0EF3            RET                     ;vrať se
```

CLOSEZEROSTR

Uzavře kanál 0. Tento podprogram není odnikud volán, takže ho můžete používat jen jako rutinu MDOSu.

```
0EF4 CLOSE0STR  XOR  A                  ;kanál 0
0EF5            JP   #0C24, CLOUTSTRF   ;uzavři kanál
```

ROMDRPAR

Tabulka základních parametrů disků A: a B: (24 bytů) přesunovaná do SRAM na DRPARZN.

Disk A:

```
0EF8 ROMDRPAR   DB   00 00 00 00 00     ;parametry diskety v A: jsou při inicializaci prázdné
0EFD            DB   #18                ;%00011000 – parametry mechaniky (6. byte)
0EFE            DB   #28                ;40-ti stopá mechanika
0EFF            DB   #09                ;9 sektorů na stopu
0F00            DB   00 00 00 00        ;zbývající byty, nevyužité
```

Disk B:

```
0F04            DB   00 00 00 00 00     ;parametry diskety v B: jsou při inicializaci prázdné
0F09            DB   #14                ;%00010100 – parametry mechaniky (6. byte)
0F0A            DB   #50                ;80-ti stopá mechanika
0F0B            DB   #09                ;9 sektorů na stopu
0F0C            DB   00 00 00 00        ;zbývající byty, nevyužité
```

TXTSDOS

Text SDOS.

```
0F10 TXTSDOS    DB   53 44 4F 53        ;SDOS
```

NUM24B

Převede 24 bitové číslo na řetězec a vytiskne ho na obrazovku.

IN: IX adresa uložení čísla (3 byty) a pomocné proměnné (další 3 byty) (v MDOSu to je SV24NM)

OUT: číslo je vytištěno na otevřený kanál v ASCIINM je uloženo ASCII vyjádření čísla (8 znaků)

```
0F14 NUM24B     LD   C, #08             ;bude osm znaků
0F16            LD   HL, #3EDA ASCIINM  ;do HL adresa uložení ASCII vyjádření čísla v SRAM
```

Nejdříve si vypočteme řády.

```
0F19 CALCDIV    PUSH HL                 ;ulož si ukazatel na ASCII vyjádření
```

Inicializujeme pomocné proměnné.

```
0F1A            LD   (IX+#03), #01      ;budeme začínat od jednotek
0F1E            LD   (IX+#04), #00      ;tedy #000001
0F22            LD   (IX+#05), #00
0F26            LD   A, C               ;do A počet znaků
0F27            DEC  A                  ;sniž o 1
0F28            JR   Z, #0F53, SETNUM   ;poslední znak? ano → skoč
```

```
0F2A          LD   B, A                ;ulož si počet znaků do A
0F2B          LD   A, (IX+#03)         ;vyzvedni si do „registru HLA“ řády
0F2E          LD   L, (IX+#04)
0F31          LD   H, (IX+#05)
```

Vynásobíme registr HLA deseti B-krát → dostaneme řád.

```
0F34 MULT10   ADD  A, A
0F35          ADC  HL, HL
0F37          ADD  A, A
0F38          ADC  HL, HL
0F3A          ADD  A, (IX+#03)
0F3D          LD   E, (IX+#04)
0F40          LD   D, (IX+#05)
0F43          ADC  HL, DE
0F45          ADD  A, A
0F46          ADC  HL, HL
0F48          LD   (IX+#03), A         ;a ulož si obsah „registru HLA“ do pomocné
0F4B          LD   (IX+#04), L         ;proměnné
0F4E          LD   (IX+#05), H
0F51          DJNZ #0F34, MULT10       ;opakuj B-krát
```

Nyní budeme od vypisovaného číslo odečítat řády a získáme tak číslici řádu.

```
0F53 SETNUM   POP  HL                  ;obnov si ukazatel na ASCII vyjádření
0F54          LD   (HL), #30           ;uložíme si tam znak „0“
0F56          PUSH HL                  ;uložíme si ukazatel na ASCII vyjádření
0F57 SUBNUM   LD   A, (IX+#00)         ;do „registru HLA“ si vyzvedneme vypisované číslo
0F5A          LD   L, (IX+#01)
0F5D          LD   H, (IX+#02)
0F60          SUB  (IX+#03)            ;a odečteme řády
0F63          LD   E, (IX+#04)
0F66          LD   D, (IX+#05)
0F69          SBC  HL, DE
0F6B          JR   C, #0F7B, ISLOW     ;je výsledek záporný? ano → skoč
0F6D          LD   (IX+#00), A         ;ulož si výsledek jako nové číslo
0F70          LD   (IX+#01), L
0F73          LD   (IX+#02), H
0F76          POP  HL                  ;obnov ukazatel na ASCII vyjádření čísla
0F77          INC  (HL)                ;zvyš kód znaku
0F78          PUSH HL                  ;a ulož si ukazatel na ASCII vyjádření
0F79          JR   #0F57, SUBNUM       ;skoč na další odečtení řádu
```

Získali jsme číslici řádu, posuneme se tady na další pozici v ASCII vyjádření, snížíme řád a opakujeme.

```
0F7B ISLOW    POP  HL                  ;obnov ukazatel na ASCII
0F7C          INC  HL                  ;posuň se na další znak
0F7D          DEC  C                   ;sniž počet znaků a řád
0F7E          JR   NZ, #0F19, CALCDIV  ;ještě nejsou všechny → skoč
0F80          LD   HL, #3EDA ASCIINM   ;do HL adresa začátku ASCII vyjádření čísla v SRAM
0F83          LD   B, #07              ;budeme kontrolovat 7 znaků
```

Odstraníme nevýznamné nuly na začátku čísla a nahradíme je mezerami.

```
0F85 CLSNUL   LD   A, (HL)             ;vyzvedni znak
0F86          CP   #30                 ;je to nula?
```

```
0F88            JR    NZ, #0F8F, PRNUM    ;ne → skoč
0F8A            LD    (HL), #20           ;ulož místo nuly mezeru
0F8C            INC   HL                  ;posuň se na další znak
0F8D            DJNZ  #0F85, CLSNUL       ;opakuj B-krát
```

A vytiskneme číslo na obrazovku.

```
0F8F PRNUM      LD    HL, #3EDA ASCIINM   ;do HL adresa ASCII vyjádření čísla
0F92            LD    B, #08              ;tiskni 8 znaků
0F94 PRNUM1     LD    A, (HL)             ;vyzvedni znak
0F95            PUSH  HL                  ;ulož si ukazatel
0F96            PUSH  BC                  ;a čítač
0F97            RST   #10                 ;piš znak
0F98            POP   BC                  ;obnov čítač
0F99            POP   HL                  ;a ukazatel
0F9A            INC   HL                  ;posuň se na další znak
0F9B            DJNZ  #0F94, PRNUM1       ;opakuj B-krát
0F9D            RET                       ;vrať se
```

TESTNM

Zjistí, jestli je ve FNZONE1 vloženo jméno souboru.

IN: ve FNZONE1 jméno souboru
OUT: Z – je vloženo jméno souboru ve FNZONE1

```
0F9E TESTNM     LD    A, (#3E8A) FNZONE1  ;vyzvedni první znak jména souboru
0FA1            AND   A                   ;je prázdné?
0FA2            JP    Z, #1019, REPORTF   ;ano → skoč na REPORT F „Invalid file name“
0FA5            RET                       ;vrať se
```

BCPRT

Rutina pro výpis obsahu registru BC do otevřeného kanálu.

IN: BC číslo 0 – 65535
OUT: vystup čísla BC do otevřeného kanálu

```
0FA6 BCPRT      RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0FA7            DW    #2D2B               ;podprogram STACK-BC uložení registru BC do
                                          ;zásobníku kalkulátoru
0FA9            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0FAA            DW    #2DE3               ;podprogram PRINT-FP výpis čísla ze zásobníku
                                          ;kalkulátoru
0FAC            RET                       ;vrať se
```

ADDHLA

Přičte k registru HL obsah registru A.

IN: HL 1. číslo
A 2. číslo
OUT: HL=HL+A

```
0FAD ADDHLA     ADD   A, L                ;přičti L k A
0FAE            LD    L, A                ;dej A do L
0FAF            RET   NC                  ;jestli nedošlo k přetečení, vrať se
0FB0            INC   H                   ;přičti k H jedničku
0FB1            RET                       ;vrať se
```

ANALSTE

Vymaže oblast pro uložení jména disku a souboru a pokračuje ve rozdělení řetězce, jehož parametry jsou v BC a DE, na jméno souboru a disku.

IN: DE adresa uložení řetězce
BC délka řetězce
OUT: v DNZONE1 je jméno disku, v FNZONE1 jméno souboru

```
0FB2            LD    HL, #3E80 DNZONE1   ;do HL adresa 1. jména disku pro I/O
0FB5            PUSH  BC                  ;ulož si parametry řetězce
0FB6            PUSH  DE
0FB7            LD    B, #15              ;21 bytů (jméno disku + jméno souboru)
0FB9            CALL  #104B, BNULHL       ;vymaž oblast
0FBC            POP   DE                  ;obnov parametry řetězce
0FBD            POP   BC
0FBE            JR    #0FDA, DIVSTRING1   ;pokračuj v rozdělení řetězce na jméno souboru a disku
```

DIVSTRINGCAT

Otestuje, jestli byly nějaké parametry, pokud ano, zpracuje je (rozdělí řetězec na zásobníku na jméno souboru a disku, uloží je do pracovních oblastí FNZONE1 a DNZONE1), pokud ne, nastaví jméno aktuálního disku do DNZONE1 a FNZONE1 vyplní znakem „?“.

```
0FC0 DIVSTRCAT  RST   #18                ;vezmi aktuální znak
0FC1            CP    #0D                 ;konec řádku?
0FC3            JR    Z, #1027, NOPARCAT  ;ano → skoč na vyplnění znakem „?“
0FC5            CP    „ : “               ;další příkaz?
0FC7            JR    Z, #1027, NOPARCAT  ;ano → skoč na vyplnění znakem „?“
0FC9            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0FCA            DW    #1C8C               ;podprogram EXPT-EXP vyhodnocení řetězce
                                          ;vyzvedne parametr příkazu
0FCC            CALL  #1051, TESTSYN1     ;otestuj, jestli není kontrola syntaxe
```

DIVSTRING

Tento podprogram vyzvedne parametry řetězce ze zásobníku, rozdělí ho na jméno disku a jméno souboru, které uloží do DNZONE1 a FNZONE1.

IN: parametry řetězce na zásobníku kalkulátoru
OUT: v DNZONE1 je jméno disku, v FNZONE1 jméno souboru

```
0FCF DIVSTRING  LD    HL, #3E80 DNZONE1   ;do HL adresa 1. jména disku pro I/O
0FD2            LD    B, #15              ;jméno disku a jméno souboru je celkem 21 bytů
0FD4            CALL  #104B, BNULHL       ;vymažeme tuto oblast
0FD7            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
0FD8            DW    #2BF1               ;podprogram STK-FETCH vyhodnocení řetězce
                                          ;ze zásobníku je vyzvednut řetězec, v BC je délka, v DE
                                          ;je adresa uložení
```

Vyhodnotíme řetězec, v DE je adresa uložení, v BC je délka řetězce, jméno disku se uloží do DNZONE1 a jméno souboru do FNZONE1.

```
0FDA DIVSTRING1 EX    DE, HL              ;dej do HL začátek řetězce
0FDB            INC   B                   ;zvyš horní byte délky o 1
0FDC            DEC   B                   ;je řetězec dlouhý?
0FDD            JP    NZ, #1019, REPORTF  ;ano → skoč na REPORT F „Invalid file name“
0FE0            LD    B, C                ;do B dej délku řetězce
```

```
0FE1              INC  B                    ;zvyš o jedničku
0FE2 DIVLOOP      DEC  B                    ;je konec řetězce?
0FE3              RET  Z                    ;ano → vrať se
0FE4              LD   DE, #3E80 DNZONE1    ;do DE adresa 1. jména disku pro I/O
0FE7              CALL #1064, GETNAME       ;vyzvedni první jméno po znaky „ : “, „ . “ nebo do
                                            ;konce řetězce
```

Nyní zjistíme, co bylo vybráno jako první (jméno disku nebo souboru).

```
0FEA              JR   NZ, #0FF6, MOVENAME  ;je konec řetězce → skoč
0FEC              JR   NC, #100A, NEXTANAL  ;bylo první jméno disku → skoč
```

Bylo vybráno jméno souboru.

```
0FEE              LD   A, (HL)              ;vyzvedni příponu souboru
0FEF              LD   (#3E94), A EXTE1     ;a ulož ji do jména souboru ve FNZONE1
0FF2              DEC  B                    ;sniž B o 2 (tečka a přípona)
0FF3              DEC  B                    ;jestli není konec řetezce → skoč na REPORT F „Invalid
0FF4              JR   NZ, #1019, REPORTF   ;file name“, protože za jménem souboru jsou ještě znaky
```

Protože nebylo vloženo jméno disku a jako první bylo vybráno jméno souboru, které je uloženo ve DNZONE1, je třeba ho přesunout do FNZONE1.

```
0FF6 MOVENAME     LD   HL, #3E80 DNZONE1    ;do HL adresa uložení analyzovaného jména
0FF9              LD   DE, #3E8A FNZONE1    ;do DE adresa 1. jména souboru pro I/O
0FFC              LD   BC, #000A            ;délka je 10 znaků
0FFF              LDIR                      ;přesuň jméno souboru na správné místo
1001              LD   HL, #3E80 DNZONE1    ;do HL adresa 1. jména disku pro I/O
1004              LD   B, #0A               ;10 znaků
1006              CALL #104B, BNULHL        ;vymaž jméno disku, protože nebylo vloženo
1009              RET                       ;vrať se
```

Jako první bylo vloženo jméno disku, proto budeme pokračovat.

```
100A              LD   A, C                 ;do A délka zpracovaného řetězce
100B              AND  A                    ;je délka nulová
100C              JR   Z, #0FE2, DIVLOOP    ;ano → skoč na pokračování ve výběru
100E              DEC  B                    ;sniž délku zpracovaného řetězce o jedničku (:)
100F              RET  Z                    ;byl už zpracovaný celý řetezec → vrať se
```

Vybereme zbývající část řetězce.

```
1010              LD   DE, #3E8A FNZONE1    ;do DE adresa 1. jména souboru pro I/O
1013              CALL #1064, GETNAME       ;vyzvedni jméno po znaky „ : “, „ . “ nebo do konce
                                            ;řetězce
1016              JR   C, #101E, ANALFNM    ;byl to jméno souboru → skoč
1018              RET  NZ                   ;byl konec řetězce → vrať se
```

Jméno souboru bylo dlouhé.

```
1019 REPORTF      LD   A, #0E               ;REPORT F „Invalid file name“
101B              JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
101E ANALFNM      LD   A, (HL)              ;vyzvedni příponu z řetězce
101F              LD   (#3E94), A EXTE1     ;a ulož ji do jména souboru v FNZONE1
1022              DEC  B                    ;sniž B o 2 (tečku a příponu)
1023              DEC  B
1024              RET  Z                    ;všechny znaky zpracovány → vrať se
1025              JR   #1019, REPORTF       ;skoč na REPORT F „Invalid file name“
```

NOPARCAT

Vyplní FNZONE1 znakem „?“ a nastaví jméno aktuálního disku do DNZONE1.

IN: –

OUT: FNZONE1 je vyplněn znakem „?“ a v DNZONE1 je jméno aktuálního disku

```
1027 NOPARCAT    CALL #1051, TESTSYN1      ;otestuj, jestli není kontrola syntaxe

   Neprovádí se kontrola syntaxe, provedeme operaci.
102A             LD   HL, #3E8A FNZONE1    ;do HL adresa 1. jména souboru pro I/O
102D             LD   DE, #3E8B            ;do DE adresa dalšího znaku v FNZONE1
1030             LD   (HL), „?“            ;vyplníme FNZONE1 i s příponou znakem „?“
1032             LD   BC, #000A            ;11 znaků
1035             LDIR                      ;vyplň
```

```
   Vyplní DNZONE1 jménem aktuálního disku.
1037 MOVEWDNM    LD   HL, #3EAA ACDRIVE    ;do HL adresa jména aktuálního drivu
103A             LD   DE, #3E80 DNZONE1    ;do DE adresa 1. jména disku pro I/O
103D             LD   BC, #000A            ;délka 10 znaků
1040             LDIR                      ;přesuň
1042             RET                       ;vrať se
```

SETWDNM

Zkontroluje, jestli je jméno disku pro I/O v DNZONE1. Pokud ne, je zde vloženo jméno aktuálního drivu.

IN: jméno disku pro I/O v DNZONE1

OUT: jméno disku pro I/O v DNZONE1

```
1043 SETWDNM     LD   HL, #3E80 DNZONE1    ;do HL adresa 1. jména disku pro I/0
1046             LD   A, (HL)              ;vezmi první znak ze jména
1047             AND  A                    ;je jméno prázdné?
1048             RET  NZ                   ;ne → vrať se
1049             JR   #1037, MOVEWDNM      ;skoč na vyplnění jménem aktuálního disku
```

BNULHL

Uloží B nul od adresy v HL.

IN: HL adresa uložení
B počet nulovaných bytů

OUT: vynuluje B bytů od adresy v HL

```
104B BNULHL      LD   (HL), #00            ;ulož na adresu v HL nulu
104D             INC  HL                   ;posuň se na další adresu
104E             DJNZ #104B, BNULHL        ;opakuj B-krát
1050             RET                       ;vrať se
```

TESTSYN1

Testuje, jestli nejde o kontrolu syntaxe. Pokud ano, vyzvedne se jednu návratovou adresu více (kontrola je volána z procedur využívaných programy) a jde se zpět do ZX ROM.

```
1051 TESTSYN1    RST  #30                  ;testuj bit
1052             RET  NZ                   ;jestli se provádí příkaz, vrať se zpět
1053             POP  BC                   ;vyzvedni návratovou adresu z procedury
1054             NOP                       ;tady si odpočiň
1055             JR   #1059, SYNCRET       ;pokračuj v návratu do ZX ROM
```

ISSYNCONTR

Testuje, jestli nejde o kontrolu syntaxe. Pokud ano, vyzvedne návratové adresy a vrátí se zpět do ZX ROM. Tento test se volá přímo z programů, které provádějí příkazy.

```
1057 ISSYNCONTR RST  #30                ;testuj bit
1058            RET  NZ                 ;jestli se provadí příkaz, vrať se
1059 SYNCRET    POP  BC                 ;vyzvedni ze zásobníku návratové adresy
105A            POP  BC
105B            POP  BC                 ;tyto dvě instrukce tady vůbec nemusely být. První
105C            PUSH HL                 ;vyzvedává návratovou adresu a druhá ukládá na
                                        ;zásobník nějakou hodnotu
105D            LD   HL, #1BF4 STMT-NEXT ;návratová adresa do ZX ROM
1060            EX   (SP), HL           ;ulož ji na zásobník
1063            JP   #1700, STANDROM    ;skoč na přestránkováni do ZX ROM
```

GETNAME

Tento podprogram vyzvedne jméno souboru nebo disku z řetězce buď po znaky „ . “ a „ : “ nebo až je do konce řetězce, pokud není jméno delší než 10 znaků.

IN: HL začátek řetězce
DE adresa, kam se bude ukládat jméno disku nebo souboru
B délka řetězce
OUT: Z, NC bylo vybráno jméno disku
Z, C bylo vybráno jméno souboru
NZ, NC byl konec řetězce
B zbývající znaky řetězce
C délka vybraného řetězce

```
1064 GETNAME    LD   C, #00             ;nastav na začátku počítadlo znaků na 0
1066 MAKENAME   LD   A, (HL)            ;vyzvedni znak z řetězce
1067            INC  HL                 ;posuň se na další znak v řetězci
1068            CP   „ : “              ;je to „ : “?
106A            RET  Z                  ;ano → vrať se s příznaky Z, NC
106B            CP   „ . “              ;je to „ . “?
106D            SCF                     ;nastav CY
106E            RET  Z                  ;ano → vrať se s příznaky Z, C
106F            LD   (DE), A            ;ulož znak do jména
1070            INC  DE                 ;posuň se na další znak ve jménu
1071            INC  C                  ;zvyš délku jména
1072            LD   A, C               ;dej délku do A
1073            CP   #0B                ;je jméno delší než 10 znaků?
1075            JP   NC, #1019, REPORTF ;ano → skoč na REPORT F „Invalid file name“
1078            DJNZ #1066, MAKENAME    ;sniž v B délku řetězce a vezmi další znak
107A            AND  A                  ;nastav NC, NZ
107B            RET                     ;vrať se
```

ARRANGNM

Tento podprogram upraví jméno souboru ve FNZONE1 na masku. Převede „*“ na otazníky do konce jména.

IN: ve FNZONE1 jméno souboru
OUT: ve FNZONE1 jméno souboru upravené na masku
C bylo použito wildchars
NZ byla vložena přípona
Z nebyla vložena přípona

```
107C ARRANGNM  LD    HL, #3E8A FNZONE1   ;do HL adresa jména souboru

   Nejdříve otestujeme, jestli bylo vůbec vloženo jméno souboru. Pokud ne, upravíme ho na „*“.
107F           LD    A, (HL)             ;vyzvedni první znak ze jména
1080           AND   A                   ;bylo vloženo jméno?
1081           JR    NZ, #1085, ARNGNM1  ;ano → skoč
1083           LD    (HL), „*“           ;dej jako první znak „*“ – wildchars,
1085 ARNGNM1   LD    B, #0A              ;budeme analyzovat deset znaků
1087           LD    C, #00              ;počítadlo znaku je na začátku 0
1089 ARRLOPPLD A, (HL)                   ;vyzvedni znak jména
108A           AND   A                   ;je konec?
108B           JR    Z, #109A, ARRANGEXT;ano → skoč na úpravu přípony
108D           CP    „*“                 ;je to hvězdička?
108F           JR    Z, #10A1, FILLMARK  ;ano → skoč na vyplnění otazníky
1091           CP    „?“                 ;je to otazník?
1093           JR    NZ, #1097, NOWILD   ;ne → skoč
1095           SET   0, C                ;nastav příznak „použito wildchars“
1097 NOWILD    INC   HL                  ;posuň se na další znak jména
1098           DJNZ  #1089, ARRLOPP      ;opakuj B-krát
109A ARRANGEXT CALL  #10B3, SETEXT       ;nastav příponu
109D           AND   A                   ;nastav NC
109E           RR    C                   ;nastav C, jestli bylo použito wildchars a Z, jestli nebyla
10A0           RET                       ;vložena přípona a vrať se

   Doplníme jméno souboru znakem „?“ místo „*“.
10A1 FILLMARK  LD    (HL), „?“           ;nahraď znak „*“ znakem „?“
10A3           INC   HL                  ;posuň se na další znak
10A4           DEC   B                   ;sniž počítadlo
10A5 FILLMARK1 LD    A, (HL)             ;vyzvedni znak ze jména
10A6           LD    (HL), „?“           ;ulož místo něj „?“
10A8           INC   HL                  ;posuň se na další znak
10A9           AND   A                   ;bylo něco za „*“ vloženo?
10AA           JP    NZ, #1019, REPORTF  ;ano → skoč na REPORT F „Invalid file name“
10AD           DJNZ  #10A5, FILLMARK1    ;opakuj B-krát
10AF           SET   0, C                ;nastav příznak „použito wildchars“
10B1           JR    #109A, ARRANGEXT    ;pokračuj v úpravě přípony souboru
```

SETEXT

Nastaví příponu v EXTE1.

IN: #3E94 (EXTE1) přípona souboru
OUT: #3E94 (EXTE1) upravená přípona („*“ a „?“)
1. bit reg. C=1 byla vložena přípona
1. bit reg. C=0 nebyla vložena přípona

```
10B3 SETEXT    SET   1, C                ;nastav signál „přípona použita“
10B5           LD    A, (#3E94) EXTE1    ;vyzvedni příponu souboru do A
10B8           CALL  #1118, UPPER        ;převeď ji na velké písmeno
10BB           CP    „*“                 ;je to hvězdička?
10BD           JR    Z, #10C4 , SETMARK  ;ano → skoč
10BF           AND   A                   ;přípona nebyla zadána?
10C0           JR    NZ, #10C6, EXTIS    ;ne → skoč
10C2           RES   1, C                ;nastav signál „není přípona“
```

```
10C4 SETMARK    LD    A, „?“              ;jako příponu uložíme „?“
```

Prohlédneme tabulku povolených přípon a zjistíme, jestli je vložená přípona povolena.

```
10C6 EXTIS      LD    (#3E94), A EXTE1    ;ulož příponu souboru
10C9            PUSH  HL                  ;ulož HL
10CA            PUSH  BC                  ;ulož BC
10CB            LD    HL, #10DB EXTTAB    ;do HL tabulka povolených přípon
10CE            LD    BC, #0007           ;7 možných přípon
10D1            CPIR                      ;porovnej vloženou příponu s tabulkou
10D3            POP   BC                  ;obnov HL
10D4            POP   HL                  ;obnov BC
10D5            RET   Z                   ;přípona existuje → vrať se zpět
10D6            LD    A, #2B              ;REPORT c „Bad file type“
10D8            JP    #0204, ERRR         ;piš hlášení a skoč do ZX ROM
```

EXTTAB

Programové přípony používané MDOSem.

```
10DB EXTTAB     DB    50 4E 43 42 51 53 3F  ;P N C B Q S ?
```

ANALWDNM

Tento podprogram analyzuje jméno disku v DNZONE1 pro I/O.

IN: jméno disku v DNZONE1
OUT: Z, C není jméno v DNZONE1
NZ, C v DNZONE1 je normální jméno disku
NZ, NC v DNZONE1 je jméno disku uvedeno jako mechanika (A, B, C, D)
v A je číslo mechaniky (0, 1, 2, 3)
pokud reg. A=255, bylo špatně vloženo jméno disku

```
10E2 ANALWDNM   LD    HL, #3E80 DNZONE1   ;do HL adresa 1. jména disku pro I/O
10E5 ANALWNM    LD    B, #0A              ;délka jména je 10 bytů
```

Nejdříve zjistíme, jestli jméno disku v DNZONE1 neobsahuje nějaký špatný znak.

```
10E7 ANALWDCH   LD    A, (HL)             ;vezmi znak ze jména do A
10E8            AND   A                   ;konec jména?
10E9            JR    Z, #10F3, ANALWDEN  ;ano → skoč
10EB            CALL  #111F, ISALFNUM     ;je to alfanumerický znak?
10EE            JR    NC, #1113, REPORTb, ;ne → skoč na REPORT b „Bad volume name“
10F0            INC   HL                  ;posuň se na další znak jména
10F1            DJNZ  #10E7, ANALWDCH     ;opakuj B-krát
```

Nyní otestujeme, jaké jméno disku bylo vloženo.

```
10F3 ANALWDEN   LD    HL, #3E80 DNZONE1   ;do HL adresa 1. jména disku pro I/O
10F6            LD    A, (HL)             ;vezmi první znak ze jména do A
10F7            AND   A                   ;bylo vůbec vloženo?
10F8            RET   Z                   ;ne → vrať se s Z, NC
```

Otestujeme, jestli je délka jména 1 nebo více znaků.

```
10F9            INC   HL                  ;posuň se na další znak
10FA            LD    A, (HL)             ;vezmi druhý znak ze jména do A
10FB ANALWDNM1  AND   A                   ;je konec jména?
10FC            SCF                       ;nastav C
```

```
10FD            RET   NZ                ;ne → vrať se NZ, C
```

Jméno disku tvoří jeden znak. Zjistíme, jestli to není určení mechaniky (A, B, C, D). Je zde však malý nedostatek, protože tato kontrola neumožní používat disketu se jménem „E“.

```
10FE            DEC   HL                ;jdi o jeden znak zpět
10FF            LD    A, (HL)           ;vyzvedni první znak jména
1100            CALL  #1118, UPPER      ;převeď na velké písmeno
1103            SUB   „A“               ;odečti 65
1105            RET   C                 ;bylo menší než „A“ → vrať se s NZ, C
1106            CP    #05               ;nastav Z pro určení kombinace
1108            JR    Z, #10FB, ANALWDNM1;není určení mechaniky → skoč na návrat s NZ, C
110A            CCF                     ;neguj CY
110B            RET   C                 ;není určení mechaniky → vrať se s NZ, C
110C            CP    #04               ;je to znak „A“–“D“?
110E            CCF                     ;neguj CY
110F            RET   NZ                ;je → vrať se s NZ, NC, v A je číslo mechaniky
```

Byla vložena špatná mechanika.

```
1110            OR    #FF               ;nastav NZ, NC, A=255
1112            RET                     ;vrať se
1113 REPORTb    LD    A, #2A            ;REPORT b „Bad volume name“
1115            JP    #0204, ERRR       ;piš hlášení a skoč do ZX ROM
```

UPPER

Pokud je znak v A alfabetický, převede ho na velké písmeno.

IN: A znak
OUT: A pokud je alfabetický, je zde převeden na velké písmeno

```
1118 UPPER      CALL  #1124, ISALFABET  ;testuj, jestli to je alfabetický znak
111B            RET   NC                ;není → vrať se
111C            AND   #DF               ;převeď na velké písmeno
111E            RET                     ;vrať se
```

ISALFNUM

Testuje, je-li znak v A alfanumerický.

IN: A znak
OUT: C je alfanumerický
NC není alfanumerický

```
111F ISALFNUM   CALL  #1132, ISNUM      ;je to číslice?
1122            CCF                     ;neguj C
1123            RET   C                 ;ano → vrať se s C
```

ISALFABET

Testuje, je-li znak v A alfabetický.

IN: A znak
OUT: C je alfabetický
NC není alfabetický

```
1124 ISALFABET  CP    „A“               ;je menší než znak „A“?
1126            CCF                     ;neguj C
1127            RET   NC                ;ano → vrať se s NC
```

```
1128            CP    „Z“+1          ;je mezi znaky „A“–„Z“?
112A            RET   C              ;ano → vrať se s C
112B            CP    „a“            ;je menší než znak „a“ a větší než znak „Z“?
112D            CCF                  ;neguj C
112E            RET   NC             ;ano → vrať se s NC
112F            CP    „z“+1          ;je mezi znaky „a“–„z“?
1131            RET                  ;vrať se s příznakem C
```

ISNUM

Testuje, je-li znak v A číslice.

IN: A znak
OUT: NC je číslice
C není číslice

```
1132 ISNUM      CP    „0“            ;je menší než znak „0“?
1134            RET   C              ;ano → vrať se s C
1135            CP    „9“+1          ;je větší než znak „9“?
1137            CCF                  ;neguj příznak C
1138            RET                  ;vrať se
```

RUN

Příkaz pro nahrání BASIC souboru se jménem „run“ do paměti. Pokud byl uložen s parametrem LINE, je také spuštěn.

Syntaxe: RUN

Příkaz prohledá adresář až najde soubor se jménem „run.p“ nebo „run.s“ a natáhne ho do paměti. V paměti nesmí být žádný jiný BASIC program, protože by ho příkaz RUN spustil, místo aby hledal soubor.

Vytvoříme si prostor v paměti pro uložení jména souboru.

```
1139 RUN        LD    BC, #0005      ;velikost prostoru je 5 bytů
113C            RST   #28            ;volej podprogram pro volání rutiny ZX ROM
113D            DW    #0030          ;podprogram BC-SPACES vytvoř prostor BC bytů
                                     ;vytvoříme prostor pro jméno souboru
```

Přesuneme jméno do paměti.

```
113F            LD    HL, #1157 TXTRUN ;adresa názvu „run“ v ROM D40
1142            LD    BC, #0003      ;délka je 3 byty
1145            PUSH  DE             ;ulož si začátek jména souboru v RAM
1146            LDIR                 ;přesuň jméno do vytvořeného prostoru
1148            LD    BC, #0003      ;délka stringu je 3
114B            POP   DE             ;obnov adresu začátku jména souboru
```

Uložíme parametry jména souboru na zásobník kalkulátoru.

```
114C            RST   #28            ;volej podprogram pro volání rutiny ZX ROM
114D            DW    #2AB2          ;podprogram STK-ST-0 uložení registry do zásobníku
                                     ;kalkulátoru – uložíme počáteční adresu jména a délku
                                     ;jména na zásobník kalkulátoru
```

A provedeme příkaz LOAD.

```
114F            LD    A, #01         ;do A příznak „LOAD“ souboru
1151            LD    (#5C74), A T_ADDR ;ulož si příznak operace
1154            JP    #1716, SMLSTART ;skoč na LOAD souboru
```

TXTRUN

Text „run“.

```
1157 TXTRUN     DB    72 75 6E                ;run
```

CATNOINF

Tento podprogram zajišťuje provádění příkazu CAT, pokud byl použit znak „-“. Vypisuje katalog bez informací o souborech.

```
115A CATNOINF   RST   #20                     ;vezmi další znak

    Nastavíme jméno drivu a masku souborů.
115B            CALL  #0FC0, DIVSTRCAT        ;vyzvedni parametry příkazu a zpracuj je
115E            CALL  #1043, SETWDNM          ;nastav jméno disku v DNZONE1 pro I/O
1161            CALL  #107C, ARRANGNM         ;uprav jméno souboru v FNZONE1 na masku
1164            CALL  #1C8F, SETACT           ;roztoč mechaniku, která má stejné jméno, jako
                                              ;v DNZONE1
1167            LD    A, #02                  ;kanál 2 (horní část obrazovky)
1169            RST   #28                     ;volej podprogram pro volání rutiny ZX ROM
116A            DW    #1601                   ;podprogram CHAN-OPEN otevření kanálu

    Vytiskneme hlavičku a jméno diskety v drivu.
116C            XOR   A                       ;položka 0 – „Catalogue of “
116D            LD    DE, #12AD TXTCAT2       ;tabulka textů příkazu CAT
1170            CALL  #01C8, PRTMES           ;piš položku textu
1173            CALL  #2199, NAMEDISK         ;do HL adresa jména drivu, na jehož parametry ukazuje IX
1176            CALL  #128D, PRTSTR           ;vytiskni jméno diskety
1179            LD    A, #0D                  ;nový řádek
117B            RST   #10                     ;tiskni
117C            LD    A, #0D                  ;nový řádek
117E            RST   #10                     ;tiskni

    Nyní postupně vytiskneme všechny neprázdné položky adresáře.
117F            LD    A, #FF                  ;začneme od první položky adresáře
1181            LD    C, #00                  ;v C bude čítač počtu obsazených položek adresáře
1183 CATNOLOOP  CALL  #212D, NEXTMASK         ;načti neprázdnou položku adresáře od A vyhovující
                                              ;masce v FNZONE1
1186            JR    NZ, #11A2, PRINTINF     ;už žádná není → skoč
1188            INC   C                       ;zvyš počet souborů na disketě
1189            PUSH  BC                      ;ulož si čítač
118A            PUSH  AF                      ;a číslo položky v adresáři

    Otestujeme, jestli nemá nastaven atribut HIDDEN.
118B            CALL  #1283, GETATR           ;vyzvedni atributy souboru
118E            BIT   7, A                    ;je HIDDEN?
1190            JR    NZ, #119E, ISHIDEEN     ;ano → nevypisuje se – skoč

    Vytiskneme příponu a jméno.
1192            LD    A, (HL)                 ;vyzvedni příponu souboru
1193            INC   HL                      ;posuň se na první znak jména souboru
1194            RST   #10                     ;tisknim příponu
1195            LD    A, #20                  ;mezera
```

```
1197                RST  #10                ;tiskni
1198                CALL #128D, PRTSTR     ;vytiskni jméno souboru
119B                LD   A, #06            ;posun na další tabelační pozici
119D                RST  #10                ;tiskni
119E ISHIDDEN       POP  AF                ;obnov registry
119F                POP  BC
11A0                JR   #1183, CATNOLOOP  ;opakuj pro další položku
```

PRINTINF

Vytiskne počet souborů na disketě a volnou kapacitu diskety.

IN: C počet souborů na disketě

OUT: vytiskne počet souborů a volnou kapacitu

```
11A2 PRINTINF       PUSH BC                ;ulož si počet souborů na disku
11A3                LD   A, #0D            ;nový řádek
11A5                RST  #10               ;tiskni
11A6                LD   A, #0D            ;nový řádek
11A8                RST  #10               ;tiskni
11A9                POP  BC                ;obnov počet souborů na disku
11AA                LD   B, #00            ;do B dej 0
11AC                CALL #0FA6, BCPRT      ;piš počet souborů na disketě na obrazovku
11AF                LD   DE, #12AD TXTCAT2 ;tabulka textů příkazu CAT
11B2                LD   A, #01            ;položka 1 – „File(s), “
11B4                CALL #01C8, PRTMES     ;piš text položky
11B7                CALL #1DC2, FREECOUNT  ;spočítej počet volných sektorů na disketě
11BA                PUSH IX                ;ulož si IX
11BC                LD   IX, #3ED4 SV24NM  ;do IX adresu místa pro výpočet kapacity disku
11C0                SLA  C                 ;BC vynásob dvěma (násobení 512-ti)
11C2                RL   B
11C4                LD   (IX+#00), #00     ;první byte je nula
11C8                LD   (IX+#01), C       ;druhý byte je C
11CB                LD   (IX+#02), B       ;třetí byte je B
11CE                CALL #0F14, NUM24B     ;převeď a tiskni kapacitu diskety
11D1                POP  IX                ;obnov si IX
11D3                LD   A, #02            ;položka 2 – „Bytes free.“
11D5                LD   DE, #12AD TXTCAT2 ;tabulka textů příkazu CAT
11D8                CALL #01C8, PRTMES     ;piš text položky
11DB                CALL #217B, ERAVAR     ;vymaž pomocné proměnné MDOSu
11DE                RET                    ;vrať se přes RETURN do ZX ROM
```

CAT

Příkaz pro výpis souborů uložených na disketě.

Syntaxe: CAT [-] ["[Zařízení:][MaskaSouboru]"]

Příkaz vypíše soubory uložené na disketě. Znak „-“ znamená, že se nebudou vypisovat podrobnější informace o souborech (atributy a délka souboru). „Zařízení“ určuje drive, pro který se příkaz provede („A:“, „disk:“ atd.). „MaskaSouboru“ určuje, které soubory se budou vypisovat. Pokud není vloženo, vypisují se všechny (ekvivalentní použití „*.*“). Příklad: „*.P“, „DOP*.B“, „DAT?.*“ atd. Může se tedy používat wildchars (hvězdičková konvence). Výsledek příkazu nelze přesměrovat do sekvenčního souboru.

Nejdříve otestujeme znak „-“.

```
11DF CATFN          RST  #18               ;vezmi aktuální znak
11E0                CP   „-“               ;je to mínus?
```

```
11E2                JP    Z, #115A, CATNOINF  ;ano → skoč na výpis adresáře souborů bez informací

   Budeme tedy vypisovat všechno.
11E5                CALL  #0FC0, DIVSTRCAT    ;vyzvedni parametry příkazu a zpracuj je
11E8                CALL  #1043, SETWDNM      ;nastav jméno disku v DNZONE1 pro I/O
11EB                CALL  #107C, ARRANGNM     ;uprav jméno souboru na masku v FNZONE1
11EE                CALL  #1C8F, SETACT       ;roztoč mechaniku, která má stejné jméno jako
                                              ;v DNZONE1
11F1                LD    A, #02              ;kanál 2 (horní část obrazovky)
11F3                RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
11F4                DW    #1601               ;podprogram CHAN-OPEN otevření kanálu

   Vytiskneme hlavičku a jméno diskety.
11F6                XOR   A                   ;položka 0 – „Directory of “
11F7                LD    DE, #129E TXTCAT1   ;tabulka textů příkazu CAT
11FA                CALL  #01C8, PRTMES       ;piš text položky
11FD                CALL  #2199, NAMEDISK     ;do HL adresa jména drivu, na jehož parametry ukazuje IX
1200                CALL  #128D, PRTSTR       ;vytiskni jméno diskety
1203                LD    A, #0D              ;nový řádek
1205                RST   #10                 ;tiskni
1206                LD    A, #0D              ;nový řádek
1208                RST   #10                 ;tiskni

   Nyní postupně vytiskneme všechny neprázdné položky adresáře.
1209                LD    A, #FF              ;začneme od 1. položky adresáře
120B                LD    C, #00              ;v C bude počítadlo souborů na disketě
120D CATFNLOOP      CALL  #212D, NEXTMASK     ;načti neprázdnou položku adresáře od A vyhovující
                                              ;masce v FNZONE1
1210                JR    NZ, #11A2, PRINTINF ;žádná není → skoč na výpis zbývajících informací
1212                INC   C                   ;zvyš počítadlo souborů
1213                PUSH  AF                  ;ulož si číslo položky

   Zjistíme, jestli není HIDDEN.
1214                CALL  #1283, GETATR       ;vyzvedni atributy souboru
1217                BIT   7, A                ;je HIDDEN?
1219                LD    B, A                ;ulož atributy do B
121A                PUSH  BC                  ;a ulož je na zásobník
121B                JR    NZ, #1276, FNISHID  ;ano → nevypisuje se – skoč
121D                PUSH  HL                  ;ulož si ukazatel na položku adresáře v buferu
121E                LD    A, (HL)             ;vyzvedni příponu souboru
121F                INC   HL                  ;posuň se na první znak jména souboru
1220                RST   #10                 ;tiskni příponu
1221                LD    A, #20              ;mezera
1223                RST   #10                 ;tiskni
1224                CALL  #128D, PRTSTR       ;vytiskni jméno souboru
1227                POP   HL                  ;obnov ukazatel na položku adresáře
1228                LD    A, #0B              ;1. byte velikosti souboru je v hlavičce na +11
122A                CALL  #0FAD, ADDHLA       ;posuň se tam

   Vytiskneme velikost souboru.
122D                PUSH  IX                  ;ulož si IX
```

```
122F            LD    IX, #3ED4 SV24NM    ;do IX adresa pro výpočet velikosti souboru
1233            LD    A, (HL)             ;vyzvedni 1. byte velikosti souboru
1234            LD    (IX+#00), A         ;ulož ho
1237            INC   HL                  ;posuň na 2. byte velikosti souboru
1238            LD    A, (HL)             ;vyzvedni ho
1239            LD    (IX+#01), A         ;a ulož ho
123C            LD    A, #09              ;3. byte velikosti souboru je v hlavičce na +9
123E            CALL  #0FAD, ADDHLA       ;posuň tam ukazatel
1241            LD    A, (HL)             ;vyzvedni 3. byte velikosti souboru
1242            LD    (IX+#02), A         ;ulož ji na 3. byte
1245            LD    A, #17              ;tabelátor
1247            RST   #10                 ;tiskni
1248            LD    A, #0E              ;na čtrnáctou pozici
124A            RST   #10                 ;nastav kurzor
124B            XOR   A                   ;do A nulu
124C            RST   #10                 ;pošli jako 2. parametr příkazu TAB
124D            CALL  #0F14, NUM24B       ;tiskni velikost souboru
1250            POP   IX                  ;obnov IX
1252            LD    A, #17              ;tabelátor
1254            RST   #10                 ;tiskni
1255            LD    A, #17              ;na 23. pozici
1257            RST   #10                 ;nastav kurzor
1258            XOR   A                   ;do A nulu
1259            RST   #10                 ;pošli jako 2. parametr příkazu TAB
125A            POP   BC                  ;obnov atribut souboru
125B            PUSH  BC                  ;a znovu ulož
125C            LD    HL, #127B DEFATTR   ;do HL adresa tabulky znaků pro atributy souboru
125F            LD    E, #08              ;8 různých atributů
```

Postupně projdeme atributy a pokud je nastaven, je vypsán jeho kód, jinak se tiskne pomlčka.

```
1261 CATFNATT   RL    B                   ;zarotuj atribut do C
1263            LD    A, (HL)             ;vezmi znak atributu
1264            INC   HL                  ;posuň se na další znak
1265            JR    C, #1269, CATFNAPR  ;atribut nastaven → skoč
1267            LD    A, „ - “            ;atribut nenastaven – vypiš pomlčku
```

Vytiskneme atribut.

```
1269 CATFNAPR   PUSH  HL                  ;ulož si registry
126A            PUSH  DE
126B            PUSH  BC
126C            RST   #10                 ;piš znak pro atribut
126D            POP   BC                  ;obnov registry
126E            POP   DE
126F            POP   HL
1270            DEC   E                   ;všechny atributy?
1271            JR    NZ, #1261, CATFNATT ;ne → testuj a piš další
1273            LD    A, #0D              ;nový řádek
1275            RST   #10                 ;tiskni
1276 FNISHID    POP   BC                  ;obnov registry
1277            POP   AF
1278            JP    #120D, CATFNLOOP    ;skoč na pokračování pro další položku
```

DEFATTR

Tabulka znaků pro atributy souboru.

H hidden (skrytý)	S system (systémový)
P program (programový)	A archive (archivní)
R readable (čtení povoleno)	W writeable (zápis povolen)
E executable (proveditelný)	D deleteable (smazatelný)

```
127B DEFATTR    DB    48 53 50 41 52 57 45 44      ;H S P A R W E D
```

GETATR

Vyzvedne atributy souboru.

IN: HL adresa položky v buferu adresáře
OUT: A atributy souboru

```
1283 GETATR     PUSH HL                    ;ulož ukazatel na položku v adresáři
1284            EX   (SP), IX              ;nyní ho dej do IX
1286            LD   A, (IX+#14)           ;vyzvedni atribut do A
1289            EX   (SP), IX              ;a vrať ukazatel zpět do HL
128B            POP  HL
128C            RET                        ;vrať se
```

PRTSTR

Vytiskne řetězec deseti znaků od HL na obrazovku. Používá se k tisku jména diskety nebo souboru.

IN: HL adresa uložení jména
OUT: jméno je vytištěno (buď 10 znaků nebo do znaku CHR$ 0)

```
128D PRTSRT     PUSH BC                    ;ulož si BC
128E            LD   B, #0A                ;tiskni až 10 znaků
1290 PRTSTRLOOP LD   A, (HL)               ;vyzvedni znak do A
1291            AND  A                     ;konec textu?
1292            JR   Z, #129C, ENDPRTSTR   ;ano → skoč
1294            PUSH HL                    ;ulož si ukazatel
1295            PUSH BC                    ;a čítač
1296            RST  #10                   ;vytiskni znak
1297            POP  BC                    ;obnov čítač
1298            POP  HL                    ;a ukazatel
1299            INC  HL                    ;posuň se na další znak
129A            DJNZ #1290, PRTSTRLOOP     ;opakuj B-krát
129C ENDPRTSTR  POP  BC                    ;obnov BC
129D            RET                        ;vrať se
```

TXTCAT

Tabulka textů příkazu CAT.

```
129E TXTCAT1    DB    #FF                                  ;invertovaný znak
   Položka 0 – používá se, pokud je tištěn výpis s informacemi o souboru
129F            0D 44 69 72 65 63 74 6F 72 79 20 6F 66 A0  ;Directory of
12AD TXTCAT2    DB    #FF                                  ;invertovaný znak
   Položka 0 – používá se, pokud je tištěn výpis bez bližších informací o souboru
12AE            0D 43 61 74 61 6C 6F 67 75 65 20 6F 66 A0  ;Catalogue of
   Položka 1 – zbývající texty jsou pro obě varianty stejné
12BC            20 46 69 6C 65 28 73 29 2C A0              ;File(s),
```

```
Položka 2
12C6            20 42 79 74 65 73 20 66 72 65 65 2E 8D      ;Bytes free.
```

ERASE

Příkaz pro vymazání souborů z disku.

Syntaxe: ERASE "[Zařízení:]MaskaSouboru"

Příkaz slouží ke smazání jednoho nebo více souborů z disku. „Zařízení“ určuje disk a „MaskaSouboru“ určuje, které soubory z disku vymazat. Lze používat wildchars (hvězdičkovou konvenci) pro smázání skupiny souborů. Je možno mazat pouze ty soubory, které mají nastaven atribut D. Pokud budete chtít smazat všechny soubory z disku (použijete „*.*“), systém se Vás zeptá: **Erase all files? (Proceed = P)**. Pokud stisknete klávesu P (nebo R), příkaz se provede. Jestli stisknete jinou klávesu, neprovede se.

```
12D3 ERASE      CALL #0FCF, DIVSTRING       ;rozděl řetězec na zásobníku na jméno disku a souboru
12D6            CALL #0F9E, TESTNM          ;otestuj, jestli bylo zadáno jméno souboru
12D9            CALL #1043, SETWDNM         ;nastav jméno disku v DNZONE1 pro I/O
12DC            CALL #107C, ARRANGNM        ;uprav jméno souboru v FNZONE1 na masku
12DF            JP   Z, #1019, REPORTF      ;chybné → skoč na REPORT F „Invalid file name“
12E2            LD   HL, #3E8A FNZONE1      ;do HL adresa 1. jména souboru pro I/O
12E5            LD   B, #0A                 ;10 bytů
```

Zjistíme, jestli nebylo použito jako masky souboru „*.*“.

```
12E7 ERASEALL   LD   A, (HL)                ;vyzvedni první znak ze jména souboru
12E8            INC  HL                     ;posuň ukazatel na další znak
12E9            CP   „?“                    ;je to „?“ (jestli se nemažou všechny soubory)
12EB            JR   NZ, #12F8, ERANOALL    ;není → skoč
12ED            DJNZ #12E7, ERASEALL        ;opakuj B-krát
```

Protože jde o mazání všech souborů, zeptáme se uživatele, jestli je chce všechny skutečně smazat.

```
12EF            LD   A, #BD                 ;do A číslo hlášení „Erase all files?“
                                            ;a dotaz „(Proceed = P)“
12F1            LD   DE, #03AF SYSMSG       ;do DE tabulka systémových hlášení
12F4            CALL #21BF, KEYMSG          ;tiskni a čekej na klávesu
12F7            RET  NC                     ;nebylo stisknuto P nebo R → vrať se
```

Provedeme mazání.

```
12F8 ERANOALL   CALL #1F66, DELALLFIL       ;smaž všechny soubory, vyhovující masce v FNZONE1
12FB            PUSH AF                     ;ulož příznak
12FC            CALL #217B, ERAVAR          ;vymaž pomocné proměnné MDOSu
12FF            POP  AF                     ;obnov příznak
1300            RET  Z                      ;vše v pořádku. → vrať se přes RETURN do ZX ROM
```

Na disketě požadovaný soubor není.

```
1301            LD   A, #1B                 ;REPORT S „File not found“
1303            JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
```

MOVE

První varianta příkazu MOVE, nastavuje implicitní zařízení.

Syntaxe: MOVE "Zařízení:"

Příkaz slouží k nastavení aktuálního disku. S tímto diskem se potom budou vykonávat všechny operace, u kterých není přímo zadán disk. Jako „Zařízení“ lze použít buď jméno drivu nebo určení A:, B:.

```
1306 MOVEACT    CALL #1057, ISSYNCONTR      ;otestuj, jestli není kontrola syntaxe
```

Neprovádí se kontrola syntaxe, provedeme změnu implicitního zařízení.

```
1309            CALL #0FCF, DIVSTRING     ;rozděl řetězec na zásobníku na jméno disku a souboru
130C            CALL #10E2, ANALWDNM      ;analyzuj jméno disku v DNZONE1
130F            LD   HL, #3E80 DNZONE1    ;do HL adresa 1. jména disku pro I/O
1312            LD   A, (HL)              ;vyzvedni první znak ze jména disku
1313            AND  A                    ;nebylo zadáno jméno disku?
1314            JP   Z, #1113, REPORTb    ;ano → skoč na REPORT b „Bad volume name“
1317            LD   DE, #3EAA ACDRIVE    ;do DE adresa jména aktuálního disku
131A            LD   BC, #000A            ;délka jména je 10 bytů
131D            LDIR                      ;přesuň nové jméno
131F            RET                       ;vrať se přes RETURN do ZX ROM
```

FORMAT

Příkaz pro formátování disket.

Syntaxe: FORMAT "Zařízení:JménoDisku[.S]"

Příkaz naformátuje disketu vloženou v „Zařízení“. „Zařízení“ je typu A: nebo B:. Potom následuje jméno disku, které bude mít naformátovaná disketa. Vhodnými pouky můžete libovolně měnit formát diskety. Pokud použijete „.S“, bude disketa naformátována jako jednostranná. Klasická formát je 512 bytů na sektor, 9 sektorů na stopu, 40/80 stop na stranu. Po vykonání příkazu jsou na obrazovku vytištěny informace o disketě (počet dobrých a špatných sektorů, kapacita diskety). Před začátkem formátování se systém zeptá otázkou: **All data will be discarded!** Pokud stisknete P nebo R, bude se provádět formátování. Po formátování jsou všechny data na disketě ztracena a nelze je již nijak obnovit.

```
1320 FORMAT     CALL #0FCF, DIVSTRING     ;rozděl řetězec na zásobníku na jméno disku a souboru
1323            CALL #10E2, ANALWDNM      ;analyzuj jméno disku v DNZONE1
1326            JR   Z, #132A, REPORTY    ;nebylo vloženo jméno mechaniky.→ skoč na
                                          ;REPORTY „Device ident missing“
1328            JR   NC, #132F, FORDROK   ;bylo vloženo určení mechaniky → skoč
132A REPORTY    LD   A, #21               ;REPORT Y „Device ident missing“
132C            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
132F FORDROK    INC  A          ;bylo určení mechaniky správné?
1330            JR   NZ, #1337, FORDISK   ;ano → skoč
1332            LD   A, #20               ;REPORT X „Bad device type“
1334            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
1337 FORDISK    DEC  A                    ;vrať zpět číslo mechaniky
1338            LD   (#3E6B), A WORKDR    ;ulož jako drive, se kterým se pracuje
133B            CALL #21AC, DRVCMP        ;zjisti adresu parametrů mechaniky v A
133E            JR   NZ, #1345, FORMAT1   ;je připojena → skoč
1340            LD   A, #22               ;REPORT Z „Device unavailable“
1342            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Nastávíme výchozí parametry diskety z parametrů mechaniky a nastavíme formát diskety (jednostranný nebo oboustranný).

```
1345 FORMAT1    CALL #107C, ARRANGNM      ;uprav jméno diskety v FNZONE1
1348            LD   HL, #3E8A FNZONE1    ;do HL adresa uložení jména diskety
134B            CALL #10E5, ANALWNM       ;analyzuj jméno diskety na nežádoucí znaky
134E            JP   C, #1113, REPORTb    ;chybné → skoč na REPORT b „Bad volume name“
1351            LD   A, (IX+#05)          ;okopíruj informace z parametrů mechaniky
1354            LD   (IX+#01), A          ;do informací o disketě
1357            LD   A, (IX+#06)          ;okopíruj počet stop na straně z parametrů mechaniky
135A            LD   (IX+#02), A          ;do parametrů diskety
135D            LD   A, (IX+#07)          ;okopíruj počet sektorů na stopu z informací o mechanice
```

```
1360            LD    (IX+#03), A           ;do parametrů diskety
1363            LD    A, (#3E94) EXTE1      ;vyzvedni počet stran, kolik se má formátovat
1366            CP    „S“                   ;jednostranný formát?
1368            JR    NZ, #136E, RFO401     ;ne → skoč
136A            RES   4, (IX+#01)           ;nastav příznak jednostranný formát
```

Tiskneme dotaz a čekáme na odpověď.

```
136E RFO41      CALL  #217B, ERAVAR         ;vymaž pomocné proměnné MDOSu
1371            PUSH  HL                    ;ulož si registry
1372            PUSH  DE
1373            LD    DE, #03AF SYSMSG      ;do DE adresa tabulky systémových hlášení
1376            LD    A, #BF                ;do A číslo hlášení „All data will be discarded!“
                                            ;a dotaz „(Proceed = P)“
1378            CALL  #21BF, KEYMSG         ;tiskni a čekej na klávesu
137B            POP   DE                    ;obnov registry
137C            POP   HL
137D            RET   NC                    ;nebylo stisknuto P nebo R → vrať se
137E            LD    A, (#3E6B) WORKDR     ;do A číslo drivu, se kterým se pracuje
1381            CALL  #254B, DRVSEL         ;roztoč mechaniku a nastav stopu podle (IX+#04)
1384            CALL  #234B, HOME           ;vystav hlavu na 0-tou stopu
```

Nastavíme počet formátovaných stop. Pokud je oboustranný formát, musí se počet zdvojnásobit.

```
1387            LD    C, (IX+#02)           ;do C počet stop na stranu
138A            BIT   4, (IX+#01)           ;jednostranný formát?
138E            JR    Z, #1392, RFORM5      ;ano → skoč
1390            RLC   C                     ;zdvojnásob počet stop
1392 RFORM5     LD    B, #00                ;v B bude počítadlo stop
```

Naformátujeme stopu.

```
1394 RFORM6     PUSH  BC                    ;ulož si počítadlo a počet stop
1395            LD    DE, #0100             ;jeden zápis, 255 opakování
1398            LD    A, (#3E6B) WORKDR     ;do A drive, se kterým se pracuje
139B            CALL  #229C, BFORMA         ;formátuj stopu
139E            POP   BC                    ;obnov čítač a počet stop
139F            INC   B                     ;zvyš počet zformátovaných stop
13A0            LD    A, B                  ;dej do A počet zformátovaných stop
13A1            CP    C                     ;už byly naformátovány všechny?
13A2            JR    NZ, #1394, RFORM6     ;ne → opakuj
13A4            DEC   B                     ;uprav počet stop na rozsah 0–39/79/159
```

Nyní otestujeme všechny sektory, jestli jsou v pořádku. Provádíme to tak, že do zásobníku si dáme koncovou značku a postupně kontrolujeme sektory. Pokud je vadný, uložíme si jeho číslo na zásobník. Postupujeme od posledního k prvnímu sektoru. Je třeba tedy dát pozor, že pokud budeme mít nízko zásobník a budeme formátovat disketu, která bude mít hodně vadných sektorů, může nám zásobník přetéct a systém se zhroutí.

```
13A5 FORMTEST   LD    HL, #FFFF             ;do HL koncová značka špatných sektorů
13A8            PUSH  HL                    ;ulož ji na zásobník
13A9 FORMTEST1  LD    C, #00                ;nastav nultý sektor
13AB FORMTST2   PUSH  BC                    ;ulož si čítač sektorů a stop
13AC            LD    HL, #4900             ;do HL adresa, kam budeme zapisovat sektor
13AF            LD    DE, #0001             ;jeden sektor, žádné opakování
13B2            LD    A, C                  ;podle sektoru změníme border
```

```
13B3                OUT  (#FE), A                ;teď
13B5                LD   A, (#3E6B) WORKDR      ;do A drive, se kterým se pracuje
13B8                CALL #236A, DREAD           ;načti sektor
13BB                LD   A, #02                 ;do A příznak „testování chybových bitů při čtení“
13BD                CP   B                      ;bylo provedeno čtení?
13BE                JR   Z, #13C5, FORMNTEST    ;ano→skoč
```

Došlo k chybě při hledání stopy.

```
13C0                LD   A, #29                 ;REPORT a „Device I/O error“
13C2                JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
13C5 FORMNTEST      LD   A, #85                 ;do A bity technických chyb
13C7                AND  C                      ;byla technická chyba?
13C8                JR   Z, #13CF , FNOTECH     ;ne → skoč
```

Došlo k chybě v komunikaci s řadičem.

```
13CA                LD   A, #3B                 ;REPORT s „Internal error“
13CC                JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
```

Nyní zkontrolujeme, jestli nedošlo k chybě při čtení sektoru.

```
13CF FNOTECH        LD   A, #18                 ;do A bity chyb při čtení sektoru
13D1                AND  C                      ;byla chyba při čtení?
13D2                POP  BC                     ;obnov čislo stopy a sektoru
13D3                JR   Z, #13DB, FORMSOK      ;ne → skoč
```

Sektor je špatný, proto ho převedeme na logický a uložíme do zásobníku.

```
13D5                PUSH BC                     ;ulož si číslo sektoru a stopy
13D6                CALL #1DE9, FYZLOG          ;převeď fyzický sektor a stopu na logický sektor
13D9                POP  BC                     ;obnov číslo stopy a sektoru
13DA                PUSH HL                     ;ulož špatný sektor do zásobníku
```

Provedeme posun na další sektor.

```
13DB FORMSOK        INC  C                      ;posuň se na další sektor
13DC                LD   A, (IX+#03)            ;do A počet sektorů na stopu
13DF                CP   C                      ;byly všechny sektory na stopě testovány?
13E0                JR   NZ, #13AB, FORMTST2    ;ne → skoč na test dalšího
13E2                DJNZ #13A9, FORMTEST1       ;opakuj B-krát
```

Nyní vytvoříme v SRAM BOOT diskety a uložíme ho na disketu do 0. sektoru 0. stopy pro MDOS (1. sektoru 0. stopy pro WD)

```
13E4                LD   HL, #3C00 FATBUF       ;do HL adresa buferu pro vytvoření BOOTu
13E7                LD   E, L                   ;dej ji i do DE
13E8                LD   D, H
13E9                INC  DE                     ;ale posuň ji v DE o jednu dále
13EA                LD   (HL), #00              ;vyplníme bufer hodnotou 0
13EC                LD   BC, #01FF              ;511 byte
13EF                LDIR                        ;vyplň bufer
13F1                LD   DE, #3C80              ;do DE místo v BOOTu pro informace o všech
                                                ;připojených mechanikách
13F4                LD   HL, #3E00 DRPARZN      ;do HL začátek tabulek parametrů disků
13F7                LD   BC, #0030              ;48 bytů
13FA                LDIR                        ;přesuň do BOOTu
13FC                PUSH IX                     ;ulož ukazatel na parametry disku, se kterým se pracuje
```

```
13FE            POP   HL                    ;a dej ho do HL
13FF            LD    BC, #000C             ;12 bytů informací o disketě
1402            LDIR                        ;přesň do BOOTu
1404            LD    DE, #3CC0             ;do DE adresa uložení jména diskety v BOOTu
1407            LD    HL, #3E8A FNZONE1     ;do HL adresa uložení jména diskety v paměti
140A            LD    BC, #000A             ;délka 10 znaků
140D            LDIR                        ;přesuň do BOOTu
140F            LD    A, R                  ;ulož dva náhodné byty
1411            LD    (DE), A               ;do BOOTu
1412            INC   DE
1413            HALT                        ;čekej na přerušení mezi jednotlivými byty
1414            LD    A, R
1416            LD    (DE), A
1417            INC   DE
1418            LD    HL, #0F10 TXTSDOS     ;do HL adresa uloženi textu „SDOS“ v ROM
141B            LD    BC, #0004             ;4 byty
141E            LDIR                        ;přesň do BOOTu
1420            LD    DE, #0101             ;1 sektor, žádné opakování
1423            LD    HL, #3C00 FATBUF      ;do HL adresa uložení BOOTu v SRAM
1426            LD    BC, #0000             ;nultý sektor, nultá stopa
1429            LD    A, (#3E6B) WORKDR     ;do A číslo drivu, se kterým se pracuje
142C            CALL  #2296, BWRITE         ;zapiš BOOT
142F            LD    HL, #3C00 FATBUF      ;do HL adresa buferu pro FAT v SRAM
1432            PUSH  HL                    ;ulož si ji na zásobník
1433            LD    DE, #3C01             ;do DE adresa buferu +1
1436            LD    (HL), #DD             ;vyplň hodnotou #DD
1438            LD    BC, #01FF             ;511 bytů
143B            LDIR                        ;vyplň
143D            POP   HL                    ;obnov adresu na bufer
143E            LD    C, #01                ;do C číslo prvního sektoru FAT
```

Vyplníme všechny položky FAT tabulky hodnotou #0DDD (3549) (systémový sektor nebo nedostupný) a zapíšeme na disketu.

```
1440 WEMPFAT    PUSH  HL                    ;ulož si adresu buferu
1441            LD    DE, #0101             ;1 sektor, žádné opakování
1444            LD    B, #00                ;stopa 0
1446            PUSH  BC                    ;ulož si číslo sektoru a stopy
1447            LD    A, (#3E6B) WORKDR     ;do A číslo drivu, se kterým se pracuje
144A            CALL  #2296, BWRITE         ;zapiš sektor
144D            POP   BC                    ;obnov číslo sektoru a stopy
144E            POP   HL                    ;obnov adresu buferu
144F            INC   C                     ;posuň se na další sektor
1450            LD    A, #06                ;do A počet sektorů FAT +1
1452            CP    C                     ;byly už zapsány všechny sektory?
1453            JR    NZ, #1440, WEMPFAT    ;ne → opakuj
1455            CALL  #1DDC, SECPERDISK     ;vypočti počet sektorů na disketě
1458            AND   A                     ;nuluj CY
1459            LD    DE, #000EH            ;do DE 14 sektorů pro systém
145C            SBC   HL, DE                ;odečti od celkového počtu
145E            EX    DE, HL                ;a dej celkový počet do DE a do HL číslo prvního
                                            ;sektoru, který se bude testovat.
145F            PUSH  DE                    ;ulož si počítadlo sektorů
```

Nyní budeme postupně zapisovat do všech přístupných položek FAT pro uložení dat hodnoty 0 (volný sektor) a zapíšeme je na disk.

```
1460 WTESTFAT  PUSH DE                    ;ulož si počítadlo sektorů
1461           LD   DE, #0000             ;do DE hodnota „sektor volný“
1464           CALL #1D1E, WRTOFAT        ;zapiš do položky FAT v HL obsah DE
1467           POP  DE                    ;obnov počítadlo sektorů
1468           INC  HL                    ;posuň se na další sektor
1469           DEC  DE                    ;a sniž počítadlo sektorů
146A           LD   A, D                  ;už byly zapsány všechny?
146B           OR   E
146C           JR   NZ, #1460, WTESTFAT   ;ne → skoč
146E           POP  BC                    ;obnov si počet sektorů
146F           LD   DE, #0000             ;do DE počítadlo vadných sektorů
```

Nyní budeme vyzvedávat ze zásobníku čísla sektorů, které jsou vadné, až do koncové značky a budeme do jejích umístění ve FAT zapisovat informaci, že jsou vadné.

```
1472 WFAILSEC  POP  HL                    ;vyzvedni číslo vadného sektoru ze zásobníku
1473           LD   A, H                  ;je to koncová značka?
1474           AND  L
1475           INC  A
1476           JR   Z, #1484, WFATEND     ;ano → skoč
1478           PUSH DE                    ;ulož si počet vadných sektorů
1479           LD   DE, #DFF              ;do DE hodnotu „vadný sektor“
147C           CALL #1D1E, WRTOFAT        ;zapiš do položky FAT v HL obsah DE
147F           POP  DE                    ;obnov si počet vadných sektorů
1480           INC  DE                    ;zvyš počet vadných sektorů
1481           DEC  BC                    ;sniž počet dobrých sektorů
1482           JR   #1472, WFAILSEC       ;skoč na testování dalšího
1484 WFATEND   CALL #1D9D, WFATIFCH       ;zapiš FAT, pokud byla změněna
```

Nyní vypíšeme informace o dobrých a špatných sektorech a volnou kapacitu disku.

```
1487           PUSH DE                    ;ulož si počet vadných sektorů
1488           PUSH BC                    ;ulož si počet dobrých sektorů
1489           RST  #28                   ;volej podprogram pro volání rutiny ZX ROM
148A           DW   #0D6B                 ;podprogram CLS smazání obrazovky
148C           LD   A, #FE                ;kanál – 2 (horní část obrazovky)
148E           RST  #28                   ;volej podprogram pro volání rutiny ZX ROM
148F           DW   #1601                 ;podprogram CHAN-OPEN otevření kanálu
1491           XOR  A                     ;položka 0 – „Format complete. Formatted “
1492           LD   DE, #14E1 TXTFORM     ;tabulka textů příkazu FORMAT
1495           CALL #01C8, PRTMES         ;piš text položky
1498           POP  BC                    ;obnov počet dobrých sektorů
1499           PUSH BC                    ;a zase ho ulož
149A           CALL #0FA6, BCPRT          ;a vytiskni ho na obrazovku
149D           LD   A, #01                ;položka 1 – „ good blocks and “
149F           LD   DE, #14E1 TXTFORM     ;tabulka textů příkazu FORMAT
14A2           CALL #01C8, PRTMES         ;piš text položky
14A5           POP  BC                    ;obnov počet dobrých sektorů
14A6           POP  DE                    ;a počet vadných sektorů
14A7           PUSH BC                    ;ulož si počet dobrých sektorů
14A8           LD   B, D                  ;dej počet vadných sektorů do BC
```

```
14A9            LD    C, E
14AA            CALL  #0FA6, BCPRT      ;piš počet vadných sektorů na obrazovku
14AD            LD    A, #02            ;položka 2 – „ bad blocks. Total capacity is “
14AF            LD    DE, #14E1 TXTFORM ;tabulka textů příkazu FORMAT
14B2            CALL  #01C8, PRTMES     ;piš text položky
14B5            POP   BC                ;obnov počet dobrých sektorů
14B6            SLA   C                 ;vynásob dvěma (násobení 512-ti)
14B8            RL    B
14BA            LD    IX, #3ED4 SV24NM  ;do IX adresa pro výpočet kapacity disku
14BE            LD    (IX+#00), #00     ;první byte bude 0
14C2            LD    (IX+#01), C       ;druhý byte je C
14C5            LD    (IX+#02), B       ;třetí byte je B
14C8            CALL  #0F14, NUM24B     ;tiskni kapacitu disku
14CB            LD    A, #03            ;položka 3 – „ Bytes.“
14CD            LD    DE, #14E1 TXTFORM ;tabulka textů příkazu FORMAT
14D0            CALL  #01C8, PRTMES     ;piš text položky
14D3            CALL  #217B, ERAVAR     ;vymaž pomocné proměnné MDOSu
14D6            LD    A, (#5C8D) ATTR_P ;do A nastavené barvy BASICu
14D9            RRCA                    ;zarotuj
14DA            RRCA
14DB            RRCA
14DC            OR    #F8               ;ponech barvy pro border
14DE            OUT   (#FE), A          ;a nastav zpět BORDER
14E0            RET                     ;vrať se
```

TXTFORM

Tabulka textů příkazu FORMAT

```
14E1 TXTFORM    DB    #FF                                       ;invertovaný znak
  Položka 0
14E2            46 6F 72 6D 61 74 20 63 6F 6D 70 6C 65 74 65 2E 0D ;Format complete.
14F3            46 6F 72 6D 61 74 74 65 64 A0                   ;Formatted
  Položka 1
14FD            20 67 6F 6F 64 20 62 6C 6F 63 6B 73 0D 61 6E 64 A0 ;good blocks and
  Položka 2
150E            20 62 61 64 20 62 6C 6F 63 6B 73 2E 0D           ;bad blocks
151B            54 6F 74 61 6C 20 63 61 70 61 63 69 74 79 20 69 F3 ;Total capacity is
  Položka 3
152C            20 42 79 74 65 73 2E 8D                         ;Bytes.
```

NOTUSED

```
1534            DS    #01CC             ;oblast délky 460 bytů vyplněná hodnotou 0. Nevyužitá.
```

STANDROM

Tato rutina na první pohled nevykoná nic, ale má veliký význam. Pokud je přistránkována ROM D40, dojde při skoku na tento podprogram k přestránkování do ZX ROM. Je to tedy vstupní brána z ROM D40 do ZX ROM. V ZX ROM je také na adrese #1700 instrukce RET.

```
1700 STANDROM   RET                   ;přestránkuj do ZX ROM
```

RSAVE

Příkaz pro uložení bloku dat na disketu.

Syntaxe: SAVE * "[Zařízení:] JménoSouboru [.PříponaSouboru]" [LINE n] [CODE adresa, délka] [DATA pole] [SCREEN$]

Slouží k uchování dat na disketě. Má stejnou syntaxi jako pro ukládání dat na pásku, ale musí obsahovat znak „*“. Nelze ukládat bezhlavičkové soubory. Každý soubor má jméno. Nesmí obsahovat wildchars. Na disketě nelze mít dva soubory se stejným jménem.

```
1701 RSAVE      LD    A, #00              ;do A příznak SAVE
```

Nyní je to velice zajímavé. Tím, že je zde byte #33, jsou další vstupní body změněny na neškodné instrukce LD HL, hodnota.

```
1703            DB    #33
```

Místo vstupních bodů příkazů LOAD a MERGE je toto:

```
    1703        LD    HL, #013E
    1706        LD    HL, #033E
```

RLOAD

Příkaz pro nahrání bloku dat ze souboru do paměti.

Syntaxe: LOAD * "[Zařízení:] JménoSouboru [.PříponaSouboru]" [DATA pole] [SCREEN$] [CODE [adresa] [, délka]]

Slouží k nahrání dat do paměti. Musí být vloženo jméno souboru. Nesmí obsahovat wildchars.

```
1704 RLOAD      LD    A, #01              ;do A příznak LOAD
```

Nyní je použito něco podobného jako u vstupního bodu příkazu SAVE.

```
1706            DB    #33
```

Takže místo vstupního bodu pro MERGE je zde

```
    1706        LD    HL, #033E
```

RMERGE

Příkaz pro přihrání dat do paměti.

Syntaxe: MERGE * "[Zařízení:] JménoSouboru [.PříponaSouboru]"

Slouží k prihrání BASIC programu, k už k existujícímu programu v paměti, se jménem JménoSouboru. Nelze použít wildchars.

```
1707 RMERGE     LD    A, #03              ;do A příznak MERGE
```

Společná část pro rutiny SAVE, LOAD, MERGE. kontroluje se syntaxe a povolené kombinace. Vychází se z ZX ROM.

```
1709 SLMSYNTAX  LD    (#5C74), A T_ADDR   ;ulož si kód operace
170C            RST   #18                 ;vezmi aktuální znak
170D            CP    „*“                 ;je to hvězdička?
170F            JP    NZ, #06C6, REPORTC  ;ne → skoč na REPORT C „Nonsense in BASIC“
1712            RST   #20                 ;vezmi další znak
1713            RST   #28                 ;volej podprogram pro volání rutiny ZX ROM
1714            DW    #1C8C               ;podprogram EXPT-EXP vyhodnocení řetězce
```

Zkontrolujeme povolené kombinace.

```
1716 SLMSTART   RST   #30                 ;otestuj, jestli je kontrola syntaxe
```

```
1717            JR   Z, #1739, SAVEDATA  ;ano → skoč
1719            LD   BC, #0011          ;prostor 17 bytů pro hlavičku operace SAVE
171C            LD   A, (#5C74) T_ADDR  ;do A typ operace
171F            AND  A                  ;je SAVE?
1720            JR   Z, #1724, SAVESPACE ;ano → skoč
1722            LD   C, #22             ;prostor 34 bytu pro hlavičku operací LOAD, MERGE
1724 SAVESPACE  RST  #28                ;volej podprogram pro volání rutiny ZX ROM
1725            DW   #0030              ;podprogram BC-SPACE vytvoření BC prostoru pro
                                        ;hlavičku v pracovní oblasti
1727            PUSH DE                 ;ulož si ukazatel na začátek vytvořeného prostoru
1728            POP  IX                 ;a dej ho do IX
172A            LD   B, #0B             ;délka názvu souboru je 11 bytů
172C            LD   A, #20             ;vyplň mezerami
172E SAVEBLANK  LD   (DE), A            ;ulož mezeru
172F            INC  DE                 ;posuň se na další byte
1730            DJNZ #172E, SAVESPACE   ;opakuj B-krát
1732            LD   (IX+#01), #FF      ;jako první znak jména si ulož 255
1736            CALL #1A3D, SLMASTR     ;rozděl řetězec na zásobníku na jméno disku a souboru
1739 SAVEDATA   RST  #18                ;vezmi aktuální znak
173A            CP   #E4                ;DATA?
173C            JR   NZ, #1788, SAVESCRN ;ne → testuj další možnosti
173E            LD   A, (#5C74) T_ADDR  ;do A kód operace
1741            CP   #03                ;je MERGE?
1743            JP   Z, #06C6, REPORTC  ;ano → skoč na REPORT C „Nonsense in BASIC“
                                        ;MERGE DATA je nepřípustná kombinace
1746            RST  #20                ;vezmi další znak
1747            RST  #28                ;volej podprogram pro volání rutiny ZX ROM
1748            DW   #28B2              ;podprogram LOOK-VARS hledej proměnnou podle
                                        ;CH_ADD – najdeme adresu uložení pole v paměti
174A            SET  7, C               ;nastav 7. bit v názvu pole
174C            JR   NC, #175C, SAVEOLD ;existuje už taková proměnná → skoč
174E            LD   HL, #0000          ;do HL signál – „užití nového pole“
1751            LD   A, (#5C74) T_ADDR  ;do A kód operace
1754            DEC  A                  ;je LOAD?
1755            JR   Z, #176D, SAVENEW  ;ano → skoč
```

Pokud jede o SAVE neexistujícího pole, tiskne se chybové hlášení.

```
1757            LD   A, #01             ;REPORT 2 „Variable not found“
1759            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

Pokračujeme v práci s existujícím polem.

```
175C SAVEOLD    JP   NZ, #06C6, REPORTC ;není to pole nebo řetězec → skoč na REPORT C
                                        ;„Nonsense in BASIC“
175F            RST  #30                ;otestuj, jestli je kontrola syntaxe
1760            JR   Z, #177A, SAVEDATA1 ;ano → skoč
1762            INC  HL                 ;nižší byte délky pole
1763            LD   A, (HL)            ;dáme do A
1764            LD   (IX+#0B), A ;a uložíme do hlavičky jako délku souboru
1767            INC  HL                 ;nyní vyšší byte délky proměnné
1768            LD   A, (HL)            ;dáme do A
1769            LD   (IX+#0C), A        ;a uložíme do hlavičky jako délku souboru
176C            INC  HL
```

Další část je stejná pro „nová“ i „stará“ pole.

```
176D SAVENEW    LD    (IX+#0E), C      ;kopie jména pole do hlavičky
1770            LD    A, #01           ;do A typ pole – „číselné pole“
1772            BIT   6, C             ;bylo pole číselné?
1774            JR    Z, #1777, SAVETYPE ;ano → skoč
1776            INC   A                ;do A typ pole – „znakové pole“
1777 SAVETYPE   LD    (IX+#00), A      ;ulož typ pole do hlavičky na první adresu
```

Test poslední části příkazu.

```
177A SAVEDATA1  EX    DE, HL           ;ulož ukazatele do DE
177B            RST   #20              ;vezmi další znak
177C            CP    „)“              ;je to pravá závorka?
177E            JR    NZ, #175C, SAVEOLD ;ne → skoč na REPORT C „Nonsense in BASIC“
1780            RST   #20              ;vezmi další znak
1781            CALL  #1057, ISSYNCONTR ;otestuj, jestli není kontrola syntaxe
1784            EX    DE, HL           ;začátek uložení dat zpět do HL
1785            JP    #1842, SAVEALL   ;skoč na výběr operace
```

Test na SCREEN$.

```
1788 SAVESCRN   CP    #AA              ;SCREEN$?
178A            JR    NZ, #17AB, SAVECODE;ne → skoč na další možnosti
178C            LD    A, (#5C74) T_ADDR ;do A kód operace
178F            CP    #03              ;je MERGE?
1791            JP    Z, #06C6, REPORTC ;ano → skoč na REPORT C „Nonsense in BASIC“
                                       ;MERGE SCREEN$ je nepřípustná kombinace
1794            RST   #20              ;vezmi další znak
1795            CALL  #1057, ISSYNCONTR ;otestuj, jestli není kontrola syntaxe
1798            LD    (IX+#0B), #00    ;do hlavičky ulož délku dat 6912 bytů
179C            LD    (IX+#0C), #1B
17A0            LD    HL, #4000        ;do HL počáteční adresa 16384
17A3            LD    (IX+#0D), L      ;ulož do hlavičky jako počátek dat
17A6            LD    (IX+#0E), H
17A9            JR    #17F8, SAVETYPE3 ;skoč na nastavení typu dat a výběr operace
```

Test na CODE.

```
17AB SAVECODE   CP    #AF              ;CODE?
17AD            JR    NZ, #17FE, SAVELINE ;ne → skoč na další možnosti
17AF            LD    A, (#5C74) T_ADDR ;do A kód operace
17B2            CP    #03              ;je MERGE?
17B4            JP    Z, #06C6, REPORTC ;ano → skoč na REPORT C „Nonsense in BASIC“
                                       ;MERGE CODE je nepřípustná kombinace
17B7            RST   #20              ;vezmi další znak
17B8            RST   #28              ;volej podprogram pro volání rutiny ZX ROM
17B9            DW    #2048            ;podprogram PR-ST-END test konce příkazu
17BB            JR    NZ, #17C9, SAVECODE1;není konec → skoč
```

Příkaz nemá parametry.

```
17BD            LD    A, (#5C74) T_ADDR ;do A kód operace
17C0            AND   A                ;je SAVE?
17C1            JP    Z, #06C6, REPORTC ;ano → skoč na REPORT C „Nonsense in BASIC“
                                       ;při SAVE musí být parametry
17C4            RST   #28              ;volej podprogram pro volání rutiny ZX ROM
```

```
17C5                DW   #1CE6                ;podprogram USE-ZERO uložení 0 na zásobník pro
                                               ;startovací adresu
17C7                JR   #17D8, SAVECODE2     ;skoč na zpracování počtu bytů
```

Hledání startovací adresy.

```
17C9 SAVECODE1      RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
17CA                DW   #1C82                ;podprogram EXPT-1NUM vyhodnocení číselného
                                               ;výrazu
17CC                RST  #18                  ;vezmi aktuální znak
17CD                CP   „ , “                ;čárka?
17CF                JR   Z, #17DD, SAVECODE3  ;ano → skoč
17D1                LD   A, (#5C74) T_ADDR    ;do A kód operace
17D4                AND  A                    ;je SAVE?
17D5                JP   Z, #06C6, REPORTC    ;ano → skoč na REPORT C „Nonsense in BASIC“
                                               ;při SAVE musí být uvedena délka souboru
17D8 SAVECODE2      RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
17D9                DW   #1CE6                ;podprogram USE-ZERO uložení nuly na zásobník jako
                                               ;počet bytů
17DB                JR   #17E1, SAVECODE4     ;skoč na zpracovávání délky dat
```

Zjištění počtu bytů.

```
17DD SAVECODE3      RST  #20                  ;vezmi další znak
17DE                RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
17DF                DW   #1C82                ;podprogram EXPT-1NUM vyhodnocení číselného
                                               ;výrazu
17E1 SAVECODE4      CALL #1057, ISSYNCONTR    ;otestuj, jestli není kontrola syntaxe
17E4                RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
17E5                DW   #1E99                ;podprogram FIND-INT2 vyzvednutí čísla do BC
                                               ;v BC je délka dat
17E7                LD   (IX+#0B), C          ;ulož délku dat do hlavičky
17EA                LD   (IX+#0C), B
17ED                RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
17EE                DW   #1E99                ;podprogram FIND-INT2 vyzvednutí čísla do BC
                                               ;v BC je počáteční adresa dat
17F0                LD   (IX+#0D), C          ;ulož počáteční adresu do hlavičky
17F3                LD   (IX+#0E), B
17F6                LD   H, B                 ;zkopíruj do HL počáteční adresu
17F7                LD   L, C
```

SCREEN a CODE jsou stejného typu.

```
17F8 SAVETYPE3      LD   (IX+#00), #03        ;ulož typ dat – „kód“
17FC                JR   #1842, SAVEALL       ;skoč na výběr operace
```

Zjištění přítomnosti LINE a dalších možných parametrů.

```
17FE SAVELINE       CP   #CA                  ;LINE?
1800                JR   Z, #180B, SAVELINE1  ;ano → skoč
1802                CALL #1057, ISSYNCONTR    ;otestuj, jestli není kontrola syntaxe
1805                LD   (IX+#0E), #80        ;ulož „nejsou další parametry“
1809                JR   #1822, SAVETYPE0     ;skoč na nastavení BASIC parametrů
```

Zjistíme číslo, které musí následovat po LINE.

```
180B SALINE1        LD   A, (#5C74) T_ADDR    ;do A kód operace
```

```
180E            AND  A                      ;je SAVE?
180F            JP   NZ, #06C6, REPORTC     ;ne → skoč na REPORT C „Nonsense in BASIC“
                                            ;LOAD LINE nebo MERGE LINE jsou nepřípustné
1812            RST  #20                    ;vezmi další znak
1813            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
1814            DW   #1C82                  ;podprogram EXPT-1NUM vyhodnocení číselného
                                            ;výrazu – číslo startovního řádku
1816            CALL #1057, ISSYNCONTR      ;otestuj, jestli není kontrola syntaxe
1819            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
181A            DW   #1E99                  ;podprogram FIND-INT2 vyzvednutí čísla do BC
181C            LD   (IX+#0D), C            ;ulož číslo startovního řádku do hlavičky
181F            LD   (IX+#0E), B
```

LINE a žádný parametr jsou typu 0.

```
1822 SAVETYPE0  LD   (IX+#00), #00          ;ulož typ dat typ dat BASIC program
```

Vyzvedneme parametry určující BASIC program a uložíme je do hlavičky

```
1826            LD   HL, (#5C59) E-LINE     ;do HL adresa konce oblasti proměnných
1829            LD   DE, (#5C53) PROG       ;do DE adresa začátku BASIC programu
182D            SCF                         ;nastav CY
182E            SBC  HL, DE                 ;vypočti délku BASIC programu (BASIC+proměnné)
1830            LD   (IX+#0B), L            ;ulož do hlavičky jako délku dat
1833            LD   (IX+#0C), H
1836            LD   HL, (#5C4B) VARS       ;do HL adresa začátek proměnných
1839            SBC  HL, DE                 ;vypočti délku BASIC programu bez proměnných
183B            LD   (IX+#0F), L            ;ulož do hlavičky
183E            LD   (IX+#10), H
1841            EX   DE, HL                 ;přesuň ukazatel do HL
```

Byly vytvořeny hlavičkové informace.
Tvar hlavičky: IX+#00 – typ dat
IX+#01 – 10 – jméno souboru
IX+#0B – 12 – počet bytů
IX+#0D – 16 – různé parametry, které záleží na typu dat
Nyní se separuje jedna z operací

```
1842 SAVEALL    LD   A, (#5C74) T_ADDR      ;do A kód operace
1845            AND  A                      ;je SAVE?
1846            JP   Z, #19FA, SAVECONTR    ;ano → skoč
```

V případě dalších operací je prvních 17 bytů hlavičkové části pracovní oblasti vybaveno informacemi, jak bylo uvedeno výše.

```
1849            PUSH HL                     ;ulož ukazatel na začátek dat
184A            LD   BC, #0011              ;formování 1. adresy
184D            ADD  IX, BC                 ;do IX adresa 2. hlavičkové oblasti
184F            PUSH IX                     ;ulož si ukazatel na 2. hlavičkovou oblast
1851            CALL #196B, LOAR01          ;zjisti, jestli je soubor na disketě, načti jeho hlavičku do
                                            ;2. hlavičkové oblasti
1854            POP  IX                     ;obnov ukazatel na 2. hlavičku
1856            POP  HL                     ;obnov ukazatel na začátek dat
```

V IX je ukazatel na 2. hlavičku, v HL je 0, jestli se jedná o nové pole nebo začátek BASIC programu.

```
1857            LD   A, (IX+#00)          ;vyzvedni typ dat
185A            CP   #03                  ;je to CODE?
185C            JR   Z, #186A, VERIFYCONT;ano → skoč
185E            LD   A, (#5C74) T_ADDR    ;do A kód příkazu
1861            DEC  A                    ;je LOAD?
1862            JP   Z, #189E, LOADCONT   ;ano → skoč
1865            CP   #02                  ;je MERGE?
1867            JP   Z, #1949, MERGECONT;ano → skoč
```

Proces verifikace.

```
186A VERIFYCONT PUSH HL                   ;ulož si ukazatel na začátek dat
186B            LD   L, (IX–#06)          ;vyzvedni počet bytů ze staré hlavičky do HL
186E            LD   H, (IX–#05)
1871            LD   E, (IX+#0B)          ;a vyzvedni počet bytů z nové hlavičky do DE
1874            LD   D, (IX+#0C)
1877            LD   A, H                 ;je délka neurčena?
1878            OR   L
1879            JR   Z, #1888, VERCONT1   ;ano → skoč dále
187B            SBC  HL, DE               ;porovnej délky
187D            JR   C, #1899, REPORTx    ;je větší než je požadováno → skoč na REPORT x „File
                                          ;too long“
187F            JR   Z, #1888, VERCONT1   ;jsou shodné → skoč
1881            LD   A, (IX+#00)          ;chyba se bude také tisknout, jestli budeme nahrávat
1884            CP   #03                  ;blok s nestejnou délkou
1886            JR   NZ, #1899, REPORTx   ;ne → skoč na REPORT x „File too long“
1888 VERCONT1   POP  HL                   ;obnov ukazatel na začátek dat
1889            LD   A, H                 ;je roven nule?
188A            OR   L
188B            JR   NZ, #1893, VERCONT2  ;ne → skoč
188D            LD   L, (IX+#0D)          ;vyzvedni nový začátek dat z hlavičky
1890            LD   H, (IX+#0E)
```

Nahrajeme blok dat.

```
1893 VERCONT2   PUSH HL                   ;přesuň ukazatel na začátek dat
1894            POP  IX                   ;do IX
1896            JP   #19AE, LOADBLOCK     ;skoč na nahrání souboru do paměti
1899 REPORTx    LD   A, #40               ;REPORT x „File too long“
189B            JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Tento podprogram řídí čtení basicovského programu včetně proměnných

```
189E LOADCONT   LD   E, (IX+#0B)          ;vyvedni do DE délku bloku z nové hlavičky
18A1            LD   D, (IX+#0C)
18A4            PUSH HL                   ;ulož si ukazatel na začátek dat
18A5            LD   A, H                 ;je čteno již deklarované pole?
18A6            OR   L
18A7            JR   NZ, #18AF, LOADCNT1  ;ano → skoč
18A9            INC  DE                   ;přidej 3 byty pro novou proměnnou
18AA            INC  DE                   ;místo pro jméno a délku proměnné
18AB            INC  DE
18AC            EX   DE, HL               ;dej délku bloku do HL, do DE začátek dat
18AD            JR   #18BB, LOADCONT2     ;skoč na test volného místa
```

Nyní je v paměti dost místa pro blok dat.

```
18AF LOADCNT1   LD   L, (IX–#06)          ;do HL vezmi délku existujícího BASIC programu
18B2            LD   H, (IX–#05)          ;v paměti včetně proměnných
18B5            EX   DE, HL               ;a dej ji do DE
18B6            SCF                       ;nastav CY
18B7            SBC  HL, DE               ;je vyžadována paměť navíc?
18B9            JR   C, #18C4, LOADDATA   ;ne → skoč
```

Otestuje volné místo.

```
18BB LOADCONT2  LD   DE, #0005            ;5 bytů navíc
18BE            ADD  HL, DE               ;přičti k počtu bytů, které jsou navíc
18BF            LD   B, H                 ;výsledek dej do BC
18C0            LD   C, L
18C1            RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
18C2            DW   #1F05                ;podprogram TEST-ROOM test, jestli je dost prostoru
18C4 LOADDATA   POP  HL                   ;obnov ukazatel na začátek dat
18C5            LD   A, (IX+#00)          ;do A typ souboru
18C8            AND  A                    ;čte se BASIC program?
18C9            JR   Z, #1909, LOADPRG    ;ano → skoč
18CB            LD   A, H                 ;čte se nové pole?
18CC            OR   L
18CD            JR   Z, #18E2, LOADDATA1  ;ano → skoč
18CF            DEC  HL                   ;vezmi do BC délku existujícího pole
18D0            LD   B, (HL)
18D1            DEC  HL
18D2            LD   C, (HL)
18D3            DEC  HL                   ;ukazatel posuň na jméno
18D4            INC  BC                   ;přičti 3 byty pro délku a jméno
18D5            INC  BC
18D6            INC  BC
18D7            LD   (#5C5F), IX X-PTR    ;ulož si IX po dobu rušení staré proměnné
18DB            RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
18DC            DW   #19E8                ;podprogram RECLAIM-2 zrušení prostoru – proměnné
18DE            LD   IX, (#5C5F) X_PTR    ;obnov si IX
```

Nyní je vyhrazeno místo pro novou proměnnou – na konci platné oblasti pro data.

```
18E2 LOADDATA1  LD   HL, (#5C59) E-LINE   ;do HL konec oblasti proměnných
18E5            DEC  HL                   ;který je označen bytem #80
18E6            LD   C, (IX+#0B)          ;do BC ulož délku nového pole
18E9            LD   B, (IX+#0C)
18EC            PUSH BC                   ;ulož si délku
18ED            INC  BC                   ;nyní přičti 3 byty pro jméno a délku
18EE            INC  BC
18EF            INC  BC
18F0            LD   A, (IX–#03)          ;zjisti název pole
18F3            PUSH AF                   ;a uschovej ho
18F4            RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
18F5            DW   #1655                ;podprogram MAKE-ROOM vytvoř prostor pro
                                          ;proměnnou – bude zabírat BC bytů
18F7            INC  HL                   ;HL nyní ukazuje na 1. byte pole, kam bude uloženo
                                          ;jméno pole
```

```
18F8            POP   AF                    ;vyzvedni jméno pole
18F9            LD    (HL), A               ;a ulož ho
18FA            POP   DE                    ;vyzvedni délku pole
18FB            INC   HL                    ;posuň ukazatel na adresu uložení délky pole
18FC            LD    (HL), E               ;ulož délku pole
18FD            INC   HL
18FE            LD    (HL), D
18FF            INC   HL                    ;HL ukazuje na adresu, kam se už bude ukládat obsah
1900            PUSH  HL                    ;pole, přesuň ho do IX
1901            POP   IX
1903            SCF                         ;signál LOAD
1904            LD    A, #FF                ;signál blok dat
1906            JP    #19AE, LOADBLOCK      ;pokračuj nahráním souboru do paměti
```

LOAD BASICovského programu a jeho proměnných.

```
1909 LOADPROG   EX    DE, HL                ;uschovej si 1. adresu začátku programu do DE
190A            LD    HL, (#5C59) E-LINE    ;do HL adresa konce oblasti proměnných
190D            DEC   HL                    ;který je označen bytem #80
190E            LD    (#5C5F), IX X_PTR     ;ulož si IX
1912            LD    C, (IX+#0B)           ;do BC vyzvedni délku programu
1915            LD    B, (IX+#0C)
1918            PUSH  BC                    ;ulož si délku programu
1919            RST   #28                   ;volej podprogram pro volání rutiny ZX ROM
191A            DW    #19E5                 ;podprogram RECLAIM-1 zrušení prostoru zruší starý
                                            ;BASIC program
191C            POP   BC                    ;obnov délku dat
191D            PUSH  HL                    ;ulož ukazatel programové oblasti a délku programu
191E            PUSH  BC
191F            RST   #28                   ;volej podprogram pro volání rutiny ZX ROM
1920            DW    #1655                 ;podprogram MAKE-ROOM vytvoření prostoru
                                            ;pro BASIC program a jeho proměnné
1922            LD    IX, (#5C5F) X_PTR     ;obnov IX
1926            INC   HL
1927            LD    C, (IX+#0F)           ;vyzvedni délku BASIC programu bez proměnných
192A            LD    B, (IX+#10)           ;do BC
192D            ADD   HL, BC                ;přičti k začátku uložení BASIC programu
192E            LD    (#5C4B), HL VARS      ;nastav novou adresu začátku proměnných
1931            LD    H, (IX+#0E)           ;otestuj, jestli
1934            LD    A, H                  ;byl přítomen příkaz LINE?
1935            AND   #C0
1937            JR    NZ, #1943, LOADPRG1   ;ne → skoč
```

Nastavíme číslo řadku a spustíme program.

```
1939            LD    L, (IX+#0D)           ;vyzvedni do HL číslo řádky, odkud se bude startovat
193C            LD    (#5C42), HL NEWPPC    ;a ulož ho systémových proměnných
193F            LD    (IY+#0A), #00 NSPPC   ;nastav pořadová číslo příkazu v řádce na první příkaz
```

Nyní můžeme načíst blok dat.

```
1943 LOADPRG1   POP   DE                    ;obnov délku souboru
1944            POP   IX                    ;obnov ukazatel na první byte uložení souboru
1946            JP    #19AE, LOADBLOCK      ;pokračuj nahráním souboru do paměti
```

Nahraje soubor do WORKSPACE a vrátí řízení do ZX ROM pro provedení ostatních kroků příkazu MERGE.

```
1949 MERGECONT LD   C, (IX+#0B)          ;vyzvedni délku nového bloku do BC
194C           LD   B, (IX+#0C)
194F           PUSH BC                   ;ulož ji
1950           INC  BC                   ;vytvoř prostor délka+1
1951           RST  #28                  ;volej podprogram pro volání rutiny ZX ROM
1952           DW   #0030                ;podprogram BC-SPACES vytvoření BC prostoru
1954           LD   (HL), #80            ;do bytu „navíc“ ulož #80 (koncový znak)
1956           EX   DE, HL               ;přesuň začátek volného místa do HL
1957           POP  DE                   ;obnov délku souboru
1958           PUSH HL                   ;ulož si začátek volného místa
1959           PUSH HL                   ;a ještě jednou
195A           POP  IX                   ;a dej ho do IX
195C           CALL #19AE, LOADBLOCK     ;nahraj soubor do paměti
195F           CALL #2536, DSKSTP        ;zastav mechaniky
1962           POP  HL                   ;vyzvedni adresu začátku uložených dat
1963           PUSH HL                   ;a znovu hu ulož
1964           LD   HL, #08CE            ;návratová adresa do počítače, kde se dále zpracuje
                                         ;nahraný soubor
1967           EX   (SP), HL             ;dej ji na zásobník a obnov do HL začátek uložených dat
1968           JP   #1700, STANDROM      ;skoč na přestránkování do ZX ROM
```

LOAR01

Zjistí, jestli je soubor na disketě.

IN: IX adresa hlavičky hledaného souboru +17. Hlavička je pásková
BC musí obsahovat hodnotu 17
ve DNZONE1 je jméno disku, ve FNZONE1 je jméno hledaného souboru

OUT: prohledá adresář a hledá soubor, pokud existuje, natáhne od adresy v IX hlavičku (prvních 17 byte z diskové hlavičky)
#3E72 (SVADRA) adresa položky adresáře v buferu

```
196B LOAR01    PUSH HL                   ;ulož si HL
196C           PUSH IX                   ;a ukazatel na hlavičku
196E           LD   A, (IX-#11)          ;do A přečti páskový typ souboru z 1. hlavičky
1971           LD   (IX+#00), A          ;ulož ji pro převod do 2. hlavičky
1974           CALL #19D0, FINTYP        ;převeď páskovou příponu na diskovou příponu
1977           CALL #1C8F, SETACT        ;roztoč mechaniku, která má stejné jméno jako
                                         ;v DNZONE1
197A           CALL #212B, FIRSTMASK     ;načti položku adresáře vyhovující masce v FNZONE1
197D           JR   NZ, #1995, TSTSNP    ;neexistuje → skoč
197F           LD   (#3E72), HL SVADRA   ;ulož adresu položky adresáře v buferu do SRAM
1982           POP  IX                   ;obnov ukazatel na hlavičku
1984           PUSH IX                   ;a znovu ho ulož
1986           POP  DE                   ;a vyzvedni do DE
1987           LDIR                      ;přenes 17 bytů položky do 2. hlavičky
1989           LD   L, (IX-#11)          ;příponu páskové hlavičky z 1. hlavičky
198C           LD   (IX+#00), L          ;ulož do 2. hlavičky
198F           XOR  A                    ;do A nulu
1990           LD   (#3E6A), A VARIA3    ;ulož, není ale nijak použito
1993           POP  HL                   ;obnov HL
1994           RET                       ;vrať se
```

Soubor nebyl nalezen. Jestli byl hledán program v BASICu, zkusíme ještě hledat soubor s příponou „S“.

```
1995 TESTSNP    LD    A, (#3E94) EXTE1      ;vyzvedni příponu souboru z diskové hlavičky
1998            CP    „P“                   ;je to „P“ (hlavička programu v BASICu)?
199A            JP    NZ, #1FB1, REPORTS    ;ne → skoč na REPORT S „File not found“
199D            LD    A, „S“                ;zkus to se snapem
199F            LD    (#3E94), A EXTE1      ;ulož novou příponu souboru do diskové hlavičky
19A2            CALL #212B, FIRSTMASK       ;načti položku adresáře vyhovující masce v FNZONE1
19A5            JP    NZ, #1FB1, REPORTS    ;neexistuje → skoč na REPORT S „File not found“
19A8            LD    (#3E72), HL SVADRA    ;ulož adresu položky adresáře v buferu do SRAM
19AB            JP    #0394, SNPLOA         ;skoč na LOAD snapu
```

LOADBLOCK

Nahraje soubor do paměti. V IX je adresa začátku dat, v DE je délka dat, na adrese #3E72 je adresa uložení položky adresáře v buferu.

IN: IX adresa začátku uložení dat
DE délka dat
#3E72 (SVADRA) adresa položky adresáře v buferu
v DNZONE1 je jméno disku
OUT: nahraje blok do paměti

```
19AE LOADBLOCK LD   (#3E74), IX STARTADR  ;ulož si počáteční adresu uložení dat
19B2           LD   (#3E76), DE LENDAT    ;ulož si délku dat
19B6           CALL #1C8F, SETACT         ;roztoč mechaniku, která má stejné jméno jako
                                          ;v DNZONE1
19B9           LD   HL, (#3E74) STARTADR  ;do HL počáteční adresu dat
19BC           LD   DE, (#3E76) LENDAT    ;do DE délku dat
19C0           CALL #1FA5, LOAFND         ;nahrej data ze souboru
```

Zvýšíme adresu uložení dat o délku dat.

```
19C3 LOADBEND   LD   IX, (#3E74) STARTADR  ;do IX počáteční adresa uložení dat
19C7            LD   DE, (#3E76) LENDAT    ;do DE délka dat
19CB            ADD  IX, DE                ;přičti délku k počáteční adrese
19CD            XOR  A                     ;nastav NZ a A=0
19CE            SCF                        ;nastav C
19CF            RET                        ;vrať se
```

FINTYP

Převede páskovou příponu na diskovou.

IN: IX adresa páskové hlavičky souboru
OUT: A disková přípona
EXTE1 zde je také uložena disková přípona

```
19D0 FINTYP     LD   A, (IX+#00)           ;do A dej typ dat z páskové hlavičky
19D3            LD   HL, #10DB EXTTAB      ;do HL adresu přípon v ROM
19D6            CALL #0FAD, ADDHLA         ;posuň se na znak přípony
19D9            LD   A, (HL)               ;vyzvedni znak přípony
19DA            LD   (#3E94), A EXTE1      ;ulož příponu souboru
19DD            RET                        ;vrať se
```

SAVESETPAR

Nastaví příponu souboru a vyzvedne parametry bloku dat.

IN: IX adresa páskové hlavičky souboru
OUT: DE délka dat
#3E78 (VALSYX) počáteční adresa
#3E7A (VALSYY) délka BASIC programu

```
19DE SAVESETPAR CALL #19D0, FINTYP       ;převeď typ na diskovou příponu
19E1            LD   L, (IX+#0D)         ;do HL vyzvedni počáteční adresu uložení dat
19E4            LD   H, (IX+#0E)
19E7            LD   (#3E78), HL VALSYX  ;a ulož ji do SRAM
19EA            LD   L, (IX+#0F)         ;do HL vyzvedni délku BASIC programu
19ED            LD   H, (IX+#10)
19F0            LD   (#3E7A), HL VALSYY  ;a ulož ji do SRAM
19F3            LD   E, (IX+#0B)         ;do DE vyzvedni délku souboru
19F6            LD   D, (IX+#0C)
19F9            RET                      ;vrať se
```

SAVECONTR

Podprogram pro uložení bloku dat na disk.

IN: IX adresa páskové hlavičky souboru
HL adresa začátku ukládaných dat
v DNZONE1 je jméno disku a v FNZONE1 je jméno souboru
OUT: data jsou uložena do souboru, na jehož páskovou hlavičku ukazuje IX

Vyzvedneme základní parametry z hlavičky.

```
19FA SAVECONTR LD   (#3E74), HL STARTADR ;ulož si začátek dat
19FD           CALL #19DE, SAVESETPAR    ;nastav příponu jména souboru a parametry bloku
1A00 SAVRUN    PUSH DE                   ;ulož si délku dat
1A01           CALL #1C8F, SETACT        ;roztoč mechaniku, která má jméno jako v DNZONE1
1A04           CALL #212B, FIRSTMASK     ;načti 1. položku adresáře vyhovující masce v FNZONE1
1A07           JR   NZ, #1A33, SAVNODEL  ;nenalezena → skoč
```

Protože už existuje soubor se stejným jménem, otestujeme, jestli se do něj může zapisovat.

```
1A09           CALL #1283, GETATR        ;vyzvedni atributy souboru
1A0C           BIT  2, A                 ;je WRITE PROTECTED?
1A0E           JR   NZ, #1A15, SAVRUN1   ;ne → skoč
1A10 REPORTf   LD   A, #2E               ;REPORT f „File is write protected“
1A12           JP   #0204, ERRR          ;piš hlášení a skoč do ZX ROM
```

Nyní otestujeme, jestli se tiskne dotaz na přepsání souboru. Pokud ne, je přímo přepsán, jinak se systém zeptá.

```
1A15 SAVRUN1   LD   A, (#3EEE) SNAPINF   ;vyzvedni informaci, jestli jde o uložení snapu
1A18           AND  A                    ;je to uložení snapu?
1A19           JR   NZ, #1A30, SAVNOASK  ;ano → skoč přímo na přepsání bez dotazu
1A1B           LD   A, (#3E62) AIFASK    ;vyzvedni informaci, jestli se ptát na přepsání souboru
1A1E           AND  A                    ;ptát se?
1A1F           JR   NZ, #1A30, SAVNOASK  ;ne → skoč
```

Provedeme dotaz na přepsání souboru.

```
1A21           PUSH HL                   ;ulož si parametry
1A22           PUSH DE
1A23           LD   DE, #03AF SYSMSG     ;do DE adresa systémových hlášení
1A26           LD   A, #BE               ;do A číslo položky „Rewrite old file?“
                                         ;a dotaz „(Proceed = P)“
```

```
1A28          CALL #21BF, KEYMSG      ;tiskni a čekej na klávesu
1A2B          POP  DE                 ;obnov parametry
1A2C          POP  HL
1A2D          JP   NC, #1019, REPORTF ;nepřepsat → skoč na REPORT F „Invalid file name“
1A30 SAVNOASK CALL #1F88, DFILER      ;smaž starý soubor z diskety
```

Uložíme soubor na disketu.

```
1A33 SAVNODEL POP  DE                 ;obnov si délku dat
1A34          LD   HL, (#3E74) STARTADR ;do HL adresa začátku uložení dat
1A37          CALL #2046, SAVEFILE    ;ulož soubor
1A3A          JP   #19C3, LOADBEND    ;skoč na ukončení rutiny a nastavení registrů
```

SLMANALSTR

Rozdělí řetězec na zásobníku na jméno disku a souboru a analyzuje je pro příkazy SAVE, LOAD, MERGE.

```
1A3D SLMASTR  CALL #0FCF, DIVSTRING   ;rozděl řetězec na zásobníku na jméno disku a souboru
1A40          CALL #1043, SETWDNM     ;nastav jméno disku v DNZONE1 pro I/O
1A43          CALL #10E2, ANALWDNM    ;analyzuj jméno disku v DNZONE1
1A46          JR   Z, #1A49, SLMNODR  ;nebylo vloženo jméno disku → skoč
1A48          INC  A                  ;nyní měl asi být test na správnost mechaniky, ale není
```

Tady mělo být **JP Z, #2337, REPORTX**.

```
1A49 SLMNODR  EX   AF, AF'            ;nic neznamenající instrukce, ani autor sám neví
1A4A          CALL #107C, ARRANGNM    ;uprav jméno souboru v FNZONE1 na masku
1A4D          JP   NZ, #1019, REPORTF ;nebylo vloženo → skoč na REPORT F „Invalid file name“
1A50          JP   C, #1019, REPORTF  ;použity wildchars → skoč na REPORT F „Invalid file
1A53          RET                     ;name“, vrať se
```

MOVE

Druhá varianta příkazu MOVE, kopíruje soubory.

Syntaxe: MOVE "JménoDisku1:MaskaSouboru1","JménoDisku2:[JménoSouboru2]"

Příkaz slouží ke kopírování souborů z diskety na disketu. „JménoDisku1“ je jméno zdrojového disku, JménoDisku2 je jméno cílového disku. Jsou zkopírovány všechny soubory, které vyhovují první masce. Pokud není „JménoSouboru2“ zadáno, je soubor uložen pod stejným jménem na cílovém disku. V „MasceSouboru“ lze použít wildchars (hvězdičkovou konvenci).

```
1A54 COPYF    CALL #1C6B, SETCOPYNM   ;analyzuj druhý řetězec
1A57          PUSH AF                 ;ulož si příznaky zpracování cílové masky souboru
1A58          LD   HL, #3E80 DNZONE1  ;do HL adresa 1. jména disku pro I/O
1A5B          LD   DE, #3E95 DNZONE2  ;do DE adresa 2. jména disku pro I/O
1A5E          LD   BC, #15            ;jméno disku a maska má 21 bytů
1A61          LDIR                    ;přenes jméno cílového disku a nové jméno souboru
1A63          CALL #1C6B, SETCOPYNM   ;zpracuj první řetězec
1A66          PUSH AF                 ;ulož si příznaky zpracování zdrojové masky souboru
1A67          LD   HL, #3E80 DNZONE1  ;do HL adresa jména zdrojového disku
1A6A          LD   DE, #3E95 DNZONE2  ;do DE adresa jména cílového disku
1A6D          LD   BC, #0A            ;délka jména disku je 10 bytů
1A70          CALL #1F0E, VERIFY      ;porovnej je
1A73          JR   NZ, #1A9A, COPYF2  ;rozdílné → skoč
```

Kopíruje se na jedné disketě. Zkontrolujeme použití wildchars, protože při kopírování na jedné disketě se nesmí používat.

```
1A75            POP   AF                  ;obnov příznaky zpracování 1. masky
1A76            JP    C, #1019, REPORTF   ;byly použity wildchars →skoč na REPORT F „Invalid
                                          ;file name“
1A79            POP   AF                  ;obnov příznaky zpracování 2. masky
1A7A            JP    C, #1019, REPORTF   ;byly použity wildchars → skoč na REPORT F „Invalid
                                          ;file name“
```

Nyní porovnáme jména souborů, protože při kopírování na jedné disketě nesmí být stejné.

```
1A7D            LD    HL, #3E8A FNZONE1   ;do HL adresa první masky
1A80            LD    DE, #3E9F FNZONE2   ;do DE adresa druhé masky
1A83            LD    BC, #0A             ;délka jména je 10 znaků
1A86            CALL #1F0E, VERIFY        ;porovnej je
1A89            JP    Z, #1019, REPORTF   ;stejné → skoč na REPORT F „Invalid file name“
```

Nyní porovnáme přípony, protože ty musí zůstat taky stejné.

```
1A8C            LD    A, (#3E94) EXTE1    ;vyzvedni příponu první masky
1A8F            LD    HL, #3EA9 EXTE2     ;do HL adresa přípony druhé masky
1A92            CP    (HL)                ;porovnej je
1A93            JP    NZ, #1019, REPORTF  ;různé → skoč na REPORT F „Invalid file name“
1A96            LD    B, #FF              ;do B příznak kopírování jednoho souboru
1A98            JR    #1AA9, COPYF4       ;pokračuj v provádění příkazu
```

Kopíruje se z diskety na disketu. Nastavíme počet kopírovaných souborů.

```
1A9A COPYF2     POP   AF                  ;obnov příznaky zpracování 1. masky
1A9B            LD    B, #00              ;do B příznak kopírování více souborů
1A9D            JR    C, #1AA5, COPYF3    ;byly použity wildchars? ano → skoč
1A9F            POP   AF                  ;obnov příznaky zpracování 2. masky
1AA0            JR    C, #1AA9, COPYF4    ;byly použity wildchars? ano → skoč
1AA2            DEC   B                   ;do B příznak kopírování jednoho souboru
1AA3            JR    #1AA9, COPYF4       ;pokračuj v provádění
1AA5 COPYF3     POP   AF                  ;obnov příznaky zpracování 2. masky
1AA6            JP    NC, #1019, REPORTF  ;nejsou wildchars → skoč na REPORT F „Invalid file
                                          ;name“
```

Nyní zkopírujeme soubory. V B je buď #FF pro kopírování jednoho souboru, nebo #00 pro kopírování více souborů.

```
1AA9 COPYF4     LD    C, #00              ;nuluj počítadlo zkopírovaných souborů
1AAB            LD    (#C000H), BC        ;toto je jedna z nejhorších chyb v MDOSu. Sám autor
                                          ;by snad musel být v nedefinovaném stavu, aby zde
                                          ;tohle vložil. Jaké následky toto má, to si vydedukujte
                                          ;sami (jsou však hrozné)
```

V opravené verzi MDOSu 1.0 je tato instrukce opravena. Vypadá to takto:

```
1AAB            NOP
1AAC            NOP
1AAD            NOP
1AAE            NOP
```

Nyní zjistíme velikost volné paměti, kterou můžeme využívat při kopírování.

```
1AAF            PUSH BC                   ;ulož si počítadlo kopírovaných souborů
1AB0            LD    DE, (#5C65) STKEND  ;do DE začátek volné paměti
```

```
1AB4            LD    HL, (#5CB2) RAMTOP  ;do HL adresa posledního byte pro BASIC
1AB7            DEC   H                   ;sniž HL o 256
1AB8            SBC   HL, DE              ;odečti začátek volné paměti
1ABA            JR    NC, #1AC1, COPYF5   ;je nějaká volná paměť? ano → skoč
1ABC REPROT4    LD    A, #03              ;REPORT 4 „Out of memory“
1ABE            JP    #0204, ERRR         ;piš hlášení a skoč do ZX ROM
```

Otestujeme, jestli se nám tam vejde alespoň jeden sektor.

```
1AC1 COPYF5     LD    A, H                ;dej H do
1AC2            SRL   A                   ;vyděl dvěma – je to alespoň 512 bytů?
1AC4            JR    Z, #1ABC, REPORT4   ;ne → skoč na REPORT 4 „Out of memory“
1AC6            LD    (#3E78), A VALSYXlo ;ulož si počet sektorů, kolik jich jde načíst do paměti
1AC9            LD    (#3E7A), DE VALSYY  ;a ulož si začátek buferu pro kopírování
1ACD            LD    A, #FF              ;prohledávej od první položky adresáře
1ACF            LD    (#3E79), A VALSYXhi ;ulož si do SRAM
```

Najdeme soubor, který vyhovuje 1. masce.

```
1AD2 COPYLOOP   CALL  #1C8F, SETACT       ;roztoč zdrojovou mechaniku, která má stejné jméno
                                          ;jako v DNZONE1
1AD5            LD    A, (#3E79) VALSYXhi ;do A číslo položky, od které budeme prohledávat
                                          ;zdrojový disk
1AD8            CALL  #212D, NEXTMASK     ;načti položku adresáře od A vyhovující 1. masce
1ADB            LD    (#3E79), A VALSYXhi ;ulož si číslo nalezené položky
1ADE            JP    NZ, #1C06, ENDCOPY  ;nenalezena → skoč
1AE1            CALL  #1283, GETATR       ;vyzvedni atributy souboru
1AE4            BIT   3, A                ;je READ PROTECTED?
1AE6            JP    Z, #1FBD, REPORTe   ;ano → skoč na REPORT e „File is read protected“
```

Uložíme si hlavičku nalezeného souboru z buferu do SRAM.

```
1AE9            LD    DE, #3EB4 SVHEAD    ;do DE adresa pro schování hlavičky položky v SRAM
1AEC            LD    BC, #0020           ;délka jedné položky je 32 bytů
1AEF            LDIR                      ;zkopíruj ji do SRAM
1AF1            LD    HL, #3EC5 SVFSC     ;do HL adresa prvního sektoru souboru souboru
1AF4            LD    E, (HL)             ;vyzvedni ho do DE
1AF5            INC   HL
1AF6            LD    D, (HL)
1AF7            LD    (#3E74), DE STARTADR ;a ulož ho do SRAM
1AFB            CALL  #1C56, CHNGDRNM     ;zaměň jména zdrojového a cílového disku a masky
1AFE            POP   BC                  ;obnov si počítadlo kopírovaných souborů a příznak
1AFF            PUSH  BC                  ;a zase ho ulož
1B00            INC   B                   ;kopíruje se jenom jeden soubor?
1B01            JR    Z, #1B13, COPYF6    ;ano → skoč
1B03            LD    HL, #3EB4 SVHEAD    ;do HL adresa uložení hlavičky položky v SRAM
1B06            LD    A, (HL)             ;vyzvedni příponu do A
1B07            LD    (#3E94), A EXTE1    ;a ulož příponu souboru do masky cílového disku
1B0A            INC   HL                  ;posuň se na jméno souboru
1B0B            LD    DE, #3E8A FNZONE1   ;do DE adresa masky souboru cílového disku
1B0E            LD    BC, #000A           ;10 znaků
1B11            LDIR                      ;zkopíruj jméno
1B13 COPYF6     CALL  #1C8F, SETACT       ;roztoč cílovou mechaniku, která má stejné jméno jako
                                          ;v DNZONE1
1B16            CALL  #217B, ERAVAR       ;vymaž pomocné proměnné MDOSu
```

```
1B19            CALL #1F66, DELALLFIL   ;smaž soubor odpovídající 2. masce na cílovém disku
1B1C            CALL #215C, FIRSTEMPTY  ;najdi 1. volnou položku adresáře od začátku
1B1F            JP   NZ, #2031, REPORTV ;není → skoč na REPORT V „Directory full“
1B22            POP  BC                 ;obnov si počítadlo zkopírovaných souborů a příznak
1B23            PUSH BC                 ;a zase ho ulož
1B24            PUSH HL                 ;ulož si ukazatel na volnou položku adresáře v buferu
1B25            EX   DE, HL             ;a dej ji do DE
1B26            INC  B                  ;kopíruje se jeden soubor?
1B27            JR   Z, #1B33, COPYFONE ;ano → skoč
```

Přesuneme uchovanou hlavičku do volné položky adresáře.

```
1B29            LD   HL, #3EB4 SVHEAD   ;do HL adresa uložení hlavičky položky v SRAM
1B2C            LD   BC, #0020          ;délka položky je 32 bytů
1B2F            LDIR                    ;přesuň ji do volné položky adresáře
1B31            JR   #1B48, COPYFILE    ;skoč na přesun souboru
```

Přesuneme do volné položky adresáře novou hlavičku, jako jméno a příponu však.použijeme cílovou masku.

```
1B33 COPYFONE   LD   A, (#3E94) EXTE1   ;vyzvedni příponu cílové masky
1B36            LD   (DE), A            ;a ulož ji do přípony v adresáři v buferu
1B37            INC  DE                 ;posuň se na jméno souboru v adresaři v buferu
1B38            LD   HL, #3E8A FNZONE1  ;do HL adresa uložení cílového jména souboru
1B3B            LD   BC, #000A          ;kopírujeme jenom 10 znaků
1B3E            LDIR                    ;zkopíruj jméno
1B40            LD   HL, #3EBF SVINF    ;do HL adresa informací o načtené položce v SRAM
1B43            LD   BC, #0015          ;délka zbývajících informací je 21 bytů
1B46            LDIR                    ;přesuň je do volné položky adresáře
```

Najdeme první volný sektor ve FAT a vytvoříme si cílový soubor jako soubor s nulovou délkou.

```
1B48 COPYFILE   LD   HL, #0000          ;FAT tabulku budeme prohledávat od začátku
1B4B            CALL #20F6, FIEMPTYFAT  ;najdi prázdnou položku FAT od HL na cílovém disku
1B4E            JP   NZ, #20BC, RETREP  ;neexistuje → skoč na REPORT U „Disk full“
1B51            LD   (#3E76), HL LENDAT ;ulož si číslo prvního sektoru na cílovém disku do SRAM
1B54            LD   DE, #0C00          ;do DE příznak „soubor s nulovou délkou“
1B57            CALL #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE na cílovém disku
1B5A            EX   DE, HL             ;dej do DE číslo prvního sektoru
1B5B            POP  HL                 ;obnov ukazatel na položku adresáře v buferu, kterou
1B5C            LD   A, #11             ;vyplňujeme a posuň se na adr. uložení prvního sektoru
1B5E            CALL #0FAD, ADDHLA
1B61            LD   (HL), E            ;uloží si číslo prvního sektoru
1B62            INC  HL                 ;do položky adresáře
1B63            LD   (HL), D
1B64            CALL #1E65, WSCADR      ;zapiš sektor adresáře na cílovém disku
1B67            CALL #1D9D, WFATIFCH    ;zapiš FAT na cílovém disku, pokud byla změněna
1B6A COPYRD     CALL #1C56, CHNGDRNM    ;zaměň jména zdrojového a cílového disku a masky
1B6D            CALL #1C8F, SETACT      ;roztoč zdrojovou mechaniku, která má stejné jméno
                                        ;jako v DNZONE1
1B70            CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
1B73            LD   DE, (#3E7A) VALSYY ;do DE adresa buferu v paměti
1B77            LD   HL, (#3E74) STARTADR ;do HL číslo prvního sektoru uložení souboru na
                                        ;zdrojovém disku
1B7A            LD   A, (#3E78) VALSYXlo ;do A počet sektorů, kolik jich jde načíst do paměti
```

Naplníme bufer ze zdrojového disku.

```
1B7D COPYRDSC  PUSH AF                      ;ulož si počet sektorů
1B7E           PUSH HL                      ;a číslo sektoru
1B7F           CALL #1DF9, LOGFYZ           ;převeď logický sektor na fyzický sektor a stopu
1B82           EX   DE, HL                  ;dej do HL adresu pro uložení sektoru
1B83           LD   DE, #0100               ;jeden sektor, 255 opakování
1B86           CALL #22A2, BREADA           ;načti sektor
1B89           EX   (SP), HL                ;vyzvdeni číslo sektoru a ulož si adresu uložení dat
1B8A           CALL #1CF1, GETWTEST         ;načti položku FAT a otestuj chybu
1B8D           BIT  3, D                    ;poslední sektor?
1B8F           JR   NZ, #1B9C, CPYRDLST     ;ano → skoč
1B91           EX   DE, HL                  ;dej do HL číslo dalšího sektoru
1B92           POP  DE                      ;obnov adresu uložení
1B93           POP  AF                      ;a počítadlo sektorů
1B94           DEC  A                       ;sniž počítadlo
1B95           JR   NZ, #1B7D, COPYRDSC;ještě není plný bufer → skoč na načtení dalšího sektoru
1B97           LD   (#3E74), HL STARTADR    ;ulož si číslo dalšího sektoru, který se bude číst ze
                                            ;zdrojového disku
1B9A           JR   #1BB2, CPYFULB          ;skoč na uložení buferu na cílový disk
```

Byl načten poslední sektor souboru, nastavíme parametry pro ukončení souboru.

```
1B9C CPYRDLST  PUSH DE                      ;ulož si obsah poslední položky
1B9D           CALL #1C56, CHNGDRNM         ;zaměň jména zdrojového a cílového disku a masky
1BA0           CALL #1C8F, SETACT           ;roztoč cílovou mechaniku, která má stejné jméno jako
                                            ;v DNZONE1
1BA3           CALL #217B, ERAVAR           ;vymaž pomocné proměnné MDOSu
1BA6           POP  DE                      ;obnov si obsah poslední položky
1BA7           POP  AF                      ;vyzvedni hodnotu ze zásobníku
1BA8           POP  AF                      ;vyzvedni počítadlo sektorů
1BA9           PUSH DE                      ;ulož si obsah poslední položky
1BAA           LD   B, A                    ;dej počítadlo sektorů do B
1BAB           LD   A, (#3E78) VALSYXlo     ;a vyzvedni počet sektorů, které se vejdou do buferu
1BAE           SUB  B                       ;odečti počítadlo – v A je počet načtených sektorů
1BAF           INC  A                       ;zvyš o jedničku
1BB0           JR   #1BC2, COPYBUF          ;skoč na vyprázdnění buferu na disk
```

Je plný bufer, nastavíme parametry pro vyprázdnění na cílový disk.

```
1BB2 CPYFULB   CALL #1C56, CHNGDRNM         ;zaměň jména zdrojového a cílového disku a masky
1BB5           CALL #1C8F, SETACT           ;roztoč cílovou mechaniku, která má stejné jméno jako
                                            ;v DNZONE1
1BB8           CALL #217B, ERAVAR           ;vymaž pomocné proměnné MDOSu
1BBB           LD   HL, #8C00               ;do HL příznak „nebyl konec souboru“
1BBE           PUSH HL                      ;a ulož na zásobník
1BBF           LD   A, (#3E78) VALSYXlo     ;do A počet sektorů, které jdou načíst do buferu (nyní je
                                            ;bufer plný)
```

Uložíme bufer na disk.

```
1BC2 COPYBUF   LD   DE, (#3E7A) VALSYY      ;vyzvedni adresu začátku buferu
1BC6           LD   HL, (#3E76) LENDAT      ;a číslo prvního sektoru, který se bude zapisovat na
                                            ;cílový disk
1BC9           PUSH AF                      ;ulož si počítadlo souborů
```

Budeme postupně zapisovat sektory z buferu.

```
1BCA COPYWSC    PUSH HL                     ;ulož číslo sektoru
1BCB            CALL #1DF9, LOGFYZ          ;převeď logický sektor na fyzický sektor a stopu
1BCE            EX   DE, HL                 ;dej do HL adresu v buferu
1BCF            LD   DE, #0100              ;jeden sektor, 255 opakování
1BD2            LD   A, (#3E6B) WORKDR      ;do A drive, se kterým se pracuje
1BD5            CALL #2296, BWRITE          ;zapiš sektor
1BD8            POP  DE                     ;do DE číslo sektoru
1BD9            POP  AF                     ;obnov počítadlo sektorů
1BDA            DEC  A                      ;sniž o jeden
1BDB            JR   Z, #1BEF, COPYIFALL    ;byl zapsán poslední sektor → skoč
1BDD            PUSH AF                     ;ulož si počítadlo
1BDE            EX   DE, HL                 ;dej do DE adresu v buferu a do HL číslo sektoru
```

Najdeme další volnou položku ve FAT a připojíme ji do stezky souboru.

```
1BDF            PUSH DE                     ;ulož si adresu v buferu
1BE0            PUSH HL                     ;a číslo naposledy zapisovaného sektoru
1BE1            CALL #20F6, FIEMPTYFAT      ;najdi prázdnou položku FAT od HL
1BE4            JR   NZ, #1C1C, COPYNOEM    ;neexistuje → skoč
1BE6            POP  DE                     ;obnov číslo naposledy zapisovaného sektoru
1BE7            EX   DE, HL                 ;zaměň čísla sektorů
1BE8            CALL #1D1E, WRTOFAT         ;zapiš do položky FAT obsah DE
1BEB            EX   DE, HL                 ;do HL dej číslo sektoru, který se bude teď zapisovat
1BEC            POP  DE                     ;obnov si adresu v buferu
1BED            JR   #1BCA, COPYWSC         ;skoč na zapsání dalšího sektoru
```

Byl zapsán celý bufer.

```
1BEF COPYIFALL  EX   DE, HL                 ;do HL adresu v buferu
1BF0            POP  DE                     ;vyzvedni obsah poslední položky
1BF1            BIT  7, D                   ;byl konec souboru?
1BF3            RES  7, D                   ;zruš 7. bit
1BF5            JR   NZ, #1C29, CPYNOEND    ;ne → skoč
```

Při naplňování buferu se narazilo na konec souboru. Musíme tedy zapsat koncovou značku souboru.

```
1BF7            CALL #1D1E, WRTOFAT         ;zapiš do položky FAT obsah DE
1BFA            CALL #1D9D, WFATIFCH        ;zapiš FAT, pokud byla změněna
1BFD            CALL #1C56, CHNGDRNM        ;zaměň jména zdrojového a cílového disku a masky
1C00            POP  BC                     ;obnov čítač zkopírovaných souborů a příznak
1C01            INC  C                      ;zvyš počet zkopírovaných souborů
1C02            PUSH BC                     ;a ulož na zásobník
1C03            JP   #1AD2, COPYLOOP        ;a skoč na kopírování dalšího souboru
```

Konec kopírování, výpis konečných informací.

```
1C06 ENDCOPY    LD   A, #FE                 ;kanál –2 (horní část obrazovky)
1C08            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
1C09            DW   1601                   ;podprogram CHAN-OPEN otevření kanálu
1C0B            LD   A, #0D                 ;nový řádek
1C0D            RST  #10                    ;tiskni
1C0E            POP  BC                     ;vyzvedni počítadlo kopírovaných soubory
1C0F            LD   B, #00                 ;do B nula
1C11            CALL #0FA6, BCPRT           ;piš BC na obrazovku
1C14            XOR  A                      ;položka 0 – „File(s) copied.“
```

```
1C15            LD    DE, #1C44 TXTMOVE  ;do DE adresa tabulky příkazu MOVE
1C18            CALL  #01C8, PRTMES      ;piš text položky
1C1B            RET                      ;vrať se přes RETURN do ZX ROM
```

Pokud dojde k chybě při zápisu, je stezka souboru regulérně ukončena (aby na disketě nevznikl zmatek) – do posledního sektoru se zapíše #0C00.

```
1C1C COPYNOEM   POP   HL                  ;vyzvedni číslo posledního sektoru
1C1D            LD    DE, #0C00          ;do DE příznak soubor s nulovou délkou
1C20            CALL  #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE
1C23            CALL  #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
1C26            JP    #20BC, RETREP      ;skoč na REPORT U „Disk full“
```

Byl zapsán celý bufer, ale ještě nebyl zkopírován celý soubor. Ukončíme prozatím stezku souboru (aby při nějaké chybě při čtení nevznikl na cílové disketé zmatek) a pokračujeme na naplnění buferu.

```
1C29 CPYNOEND   PUSH  HL                 ;ulož si číslo naposledy zapsaného sektoru
```

Najdeme volný sektor na cílové disketě.

```
1C2A            CALL  #20F6, FIEMPTYFAT  ;najdi prázdnou položku FAT od HL
1C2D            JR    NZ, #1C1C, COPYNOEM;není → skoč
1C2F            POP   DE                 ;do DE číslo naposledy zapsaného sektoru
1C30            EX    DE, HL             ;zaměň čísla sektorů
1C31            CALL  #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE
1C34            EX    DE, HL             ;zaměň čísla zpět
```

Ukončíme stezku souboru pro bezpečnost.

```
1C35            LD    DE, #0C00          ;do DE příznak soubor s nulovou délkou
1C38            CALL  #1D1E, WRTOFAT     ;zapiš do položky FAT obsah DE
1C3B            LD    (#3E76), HL LENDAT ;ulož do SRAM jako sektor, do kterého se bude zapisovat
1C3E            CALL  #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
1C41            JP    #1B6A, COPYRD      ;skoč na načtení dalších sektorů do buferu
```

TXTMOVE

Tabulka textů příkazu MOVE.

```
1C44 TXTMOVE    DB    #FF                                          ;invertovaný znak
  Položka 0
1C45            20 46 69 6C 65 28 73 29 20 63 6F 70 69 65 64 2E 8D ;File(s) copied.
```

CHANGEDRNM

Zamění jména zdrojového a cílového disku i s maskami.

IN: jm. disku v DNZONE1 a DNZONE2, masky souborů v FNZONE1 a FNZONE2
OUT: jm. disku a masky jsou zaměněny

```
1C56 CHNGDRNM   LD    HL, #3E80 DNZONE1  ;do HL adresa jména zdrojového disku
1C59            LD    DE, #3E95 DNZONE2  ;do DE adresa jména cílového disku
1C5C            LD    BC, #0015          ;21 znaků
1C5F            PUSH  AF                 ;ulož si A
```

Provedeme přesun.

```
1C60 CHANGDR1   LD    A, (DE)            ;vyzvedni znak jména disku a masky z DNZONE2
1C61            LDI                      ;přesuň znak z DNZONE1 do DNZONE2
1C63            DEC   HL                 ;posuň ukazatel o jeden zpět v DNZONE1
```

```
1C64            LD   (HL), A                ;a ulož znak z DNZONE2 do DNZONE1
1C65            INC  HL                     ;další znak ve jménu disku a souboru v DNZONE1
1C66            JP   PE, #1C60, CHANGDR1    ;opakuj BC-krát
1C69            POP  AF                     ;obnov AF
1C6A            RET                         ;vrať se
```

SETCOPYNM

Rozdělí řetězec na zásobníku na jméno souboru a disku do FNZONE1 a DNZONE1, analyzuje jméno disku a souboru pro příkaz MOVE.

```
1C6B SETCOPYNM  CALL #0FCF, DIVSTRING       ;rozděl řetězec na zásobníku na jméno disku a souboru
1C6E            CALL #1043, SETWDNM         ;nastav jméno disku v DNZONE1 pro I/O
1C71            CALL #10E2, ANALWDNM        ;analyzuj jméno disku v DNZONE1
1C74            JR   C, #1C7B, SETCOPYN1    ;bylo vloženo jméno disku → skoč
1C76            LD   A, #2A                 ;REPORT b „Bad volume name“
1C78            JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
1C7B SETCOPYN1  INC  A                      ;chybná identifikace disku?
1C7C            JP   Z, #2337, REPORTX      ;ano → skoč na report
```

Nastavíme a zkontrolujeme jméno souboru a použití wildchars v příponě.

```
1C7F            CALL #107C, ARRANGNM        ;uprav jméno souboru v FNZONE1 na masku
1C82            PUSH AF                     ;ulož si příznaky úprav
1C83            LD   A, (#3E94) EXTE1       ;vyzvedni příponu masky
1C86            CP   „?“                    ;je to otazník?
1C88            JR   NZ, #1C8D, SETCPYN2    ;ne → skoč
1C8A            POP  AF                     ;obnov příznaky úprav
1C8B            SCF                         ;nastav příznak „použito wildchars“
1C8C            RET                         ;vrať se
1C8D SETCPYN2   POP  AF                     ;obnov příznaky
1C8E            RET                         ;vrať se
```

SETACT

Tento podprogram nastaví drive podle jména v DNZONE1 jako drive, se kterým se bude pracovat. Prohledá všechny jména drivů a pokud nenajde hledané, načte parametry všech připojených drivů z disket a zkusí to znovu.

IN: DNZONE1 jméno hledaného drivu
OUT: Z takový drive byl nalezen
NZ drive s takovým jménem není

```
1C8F SETACT     PUSH BC                     ;ulož si registry
1C90            PUSH DE
1C91            PUSH HL
1C92            PUSH AF
1C93            XOR  A                      ;nyní je nulování systémových proměnných, které ale
1C94            LD   (#3E68), A VARIA1      ;nejsou nijak systémem použity. Asi je to pozůstatek
1C97            LD   (#3E69), A VARIA2      ;z ladících rutin.
1C9A            DEC  A                      ;do A dej 255
1C9B            LD   (#3E6A), A VARIA3      ;taky bez významu
1C9E            CALL #10E2, ANALWDNM        ;analyzuj jméno disku v DNZONE1
1CA1            JR   NZ, #1CA8, OKANALW     ;je v pořádku → skoč
```

V DNZONE1 není jméno disku.

```
1CA3            LD   A, #3B                 ;REPORT s „Internal error“
```

```
1CA5                JP    #0204, ERRR                ;piš hlášení a skoč do ZX ROM
1CA8 OKANALW        JR    C, #1CB9, SETNAME          ;nebylo vloženo určení mechanika A–D → skoč
1CAA                INC   A                          ;otestuj, jestli byla vložena správná mechanika
1CAB                JR    NZ, #1CB2, SETDRIVE        ;ano → skoč na změnu mechaniky
1CAD                LD    A, #20                     ;REPORT X „Bad device type“
1CAF                JP    #0204, ERRR                ;piš hlášení a skoč do ZX ROM
```

Protože jako jméno mechaniky bylo použito určení mechaniky A–D, je třeba udělat změnu mechaniky, se kterou se pracuje.

```
1CB2 SETDRIVE       DEC   A                          ;uprav číslo mechaniky na rozsah 0–3
1CB3                LD    (#3E6B), A WORKDR          ;nastav jako disk, se kterým se bude pracovat
1CB6                JP    #1EA5, GETPAR1             ;načti BOOT a nastav parametry disku
```

Nyní se musíme nastavit disk, se kterým se bude pracovat, podle jména disku.

```
1CB9 SETNAME        CALL  #1F16, SETDRV              ;hledej drive se stejným jménem jako v DNZONE1
1CBC                JR    Z, #1CEB, SETRET           ;našel drive → skoč na návrat
1CBE                CALL  #1F49, INITALLDR           ;nastav parametry všech připojených disků z BOOTů
1CC1                CALL  #1F16, SETDRV              ;hledej drive se stejným jménem jako v DNZONE1
1CC4                JR    Z, #1CEB, SETRET           ;našel → skoč
1CC6                LD    DE, #03AF SYSMSG           ;do DE tabulka chybových hlášení
1CC9                LD    A, #3C                     ;REPORT „Please insert volume“ a dotaz „(Retry = R)“
1CCB                CALL  #21BF, KEYMSG              ;tiskni hlášení a čekej na klávesu P nebo R
1CCE                JR    C, #1CB9, SETNAME          ;stisknuto P, R → skoč na opakování
1CD0                LD    A, #2C                     ;REPORT d „Volume not found“
1CD2                JP    #0204, ERRR                ;piš hlášení a skoč do ZX ROM
```

CMPDSK

Načte BOOT z diskety a porovná jméno diskety se jménem disku v SRAM. Pokud jsou rozdílné, nastaví ho jako nové jméno disku a naství nové parametry z BOOTu.

IN: –
OUT: Z jméno diskety v drivu a jméno drivu je stejné
NZ jméno diskety bylo jiné, musely se nastavit parametry z BOOTu

```
1CD5 CMPDSK         PUSH  BC                        ;ulož si registry
1CD6                PUSH  DE
1CD7                PUSH  HL
1CD8                PUSH  AF
1CD9                CALL  #1E7E, RDBOOT              ;načti BOOT z diskety drivu, se kterým se pracuje
1CDC                CALL  #2199, NAMEDISK            ;do HL adresa jména drivu v SRAM
1CDF                LD    DE, #3AC0                  ;do DE adresa jména diskety v BOOTu
1CE2                LD    BC, #000C                  ;porovnáme 12 byte (jméno + dva náhodné byty)
1CE5                CALL  #1F0E, VERIFY              ;porovnej je
1CE8                JP    NZ, #1EAB, SETPARAM        ;jsou rozdílné → skoč na nastavení parametrů z BOOTu
1CEB SETRET         POP   AF                         ;obnov registry
1CEC                POP   HL
1CED                POP   DE
1CEE                POP   BC
1CEF                CP    A                          ;nastav Z – jméno diskety v drivu je stejné jako jméno
1CF0                RET                              ;drivu v SRAM vrať se
```

GETWITHTEST

Načte obsah položky v HL ve FAT a zjistí, jestli není poškozená FAT.

IN: HL číslo položky FAT tabulky
OUT: DE obsah položky HL ve FAT tabulce
HL číslo položky FAT tabulky

```
1CF1 GETWTEST   CALL #1D04, GETFAT      ;vyzvedni obsah položky v HL do DE
```

Nyní otestujeme, jestli to není vadný, systémový nebo prázdný sektor.

```
1CF4            PUSH AF                 ;schovej si AF
1CF5            LD   A, D               ;do A vyšší byte obsahu položky
1CF6            CP   #0D                ;systémový nebo vadný sektor?
1CF8            JR   Z, #1CFF, REPORTl  ;ano → skoč na REPORT l „Corrupted FAT structure“
1CFA            OR   E                  ;volný sektor?
1CFB            JR   Z, #1CFF, REPORTl  ;ano → skoč na REPORT l „Corrupted FAT structure“
1CFD            POP  AF                 ;obnov AF
1CFE            RET                     ;vrať se
1CFF REPORTl    LD   A, #34             ;REPORT l „Corrupted FAT structure“
1D01            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

GETFAT

Vyzvedává obsah položky v HL z tabulky FAT bez testu chyby ve FAT.

IN: HL číslo položky FAT
OUT: DE obsah položky HL ve FAT tabulce
HL číslo položky FAT

```
1D04 GETFAT     PUSH HL                 ;schovej si číslo položky
1D05            CALL #1D46, READFATSC   ;načti sektor FAT tabulky, kde se nachází daná položka
                                        ;a vypočti adresu uložení položky
1D08            JR   C, #1D16, IFODD    ;lichá adresa → skoč
```

Obsah položky začíná na sudé adrese.

```
1D0A            LD   E, (HL)            ;nižší byte obsahu položky do E
1D0B            INC  HL                 ;posuň se na byte
1D0C            LD   A, (HL)            ;vyšší byte položky do A
1D0D            AND  #F0                ;vyber horní 4 bity
1D0F            RRCA                    ;posuň je dolů
1D10            RRCA
1D11            RRCA
1D12            RRCA
1D13            LD   D, A               ;předej do D, v DE je obsah položky
1D14            POP  HL                 ;obnov HL
1D15            RET                     ;vrať se
```

Obsah položky začíná na liché adrese.

```
1D16 IFODD      LD   A, (HL)            ;vyšší byte obsahu položky do A
1D17            AND  #0F                ;vyber dolní 4 bity
1D19            LD   D, A               ;předej do D
1D1A            INC  HL                 ;posuň se na nižší
1D1B            LD   E, (HL)            ;nižší byte obsahu položky do E, v DE je obsah položky
1D1C            POP  HL                 ;obnov HL
1D1D            RET                     ;vrať se
```

WRTOFAT

Zapíše do FAT obsah DE do položky v HL.

IN: HL číslo logického sektoru, do něhož se bude zapisovat ve FAT
DE hodnota, která se do FAT zapíše

OUT: zapíše DE do položky FAT v HL

```
1D1E WRTOFAT    PUSH HL                     ;ulož registry
1D1F            PUSH DE
1D20            CALL #1D46, READFATSC       ;načti sektor FAT tabulky, kde je uložen sektor v HL
1D23            LD   A, #FF                 ;do A příznak „FAT tabulka změněna“
1D25            LD   (#3E6C), A CHNGFLAG    ;ulož
1D28            POP  DE                     ;obnov obsah DE
1D29            PUSH DE                     ;a zase ho ulož
1D2A            JR   C, #1D3C, WISODD       ;lichá položka → skoč
```

Číslo sektoru v HL je sudé, zapisuje se do vyšších bytů.

```
1D2C            LD   (HL), E                ;ulož nižší část
1D2D            INC  HL                     ;posuň se na vyšší část
1D2E            LD   A, D                   ;dej ji do A
1D2F            RRCA                        ;posuň ji nahoru
1D30            RRCA
1D31            RRCA
1D32            RRCA
1D33            LD   D, A                   ;dej ji do D
1D34            LD   A, (HL)                ;vyzvedni do A informace z vedlejší položky
1D35            AND  #0F                    ;ponech jen spodní bity
1D37            OR   D                      ;přidej vyšší část
1D38            LD   (HL), A                ;ulož ji zpět do FAT
1D39            POP  DE                     ;obnov registry
1D3A            POP  HL
1D3B            RET                         ;vrať se
```

Číslo sektoru v HL je liché, zapisuje se do nižších bitů.

```
1D3C WIFODD     LD   A, (HL)                ;vyzvedni informace z vedlejší položky
1D3D            AND  #F0                    ;ponech horní bity
1D3F            OR   D                      ;přidej nižší část
1D40            LD   (HL), A                ;ulož ji do FAT
1D41            INC  HL                     ;posuň se na vyšší část
1D42            LD   (HL), E                ;ulož vyšší část
1D43            POP  DE                     ;obnov registry
1D44            POP  HL
1D45            RET                         ;vrať se
```

READFATSC

Načte sektor FAT, ve kterém je uložen daná položka v HL, do buferu.

IN: HL číslo položky ve FAT

OUT: sektor FAT, ve kterém se nachází daná položka, je načten do FATBUF
HL adresa uložení položky v FATBUF
NC sudá
C lichá

Nejdříve otestujeme, jestli vůbec existuje taková položka FAT.

```
1D46 READFATSC  PUSH BC                     ;ulož si BC
1D47            LD   BC, #06A9              ;do BC 1705 – počet položek FAT tabulky
1D4A            AND  A                      ;nuluj CY
1D4B            SBC  HL, BC                 ;odečti BC od HL
1D4D            JR   C, #1D54, NOHIGHER     ;je číslo položky větši než je maximum? ne → skoč
```

Taková položka neexistuje.

```
1D4F            LD   A, #3B                 ;REPORT s „Internal error“
1D51            JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
```

Najdeme požadovaný sektor.

```
1D54 NOHIGHER   ADD  HL, BC                 ;přičti BC zpět k HL
1D55            LD   C, #00                 ;v C bude počítadlo sektorů – na začátku nula
1D57            LD   DE, #0155              ;do DE 341-počet položek v jednom sektoru FAT
1D5A            AND  A                      ;nuluj příznak CY
1D5B CALCSCFAT  INC  C                      ;zvětši číslo sektoru
1D5C            SBC  HL, DE                 ;odečti od HL počet položek jednoho sektoru
1D5E            JR   NC, #1D5B, CALCSCFAT;ještě to není ten sektor → skoč
1D60            ADD  HL, DE                 ;přičti zpět
```

Vypočteme relativní adresu uložení položky v buferu.

```
1D61            LD   D, H                   ;dej relativní položku do DE
1D62            LD   E, L
1D63            ADD  HL, HL                 ;vynásob HL třemi
1D64            ADD  HL, DE
1D65            SRL  H                      ;a vyděl dvěma
1D67            RR   L                      ;v HL je nyní adresa položky v paměti
```

Nyní zkontrolujeme, jestli již takový sektor není v buferu.

```
1D69            PUSH AF                     ;ulož si příznak sudá/lichá
1D6A            LD   A, (#3E6E) FATDR       ;do A číslo drivu pro poslední práci s FAT
1D6D            LD   B, A                   ;ulož do B
1D6E            LD   A, (#3E6B) WORKDR      ;do A číslo drivu, se kterým se pracuje
1D71            CP   B                      ;jsou stejné?
1D72            JR   NZ, #1D7A, MUSTREAD;ne → skoč na načtení sektoru
1D74            LD   A, (#3E6D) FATSC       ;do A číslo naposledy čteného sektoru FAT
1D77            CP   C                      ;porovnej s vypočítaným
1D78            JR   Z, #1D96, RDFATPOL     ;stejné → skoč
```

Musíme načíst sektor FAT do buferu.

```
1D7A MUSTREAD   PUSH HL                     ;ulož registry
1D7B            PUSH BC
1D7C            CALL #1D9D, WFATIFCH        ;zapiš FAT, pokud byla změněna
1D7F            POP  BC                     ;obnov číslo sektoru
1D80            LD   A, C                   ;dej číslo sektoru do A
1D81            LD   (#3E6D), A FATSC       ;ulož číslo sektoru FAT jako naposledy čtený
1D84            LD   B, #00                 ;číslo stopy 0
1D86            LD   DE, #0101              ;jeden sektor, žádné opakování
1D89            LD   HL, #3C00 FATBUF       ;do HL adresa pro uložení
1D8C            CALL #22A2, BREADA          ;načti sektor FAT
```

```
1D8F            POP  HL                ;obnov relativní adresu položky
1D90            LD   A, (#3E6B) WORKDR  ;do A drive, se kterým se pracuje
1D93            LD   (#3E6E), A FATDR   ;ulož ho jako drive pro práci s FAT
```

Vypočteme absolutní adresu uložení položky v paměti.

```
1D96 RDFATPOL   LD   DE, #3C00 FATBUF   ;do DE adresa uložení FAT sektoru
1D99            ADD  HL, DE             ;připočti k relativní adrese, v HL je teď adresa uložení
                                        ;položky
1D9A            POP  AF                 ;obnov příznak sudá/lichá
1D9B            POP  BC                 ;obnov BC
1D9C            RET                     ;vrať se
```

WFATIFCH

Zapíše sektor FAT, pokud byly informace v buferu změněny.

IN: –

OUT: zapíše bufer FAT na disketu, pokud od jeho posledního načtení bylo do něj zapisováno

```
1D9D WFATIFCH   PUSH BC                 ;ulož si registry
1D9E            PUSH DE
1D9F            PUSH HL
1DA0            PUSH AF
```

Zkontrolujeme, jestli byl od posledního zápisu nebo načtení sektoru FAT do buferu, zapisováno do buferu.

```
1DA1            LD   A, (#3E6C) CHNGFLAG ;vyzvedni informaci, jestli bylo zapisováno do buferu FAT
1DA4            AND  A                  ;bylo zapisováno do buferu FAT?
1DA5            JR   Z, #1DBD, NOWFAT   ;ne → skoč
```

Musíme zapsat změny na disk.

```
1DA7            LD   A, (#3E6D) FATSC   ;vyzvedni číslo sektoru FAT, který byl naposledy načten
1DAA            LD   C, A               ;a dej ho C
1DAB            LD   A, (#3E6E) FATDR   ;vyzvedni číslo drivu, ze kterého byla naposledy čtena FAT
1DAE            LD   DE, #0101          ;1 sektor, žádné opakování
1DB1            LD   HL, #3C00 FATBUF   ;do HL adresa začatku uložení buferu
1DB4            LD   B, #00             ;stopa 0
1DB6            CALL #2296, BWRITE      ;zapiš sektor na disk
1DB9            XOR  A                  ;příznak „nebyly změny v buferu FAT“
1DBA            LD   (#3E6C), A CHNGFLAG ;ulož ho CHNGFLAG
1DBD NOWFAT     POP  AF                 ;obnov registry
1DBE            POP  HL
1DBF            POP  DE
1DC0            POP  BC
1DC1            RET                     ;vrať se
```

FREECOUNT

Spočítá všechny volné sektory na disketě.

IN: –

OUT: BC počet volných sektorů

```
1DC2 FREECOUNT  LD   BC, #0000          ;počet volných sektorů je na začátku nula
1DC5            LD   HL, #000E          ;začni od sektoru č.14
```

Postupně projdeme všechny sektory od prvního sektoru datové oblasti až po první nevyužitý sektor na konci diskety.

```
1DC8 FRCOUNT1  CALL #1D04, GETFAT       ;vyzvedni obsah položky do DE
1DCB           INC  HL                  ;a posuň se na další sektor
1DCC           LD   A, D                ;do A vyšší část obsahu
1DCD           CP   #0D                 ;systémový nebo vadný sektor?
1DCF           JR   NZ, #1DD6, NOSYS    ;ne → skoč
1DD1           LD   A, E                ;do A nižší část obsahu
1DD2           CP   #DD                 ;systémový sektor na konci diskety?
1DD4           RET  Z                   ;ano → konec prohledávání – vrať se
1DD5           OR   D                   ;volný sektor?
1DD6 NOSYS     OR   E                   ;volný sektor?
1DD7           JR   NZ, #1DC8, FRCOUNT1 ;ne → skoč
1DD9           INC  BC                  ;zvyš počet volných sektorů
1DDA           JR   #1DC8, FRCOUNT1     ;skoč na testování dalšího sektoru
```

SECPERDSK

Vypočte, kolik sektorů je na disketě.

IN: IX adresa parametrů disku
OUT: HL počet sektorů na disketě

```
1DDC SECPERDSK LD   B, (IX+#02)         ;do B počet stop na disketě
1DDF           BIT  4, (IX+#01)         ;jednostranný formát?
1DE3           JR   Z, #1DE7, SECPD1    ;ano → skoč
1DE5           RLC  B                   ;vynásob počet stop dvěma
1DE7 SECPD1    LD   C, #00              ;nastav počet sektorů na nulu
```

FYZLOG

Převede fyzický sektor a stopu na logický sektor.

IN: B číslo stopy
C číslo sektoru
IX adresa parametrů disku
OUT: HL logický sektor

```
1DE9 FYZLOG    PUSH DE                  ;ulož si DE
1DEA           LD   E, (IX+#03)         ;do E počet sektorů na stopu z parametrů diskety
1DED           LD   D, #00              ;do D nula
1DEF           LD   H, D                ;do H taky nula
1DF0           LD   L, C                ;do L číslo sektoru
1DF1           INC  B                   ;zvyš číslo stopy
1DF2           JR   #1DF5, CALCLOG1     ;skoč na výpočet
```

Budeme postupně přičítat počet sektorů na stopu.

```
1DF4 CALCLOG   ADD  HL, DE              ;přičti počet sektorů na stopu
1DF5 CALCLOG1  DJNZ #1DF4, CALCLOG      ;opakuj B-krát
1DF7           POP  DE                  ;obnov DE
1DF8           RET                      ;vrať se
```

LOGFYZ

Přepočte logický sektor na fyzický sektor a stopu.

IN: HL logický sektor
IX adresa parametrů disku

OUT: B číslo stopy
C číslo sektoru ve stopě

```
1DF9 LOGFYZ   PUSH DE              ;ulož si DE
1DFA          LD   E, (IX+#03)     ;do E počet sektorů na stopu z parametrů diskety
1DFD          LD   D, #00          ;do D dej nulu
1DFF          LD   B, #FF          ;v B bude počítadlo stop, začnem od –1 (stopy se
1E01          AND  A               ;totiž číslují od 0), nuluj CY
```

Nyní budeme odčítat počet sektorů na stopu od logického sektoru a posunovat počítadlo stop, dokud nebude výsledek záporný.

```
1E02 CALCLF   INC  B               ;zvyš počítadlo stop o 1
1E03          SBC  HL, DE          ;odečti počet sektorů na stopu
1E05          JR   NC, #1E02, CALCLF ;ještě málo → skoč
```

Upravíme zpět na sektory.

```
1E07          ADD  HL, DE          ;přičti zpět
1E08          LD   C, L            ;do C dej číslo sektoru
1E09          POP  DE              ;obnov DE
1E0A          RET                  ;vrať se
```

READADR

Načte sektor adresáře, ve kterém se nachází položka adresáře do buferu a vrátí adresu uložení položky v buferu.

IN: A číslo položky–1
IX adresa parametrů drivu
OUT: HL adresa uložení položky v buferu adresáře
Z taková položka existuje
A číslo dané položky

```
1E0B READADR  INC  A               ;zvyš číslo položky o 1
1E0C          BIT  7, A            ;je více než 128 položek? (0 – 127)
1E0E          RET  NZ              ;ano → vrať se
1E0F          PUSH AF              ;ulož registry
1E10          PUSH BC
1E11          PUSH DE
1E12          LD   B, A            ;uschovej si číslo položky do B
1E13          AND  #0F             ;nech zbytek po dělení 16 (počet položek v jednom sek.)
1E15          LD   C, A            ;ulož ho do C
1E16          PUSH BC              ;ulož BC
1E17          LD   A, B            ;dej do A zpět číslo položky
1E18          AND  #70             ;A vydělíme 16
1E1A          RLCA                 ;vynásob A dvěma
1E1B          RLA                  ;znovu vynásob A dvěma a 8. bit dej do příznaku C
1E1C          PUSH AF              ;ulož příznak C
1E1D          RLCA                 ;vynásob A dvěma
1E1E          RLCA                 ;a ještě jednou dvěma
1E1F          LD   B, A            ;ulož si A do B
1E20          POP  AF              ;obnov si příznak C
1E21          LD   A, B            ;dej B zpět do A
1E22          RLA                  ;a zarotuj
```

A jsem vynásobili 32-mi a přičetli nulu nebo jedničku podle toho, jestli se položka nachází v lichém nebo sudém sektoru), v A je teď relativní adresa sektoru, ve kterém se nachází daná položka

```
1E23            LD    HL, #0006              ;adresář je od 6. sektoru
1E26            CALL  #0FAD, ADDHLA          ;přičti A k HL, v HL je teď logický sektor
1E29            CALL  #1DF9, LOGFYZ          ;převeď logický sektor na fyzický sektor a stopu
```

Nyní porovnáme, jestli není daný sektor adresáře v buferu adresáře.

```
1E2C            LD    A, (#3E6B) WORKDR      ;do A drive, se kterým se pracuje
1E2F            LD    H, A                   ;ulož do H
1E30            LD    A, (#3E71) ADRDR       ;vyzvedni drive, z kterého byl naposledy přečten adresář
1E33            CP    H                      ;porovnej, jestli jsou stejné
1E34            JR    NZ, #1E3D, RDSFDR      ;rozdílné → skoč na načtení sektoru adresáře
1E36            LD    HL, (#3E6F) ADRSCTR    ;vyzvedni číslo stopy a sektoru adresáře, který byl
                                             ;naposledy načten
1E39            SBC   HL, BC                 ;porovnej s vypočtenými hodnotami
1E3B            JR    Z, #1E50 , RDADRCALC;stejné → skoč
```

Musíme načíst sektor adresáře.

```
1E3D RDSFDR     LD    A, (#3E6B) WORKDR      ;do A, se kterým se pracuje
1E40            LD    (#3E71), A ADRDR       ;ulož si ho jako drive, z kterého byl naposledy přečten
                                             ;adresář
1E43            LD    (#3E6F), BC ADRSCTR    ;ulož si číslo sektoru a stopy adresáře, který se načte
                                             ;jako poslední z disku
1E47            LD    HL, #3800 DIRBUF       ;do HL adresa, kam se bude ukládat sektor adresáře
1E4A            LD    DE, #0101              ;jeden sektor, žádné opakování
1E4D            CALL  #22A2, BREADA          ;načteme sektor
```

Sektor máme v paměti. Vypočteme nyní adresu položky.

```
1E50 RDADRCALC  POP   BC                     ;obnov BC
1E51            LD    A, C                   ;dej do A zbytek po dělení 16
1E52            RLCA                         ;vynásobíme 16-ti
1E53            RLCA
1E54            RLCA
1E55            RLCA
1E56            LD    D, #00                 ;do D dej 0
1E58            RLA                          ;vynásob dvěma a příznak přetečeni do CY
1E59            LD    E, A                   ;do E dej tedy nižší adresu
1E5A            RL    D                      ;posuň CY do D – v DE je relativní adresa položky
1E5C            LD    HL, #3800 DIRBUF       ;do HL začátek dat adresáře
1E5F            ADD   HL, DE                 ;přičti relativní adresu k HL
1E60            POP   DE                     ;obnov registry
1E61            POP   BC
1E62            POP   AF
1E63            CP    A                      ;nastav Z – položka nalezena
1E64            RET                          ;vrať se
```

WSCADR

Zapíše sektor adresáře.

IN: ADRDR číslo drivu, kam se bude zapisovat
ADRSCTR číslo stopy a sektoru, který se bude zapisovat
OUT: zapíše se daný sektor na disk

```
1E65 WSCADR     PUSH  AF                     ;ulož si registry
1E66            PUSH  BC
```

```
1E67            PUSH DE
1E68            PUSH HL
1E69            LD   A, (#3E71) ADRDR      ;vyzvedni drive, na který se bude zapisovat sektor adresáře
1E6C            LD   BC, (#3E6F) ADRSCTR   ;vyzvedni číslo sektoru a stopy do BC
1E70            LD   DE, #0101             ;1 sektor, žádné opakování
1E73            LD   HL, #3800 DIRBUF      ;do HL adresa, kde je uložen začátek sektoru adresáře
1E76            CALL #2296, BWRITE         ;zapiš sektor
1E79            POP  HL                    ;obnov registry
1E7A            POP  DE
1E7B            POP  BC
1E7C            POP  AF
1E7D            RET                        ;vrať se
```

RDBOOT

Načte BOOT z drivu, se kterým se pracuje do AUXBUF a porovná, jestli disketa v drivu je MDOSovská.

IN: –
OUT: načtený BOOT v AUXBUF

```
1E7E RDBOOT     LD   HL, #3A00 AUXBUF      ;do HL adresa, kam nahrát BOOT v SRAM
1E81            LD   DE, #0101             ;1 sektor, žádné opakování
1E84            LD   BC, #0000             ;0. sektor, 0. stopa
1E87            CALL #22A2, BREADA         ;načti sektor
1E8A            LD   HL, #3ACC             ;do HL adresa, kde je uložen text „SDOS“ v BOOTu
1E8D            LD   DE, #0F10 TXTSDOS     ;do DE adresa, kde je uložen text „SDOS“ v paměti
1E90            LD   BC, #0004             ;délka je 4 byty
1E93            CALL #1F0E, VERIFY         ;porovnej je
1E96            RET  Z                     ;jsou stejné → vrať se
```

Disketa v drivu není MDOSovská.

```
1E97            LD   A, #FF                ;do A dej 255
1E99            LD   (#3E6E), A FATDR      ;zruš číslo drivu, ze kterého byla naposledy čtena FAT
1E9C            LD   A, #20                ;REPORT X „Bad device type“
1E9E            JP   #0204, ERRR           ;piš hlášení a skoč do ZX ROM
```

GETPAR

Přečte BOOT, nastaví parametry, z BOOTu, uloží jméno drivu do SRAM pro disk, se kterým se pracuje.

IN: #3E6B (WORKDR) číslo drivu, odkud číst BOOT a nastavovat parametry
OUT: NZ
nastavené parametry a jméno drivu

```
1EA1 GETPAR     PUSH BC                    ;ulož si registry
1EA2            PUSH DE
1EA3            PUSH HL
1EA4            PUSH AF
1EA5 GETPAR1    CALL #21A1, DRVSYS         ;zjisti adresu parametrů disku, se který se pracuje
1EA8            CALL #1E7E, RDBOOT         ;načti BOOT do paměti a porovnej značku „SDOS“
```

Nastavíme parametry a jméno diskety z BOOTu, který je načten v BOOTBUF, jako parametry drivu, na jehož parametry ukazuje IX.

```
1EAB SETPARAM   LD   A, (IX+#05)           ;vyzvedni informace o mechanice
1EAE            LD   (IX+#01), A           ;ulož do informací o disketě
1EB1            BIT  4, (IX+#05)           ;jaký je formát mechaniky? (jednostranny/oboustranný)
```

```
1EB5            JR    NZ, #1EBE, DSIDE  ;oboustranný → skoč
1EB7            LD    A, (#3AB1)        ;vyzvedni informace z BOOTu
1EBA            BIT   4, A              ;je disketa jednostranná?
1EBC            JR    NZ, #1F09, REPORTX ;ne → skoč na REPORT X „Bad device type“
1EBE DSIDE      LD    A, (#3AB2)        ;do A počet stop z BOOTu
1EC1            CP    (IX+#06)          ;porovnej s počtem stop mechaniky
1EC4            JR    Z, #1ED2, TRACKOK ;stejné → skoč
1EC6            JR    C, #1ED2, TRACKLOW ;disketa má méně stop → skoč
```

Zjistíme, jestli není 80-ti stopá disketa v 40-ti stopé mechanice.

```
1EC8            SUB   (IX+#06)          ;odečti počet stop mechaniky
1ECB            CP    #08               ;je rozdíl větší než 8 (nejvíce může být 8 stop navíc)
1ECD            JR    NC, #1F09, REPORTX ;ano → skoč na REPORT X „Bad device type“
1ECF            LD    A, (#3AB2)        ;vyzvedni počet stop z BOOTu
1ED2 TRACKOK    LD    (IX+#02), A       ;ulož počet stop z BOOTU do parametrů diskety
```

Nyní otestujeme, jestli není v 40-ti stopá disketa v 80-ti stopé mechanice. Tento test však vylučuje možnost, kdy je disketa naformátována na poloviční počet stop.

```
1ED5            ADD   A, A             ;vynásob počet stop na disketě dvěma
1ED6            CP    (IX+#06)          ;porovnej s počtem stop mechaniky
1ED9            JR    NZ, #1EDF, NOLINH ;různé → skoč
1EDB            SET   5, (IX+#01)       ;nastav bit v 80-ti stopé mechanice je 40-ti stopá disketa
1EDF NOLINH     LD    A, (#3AB1)        ;do A informace o disketě z BOOTu
1EE2            AND   #13               ;%00010011 (ponech pouze potřebné bity pro disketu)
1EE4            LD    B, A              ;dej do B
1EE5            LD    A, (IX+#01)       ;vyzvedni informace o disketě z SRAM
1EE8            AND   #EC               ;%11101100 (ponech pouze potřebné bity pro mechaniku)
1EEA            OR    B                 ;přidej informace o disketě
1EEB            LD    (IX+#01), A       ;ulož nové informace pro disketu
1EEE            LD    A, (#3AB3)        ;počet sektorů z BOOTu do A
1EF1            LD    (IX+#03), A       ;ulož do SRAM
```

Nyní přesuneme jméno diskety do jména drivu.

```
1EF4            CALL  #2199, NAMEDISK   ;vypočti adresu jména drivu v SRAM podle IX
1EF7            EX    DE, HL            ;dej ji do DE
1EF8            LD    HL, #3AC0         ;do HL adresa jména diskety v BOOTu
1EFB            LD    BC, #000C         ;12 bytů (jméno + náhodné byty)
1EFE            LDIR                    ;přesuň nové jméno drivu
1F00            POP   AF                ;obnov A
1F01            LD    L, A              ;uschovej A do L
1F02            OR    #FF               ;nastav NZ
1F04            LD    A, L              ;obnov A z L
1F05            POP   HL                ;obnov registry
1F06            POP   DE
1F07            POP   BC
1F08            RET                     ;vrať se
1F09 REPORTX    LD    A, #20            ;REPORT X „Bad device type“
1F0B            JP    #0204, ERRR       ;piš hlášení a skoč do ZX ROM
```

VERIFY

Porovnává 2 bloky o délce BC bytů.

IN: BC počet porovnávaných bytů
HL adresa 1. bloku
DE adresa 2. bloku
OUT: Z bloky jsou stejné
NZ bloky nejsou stejné

```
1F0E VERIFY   LD   A, (DE)          ;vyzvedni obsah do A z (DE)
1F0F          INC  DE               ;posuň na se na další
1F10          CPI                   ;porovnej A s obsahem (HL)
1F12          RET  NZ               ;různé obsahy → vrať se s NZ
1F13          RET  PO               ;je B=0 → vrať se s Z
1F14          JR   #1F0E, VERIFY    ;pokračuj pro další byte
```

SETDRV

Nastaví drive, se kterým se bude pracovat, podle jména drivu v DNZONE1.

IN: jméno disku v DNZONE1
OUT: Z disk s takovým jménem byl nalezen
NZ disk s takovým jménem nebyl nalezen
na #3E6B (WORKDR) je uloženo číslo drivu

```
1F16 SETDRV   XOR  A                ;budeme prohledávat od mechaniky A:
1F17 FINDNMDR PUSH AF               ;uschovej si číslo disku
1F18          LD   (#3E6B), A WORKDR ;nastav ho jako disk, se kterým se bude pracovat
1F1B          CALL #21AC, DRVCMP    ;zjisti adresu parametrů drivu
1F1E          JR   Z, #1F3B, NEXTNM ;mechanika není připojena → skoč
1F20          CALL #2199, NAMEDISK  ;do HL adresa jména drivu
1F23          LD   DE, #3E80 DNZONE1 ;do DE adresa 1. jména disku pro I/O
1F26          LD   BC, #000A        ;délka jmen je 10 bytů
1F29          CALL #1F0E, VERIFY    ;porovnáme je
1F2C          JR   NZ, #1F3B, NEXTNM ;rozdílné → skoč
```

Našel drive se stejným jménem.

```
1F2E          LD   A, (#3E6B) WORKDR ;do A číslo nalezeného drivu
1F31          CALL #256D, TESTDR    ;je drive připraven?
1F34          JR   Z, #1F3B, NEXTNM ;není → skoč
1F36          CALL #1CD5, CMPDSK    ;porovnej jméno diskety s jménem disku, pokud je různé
                                    ;načti BOOT a nastav parametry a jméno diskety z BOOTu
1F39          JR   Z, #1F43, FINDNMOK ;byly shodné → skoč
```

Jméno nesouhlasí, zkusíme tedy otestovat další jméno disku.

```
1F3B NEXTNM   POP  AF               ;obnov číslo disku
1F3C          INC  A                ;posuň se na další disk
1F3D          CP   #04              ;už byly 4 disky?
1F3F          JR   C, #1F17, FINDNMDR ;ne → opakuj
1F41          OR   A                ;nastav NZ
1F42          RET                   ;vrať se

1F43 FINDNMOK POP  AF               ;obnov číslo disku
1F44          LD   (#3E6B), A WORKDR ;nastav ho jako drive, se kterým se bude pracovat
1F47          CP   A                ;nastav Z
1F48          RET                   ;vrať se
```

INITALLDR

Nastaví parametry všech připojených drivů z BOOTů disket

IN: –

OUT: nastavené parametry všech drivů z BOOTů

```
1F49 INITALLDR  XOR  A                  ;začíná se od mechaniky A
1F4A INITDR     PUSH AF                 ;uschovej si číslo drivu
1F4B            CALL #21AC, DRVCMP      ;zjisti adresu parametrů drivu v A
1F4E            JR   Z, #1F5F, NOINITDR ;není připojen → skoč
1F50            POP  AF                 ;vyzvedni si číslo drivu
1F51            PUSH AF                 ;ještě si ho uschovej
1F52            CALL #256D, TESTDR      ;je drive připraven?
1F55            JR   Z, #1F5F, NOINITDR ;není → skoč
1F57            POP  AF                 ;vyzvedni číslo drivu
1F58            PUSH AF                 ;a znovu ho ulož
1F59            LD   (#3E6B), A WORKDR  ;nastav ho jako drive, se kterým se pracuje
1F5C            CALL #1EA1, GETPAR      ;načti BOOT a nastav parametry
1F5F NOINITDR   POP  AF                 ;vyzvedni číslo drivu
1F60            INC  A                  ;posuň se na další
1F61            CP   #04                ;už byly čtyři
1F63            JR   C, #1F4A, INITDR   ;ne → opakuj
1F65            RET                     ;vrať se
```

DELALLFIL

Smaže všechny soubory vyhovující masce z disku.

IN: ve FNZONE1 je maska souboru
v DNZONE1 je jméno disku

OUT: Z byl nalezen nějaký soubor
NZ na disketě nebyl žádný soubor se jménem, které by vyhovovalo masce

```
1F66 DELALLFIL  CALL #1C8F, SETACT      ;roztoč mechaniku, která má jméno jako v DNZONE1
1F69            CALL #212B, FIRSTMASK   ;načti 1. položku adresáře vyhovující masce v FNZONE1
1F6C            RET  NZ                 ;není taková → vrať se
```

Postupně vymažeme všechny soubory.

```
1F6D DELFIND    PUSH AF                 ;ulož si číslo položky
1F6E            CALL #1283, GETATR      ;vyzvedni atributy souboru
1F71            BIT  0, A               ;je chráněn proti smazání?
1F73            JR   NZ, #1F7A, NODELPR ;ne → skoč
1F75            LD   A, #30             ;REPORT h „File is delete protected“
1F77            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
1F7A NODELPR    CALL #1F88, DFILER      ;proveď vymazání vyhledaného souboru v adresáři i FAT
1F7D            POP  AF                 ;obnov číslo položky
1F7E            CALL #212D, NEXTMASK    ;načti další položku adresáře (od A) vyhovující masce
                                        ;v FNZONE1
1F81            JR   Z, #1F6D, DELFIND  ;našel → skoč
1F83            CALL #1D9D, WFATIFCH    ;zapiš FAT, pokud byla změněna
1F86            XOR  A                  ;nastav Z
1F87            RET                     ;vrať se
```

DFILER

Provede smazání vyhledané položky adresáře.

IN: HL adresa položky adresáře v buferu
OUT: soubor je smazán z diskety

Nejdříve vymažeme soubor z adresáře.

```
1F88 DFILER   LD   (HL), #E5          ;ulož do hlavičky příznak „prázdná položka“
1F8A          CALL #1E65, WSCADR      ;zapiš sektor adresáře
1F8D          LD   DE, #0011          ;do DE relativní adresa uložení prvního sektoru v hlavičce
1F90          ADD  HL, DE             ;posuň ukazatel na hlavičku
1F91          LD   A, (HL)            ;do HL dej číslo prvního sektoru souboru
1F92          INC  HL
1F93          LD   H, (HL)
1F94          LD   L, A
```

Nyní vymažeme soubr z FAT. V HL je číslo prvního sektoru.

```
1F95 DFILER1  CALL #1CF1, GETWTEST    ;načti položku z FAT a otestuj chybu ve FAT
1F98          PUSH DE                 ;ulož si obsah položky
1F99          LD   DE, #0000          ;do DE nulu – „prázdný sektor“
1F9C          CALL #1D1E, WRTOFAT     ;zapiš do položky FAT v HL obsah DE
1F9F          POP  HL                 ;vyzvedni si obsah položky
1FA0          BIT  3, H               ;poslední sektor?
1FA2          RET  NZ                 ;ano → vrať se
1FA3          JR   #1F95, DFILER1     ;opakuj pro další sektor
```

LOAFND

Nahraje data ze souboru.

IN: HL počáteční adresa uložení dat v paměti
DE délka dat
IX adresa parametrů disku, ze kterého se budou načítat data
#3E72 (SVADRA) adresa uložení položky adresáře v buferu
v buferu je načten sektor adresáře, ve kterém jsou informace o souboru
OUT: nahraje data ze souboru do paměti

```
1FA5 LOAFND   PUSH HL                 ;ulož si počáteční adresu dat
1FA6          LD   HL, (#3E72) SVADRA ;do HL adresa položky adresáře v buferu
1FA9          JR   #1FB6, LOAFND1     ;pokračuj v nahrávání
```

LOAWITHF

Vyhledá soubor se jménem ve FNZONE1 a nahraje data ze souboru.

IN: HL počáteční adresa uložení dat v paměti
DE délka dat
IX adresa parametrů drivu
ve FNZONE1 je jméno souboru
OUT: vyhledá daný soubor na disketě a nahraje ho do paměti

```
1FAB LOAWITHF PUSH HL                 ;ulož si počáteční adresu dat
1FAC          CALL #212B, FIRSTMASK   ;najdi 1. položku adresáře vyhovující masce v FNZONE1
1FAF          JR   Z, #1FB6, LOAFND1  ;našel → skoč
1FB1 REPORTS  LD   A, #1B             ;REPORT S „File not found“
1FB3          JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

Vstupní bod pro nahrávání souboru. Na zásobníku je uložena počáteční adresa uložení dat, v DE je délka dat, v HL je adresa položky adresáře v buferu a v IX je ukazatel na parametry disku, ze kterého se budou načítat data.

```
1FB6 LOAFND1    CALL #1283, GETATR          ;načti atributy souboru
1FB9            BIT  3, A                   ;je READ PROTECTED?
1FBB            JR   NZ, #1FC2, LOAFND2     ;ne → skoč
1FBD REPORTe    LD   A, #2D                 ;REPORT e „File is read protected“
1FBF            JP   #0204, ERRR            ;piš hlášení a skoč do ZX ROM
1FC2 LOAFND2    LD   A, #11                 ;do A posun na číslo prvního sektoru
1FC4            CALL #0FAD, ADDHLA          ;posuň ukazatel na číslo prvního sektoru
1FC7            LD   A, (HL)                ;vyzvedni číslo prvního sektoru do HL
1FC8            INC  HL
1FC9            LD   H, (HL)
1FCA            LD   L, A
```

Nyní budeme postupně načítat sektory nebo skupiny sektorů, pokud následují za sebou.

```
1FCB LOAFNDLP   PUSH HL                     ;ulož si číslo prvního sektoru
1FCC            CALL #1DF9, LOGFYZ          ;převeď logický sektor na fyzický sektor a stopu
1FCF            POP  HL                     ;obnov logické číslo prvního sektoru
1FD0            PUSH BC                     ;a ulož si fyzické číslo sektoru a stopy
1FD1            CALL #20C4, COUNTCSEC       ;spočítej počet sektorů souboru následující za sebou
1FD4            EX   DE, HL                 ;ulož si číslo prvního sektoru mimo posloupnost do HL
1FD5            BIT  3, H                   ;testuj, jestli je to poslední sektor souboru
1FD7            LD   D, B                   ;do D počet sektorů jdoucích za sebou
1FD8            LD   E, #00                 ;do E 255 opakování
1FDA            POP  BC                     ;obnov fyzické číslo stopy a sektoru
1FDB            JR   NZ, #1FE4, LOAFND4     ;je to poslední sektor souboru → skoč
1FDD            EX   (SP), HL               ;vyzvedni si adresu uložení dat a ulož si číslo prvního
                                            ;sektoru mimo posloupnost
1FDE            CALL #22A2, BREADA          ;načti sektor
1FE1            EX   (SP), HL               ;ulož si adresu uložení dat a obnov číslo sektoru
1FE2            JR   #1FCB, LOAFNDLP        ;skoč na hlednání další posloupnosti
```

Dočteme zbytek posloupnosti a načteme poslední sektor souboru.

```
1FE4 LOAFND4    BIT  1, H                   ;je to sektor souboru s nulovou délkou?
1FE6            JR   Z, #201B, LFNDNULL     ;ano → skoč
1FE8            LD   A, H                   ;do A vrchní byte délky dat v sektoru
1FE9            AND  #01                    ;ponech jen 0. bit
1FEB            LD   H, A                   ;a vrať zpět do H – v HL je teď délka dat v posledním
1FEC            LD   A, H                   ;sektoru – je rovna 512?
1FED            OR   L
1FEE            JR   NZ, #1FF8, LOAFND3     ;ne → skoč
```

Načteme zbytek posloupnosti a všech 512 bytů z posledního sektoru.

```
1FF0            POP  HL                     ;vyzvedni si adresu uložení dat
1FF1            CALL #22A2, BREADA          ;načti zbývající sektory i s posledním
1FF4            CALL #217B, ERAVAR          ;vymaž pomocné proměnné
1FF7            RET                         ;vrať se
```

V posledním sektoru je méně než 512 bytů.

```
1FF8 LOAFND3    DEC  D                      ;sniž počet sektorů v posloupnosti o 1
1FF9            JR   Z, #2000, LOAFNDLS     ;byl jenom jeden → skoč
```

Musíme dohrát posloupnost zbývajících sektorů bez posledního.

```
1FFB            EX   (SP), HL               ;vyzvedni adresu uložení dat
```

```
1FFC            CALL #22A2, BREADA      ;načti zbývající sektory bez posledního
1FFF            EX   (SP), HL           ;ulož zpět adresu uložení dat
```

Nyní načteme poslední sektor do buferu a přeneseme jen potřebnou část.

```
2000 LOAFNDLS   PUSH HL                 ;ulož si délku dat v posledním sektoru
2001            LD   HL, (#3E7E) SVFRSC ;vezmi do HL číslo posledního sektoru souboru
2004            CALL #1DF9, LOGFYZ      ;převeď logický sektor na fyzický sektor a stopu
2007            LD   HL, #3A00 AUXBUF   ;do HL adr. buferu v SRAM pro uložení posledního sek.
200A            LD   DE, #0101          ;jeden sektor, žádné opakování
200D            CALL #22A2, BREADA      ;načti sektor
2010            POP  BC                 ;do BC délka dat v posledním sektoru
2011            POP  DE                 ;do DE adresa uložení dat
2012            LD   HL, #3A00 AUXBUF   ;od začátku buferu
2015            LDIR                    ;přesuň do paměti jen potřebné byty
2017 LOAFNDEND  CALL #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
201A            RET                     ;vrať se
```

Návrat, pokud je poslední sektor bez dat (soubor s nulovou délkou).

```
201B LFNDNULL   POP  HL                 ;vyzvedni adreu uložení dat
201C            JR   #2017, LOAFNDEND   ;skoč na dohrání posloupnosti sektorů
```

TRANSTOSEC

Vypočte, kolik sektorů zabere soubor na disku.

IN: DE délka souboru
OUT: B počet sektorů
DE délka dat v posledním sektoru

```
201E TRANSTOSEC LD   A, D               ;dej do A vrchní byte délky dat
201F            AND  #FE                ;zruš 0. bit
2021            RRCA                    ;a vyděl A dvěma – v A je nyní počet sektorů, které zabere
                                        ;soubor na disku a které jsou zcela vypněny
2022            LD   B, A               ;dej počet sektorů z A do B
2023            LD   A, D               ;dej do A vrchní byte délky dat
2024            AND  #01                ;ponech pouze 0. bit
2026            LD   D, A               ;dej zpět do D – v DE je nyní zbytek po dělení 512
                                        ;je to vlastně počet bytů v posledním sektoru
2027            LD   A, D               ;je délka dat v posledním sektoru nulova?
2028            OR   E
2029            RET  Z                  ;ano → vrať se
202A            INC  B                  ;zvyš počet sektorů o poslední sektor
202B            RET                     ;vrať se
```

FINDANDFILL

Najde první prázdnou položku adresáře a přesune do položky adresáře v buferu jméno souboru z FNZONE1.

IN: jméno ve FNZONE1
IX adresa parametrů drivu
OUT: HL ukazatel za jméno souboru v položce v buferu adresáře

```
202C FINDANDFILL CALL #215C, FIRSTEMPTY ;najdi 1. prázdnou položku adresáře od začátku
202F            JR   Z, #2036, IFFIND   ;nalezena → skoč
2031 REPORTV    LD   A, #1E             ;REPORT V „Directory full“
2033            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

Přesuneme jméno a příponu z FNZONE1 do nalezené prázdné hlavičky.

```
2036 IFFIND     LD   A, (#3E94) EXTE1      ;vyzvedni příponu souboru
2039            LD   (HL), A               ;uloži ji na první místo v položce adresáře
203A            INC  HL                    ;posuň se na další znak
203B            LD   DE, #3E8A FNZONE1     ;do DE adresa 1. jména souboru pro I/O
203E            LD   BC, #000A             ;10 znaků
2041            EX   DE, HL                ;zaměň ukazatele
2042            LDIR                       ;přesuň jméno do položky
2044            EX   DE, HL                ;a vrať zpět ukazatele
2045            RET                        ;vrať se
```

SAVEFILE

Uloží soubor na disk.

IN: HL adresa začátku dat
DE délka dat
IX adresa parametrů drivu
#3E78 (VALSYX) počáteční adresa uložení dat
#3E7A (VALSYY) délka BASICu pez proměnných
ve FNZONE1 je jméno souboru
OUT: data jsou z paměti uložena do souboru se jménem ve FNZONE1

Vytvoříme diskovou hlavičku souboru.

```
2046 SAVEFILE   PUSH HL                    ;ulož si adresu uložení dat
2047            PUSH DE                    ;a délku dat
2048            CALL #202C, FINDANDFILL    ;najdi první neprázdnou položku adresáře a dej do ní
                                           ;jméno souboru z FNZONE1
204B            POP  DE                    ;obnov délku dat
204C            LD   (HL), E               ;a ulož ji do hlavičky položky adresáře
204D            INC  HL
204E            LD   (HL), D
204F            INC  HL
2050            LD   BC, (#3E78) VALSYX    ;vyzvedni počáteční adresu
2054            LD   (HL), C               ;a ulož ji do hlavičky položky adresáře
2055            INC  HL
2056            LD   (HL), B
2057            INC  HL
2058            LD   BC, (#3E7A) VALSYY    ;vyzvedni délku BASIC programu bez proměnných
205C            LD   (HL), C               ;a ulož do hlavičky položky adresáře
205D            INC  HL
205E            LD   (HL), B
205F            INC  HL
2060            PUSH HL                    ;ulož si ukazatel do hlavičky položky adresáře
2061            INC  HL                    ;posuň se 2 byte položky adresáře
2062            INC  HL
2063            LD   A, (#3E7C) HEAD20     ;vyzvedni hodnotu do A (je zde 0)
2066            LD   (HL), A               ;a ulož ji do položky adresáře
2067            INC  HL                    ;posuň se na atributy
2068            LD   (HL), #0F             ;ulož atributy „RWED“
206A            INC  HL                    ;posuň se na třetí byte délky souboru
206B            LD   (HL), #00             ;ulož tam nulu
206D            CALL #201E, TRANSTOSEC     ;vypočti počet sektorů, které zabere soubor, a počet
                                           ;bytů v posledním sektoru
```

```
2070                OR   B                    ;zapisuje se nějaký sektor?
2071                LD   A, D                 ;do A vyšší byte počtu bytů v posledním sektoru
2072                JR   NZ, #2079, SAVEFILE1 ;zapisuje → skoč
```

Je to soubor s nulovou délkou.

```
2074                LD   A, #0C               ;do A příznak soubor s nulovou délkou
2076                INC  B                    ;bude se zapisovat 1 sektor
2077                JR   #207B, SAVEFILE2     ;skoč na zápis souboru
```

V posledním sektoru budou data.

```
2079 SAVEFILE1      OR   #0E                  ;přidej značku poslední sektor souboru
207B SAVEFILE2      LD   D, A                 ;dej do D obsah A – DE obsahuje hodnotu, která se
                                              ;zapíše do posledního sektoru stezky souboru
207C                CALL #210A, FINDBESEC     ;zjisti, jestli na disketě existuje blok volných sektorů
                                              ;jdoucích za sebou o délce B
207F                JR   Z, #2089, SAVEFILE3  ;ano → skoč
```

Protože není takový dlouhý blok, musíme soubor rozházet do více bloků.

```
2081                LD   HL, #0000            ;budeme prohledávat FAT od první položky FAT
2084                CALL #20F6, FIEMPTYFAT    ;najdi prázdnou položku FAT od HL
2087                JR   NZ, #20BC, RETREP    ;nenalezena → skoč na REPORT U „Disk full“
```

Zapíšeme číslo prvního sektoru do hlavičky souboru.

```
2089 SAVEFILE3      EX   DE, HL               ;dej číslo prvního nalezeného sektoru do DE
208A                EX   (SP), HL             ;do HL adresa hlavičky položky adresáře
208B                LD   (HL), E              ;ulož do ní číslo prvního sektoru
208C                INC  HL
208D                LD   (HL), D
208E                CALL #1E65, WSCADR        ;zapiš sektor adresáře na disk
2091                EX   DE, HL               ;vrať do HL číslo prvního sektoru
```

Nyní zapíšeme stezku souboru do FAT.

```
2092                POP  DE                   ;do DE délku dat v posledním sektoru
2093                PUSH HL                   ;ulož si číslo prvního sektoru
2094                CALL #20DB, SAVETOFAT     ;ulož stezku souboru do FAT
2097                INC  B                    ;byly zapsány všechny sektory do FAT?
2098                DEC  B
2099                JR   NZ, #20BC, RETREP    ;ne → skoč na REPORT U „Disk full“
209B                POP  HL                   ;obnov číslo prvního sektoru
```

Nyní uložíme soubor na disketu do vytvořené stezky. Budeme postupně zapisovat soubor do vytvořených bloků.

```
209C SAVEFILE4      PUSH HL                   ;ulož si číslo sektoru
209D                CALL #1DF9, LOGFYZ        ;převeď logický sektor na fyzický sektor a stopu
20A0                POP  HL                   ;obnov si číslo logického sektoru
20A1                PUSH BC                   ;ulož si fyzickou stopu a sektor
20A2                CALL #20C4, COUNTCSEC     ;spočti sektory následující za sebou ve stezce souboru
20A5                PUSH DE                   ;ulož si číslo posledního sektoru
20A6                LD   D, B                 ;do D počet sektorů za sebou
20A7                LD   E, #00               ;do E 255 opakování
20A9                POP  HL                   ;obnov si počet bytů v posledním sektoru
20AA                POP  BC                   ;obnov si fyzickou stopu a sektor
20AB                EX   (SP), HL             ;ulož obsah HL a vyzvedni adresu dat
```

```
20AC            LD    A, (#3E6B) WORKDR  ;do A drive, se kterým se pracuje
20AF            CALL  #2296, BWRITE      ;zapiš sektory
20B2            EX    (SP), HL           ;ulož si adresu uložení dat a vyzvedni číslo posledního
                                         ;sektoru bloku
20B3            BIT   3, H               ;je to poslední sektor souboru?
20B5            JR    Z, #209C, SAVEFILE4 ;ne → skoč
20B7            POP   HL                 ;obnov číslo sektoru
20B8            CALL  #217B, ERAVAR      ;vymaž pomocné proměnné MDOSu
20BB            RET                      ;vrať se
20BC RETREP     CALL  #217B, ERVAR       ;vymaž pomocné proměnné MDOSu
20BF            LD    A, #1D             ;REPORT U „Disk full“
20C1            JP    #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

COUNTCSEC

Vypočte počet sektorů ve stezce souboru, které jdou souvisle za sebou (22, 23, 24, 25, …).

IN: HL číslo sektoru, od kterého se bude začínat
OUT: HL číslo prvního sektoru v posloupnosti
DE číslo prvního sektoru mimo posloupnost nebo obsah posledního sek.
B počet sektorů ve stezce jdoucí za sebou
#3E7E (SVFRSC) číslo posledního sektoru

```
20C4 COUNTCSEC  LD    B, #00             ;v B bude počítadlo sektorů
20C6            PUSH  HL                 ;ulož si číslo logického sektoru
```

Budeme hledat první sektor, který již nenásleduje těsně po předchozím.

```
20C7 COUNTSEC1  LD    (#3E7E), HL SVFRSC ;ulož si číslo sektoru do SRAM
20CA            CALL  #1CF1, GETWTEST    ;načti položku FAT a otestuj chybu
20CD            DEC   DE                 ;sniž číso následujíciho sektoru ve stezce o 1
20CE            AND   A                  ;nuluj CY
20CF            SBC   HL, DE             ;odečti od aktuálního sektoru snížený následující sektor
20D1            ADD   HL, DE             ;a přičti zpět
20D2            INC   DE                 ;vrať do původního stavu následující sektor
20D3            INC   HL                 ;a posuň číslo sektoru
20D4            PUSH  AF                 ;ulož si příznak
20D5            INC   B                  ;zvyš počítadlo sektorů
20D6            POP   AF                 ;obnov příznak
20D7            JR    Z, #20C7, COUNTSEC1 ;sektory jdou za sebou → skoč na opakování
20D9            POP   HL                 ;obnov číslo
20DA            RET                      ;vrať se
```

SAVETOFAT

Zapíše cestu souboru do FAT.

IN: HL číslo počátečního sektoru
DE obsah posledního sektoru FAT
B počet sektorů
OUT: zapíše stezku souboru do FAT, počáteční sektor je v HL
B počet nezapsaných sektorů

```
20DB SAVETOFAT  PUSH  DE                 ;ulož si obsah položky posledního sektoru
```

Budeme postupně zapisovat všechny sektory do FAT.

```
20DC SAVE2FLP   PUSH  HL                 ;ulož si číslo sektoru
```

```
20DD                DEC  B                    ;sniž počítadlo sektorů
20DE                JR   Z, #20ED, SAVETOF1   ;zapsány všechny → skoč
20E0                CALL #20F6, FIEMPTYFAT    ;najdi prázdnou položku FAT od HL
20E3                JR   NZ, #20ED, SAVETOF1  ;nenalezena → skoč
20E5                EX   DE, HL               ;dej číslo nového sektoru do DE
20E6                POP  HL                   ;obnov číslo starého sektoru
20E7                CALL #1D1E, WRTOFAT       ;zapiš do položky FAT obsah DE
20EA                EX   DE, HL               ;zaměň čísla sektorů
20EB                JR   #20DC, SAVE2FLP      ;opakuj zápis do FAT tabulky
20ED SAVETOF1       POP  HL                   ;obnov číslo posledního sektoru
20EE                POP  DE                   ;obnov obsah položky posledního sektoru
20EF                CALL #1D1E, WRTOFAT       ;zapiš do položky FAT obsah DE
20F2                CALL #1D9D, WFATIFCH      ;zapiš FAT, pokud byla změněna
20F5                RET                       ;vrať se
```

FINDEMPTYFAT

Najde prázdnou položku FAT tabulky od HL.

IN: HL logický sektor, od kterého se bude prohledávat FAT
OUT: HL logický sektor, který je prázdný
Z taková položka existuje
NZ taková položka neexistuje

```
20F6 FIEMPTYFAT  PUSH DE                   ;ulož si DE
```

Postupně budeme načítat obsahy položek od HL, dokud nenarazíme na prázdnou.

```
20F7 FINDEFAT1   INC  HL                   ;zvyš číslo logického sektoru
20F8             CALL #1D04, GETFAT        ;vyzvedni obsah položky
20FB             LD   A, D                 ;je prázdná?
20FC             OR   E
20FD             JR   Z, #2107;IFFATEMPTY  ;ano → skoč
20FF             LD   DE, #06A8            ;do DE počet položek FAT
2102             SBC  HL, DE               ;už byla prohlédnuta poslední položka?
2104             ADD  HL, DE               ;přičti zpět
2105             JR   C, #20F7, FINDEFAT1  ;ne → skoč
2107 IFFATEMPTY  AND  A                    ;nastav Z podle výsledku
2108             POP  DE                   ;obnov DE
2109             RET                       ;vrať se
```

FINDBESEC

Hledá ve FAT, jestli v ní není B volných sektorů nepřetržitě za sebou.

IN: B počet hledaných sektorů.
OUT: HL číslo prvního sektoru v posloupnosti
Z taková posloupnost existuje

```
210A FINDBESEC   PUSH DE                   ;ulož si registry
210B             PUSH BC
210C             LD   HL, #0000            ;budeme prohledávat FAT od začátku
```

Hledáme posloupnost.

```
210F FINDBE1     CALL #20F6, FIEMPTYFAT    ;najdi prázdnou položku FAT od HL
2112             JR   NZ, #2123, FINDBEERR ;nenalezena → skoč
2114             PUSH HL                   ;ulož si číslo položky ve FAT
```

```
2115 FINDBE2    DEC  B                    ;sniž čítač počtu sektorů
2116            JR   Z, #2122, FINDBEOK   ;už jsi našel takový blok → skoč
2118            INC  HL                   ;zvyš číslo položky ve FAT
2119            CALL #1D04, GETFAT        ;vyzvedni obsah položky
211C            LD   A, D                 ;je následující položka FAT prázdná?
211D            OR   E
211E            JR   Z, #2115, FINDBE2    ;ano → skoč na test další položky
2120            JR   #2126, FINDBNBL      ;nenašel B volných→ skoč
```

Našel takovou posloupnost.

```
2122 FINDBEOK   POP  HL                   ;vyzvedni do HL číslo první volné položky
2123 FINDBERR   POP  BC                   ;obnov registry
2124            POP  DE
2125            RET                       ;vrať se
```

Pokud blok není dlouhý B sektorů, nastavíme počítadlo na původní stav a zkusíme hledat další blok.

```
2126 FINDBNBL   POP  BC                   ;vyzvedni hodnotu ze zásobníku
2127            POP  BC                   ;vyzvedni počet hledaných sektorů v bloku
2128            PUSH BC                   ;a znovu ho ulož
2129            JR   #210F, FINDBE1       ;a zkus hledat znovu
```

FIRSTMASK

Načte první položku adresáře, která vyhovuje masce souboru v FNZONE1.

IN: maska ve FNZONE1
IX adresa parametrů drivu
OUT: HL adresa uložení položky v buferu adresáře
Z byla nalezena
NZ taková položka není
A číslo dané položky

```
212B FIRSTMASK  LD   A, #FF               ;začínáme od první položky
```

NEXTMASK

Načte další položku od A vyhovující masce v FNZONE1.

IN: maska ve FNZONE1
A číslo položky-1, od které se bude prohledávat
IX adresa parametrů drivu
OUT: HL adresa uložení položky v buferu adresáře
Z byla nalezena
NZ taková položka není
A číslo dané položky

Postupně budeme procházet všechny neprázdné položky a testovat je na masku.

```
212D NEXTMASK   CALL #216C, RDNOEMPTY     ;načti 1. neprázdnou položku adresáře od A
2130            RET  NZ                   ;není → vrať se
2131            CALL #2137, TESTMSK       ;vyhovuje jméno souboru masce v FNZONE1?
2134            JR   NZ, #212D, NEXTMASK  ;ne → skoč na opakování hledání
2136            RET                       ;vrať se
```

TESTMSK

Zjistí, jestli jméno souboru adresovaného HL odpovídá masce v FNZONE1.

IN: maska v FNZONE1
HL adresa jména souboru
OUT: Z vyhovuje masce

```
2137 TESTMSK    PUSH HL                 ;ulož si registry
2138            PUSH BC
2139            PUSH DE
```

Nejdříve porovnáme příponu.

```
213A            LD   C, A               ;ulož si číslo položky do C
213B            LD   A, (#3E94) EXTE1   ;vyzvedni příponu masky
213E            CP   „?“                ;je to otazník?
2140            JR   Z, #2145 , TESTNM  ;ano → skoč (přeskočíme test přípony)
2142            CP   (HL)               ;porovnej s příponou ve jméně souboru
2143            JR   NZ, #2157, NONAME  ;různé → skoč
```

Nyní porovnáme jméno.

```
2145 TESTNM     INC  HL                 ;posuň se na první znak ve jméně
2146            LD   B, #0A             ;porovnáme deset znaků
2148            LD   DE, #3E8A FNZONE1  ;do DE adresa masky
214B TSTNMLOOP  LD   A, (DE)            ;vyzvedneme znak z masky
214C            CP   „?“                ;otazník?
214E            JR   Z, #2153 , NEXTTEST ;ano → skoč
2150            CP   (HL)               ;porovnej se znakem ve jméně souboru
2151            JR   NZ, #2157, NONAME  ;různé → skoč
2153 NEXTTEST   INC  DE                 ;posuň se na další znak v masce
2154            INC  HL                 ;posuň se na další znak ve jméně souboru
2155            DJNZ #214B, TSTNMLOOP   ;opakuj B-krát
2157 NONAME     LD   A, C               ;obnov si číslo položky
2158            POP  DE                 ;obnov si registry
2159            POP  BC
215A            POP  HL
215B            RET                     ;vrať se
```

FIRSTEMPTY

Najde první volnou položku v adresáři od začátku adresáře.

IN: IX adresa parametrů drivu
OUT: A číslo položky, která je první
HL adresa položky v buferu adresáře
Z položka byla nalezena
NZ položka nebyla nalezena

```
215C FIRSTEMPTY LD   A, #FF             ;začínáme od první položky adresáře
```

NEXTEMPTY

Najde další volnou položku v adresáři od položky v A.

IN: A číslo položky –1, od které se bude prohledávat adresář
IX adresa parametrů drivu
OUT: A číslo nalezené položky v adresáři
HL adresa položky v buferu adresáře
Z položka byla nalezena
NZ položka nebyla nalezena

Budeme postupně procházet všechny položky adresáře, dokud nenarazíme na prázdnou.

```
215E NXTEMPT  CALL #1E0B, READADR      ;načti položku adresáře v A
2161          RET  NZ                  ;neexistuje → vrať se
2162          PUSH BC                  ;ulož si BC
2163          LD   B, A                ;ulož si číslo položky do B
2164          LD   A, (HL)             ;vyzvedni příponu
2165          CP   #E5                 ;je to prázdná položka?
2167          LD   A, B                ;obnov číslo položky
2168          POP  BC                  ;a BC
2169          JR   NZ, #215E, NXTEMPT  ;ne → skoč na hledání další položky
216B          RET                      ;vrať se
```

RDNOEMPTY

Přečte od A první neprázdnou položku adresáře.

IN: A číslo položky –1, od které se bude prohledávat
IX adresa parametrů drivu
OUT: HL adresa položky v buferu adresáře
Z byla nalezena
NZ nebyla nalezena
A číslo položky

```
216C RDNOEMPTY CALL #1E0B, READADR      ;načti danou položku v A adresáře z disku
216F           RET  NZ                  ;neexistuje → vrať se
2170           PUSH BC                  ;ulož BC
2171           LD   B, A                ;ulož si číslo položky do B
2172           LD   A, (HL)             ;vyzvedni první znak z názvu
2173           CP   #E5                 ;prázdná položka adresáře?
2175           LD   A, B                ;vrať číslo položky zpět do A
2176           POP  BC                  ;obnov BC
2177           JR   Z, #216C, RDNOEMPTY;je prázdná → skoč na hledání další
2179           CP   A                   ;nastav Z
217A           RET                      ;vrať se
```

ERAVAR

Vymaže pomocné proměnné MDOSu pro práci s adresářem a FAT.

IN: –
OUT: vymazaní všech proměnných MDOSu

```
217B          XOR  A                   ;nastav do A nulu
217C          LD   (#3E6D), A FATSC    ;nuluj číslo posledně čteného sektoru FAT
217F          LD   (#3E6C), A CHNGFLAG ;nastav příznak „žádné změny ve FAT“
2182          LD   (#3E6F), A ADRSCTRhi ;nuluj sektor, odkud byl naposledy čten adresář
2185          LD   (#3E70), A ADRSCTRlo ;nuluj stopu, odkud byl naposledy čten adresář
2188          LD   (#3E68), A VARIA1   ;bez významu
218B          LD   (#3E69), A VARIA2   ;bez významu
218E          DEC  A                   ;do A dej 255
218F          LD   (#3E6A), A VARIA3   ;bez významu
2192          LD   (#3E6E), A FATDR    ;nuluj číslo drivu, ze kterého byla naposledy čtena FAT
2195          LD   (#3E71), A ADRDR    ;vymaž číslo disku, odkud byl naposledy čten adresář
2198          RET
```

NAMEDISK

Do HL vypočte adresu jména diskety v drivu, na jehož parametry ukazuje IX.

IN: IX adresa parametrů drivu
OUT: HL adresa uložení jeho jména

```
2199 NAMEDISK   PUSH IX                 ;přenes ukazatel na parametry drivu do HL
219B            POP  HL
219C            LD   DE, #0030          ;do DE dej 48 (délka parametrů drivů)
219F            ADD  HL, DE             ;přičti k HL
21A0            RET                     ;vrať se
```

DRVSYS

Vypočte adresu tabulky parametrů disku, se kterým se pracuje.

IN: WORKDR číslo drivu, se kterým se pracuje
OUT: IX adresa parametrů disku

```
21A1 DRVSYS     PUSH BC                 ;ulož si registry
21A2            PUSH AF
21A3            LD   A, (#3E6B) WORKDR  ;do A drive, se kterým se pracuje
21A6            CALL #21AC, DRVCMP      ;zjisti adresu parametrů drivu v A
21A9            POP  AF                 ;obnov registry
21AA            POP  BC
21AB            RET                     ;vrať se
```

DRVCMP

Vypočte adresu tabulky parametrů disku.

IN: A číslo drivu
OUT: IX adresa tabulky parametrů drivu
Z drive není připojen
NZ drive je připojen

```
21AC DRVCMP     RLCA                    ;délka parametrů jednoho disku je 12 bytů
21AD            RLCA                    ;vynásobíme tedy číslo drivu dvanácti
21AE            LD   C, A               ;získáme tak relativní adresu začátku parametrů
21AF            RLCA                    ;drivu v A
21B0            ADD  A, C
21B1            LD   C, A               ;dáme ji do BC
21B2            LD   B, #00
21B4            LD   IX, #3E00 DRPARZN  ;do IX začátek tabulek parametrů disků
21B8            ADD  IX, BC             ;přičti relativní adresu
21BA            BIT  0, (IX+#00)        ;testuj, jestli je mechanika připojena
21BE            RET                     ;vrať se
```

KEYMSG

Tiskne hlášení a dotaz v A a čeká na stisk klávesy R nebo P.

IN: A 0. až 6. bit určují číslo hlášení v systémových hlášeních a 7. bit určuje typ dotazu
DE tabulka systémových hlášení
OUT: vytiskne hlášení s dotazem na obrazovku a čeká na stisk klávesy
C bylo stisknuto P nebo R
NC nebylo stisknuto ani P ani R

```
21BF KEYMSG     PUSH HL                     ;ulož si registry
21C0            PUSH BC
```

Uložíme si aktuální kanál pro vstup a výstup, aby ho bylo možno vrátit do původního stavu.

```
21C1            LD   HL, (#5C51) CURCHL     ;vyzvedni adresu kanálu, který je používan pro vstup
21C4            PUSH HL                     ;a výstup – ulož si ji
21C5            PUSH AF                     ;a ještě další registry
21C6            PUSH DE
21C7            LD   A, #FD                 ;kanál –3 (dolní část obrazovky)
21C9            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
21CA            DW   #1601                  ;podprogram CHAN-OPEN otevření kanálu
```

Nejdříve vytiskneme hlášení.

```
21CC            POP  DE                     ;obnov si ukazatel na tabulku hlášení
21CD            POP  AF                     ;a číslo hlášení
21CE            PUSH AF                     ;znovu si ulož číslo hlášení
21CF            AND  #7F                    ;zruš 7. bit
21D1            INC  A                      ;a zvyš o 1
21D2            CALL #01C8, PRTMES          ;tiskni hlášení na obrazovku
```

Nyní vytiskneme dotaz.

```
21D5            POP  AF                     ;vyzvedni číslo hlášení
21D6            AND  #80                    ;ponechej pouze 7. bit
21D8            RLCA                        ;a posuň 7. bit do 0. bitu – dostaneme tak číslo dotazu
21D9            LD   DE, #21FB TXTQUE       ;do DE adresa dotazů
21DC            CALL #01C8, PRTMES          ;piš text dotazu
21DF            SET  5, (IY+#02) TVFLAG     ;nastav signál, že spodní část obrazvky bude potřeba vyčistit
```

Nyní čekáme na stisk klávesy. To není celkem dobře vyřešeno, protože pokud v době volání byl kurzor nastaven v jiném módu, než L nebo C, jsou špatně interpretovány kódy (UDG kódy nebo kódy klíčových slov).

```
21E3            RST  #28                    ;volej podprogram pro volání rutiny ZX ROM
21E4            DW   #15D4                  ;podprogram WAIT_KEY čekání na stisk klávesy
21E6            POP  HL                     ;obnov do HL adresu kanálu pro vstup a výstup
21E7            LD   (#5C51), HL CURCHL     ;ulož ji zpět (vrátíme vše do původního stavu)
21EA            POP  BC                     ;obnov ostatní registry
21EB            POP  HL
21EC            LD   A, (#5C08) LAST_K      ;do A kód naposledy stisknuté klávesy
21EF            AND  #DF                    ;převeď na velké písmeno
21F1            CP   „R“                    ;bylo stisknuto „R“?
21F3            SCF                         ;nastav CY
21F4            RET  Z                      ;ano → vrať se
21F5            CP   „P“                    ;bylo stisknuto „P“?
21F7            SCF                         ;nastav CY
21F8            RET  Z                      ;ano → vrať se
21F9            AND  A                      ;nuluj CY – nastav signál „neplatná klávesa“
21FA            RET                         ;vrať se
```

TXTQUE

Tabulka textů dotazů.

```
21FB TXTQUE     DB   #80                    ;invertovaný znak
```

```
   Dotaz 0
21FC            20 28 52 65 74 72 79 20 3D 20 52 A9      ;(Retry-R)
   Dotaz 1
2208            20 28 50 72 6F 63 65 65 64 20 3D 20 50 A9 ;(Proceed-P)
```

HWINIT

Test disketových mechanik, zjištění počtu stop, inicializace obvodu 8255.

IN: –

OUT: nastavení parametrů mechanik a inicializace obvodu 8255

Nejdříve inicializujeme řadič.

```
2216 HWINIT     LD   A, #D0          ;příkaz přerušení činnosti řadiče
2218            OUT  (#81), A       ;vyšli příkaz řadiči
221A            XOR  A              ;bude se testovat od mechaniky A
```

Nyní budeme postupně testovat jednotlivé mechaniky a zapisovat jejich parametry do SRAM.

```
221B HWINI0     PUSH AF             ;schovej si číslo mechaniky
221C            CALL #21AC, DRVCMP  ;vypočti adresu parametrů mechaniky v A
221F            PUSH IX             ;předej ji do HL
2221            POP  HL
2222            INC  HL             ;zvyš ji o 1, ukazuje teď na parametry diskety
2223            LD   E, L           ;a dej ji do DE
2224            LD   D, H
2225            INC  HL             ;zvyš ji o 4, ukazuje teď na parametry mechaniky
2226            INC  HL
2227            INC  HL
2228            INC  HL
2229            LD   BC, #0003      ;zkopíruj 3 parametry z 5. na 1.
222C            LDIR
222E            RES  0, (IX+#00)    ;nastav signál drive nepřipojen
2232            RES  7, (IX+#00)    ;nastav příznak neporovnávat číslo stopy s registrem
                                    ;řadiče (tady to je z důvodů inicializace mechanik)
2236            LD   A, (IX+#02)    ;vyzvedni počet stop do A
2239            AND  A              ;je drive definován?
223A            JR   Z, #2271, HWINI1 ;ne → skoč
223C            POP  AF             ;obnov si číslo mechaniky
223D            PUSH AF             ;a znovu si ho schovej
223E            CALL #254B, DRVSEL  ;roztoč mechaniku
2241            CALL #234B, HOME    ;pošli hlavu na stopu 0
2244            OUT  (#87), A       ;vyšli získaný status jako číslo stopy do registru řadiče
2246            AND  #04            ;je disk připojen? (test signálů BUSY a TRACK 00)
2248            JR   Z, #2271, HWINI1 ;ne → skok na test dalšího disku
224A            SET  0, (IX+#00)    ;nastav signál disk připojen
224E            LD   (IX+#04), #00  ;ulož informaci, že hlava je na stopě nula
```

Nyní probíha zajímave zjišťování, o jakou mechaniku se vlastně jedná. Jestli o 40-ti nebo 80-ti stopou mechaniku. Probíhá to asi tak, že se nejdříve vystaví hlava na 54 stopu a potom hned na 2 stopu. Pokud je mechanika 40-ti stopá, hlásí ve statusu, že je na stopě 0. To, co slyšíte při resetu počítače (hrk, hrk, hrk, vrrrrrrr…) je vystavování hlavy právě na stopu 54. Tam už totiž žádná 40-ti stopá mechanika nedosáhne. A to se potom může zbláznit.

```
2252            LD   A, #36         ;do A stopa 54
2254            LD   D, #10         ;do D příkaz SEEK (nastav na stopu)
```

```
2256            CALL #2340, SEEK        ;proveď vystavení na stopu 54
2259            LD   A, #02             ;stopa 2
225B            LD   D, #10             ;do D příkaz SEEK (nastav na stopu)
225D            CALL #2340, SEEK        ;proveď vystavení na stopu 2
2260            AND  #04                ;byla vystavení hlavy v pořádku?
2262            LD   A, #28             ;do A si dáme 40 stop
2264            JR   NZ, #2268, TRK40   ;ne → skoč na disk 40 stop
2266            LD   A, #50             ;disk má 80 stop
2268 TRK40      LD   (IX+#06), A        ;ulož informaci o počtu stop do parametrů drivu
226B            LD   (IX+#03), A        ;a také do parametrů diskety
226E            CALL #234B, HOME        ;proveď vystavení hlavy na stopu 0, ať se 40-ti stopé
                                        ;mechaniky dají dohromady
2271 HWINI1     POP  AF                 ;obnov číslo mechaniky
2272            INC  A                  ;posuň na další mechaniku
2273            CP   #02                ;už byly otestovány dva disky?
2275            JR   C, #221B, HWINI0   ;ne → testuj další
2277            CALL #2536, DSKSTP      ;vypni mechaniky
```

Nyní otestujeme, jestli není připojen nějaký jiný obvod 8255. Pokud ano, je obvod v mechanice zablokován. Tento test je ale chybný, protože ať je nebo není připojen jiný obvod 8255, dojde vždy k zablokování obvodu v disketové jednotce.

```
227A            LD   A, #88             ;inicializační hodnota interface v D40
                                        ;A0, PA=výstupní, PCH=vstupní, B0, PB=výstupní,
                                        ;PCL=výstupní. Toto však není nejlepší. Spíše by mělo
                                        ;být LD A, #80
227C            OUT  (#7F), A           ;inicializuj obvod 8255
227E            LD   A, #0F             ;nastav PC7=1
2280            OUT  (#7F), A
2282            LD   A, #0B             ;nastav PC5=1
2284            OUT  (#7F), A
2286            IN   A, (#5F)           ;čti PC
2288            AND  #F0                ;ponech PC4 – PC7
228A            CP   #A0                ;byl zjištěn jiný 8255?
228C            LD   A, #9B             ;všechny porty jako vstupní
228E            OUT  (#7F), A
2290            RET  Z                  ;ano → vrať se při jiném 8255
2291            LD   A, #20             ;povol činnost portu
2293            OUT  (#91), A
2295            RET                     ;vrať se
```

V opravené verzi MDOSu 1.0 je test obvodu 8255 zrušen. Vypadá to takto:

```
227A            LD   A, #C7             ;význam těchto instrukcí mi není znám
227C            LD   (#0066), A         ;toto je skutečně nějaká…
227F            RET                     ;ale tady už to dává smysl, je to návrat zpět
2280            DB   #FF, #FF, #FF, #FF, #FF
2285            DB   #FF, #FF, #FF, #FF, #FF
228A            DB   #FF, #FF, #FF, #FF, #FF
2290            DB   #FF, #FF, #FF, #FF, #FF
2295            RET
```

Komentovaný výpis MDOSu – skvělý zdroj informací

BWRITE

Zapíše sektor nebo řadu sektorů na disk.

IN: HL adresa, odkud zapisovat
B číslo stopy
C číslo sektoru
D kolik sektorů se má zapisovat
E počet opakování při chybě CRC
A číslo drivu, na který se bude zapisovat

```
2296 BWRITE     PUSH HL                     ;ulož si datový ukazatel
2297            LD   HL, #23BE DWRITE       ;do HL adresa rutiny pro načtení sektoru
229A            JR   #22A9, BRWR0           ;skoč na společnou část
```

BFORMA

Formátuje stopu, parametry jsou shodné jako u BWRITE (mimo počet sektorů a čísla sektoru).

```
229C BFORMA     PUSH HL                     ;ulož si datový ukazatel
229D            LD   HL, #23D8 DFORMA       ;do HL adresa rutiny pro formátování stopy
22A0            JR   #22A9BRWR0             ;skoč na společnou část
```

BREADA

Načte sektor nebo řadu sektorů z disku. Parametry jsou shodné jako u BWRITE až na registr A.

```
22A2 BREADA     LD   A, (#3E6B) WORKDR      ;do A drive, se kterým se pracuje
```

BREAD

Má stejnou funkci jako BREADA, ale v registru A je číslo drivu, ze kterého se bude číst.

```
22A5 BREAD      PUSH HL                     ;ulož si datový ukazatel
22A6            LD   HL, #236A DREAD        ;do HL adresa rutiny pro čtení sektoru
```

Vytvoříme si v SRAM skok na požadovanou rutinu.

```
22A9 BRWR0      LD   (#3E64), HL MODJPA2    ;nastav skok na rutinu přes SRAM
22AC            LD   HL, #3E63 MODJP1       ;na adresu #3E63
22AF            LD   (HL), #C3              ;ulož kód instrukce JP
22B1            POP  HL                     ;obnov datový ukazatel
```

Provedeme operaci pro jeden sektor.

```
22B2 BRWL0      PUSH BC                     ;ulož si registry
22B3            PUSH AF
22B4            PUSH DE
22B5            PUSH HL
22B6            CALL #3E63, MODJP1          ;zavolej rutinu čtení/zápis/formatování
22B9            INC  C                      ;byla nějaká chyba??
22BA            DEC  C
22BB            JR   Z, #22E7, BREAD1       ;ne → skoč
22BD            DEC  B                      ;budeme testovat základní chyby?
22BE            JR   NZ, #2300, BREAD2      ;ne → skoč
22C0            BIT  7, C                   ;byla vložena diskety v mechanice?
22C2            JR   Z, #22D3, BREAD3       ;ano → skoč
```

V mechanice nebyla disketa.

```
22C4 BREA71     LD    A, #36                ;REPORT n „Drive is not ready“ a dotaz „(Retry = R)“
22C6 BREA72     LD    DE, #03AF SYSMSG      ;do DE adresa systémových hlášení
22C9 BREAD6     CALL  #21BF, KEYMSG         ;tiskni hlášení a čekej na stisk klávesy
22CC            JR    C, #22E1, BREAD4      ;bylo stisknuto P nebo R → skoč na opakování operace
22CE            LD    A, #29                ;REPORT a „Device I/O error“
22D0            JP    #0204, ERRR           ;piš hlášení a skoč do ZX ROM
```

Otestujeme chyby při operaci.

```
22D3 BREAD3     LD    A, C                  ;dej výsledek operace do A
22D4            AND   #18                   ;nastaven signál SEEK error, CRC error nebo RNF?
22D6            JR    NZ, #22DD, BREA31     ;ano → skoč
```

Technická chyba řadiče nebo mechaniky.

```
22D8 ERR45      LD    A, #3B                ;REPORT s „Internal error“
22DA            JP    #0204, ERRR           ;piš hlášení a skoč do ZX ROM
```

Tiskni chybu při hledání stopy nebo sektoru a čekej na stisk klávesy pro opakování operace.

```
22DD BREA31     LD    A, #37                ;REPORT o „Seek error“
22DF            JR    #22C6, BREA72         ;skoč na tisk hlášení s volbou
```

Bylo stisknuto R nebo P, proto obnovíme registry a opakujeme operaci.

```
22E1 BREAD4     POP   HL                    ;obnov si registry
22E2            POP   DE
22E3            POP   AF
22E4            POP   BC
22E5            JR    #22B2, BRWL0          ;zkus to znovu
```

Tudy se jde, pokud byla operace v pořádku.

```
22E7 BREAD1     POP   HL                    ;obnov si registry
22E8            POP   DE
22E9            POP   AF
22EA            LD    BC, #0200             ;zvyš adresu dat o 512
22ED            ADD   HL, BC                ;a přitom stačilo jen INC H, INC H
22EE            POP   BC                    ;obnov číslo stopy a sektoru
22EF            PUSH  AF                    ;schovej si číslo drivu
22F0            INC   C                     ;posuň se na další sektor
22F1            LD    A, (IX+#03)           ;do A počet sektorů na stopu
22F4            CP    C                     ;jsi na konci stopy?
22F5            JR    NZ, #22FA, BREA11     ;ne → skoč
22F7            LD    C, #00                ;do C dej nultý sektor
22F9            INC   B                     ;a posuň se na další stopu
22FA BREA11     POP   AF                    ;obnov číslo drivu
22FB            DEC   D                     ;načteny všechny sektory?
22FC            JP    NZ, #22B2, BRWL0      ;ne → opakuj
22FF            RET                         ;vrať se
2300 BREAD2     DEC   B                     ;testování chyb příkazu READ SECTOR?
2301            JR    NZ, #231C, BREAD7     ;ne → skoč dál
```

Budeme testovat chyby při čtení sektoru.

```
2303            LD    A, C                  ;dej výsledek operace do A
```

```
2304            AND  #9D                ;byla nějaká chyba při operaci READ SECTOR?
2306            JR   Z, #22E7, BREAD1   ;ne → skoč
2308 BREA81     BIT  7, C               ;testuj, jestli byla vložena disketa
230A            JR   NZ, #22C4, BREA71  ;ano → skoč
230C            BIT  4, C               ;byl nalezen sektor?
230E            JR   Z, #2314, BREAD8   ;ano → skoč
2310            LD   A, #38             ;REPORT p „Sector not found“
2312            JR   #22C6, BREA72      ;skoč na tisk hlášení s volbou
2314 BREAD8     BIT  3, C               ;byl CRC error?
2316            JR   Z, #22D8, ERR45    ;ne → skoč
2318            LD   A, #39             ;REPORT q „CRC error“
231A            JR   #22C6, BREA72      ;skoč na tisk hlášení s volbou
231C BREAD7     DEC  B                  ;testování chyb při příkazu WRITE SECTOR?
231D            JR   NZ, #232A, BREAD9  ;ne → skoč dál
```

Budeme testovat chyby při zápisu sektoru.

```
231F            LD   A, C               ;dej výsledek operace do A
2320            AND  #DD                ;byla nějaká chyba při operaci WRITE SECTOR?
```

Tady chybí **JR Z, BREAD1**. Nemá to však žádný vliv na práci.

```
2322            BIT  6, C               ;byla disketa chráněna proti zápisu?
2324            JR   Z, #2308, BREA81   ;ne → skoč
```

A toto je asi nejzákeřnější chyba, která se zde vyskytuje (je snad ještě horší, než v příkazu MOVE). Pokud totiž máte v mechanice disketu, která je chráněná proti zápisu, a budete si chtít něco uložit, systém vám hlásí, že v mechanice je disketa chráněná proti zápisu, a čeká na stisk klávesy R nebo P. Pokud nyní disketu vyměníte a stisknete klávesu R nebo P, stane se něco velice nepříjemného. Dojde totiž k přepsání sektoru adresáře. V buferu je totiž uložen celý sektor a když vy vyměníte disketu, zapíše se tento sektor. Takže místo souborů, které byly na nové disketě, se vám při výpisu objeví soubory ze zalepené diskety. Nabídka volby s návratem je zde proto, aby byla možnost disketu zbavit ochrany. Uživatel si to musí pamatovat.

```
2326            LD   A, #3A             ;REPORT r „Disk is write protected“
2328            JR   #22C6, BREA72      ;skoč na tisk hlášení s volbou
```

Správně mělo být **JP #0204, ERRR.**

Tady se testují chyby, ke kterým došlo při FIND TRACK a jsou způsobeny hardwarovou chybou při komunikaci mezi řadičem a mechanikou.

```
232A BREAD9     BIT  0, C               ;nedošlo k chybě v komunikaci s mechanikou?
232C            JR   Z, #2333, BREA91   ;ne → skoč
232E            LD   A, #22             ;REPORT Z „Device unavailable“
2330            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
2333 BREA91     BIT  1, C               ;je disketa vůbec formátována pod MDOSem (512 bytů
                                        ;na sektor)?
2335            JR   Z, #22D8, ERR45    ;ano → skoč
2337 BADTYP     LD   A, #20             ;REPORT X „Bad device type“
2339            JP   #0204, ERRR        ;piš hlášení a skoč do ZX ROM
```

Toto je zapomenutý kód.

```
233C            OR   #01
233E            EI
233F            RET
```

SEEK

Nastaví hlavu na stopu.

IN: A číslo stopy
D příkaz SEEK
IX adresa parametrů drivu
OUT: nastaví hlavu na požadovanou stopu
A výsledek operace

```
2340 SEEK       OUT  (#87), A          ;pošli číslo stopy do registru řadiče
2342            AND  A                 ;je to stopa 0?
2343            JR   Z, #234B, HOME    ;ano → skoč na povel RESTORE
2345            CALL #25C7, DELAY      ;časová prodleva
2348            LD   A, D              ;vezmi příkaz pro řadič z D
2349            JR   #234D, TRACKSEEK  ;proveď příkaz
```

HOME

Pošle hlavu disku na stopu 0.

IN: IX adresa parametrů drivu
OUT: hlava je vystavena na stopu 0
A výsledek operace

```
234B HOME       LD   A, #08            ;do A příkaz RESTORE (vystav hlavu na stopu 0)
```

Provedme vystavení hlavy.

```
234D TRACKSEEK  LD   B, A              ;dej příkaz do B
234E            LD   A, (IX+#01)       ;vezmi parametr diskety
2351            AND  #C0               ;ponech rychlost krokování
2353            RLCA                   ;a posuň je do 0. a 1. bitu
2354            RLCA
2355            OR   B                 ;přidej příkaz pro řadič
2356            OUT  (#81), A          ;vyšli příkaz do řadiče
2358            CALL #25C7, DELAY      ;časová prodleva
```

Nyní budeme čekat na ukončení operace.

```
235B WAITBUSY   IN   A, (#81)          ;čti status
235D            BIT  0, A              ;je konec operace?
235F            JR   NZ, #235B, WAITBUSY ;ne → čekej
```

A ještě chvilku počkáme.

```
2361            LD   BC, #000F         ;časová prodleva po skončení BUSY
2364 WAITHOME   DJNZ #2364, WAITHOME   ;opakuj B-krát
2366            DEC  C                 ;sniž C
2367            JR   NZ, #2364, WAITHOME;a ještě C-krát
2369            RET                    ;vrať se
```

DREAD

Načte jeden sektor z diskety.

IN: B číslo stopy
C číslo sektoru
E počet opakování při chybě
HL adresa uložení dat
OUT: C výsledek operace

B hodnota 2
načte sektor do paměti

```
236A DREAD      CALL #2493, FINDTRACK     ;najdi stopu
236D            JR   NC, #23BA, DOOPRET   ;nenalezena → skoč
236F            LD   A, #88               ;do A příkaz READ SECTOR
2371            LD   B, #02               ;do B příznak pro testování výsledných bitů příkazu
                                          ;READ SECTOR
2373            LD   IX, #25EA, REANMI    ;do IX adresa programu pro čtení dat
```

DOWDCOM

Rutina pro vykonání povelu řadiče WD2797.

IN: A povel pro WD2797
B 02 – READ SECTOR
03 – WRITE SECTOR
E počet opakování při chybě
HL adresa uložení dat
IX použitá rutina #25EA INI+RET
#25ED OUTI+RET

Nejdříve nastavíme podmínky přenosu a stranu diskety.

```
2377 DOWDCOM    PUSH AF                   ;ulož si povel
2378            LD   A, (#3EE9) SELSTA1   ;do A bity výběru a rozběhnutí mechaniky
237B            SET  6, A                 ;povol NMI
237D            CALL #25BC, OUTTODR       ;nastav podmínky NMI
2380            POP  AF                   ;vyzvedni příkaz pro řadič
2381            PUSH HL                   ;schovej adresu buferu dat
2382            LD   HL, #3EEB SVSIDE     ;do HL adr. uložení informace o používané straně diskety
2385            OR   (HL)                 ;vlož tuto informaci do povelu
2386            POP  HL                   ;vyzvedni adresu buferu dat
```

Provedeme příkaz.

```
2387 DOWDCREP   PUSH AF                   ;ulož si registry
2388            PUSH HL
2389            PUSH DE
238A            PUSH BC
238B            LD   C, #87               ;nastav C na datový registr řadiče
238D            LD   D, #01               ;do D maska pro signál BUSY
238F            LD   B, C                 ;dej do B nenulovou hodnotu
2390            OUT  (#81), A             ;vyšli povel do řadiče
```

Nyní budeme čekat na zahájení operace.

```
2392 DOWDL1     IN   A, (#81)             ;čti status operace
2394            LD   B, C                 ;dej do B nenulovou hodnotu
2395            AND  D                    ;začala probíhat operace?
2396            JR   Z, #2392, DOWDL1     ;ne → čekej na zahájení operace
```

Nyní je zahájen provoz smyčky, která čeká na konec operace. Během jejího provozu dochází k požadavku DRQ, který vyvolá NMI a samotná operace přístupu na disk je provedena na adrese (IX). Po ukončení operace se řízení vrátí opět sem.

```
2398 DOWDL2     IN   A, (#81)             ;čti status operace
239A            LD   B, C                 ;dej do B nenulovou hodnotu
```

```
239B            AND  D                      ;probíhá operace?
239C            JR   NZ, #2398, DOWDL2      ;ano → čekej na konec operace
```

Operace je u konce, zrušíme požadavek NMI a otestujeme chybu CRC.

```
239E            LD   A, (#3EE9) SELSTA1     ;do A stav NMI a mechanik
23A1            RES  6, A                   ;nuluj požadavek NMI
23A3            CALL #25BC, OUTTODR         ;a odešli ho řadiči, mechaniky se nezastaví
23A6            IN   A, (#81)               ;čti výsledek operace
23A8            POP  BC                     ;obnov registry
23A9            POP  DE
23AA            POP  HL
23AB            LD   D, A                   ;ulož výsledek do A
23AC            IN   A, (#85)               ;načti číslo sektoru z registu řadiče do A
23AE            DEC  A                      ;sniž o jeden (posuneme se zpět na daný sektor)
23AF            OUT  (#85), A               ;a odešli ho zpět do registru řadiče
23B1            POP  AF                     ;obnov AF
23B2            BIT  3, D                   ;byla chyba CRC?
23B4            JR   Z, #23B9, DONOCRC      ;ne → skoč
23B6            DEC  E                      ;sniž počet opakování
23B7            JR   NZ, #2387, DOWDCREP    ;další pokus? ano → skoč
```

Operace je u konce.

```
23B9 DONOCRC    LD   C, D                   ;předej status do C
23BA DOOPRET    CALL #2520, DISKRET         ;uveď do původního stavu NMI a IX
23BD            RET                         ;vrať se
```

DWRITE

Zapíše jeden sektor na disketu.

IN: **B číslo stopy**
C číslo sektoru
E počet opakování při chybě
HL adresa uložení dat

OUT: **C výsledek operace**
B hodnota 3
zapíše sektor na disketu

```
23BE DWRITE     CALL #2493, FINDTRACK       ;nastav hlavu na stopu v B
23C1            JR   NC, #23B9, DONOCRC     ;nenalezena → skoč
23C3            BIT  5, (IX+#01)            ;otestuj, jestli je 40-ti stopá disketa v 80-ti stopé mechanice
23C7            JR   Z, #23CE, DWRITE1      ;ne → skoč
```

Nelze tedy zapsat na takovouto disketu.

```
23C9 ERR40IN80  LD   BC, #0402              ;do B chyba při FINDTRACK a do C příznak chyby
                                            ;„Internal error“
23CC            JR   #23BA, DOOPRET         ;skoč na nastavení NMI, IX a vrať se zpět
23CE DWRITE1    LD   IX, #25ED, WRINMI      ;do IX adresa programu pro zápis dat
23D2            LD   A, #A8                 ;do A příkaz WRITE SECTOR
23D4            LD   B, #03                 ;budou se testovat výsledné bity příkazu WRITE SECTOR
23D6            JR   #2377, DOWDCOM         ;skoč na vykonání příkazu
```

DFORMA

Vytvoří obraz stopy ve VIDEORAM a formátuje stopu disku.

IN: B číslo stopy
E počet opakování při chybě
OUT: C výsledek operace
B hodnota 3
naformátuje stopu

```
23D8 DFORMA     CALL #2518, FORFINDTR     ;nastav hlavu na stopu v B
23DB            JR   NC, #23BA, DOOPRET   ;nenalezena → skoč
23DD            BIT  5, (IX+#01)          ;otestuj, jestli je 40-ti stopá disketa v 80-ti stopé mechanice
23E1            JR   NZ, #23C9, ERR40IN80 ;ano → skoč
```

Nelze tedy ani formátovat takovouto disketu.

Nastavíme stejnou barvu podkladu a inkoustu v atributech.

```
23E3            PUSH BC                   ;ulož si číslo stopy
23E4            LD   HL, #5800            ;do HL adresa začátku atributů obrazovky
23E7            LD   A, (#5C8D) ATTR_P    ;vezmi platný atribut do A
23EA            AND  #38                  ;vyber PAPER
23EC            LD   B, A                 ;schovej si ho do B
23ED            RRCA                      ;posuň na INK
23EE            RRCA
23EF            RRCA
23F0            OR   B                    ;a přidej PAPER – PAPER=INK
23F1            LD   BC, #0003            ;celá obrazovka
23F4 FSECOLOR   LD   (HL), A              ;změň atribut
23F5            INC  HL                   ;posuň se na další
23F6            DJNZ #23F4, FSECOLOR      ;opakuj B-krát
23F8            DEC  C                    ;sníž počítadlo třetin
23F9            JR   NZ, #23F4, FSECOLOR  ;nebyly všechny → skoč
```

Nyní ve VIDEORAM vytvoříme obraz stopy.

```
23FB            LD   HL, #4000            ;do HL adresa buferu pro vytvoření obrazu stopy
                                          ;je použit bufer obrazovky
23FE            IN   A, (#83)             ;načti do A číslo stopy z registru řadiče
2400            OUT  (#FE), A             ;nastav BORDER podle čísla stopy
```

sudé stopy mají fyzické pořadí sektorů 1-2-3-4-5-6-7-8-9
liché stopy mají fyzické pořadí sektorů 5-6-7-8-9-1-2-3-4

```
2402            AND  #01                  ;vyber 1. bit čísla stopy
2404            RLCA                      ;vynásob pěti
2405            RLCA                      ;A×4
2406            INC  A
2407            LD   E, A                 ;počáteční číslo sektoru do E
2408            LD   D, #00               ;nuluj čítač sektorů na stopě
```

Příprava dat pro formátování stopy.

```
240A MKFDATA    LD   A, #4E               ;úvodní mezera na stopě
240C            LD   B, #0A               ;deset bytů
```

Úvodní mezera před prvním sektorem je dost malá, takže může dojít k tomu, že nenačtete první sektor na stopě.

```
240E            CALL #248E, FILLCONST     ;ulož do dat
2411            LD   A, #00               ;volné místo
2413            LD   B, #0C               ;12 bytů
```

```
2415        CALL #248E, FILLCONST   ;ulož do dat
2418        LD   A, #F5             ;příznak – zapisuje 0A1
241A        LD   B, #03             ;tři byty
241C        CALL #248E, FILLCONST   ;ulož do dat
241F        LD   A, #FE             ;adresová značka ID pole
2421        LD   (HL), A            ;ulož do dat
2422        INC  HL                 ;a posuň se na další
2423        IN   A, (#83)           ;načti do A číslo stopy z registru řadiče
2425        LD   (HL), A            ;ulož do dat
2426        INC  HL                 ;a posuň se na další
```

Nyní se do ID pole zapisuje číslo strany. Na první phled se to zdá v pořádku (MDOS pracuje normálně). Ale pokud budete chtít takto naformátovanou disketu číst na počítači PC (Personal Computer), tak to nepůjde. Abych to upřesnil, tak nepůjde přečíst 2. strana diskety. U 1. strany to není problém, protože se zde ukládá 0, ale pro 2. stranu se zde ukládá 2. MDOSu to nevadí, protože neprovádí kontrolu strany, ale MS-DOS ji provádí a pro něj je tato hodnota nepřípustná (správně by tam měla být 1). Takže pokud budete chtít diskety D40 používat na PC, musíte si napsat vlastní formátovací rutinu, která už bude upravená.

```
2427        LD   A, (#3EEB) SVSIDE  ;vyzvedni číslo strany
242A        LD   (HL), A            ;ulož do dat
242B        INC  HL                 ;a posuň se na další
242C        LD   A, E               ;do A číslo sektoru
242D        LD   (HL), A            ;ulož do dat
242E        INC  HL                 ;a posuň se na další
242F        LD   A, #02             ;údaj délky sektoru (512 bytů)
2431        LD   (HL), A            ;ulož do dat
2432        INC  HL                 ;a posuň se na další
2433        LD   A, #F7             ;příznak pro vyvolání uložení CRC pro ID pole
2435        LD   (HL), A            ;ulož do dat
2436        INC  HL                 ;a posuň se na další
2437        LD   A, #4E             ;byty oddělující hlavičku sektoru od dat
2439        LD   B, #16             ;22 bytů
243B        CALL #248E, FILLCONST   ;ulož do dat
243E        LD   A, #00             ;volné místo
2440        LD   B, #0C             ;12 bytů
2442        CALL #248E, FILLCONST   ;ulož do dat
2445        LD   A, #F5             ;příznak zapisování 0A1
2447        LD   B, #03             ;tři byty
2449        CALL #248E, FILLCONST   ;ulož do dat
244C        LD   A, #FB             ;adresová značka pole dat
244E        LD   (HL), A            ;ulož do dat
244F        INC  HL                 ;a posuň se na další
2450        LD   A, #E5             ;místo pro uložení dat
2452        LD   B, #00             ;512 bytů
2454        CALL #248E, FILLCONST   ;ulož 256 do dat
2457        CALL #248E, FILLCONST   ;ulož 256 do dat
245A        LD   A, #F7             ;vyvolá uložení CRC pro data
245C        LD   (HL), A            ;ulož do dat
245D        INC  HL                 ;a posuň se na další
245E        LD   A, #4E             ;mezera mezi sektory
2460        LD   B, #28             ;40 bytů
2462        CALL #248E, FILLCONST   ;ulož do dat
2465        LD   A, E               ;dej číslo sektoru do A
```

```
2466                INC   E                    ;zvyš číslo sektoru
2467                CP    (IX+#03)             ;je větší, než počet sektorů na stopu?
246A                JR    C, #246E, MAKENEXT   ;ne → skoč
246C                LD    E, #01               ;nastav číslo sektoru na 1
246E MAKENEXT       INC   D                    ;zvyš čítač sektorů
246F                LD    A, D                 ;dej čítač do A
2470                CP    (IX+#03)             ;už byly vytvořeny ID pole pro všechny sektory?
2473                JR    NZ, #240A, MKFDATA   ;ne → připrav ID pro další sektor
2475                LD    A, #4E               ;do konce stopy vyplníme hodnotou
2477                LD    B, #00               ;celkem 512 bytů
2479                CALL  #248E, FILLCONST     ;ulož 256 bytů
247C                CALL  #248E, FILLCONST     ;ulož 256 bytů
247F                POP   BC                   ;obnov číslo stopy a sektoru
2480                LD    HL, #4000            ;do HL adresa začátku dat pro formátování
2483                LD    IX, #25ED, WRINMI    ;do IX adresa programu pro zápis dat
2487                LD    A, #F0               ;do A příkaz WRITE TRACK
2489                LD    B, #03               ;budou se testovat výsledné bity příkazu WRITE
                                                 ;SECTOR – pokud si říkáte, ze spíše WRITE TRACK,
                                                 ;tak je to jedno, protože mají vcelku stejné příznaky
                                                 ;(WRITE TRACK má o dva míň, ale ty jsou vždy
                                                 ;nastaveny na nulu)
248B                JP    #2377, DOWDCOM       ;skoč na vykonání příkazu
```

FILLCONST

Rutina pro naplněni úseku paměti od HL dlouhého B bytů konstantou v A.

IN: HL adresa, odkud se bude vyplňovat
B počet bytů
A hodnota, kterou se bude vyplňovat
OUT: B bytů je od HL vyplněno hodnotou v A

```
248E FILLCONST      LD    (HL), A              ;ulož A na adresu v HL
248F                INC   HL                   ;posuň se na další byte
2490                DJNZ  #248E, FILLCONST     ;opakuj B-krát
2492                RET                        ;vrať se
```

FINDTRACK

Najede hlavou na stopu a nastaví do registru řadiče číslo sektoru.

IN: B číslo stopy
C číslo sektoru
OUT: BC výsledek operace
C stopa nalezena
NC stopa nenalezena

```
2493 FINDTRACK      PUSH  HL                   ;ulož si registry
2494                PUSH  DE
2495                PUSH  BC
2496                LD    D, #1C               ;příkaz SEEK s testem čísla stopy a s přiloženou hlavou
2498 FINDTRACK1     LD    A, (#3E6B) WORKDR    ;do A dej drive, se kterým se pracuje
249B                CALL  #254B, DRVSEL        ;spusť mechaniku a nastav hlavu na stopu podle (IX+#04)
249E                BIT   0, (IX+#00)          ;otestuj, jestli je drive připojen
24A2                LD    BC, #0401            ;do B chyba při FINDTRACK a do C chyba „Device
                                                 ;unavailable“
```

```
24A5            JR    Z, #24B5, FINDTRRET ;není → skoč na návrat s chybou
24A7            LD    HL, #0000           ;do HL dej nulu
24AA            LD    A, (#3E6B) WORKDR   ;do A drive, se kterým se pracuje
24AD            CALL  #256D, TESTDR       ;otestuj připravenost drivu
24B0            JR    NZ, #24B9, FINDTRRD ;je připraven? ano → skoč
24B2            LD    BC, #0180           ;do B příznak pro základní test a do C chyba „Drive is
                                          ;not ready“
```

```
Tady chybí      LD    (#3EE7), IX.
24B5 FINDTRRET  POP   HL                  ;vyzvedni hodnoty ze zásobníku
24B6            POP   HL
24B7            POP   HL
24B8            RET                       ;vrať se
```

Nastavíme hlavu na stopu.

```
24B9 FINDTRRD   POP   BC                  ;obnov číslo stopy a sektoru
24BA            INC   C                   ;zvyš číslo sektoru (číslují se od 1)
24BB            LD    A, C                ;dej číslo sektoru do A
24BC            OUT   (#85), A            ;a pošli je registru řadiči
24BE            XOR   A                   ;nuluj A
24BF            BIT   4, (IX+#01)         ;je disketa jednostranná?
24C3            JR    Z, #24C9 , FINDTRSVS ;ano → skoč
24C5            RR    B                   ;vyděl B dvěma a vem poslední bit z B (liché sektory
24C7            RLA                       ;jsou na 2 straně, sudé na 1 straně) a dej ho do A
24C8            RLCA                      ;na 1. bit
24C9 FINDTRSVS  LD    (#3EEB), A SVSIDE   ;ulož si, se kterou stranou se bude pracovat (0=1. strana,
                                          ;2=2. strana)
```

Protože v 80-ti stopé mechanice je 80-ti stopá diskety (nebo ve 40-ti stopé mechanice je 40-ti stopá disketa), nastavíme zpět číslo stopy v registru na původní stav, potože nedojde k žádným komplikacím při vystavování hlavy na stopu (nemusí se dělat dvojnásobný posun hlavy), a také se provádí testování vyhledané stopy. Pokud by byla 40-ti stopá disketa v 80-ti stopé mechanice, musí se zvolit dvojnásobný posun hlavy (hlava se posune o dvojnásobnou vzdálenost, protože to je 40-ti stopá disketa a ne 80-ti stopá) a neprovádí se test vyhledané stopy.

```
24CC            BIT   5, (IX+#01)         ;testuj, jestli je 40-ti stopá disketa v 80-ti stopé mechanice
24D0            JR    Z, #24DB, NOD40IN80 ;ne → skoč
```

Musíme upravit obsah registru stopy a příkaz pro řadič.

```
24D2            LD    D, #18              ;do D příkaz SEEK s přiloženou hlavou bez testu stopy
24D4            SLA   B                   ;vrať do původního stavu číslo stopy, protože se bude
                                          ;provádět dvojnásobný posun hlavy
24D6            IN    A, (#83)            ;načti číslo stopy, na které se nachází hlava
24D8            ADD   A, A                ;vynásob dvěma, aby to bylo v původním stavu
24D9            OUT   (#83), A            ;a pošli číslo stopy zpět do registru řadiče
```

Nyní otestujeme, jestli při poslední operaci nedošlo k chybě nebo nebyly zastaveny mechaniky.

```
24DB NOD40IN80  BIT   7, (IX+#00)         ;otestuj, jestli byla při poslední operaci chyba nebo byly
                                          ;zastaveny mechaniky
24DF            JR    Z, #24E6, FNDTRNOC  ;ano → skoč
```

Pokud nebyla chyba nebo zastaveny mechaniky, porovnáme, jestli hlava není už na požadované stopě.

```
24E1            IN    A, (#83)            ;přečti číslo stopy z registru řadiče
24E3            CP    B                   ;jsou shodné se stopu, kam chci vystavit hlavu?
```

```
24E4                JR    Z, #2504, FINDTROK   ;ano → skoč
```

Musíme vystavit hlavu.

```
24E6 FNDTRNOC       LD    A, B                 ;do A číslo stopy, kam chci vystavit hlavu
24E7                LD    E, A                 ;ulož si do E číslo stopy
24E8                CALL  #2340, SEEK          ;nastav hlavu na stopu
24EB                AND   #98                  ;byla chyba při vystavování hlavy?
24ED                JR    Z, #2504, FINDTROK   ;ne → skoč
24EF                CALL  #234B, HOME          ;pošli hlavu na nultou stopu
24F2                LD    A, E                 ;do A obnov číslo stopy
24F3                CALL  #2340, SEEK          ;zkus znovu nastavit hlavu stopu
24F6                LD    C, A                 ;dej příznaky chyb do C
24F7                AND   #98                  ;byla chyba při vystavování hlavy?
24F9                JR    Z, #2504, FINDTROK   ;ne → skoč
24FB                RES   7, (IX+#00)          ;nastav příznak neporovnávat číslo stopy s registrem
                                               ;řadiče (při operaci došlo k chybě)
Tady chybí          LD    (#3EE7), IX.
24FF                POP   DE                   ;obnov registry
2500                POP   HL
2501                LD    B, #01               ;do B příznak pro základní testování chyb
2503                RET                        ;vrať se
2504 FINDTROK       BIT   5, (IX+#01)          ;otestuj, jestli není 40-ti stopá disketa v 80-ti stopé
                                               ;mechanice
2508                JR    Z, #250F, NOD40IN801 ;ne → skoč
```

Vydělíme obsah registru řadiče dvěma, aby při dalším vystavování hlavy nedošlo k špatnému vystavení.

```
250A                IN    A, (#83)             ;načti číslo stopy z registru řadiče
250C                RRCA                       ;vyděl číslo stopy dvěma
250D                OUT   (#83), A             ;a vyšli ho zpět do registru řadiče
250F NOD40IN801     DI                         ;zakaž přerušení
2510                LD    (#3EE7), IX DOSIX2   ;ulož si adresu parametrů disku do SRAM
2514                POP   DE                   ;obnov registry
2515                POP   HL
2516                SCF                        ;nastav C
2517                RET                        ;vrať se
```

FORFINDTR

Tento vstupní bod na nastavení hlavy na stopu v registru B se používá při formátování, protože se vůbec netestuje stopa (ani to nejde, protože u nové diskety žadné stopy ještě nejsou).

IN: B číslo stopy
OUT: BC výsledek operace
C stopa nalezena
NC stopa nenalezena

```
2518 FORFINDTR      PUSH  HL                   ;ulož si registry
2519                PUSH  DE
251A                PUSH  BC
251B                LD    D, #18               ;do D příkaz SEEK s přiloženou hlavou bez testu stopy
251D                JP    #2498, FINDTRACK1    ;pokračuj v hledání stopy
```

DISKRET

Návrat z operací READ, WRITE SECTOR a FORMAT TRACK.

```
2520 DISKRET    LD   IX, (#3EE7) DOSIX2   ;vyzvedni si adresu parametrů disku
2524            IN   A, (#83)             ;načti číslo stopy z registru řadiče
2526            LD   (IX+#04), A          ;ulož na (IX+#04) jako číslo stopy, kde se nachází hlava
2529            XOR  A                    ;nuluj počet průchodů při NMI
252A            LD   (#3EE2), A INTCNT
252D            EI                        ;povol přerušení
252E            LD   A, C                 ;do A status
252F            AND  A                    ;byla chyba?
2530            RET  Z                    ;ne → vrať se zpět
2531            RES  7, (IX+#00)          ;nastav příznak neporovnávat číslo stopy s registrem
2535            RET                       ;řadiče (při operaci došlo k chybě) a vrať se
```

DSKSTP

Vypne všechny mechaniky.

IN: –

OUT: zastaví všechny mechaniky

```
2536 DSKSTP     XOR  A                    ;do A nula pro zastavení mechanik
2537            CALL #25BC, OUTTODR       ;vypni mechaniky
253A            CALL #21AC, DRVCMP        ;zjisti adresu parametrů disku A
253D            RES  7, (IX+#00)          ;nastav příznak neporovnávat číslo stopy s registrem
                                          ;řadiče (mechaniky jsou zastaveny)
2541            LD   A, #01               ;nyní mechanika B:
2543            CALL #21AC, DRVCMP        ;zjisti adresu parametrů disku B
2546            RES  7, (IX+#00)          ;nastav příznak neporovnávat číslo stopy s registrem
254A            RET                       ;řadičem (mechaniky jsou zastaveny) a vrať se
```

DRVSEL

Rozběhne mechaniku a nastaví stopu podle (IX+#04).

IN: A číslo mechaniky

OUT: roztočí mechaniku a nastaví registr řadiče na stopu, kde byla naposledy vystavena hlava
IX adresa parametrů drivu

```
254B DRVSEL     PUSH AF                   ;ulož registry
254C            PUSH BC
254D            PUSH HL
254E            CALL #21AC, DRVCMP        ;zjisti adresu systémových parametrů drivu v A
2551            BIT  2, (IX+#01)          ;test mechaniky (A–0, B–1)
2555            LD   A, #05               ;DS–0, MO–0 je mechanika A:
2557            JR   Z, #255A, DRVSELOUT  ;je to A? ano → skoč
2559            RLCA                      ;posuň pro mechaniku B: DS–1, MO–1
255A DRVSELOUT  LD   C, A                 ;uschovej si A do C
255B            LD   A, (#3EE9) SELSTA1   ;do A stav NMI a bity výběru mechaniky
255E            AND  #FC                  ;ponech bity
2560            OR   C                    ;přidej nový výběr mechaniky
2561            CALL #25BC, OUTTODR       ;zapni vybranou mechaniku
2564            LD   A, (IX+#04)          ;do A číslo stopy, na které se teď nachází hlava
2567            OUT  (#83), A             ;pošli číslo stopy do registru řadiče
2569            POP  HL                   ;obnov registry
256A            POP  BC
256B            POP  AF
```

```
256C                RET                         ;vrať se
```

TESTDR

Otestuje, jestli je mechaniky připravena.

IN: IX adresa parametrů drivu
OUT: Z není připravena
NZ je připravena

```
256D TESTDR         EI                          ;musíme povolit přerušení
256E                BIT  7, (IX+#00)            ;testuj, jestli byla při poslední operaci chyba nebo byly
2572                RET  NZ                     ;mechaniky vypnuté, ne → vrať se
2573                PUSH BC                     ;ulož registry
2574                PUSH HL
2575                CALL #254B, DRVSEL          ;rozběhni mechaniku a nastav stopu podle (IX+#04)
```

Tady se na chvíli zastavíme. Autoři vymysleli zajímavý způsob testování připravenosti drivu. Protože není zapojen signál READY, musí se připravenost testovat programově. Probíhá to asi tak, že se řadiči vyšle příkaz násilného přeušení operace. Potom se čeká na odezvu z řadiče. Doba čekání je 100 cyklů přerušení (tedy 2 sekundy). Pokud do té doby nepřijde odezva z řadiče a čítač průchodů přerušením je 0, neukončí se přerušení normálně, ale je „násilně“ ukončeno a řízení se nevrací do čekací smycky, ale zpět do programu s nastavenými příznaky.

```
2578                LD   HL, #25B6 INVALRET     ;do HL adresu, kam se bude skákat, pokud dojde
                                                ;k přetečení přerušení při testu drivu
257B                LD   (#3EE3), HL TERADR2    ;ulož ji do SRAM
257E                LD   A, #64                 ;počet přerušení je 100 (2 sekundy)
2580                LD   (#3EE2), A INTCNT      ;ulož ji do SRAM
2583                CALL #2599, TESTRDR         ;testuj připravenost drive
2586                SET  7, (IX+#00)            ;nastav příznak porovnávat číslo stopy s registrem
                                                ;řadiče (mechanika běží a je připravena)
258A                JR   NZ, #2596, TESTDRRET;pokud je drive připraven → skoč
258C                CALL #25BC, OUTTODR         ;zastav drive
258F                LD   (IX+#30), A            ;ulož do prvního znaku jména drivu nulu
2592                RES  7, (IX+#00)            ;nastav příznak neporovnávat číslo stopy s registrem
                                                ;řadiče (při poslední operaci s drivem došlo k chybě)
2596 TESTDRRET      POP  HL                     ;obnov registry
2597                POP  BC
2598                RET                         ;vrať se
```

Čekací smyčka při testování drivu.

```
2599 TESTRDR        LD   A, #D0                 ;do A příkaz přerušení činnosti řadiče
259B                OUT  (#81), A               ;vyšli příkaz řadiči
259D                LD   (#3EE5), SP HERRSP2    ;ulož si SP pro případnou obnovu při přetečení přerušení
25A1                LD   B, #02                 ;proběhnou celkem dvě smyčky čekáni na odezvy z řadiče
25A3 LOOPISDRQ      CALL #25C2, TESTDRQ         ;testuj DRQ
25A6                JR   NZ, #25A3, LOOPISDRQ;je → čekej na ukončení DRQ
25A8 LPNOTDRQ       CALL #25C2, TESTDRQ         ;testuj DRQ
25AB                JR   Z, #25A8, LPNOTDRQ     ;není → čekej na DRQ
25AD                DJNZ #25A3, LOOPISDRQ       ;opakuj B-krát
25AF                XOR  A                      ;do A 0
25B0                LD   (#3EE2), A INTCNT      ;nuluj počítadlo průchodů přerušením
25B3                DEC  A                      ;nastav signál drive je připraven (NZ)
25B4                CCF                         ;nastav NC
25B5                RET                         ;vrať se
```

Sem se skáče, pokud dojde k přetečení přerušení

```
25B6 INVALRET   LD   SP, (#3EE5) HERRSP2  ;obnov SP z SRAM
25BA            XOR  A                    ;nastav signál drive není připraven (Z)
25BB            RET                       ;vrať se
```

OUTTODR

Vyšle výběr mechaniky a podmínky pro NMI.

IN: A bity výběru mechaniky a NMI
OUT: je vybrána mechanika a podmínky NMI

```
25BC OUTTODR    OUT  (#89), A             ;vyber mechaniku a nastav podmínky NMI
25BE            LD   (#3EE9), A SELSTA1   ;ulož si aktuální stav
25C1            RET                       ;vrať se
```

TESTDRQ

Načte hodnotu z řadiče a ponechá pouze informaci o DRQ.

IN: –
OUT: Z není DRQ
NZ je DRQ

```
25C2 TESTDRQ    IN   A, (#81)             ;čti status
25C4            AND  #02                  ;vyber pouze DRQ
25C6            RET                       ;vrať se
```

DELAY

Spožďovací smyčka pro aktivaci BUSY. Slouží pro čekání mezi jednotlivými příkazy a testy.

```
25C7 DELAY      PUSH BC                   ;uschovej si BC
25C8            LD   B, #0A               ;čekáme 10 smyček (celkem 163 taktů)
25CA DELAYLOOP  DJNZ #25CA, DELAYLOOP     ;opakuj B-krát
25CC            POP  BC                   ;obnov BC
25CD            RET                       ;vrať se
```

Tady se provádí obsluha přerušení od ULA. Celkem se tady neděje nic vážného. Slouží k tomu, aby při povoleném přerušení nezamrzl počítač. Dále se využívá při testování připravenosti drivu. Pokud je obsah adresy NMICNT různý od hodnoty 0, je postupně snížen. Pokud je po snížení 0 (tzn. že došlo k přetečení přerušení), provede se návrat přes INVALRET. Tím se zamezí tomu, aby se při testování připravenosti drivu při čekání na odezvu z řadiče zablokoval systém.

```
25CE INTERRUPT  PUSH AF                   ;ulož si AF
25CF            LD   A, (#3EE2) INTCNT    ;vyzvedni počet průchodů při NMI
25D2            AND  A                    ;je nula? (neprovádí se testování připravenosti drivu)
25D3            JR   Z, #25E2, OKINTERR   ;ano → skoč na nenásilný návrat z přerušení
25D5            DEC  A                    ;sniž o 1
25D6            LD   (#3EE2), A INTCNT    ;a ulož zpět
25D9            JR   NZ, #25E2, OKINTERR  ;není teď nula → skoč na nenásilný návrat z přerušení
```

Nyní došlo k přetečení přerušení a tak se skáče na adresu, která je uložena na #3EE3 (TERADR2).

```
25DB            LD   HL, (#3EE3) TERADR2  ;adresa rutiny pro násilné přerušení testování drivu
25DE            LD   A, H                 ;je tam #0000?
25DF            OR   L
25E0            JR   NZ, #25E6, GOINTERR  ;ne → proveď rutinu
25E2 OKINTERR   POP  AF                   ;obnov AF
```

Provedeme návrat z přerušení.

```
25E3 ENDINTERR  EI                      ;povol přerušení
25E4            RETI                    ;vrať se z přerušení
```

Provedeme násilný návrat z přeušení. V HL je adresa, kam se bude skákat.

```
25E6 GOINTERR   POP   AF                ;obnov AF
25E7            EX    (SP), HL          ;adresa rutiny na zasobník a návratová do HL
25E8            JR    #25E3, ENDINTERR  ;proveď rutinu násilného ukončení přerušení
```

REANMI

Přenos dat z řadiče. Sem se skáče 512× při čtení dat z diskety.

```
25EA REANMI     INI                     ;načtení dat z řadiče
25EC            RET                     ;vrať se
```

WRINMI

Přenos dat do řadiče. Sem se skáče 512× při zápisu dat na disketu.

```
25ED WRINMI     OUTI                    ;vyslání dat do řadiče
25EF            RET                     ;vrať se
```

Kdo se nezbaví PeCe včas,

ZEMŘE!

©1997 Dron of K3L Team

Už i klony ZX Spectra lepší jsou než Opel Vectra

©1997 someone on IRC

8. Systémové proměnné MDOSu a popis SRAM.

Adr.	Dél.	Jméno	Popis
3800	512	DIRBUF	– prostor pro práci s adresářem
3A00	512	AUXBUF	– prostor pro nahrání posledního sektoru souboru a pro práci s BOOTem
3C00	512	FATBUF	– prostor pro práci s FATkou a pro tvorbu BOOTu při formátování
3E00	48	DRPARZN	– prostor pro uložení parametrů mechanik (12 bytů pro 1 mechaniku)
3E30	48	DRNAMES	– prostor pro uložení jmen připojených drivů (10 bytů jméno + 2 byty náhodné byty)
3E60	1	DEBUG	– pokud je zde uložena jiná hodnota než 0, vypisuje MDOS ladící tisky
3E61	1	SNPCOUNT	– počítadlo snapů
3E62	1	AIFASK	– pokud je zde nenulová hodnota, tak se MDOS při přepisování starého souboru nebude ptát na přepsání starého
3E63	1	MODJP1	– zde se ukládá kód instrukce JP při operacích čtení/zápis sektoru a formátování stopy
3E64	2	MODJPA2	– zde se ukládá adresa rutiny
3E66	2	SAVEDE	– místo pro dočasné uložení registru DE při volání rutiny v ZX ROM
3E68	1	VARIA1	– systém ji přepisuje, ale nijak je nevyužívá
3E69	1	VARIA2	– systém ji přepisuje, ale nijak je nevyužívá
3E6A	1	VARIA3	– systém ji přepisuje, ale nijak je nevyužívá
3E6B	1	WORKDR	– číslo drivu, se kterým se pracuje
3E6C	1	CHNGFLAG	– nula znamená, že sektor FAT ve FATBUF není třeba zapisovat
3E6D	1	FATSC	– číslo načteného sektoru FAT ve FATBUF (0 – žádný)
3E6E	1	FATDR	– číslo drivu, ze kterého byla naposledy čtena FAT
3E6F	2	ADRSCTR	– uložení čísla a stopy, odkud byl naposledy čten adresář do DIRBUF
3E71	1	ADRDR	– číslo drivu, odkud byl naposledy čten adresář do DIRBUF
3E72	2	SVADRA	– adresa nalezené položky adresáře v DIRBUF
3E74	2	STARTADR	– místo pro adresu začátku bloku dat při LOAD, SAVE bloku dat
3E76	2	LENDAT	– místo pro uložení délky bloku při LOAD, SAVE
3E78	2	VALSYX	– pomocná proměnná MDOSu
3E7A	2	VALSYY	– pomocná proměnná MDOSu
3E7C	1	HEAD20	– obsah této proměnné se ukládá do hlavičky souboru na 20. byte
3E7D	1	NONE1	– nevyužito
3E7E	2	SVFRSC	– při LOAD souboru se zde ukládá číslo posledního sektoru bloku sektorů a po ukončení operace je zde číslo posledního sektoru ve stezce souboru
3E80	10	DNZONE1	– uložení 1. jména disku pro I/O
3E8A	10	FNZONE1	– uložení 1. jména souboru, se kterým se pracuje
3E92	–	SNONMB1	– uložení vyšší číslice snapu ve jménu (desítky)
3E93	–	SNONMB2	– uložení nižší číslice snapu ve jménu (jednotky)
3E94	1	EXTE1	– přípona souboru ve FNZONE1
3E95	10	DNZONE2	– uložení 2. jména disku pro I/O (např. při kopírování, přejmenování)
3E9F	10	FNZONE2	– uložení 2. jména souboru, se kterým se pracuje
3EA9	1	EXTE2	– přípona souboru ve FNZONE2
3EAA	10	ACDRIVE	– jméno aktuálního drivu (nastavený příkazem MOVE)
3EB4	32	SVHEAD	– prostor pro uložení hlavičky při kopírování
3EBF	–	SVINF	– začátek uložení informací v uchované hlavičce při kopírování
3EC5	–	SVFSC	– uložení prvního čísla sektoru v uchované hlavičce při kopírování
3ED4	3	SV24NM	– místo pro uložení 24-bitového čísla
3ED7	3	POM24NM	– pomocná proměnná pro převod 24-bitového čísla na ASCII vyjádření
3EDA	8	ASCIINM	– místo pro uložení ASCII kódové vyjádření 24-bitového čísla
3EE2	1	INTCNT	– počítadlo průchodů přerušení IM 1 při testování připravenosti drivu
3EE3	2	TERADR2	– adresa návratu při „násilném“ návratu z přerušení
3EE5	2	HERRSP2	– místo pro uložení registru SP při testování připravenosti drivu
3EE7	2	DOSIX2	– místo pro uschování registru IX při operacích čtení/zápis/formátování

3EE9 1 SELSTA1 – místo pro uložení bitů výběrů mechanik a stavu NMI
3EEA 1 NONE2 – nevyužito
3EEB 1 SVSIDE – místo pro uložení strany, se kterou se pracuje
3EEC 2 IREG2 – místo pro uložení I registru a stavu přerušení při přestránkování ROMek
3EEE 1 SNAPINF – pokud je zde nenulová hodnota, provádí se snap
3EEF 8 SYSMRK – místo pro kontrolní tabulku, která se vytvoří při inicializaci MDOSu, pokud je narušená, provádí se reset
3EF7 1 SYSFLAG – místo pro uložení důvodu přestránkování do ROM D40 (tisk chybového hlášení, návrat z rutiny ZX ROM)
3EF8 240 NONE3–240 – nevyužito
3FE8 22 SVREG – místo pro uložení registrů při snapu

Sled instrukcí PUSH a POP

```
PUSH AF
PUSH BC
PUSH DE
PUSH HL
EXX
EX   AF, AF'
PUSH AF
PUSH BC
PUSH DE
PUSH HL
PUSH IX
PUSH IY
LD   BC, (#3EEC) IREG2      ;do BC hodnota vektoru přerušení a stav přerušení
PUSH BC
```

… zde je program pro uložení snapu, který také při návratu vyzvedává hodnotu vektoru přerušení a stav přeušení, podle obsahu potom provede návrat zpět do programu

```
POP  IY
POP  IX
POP  HL
POP  DE
POP  BC
POP  AF
EX   AF, AF'
EXX
POP  HL
POP  DE
POP  BC
POP  AF
```

3FFE 2 SAVESP – místo pro uložení registru SP při snapu

9. Jak používat rutiny MDOSu

První otázka, která se programátorovi naskytne, je, jak používat rutiny MDOSu. Klasicky se ovládá D40 z BASICu. Jak na to ale ve strojovém kódu? Varianty jsou dvě. Buď budu používat pouze příkazy BASICu a to tak, že si někde v paměti nasimuluji BASIC řádek a skočím na provedení příkazu. Toto nás však omezuje pouze používání příkazů BASICu. Pokud však budeme chtít využívat pouze určité podprogramy a některé kroky budeme chtít vynechat, musíme použit variantu druhou a to přistránkování ROM D40 místo ZX ROM a volat dané podprogramy. Jak tedy stránkovat ZX ROM? Po prostudování struktury MDOSu zjistíte, že pokud se v proměnné SYSFLAG nachází hodnota #4F, provede se návrat z ROM D40 bez skoku na adresu #1700 (stránkování ZX ROM). Zůstane tedy přistránkována ROM D40.

Samotné řešení problému stránkování již bylo publikováno několik. První dvě varianty vyšly v ZX Magazínu. První využívala BASIC příkazu **POKE #247,79**, které způsobí, že po provedení příkazu a po zavolání RST #00 zůstane přistránkována ROM D40. Vyžaduje však nutnost zachování systémových proměnných Spectra. Druhá varianta využíva přerušení **IM 2**, kdy se počítá čas od jednoho přerušení k druhému. Poté se tento čas použije k čekání a těsně před přerušením se provede RST #08 a program se „odchytí" už v ROM D40. Ani tato varianta není nejlepší (využívá se přerušení). Další varianta vyšla o něco později, ale zato je geniálnější.

```
TAB     DW   #00
        DW   #3EF7
START   LD   A, #4F              ;do A a DE potřebné parametry
        LD   DE, TAB–#001A
        CALL #25AB               ;skoč na zápis znaku
        LD   HL, #00             ;nastav původní hodnoty v TAB
        LD   (TAB), HL
        LD   HL, #3EF7
        LD   (TAB+#02), HL
        RST  #00                 ;proveď přestránkování a vrať se s ROM D40
        RET
```

Jak tato krátká rutina funguje? Základ tvoří příkaz CALL #25AB, kde v ZX ROM je instrukce RST #00. ROM D40 podle návratové adresy zjistí, že se volá rutiny pro zápis znaku do kanálu (otevřeného sekvenčního souboru). Registr DE ukazuje na začátek hlavičky buferu +1 a v A je zapisovaný znak. Do registru HL se vypočte umístění parametrů v hlavičce (na 28. a 29. bytu hlavičky je uložen počet zapsaných znaků, na 30. a 31. bytu je místo uložení znaku do buferu), do DE se vyzvedne počet zapsaných znaků a do BC se vyzvedne adresa, kam se bude znak v A ukládat. Dojde k uložení znaku v A (#4F – návrat z rutiny ZX ROM) na adresu v BC (#3EF7 – SYSFLAG), otestuje se, jestli není už není zaplněn celý bufer (v DE je 0, takže se nic neděje) a pokud ne, vrátí se zpět s přistránkovanou ZX ROM (je to jakási obdoba příkazu POKE #247,79). Nyní však je na SYSFLAG uložena hodnota #4F, která nám po zavolání RST #00 zaručí návrat s přistránkovanou ROM D40 (viz. výpis ROM D40). Je ale třeba dát pozor na to, že se data v TAB změní, takže je třeba je nastavit před každým stránkováním na správné hodnoty. Jinak uvedená rutina nepotřebuje ani systémové proménné BASICu, ani přerušení. Nemá taky žádný vedlejší vliv (kromě změny obsahu TAB). Jedinnou nevýhodou této rutiny je to, že nerozpozná, jestli je připojena disketová jednotka D40/80. Pokud není, provede reset (první dvě varianty lze upravit tak, že pokud není připojena D40, může vrátit chybové hlášení a aplikace může nabízet pouze operace s kazetovým magnetofonem).

10. A co na závěr?

No co jiného než hodnocení. Samotná disketová jednotka je celkem kvalitní výrobek, který vlastně do té doby u nás chyběl nebo byl pouze pro ty schopnější, kteří si ho dokázali vyrobit sami. Tak se vlastně stala disketová jednotka volně přístupná všem, byl na ní servis, tvořily se pro ni programy. Některé konstrukční věci byly sice řešeny narychlo: možnost připojení pouze 2 mechanik místo 3, nevyužití celé paměti EPROM – co by se asi vešlo do 4 KB *(třeba DevastAce – pozn. Tritol&Pvl)*, ale já jsem s ní celkově spokojen a zatím jsem s ní neměl velké problémy. Doporučuji ji všem, kteří se rozhodují nad koupí vhodné disketové jednotky ke svému Speccy nedo Didaktiku.

Mnoho úspěchů při tvorbě software

Vám přeje

ještě jednou

KVAKSOFT

Obsah

Podpůrné podprogramy

Další příkazy MDOSu

Další podpůrné podprogramy

Podprogramy pro práci s řadičem

Komentovaný výpis MDOSu verze 1.0 a jeho opravy

autor

sazba