





# ZPRAVODAJ ATARI klub Praha

příloha XI (1989)

## PŘÍRUČKA UŽIVATELE DISKETOVÉ JEDNOTKY

Vydává 487. 20 Svazarmu -  
ATARI KLUB v Praze 4.  
Séfredaktor a vedoucí redakční rady  
JUDr. Jan Hlaváček.  
Zástupce séfredaktora  
ing. Stanislav Borský.  
Technická redakce O. Strnadová,  
Z. Šebesta.

Adresa redakce:  
487. 20 Svazarmu - Atari klub Praha  
REDAKCE

poštovní příhrádka 51  
100 00 Praha 10

Redakční rada: M. Bayer, ing. J. Bis-  
kup, RNDr. J. Bok, CSc., ing. S. Borský,  
ing. V. Friedrich, ing. O. Hanuš, RNDr.  
L. Hejna, CSc., ing. J. Chábera, ing. P.  
Jandík, Z. Lazar, prom. fyz., CSc., P.  
Vacek.

Otisk povolen se souhlasem redakce  
při zachování autorských práv a s  
uvedením pramene. Rukopisy nevyzá-  
dané redakcí se nevracejí. Za pů-  
vodnost a věcnou správnost rutí  
autor.

Vychází šestkrát ročně. Neprodejně.  
Cleněm klubu distribuováno zdarma.

Neprvnídelné přílohy na objednávku  
jsou kompenzovány zvláštním klubo-  
vým příspěvkem  
Rozsah 229 stran.

Neprošlo jazykovou úpravou.

Do tisku předáno v IV/1988.  
Vydávání schváleno DV Svazarmu Praha  
4, DSK DNV Praha 4.  
Evidenční číslo UVEI 86 042.  
© ATARI KLUB Praha, 1989.

ing. Petr Jandík

**PŘÍRUČKA UŽIVATELE  
DISKETOVÉ JEDNOTKY**

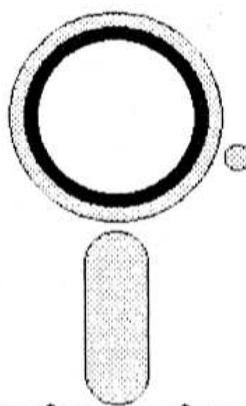
© ing. Petr Jandík, 1989

Recenze Petr Vacek

Revize rukopisu ing. Miroslav Jeríje,  
Dr. Jan Hlaváček, Luboš Bezděk

© Publication ATARI KLUB Praha, 1989

**Příručka  
uživatele  
disketové  
jednoty**



**Atari 1050 a XF551**

## Resumé:

V příručce je obsažen popis obsluhy disketové jednotky, popis DOS 2.5 pro začátečníky, principy práce diskového operačního systému a formáty záznamu, které se v praxi vyskytuji. Následují popisy ovládání diskových operačních systémů DOS 2.5, OS/A+, DOS XL, Happy DOS, Bibo DOS, MYDOS a SpartaDOS. U DOS 2.5 je popsána práce s disketovou jednotkou v Basicu. U každého systému je popsáno ovládání, funkce použitelné v Basicu (pokud se liší od DOS 2.5) a služební programy pokud nějaké má. Bibo DOS, MYDOS a Sparta DOS jsou navíc popsány z hlediska programátora pracujícího v assembleru. V závěrečných kapitolách je seznam chyb, hlášených jednotlivými systémy, vysvětlivky základních pojmu, slovníček důležitých terminů v němčině a angličtině a rejstřík.

## 1. Úvod

Zařazením disketové jednotky do počítačového systému se otvírá podstatně širší pole možnosti, než jaké má uživatel kazetového magnetofonu. Veškerá práce s počítačem se pomocí disketové jednotky podstatně zrychlí jak díky vyšší rychlosti, tak také vyšší spolehlivosti záznamu dat. Data se na disketu ukládají v podobě souborů, které lze označovat jménem, kopirovat, mazat a jiným způsobem s nimi manipulovat. K tomu, aby se počítač mohl s disketovou jednotkou "domluvit", komunikovat s ní, je třeba, aby se v paměti počítače nacházel program, který umožní manipulaci se soubory. Takový program se nazývá DOS, tedy disketový operační systém (Disk Operating System). DOS je sám také uložen na disketu v podobě jednoho nebo více souborů. Dříve, než můžeme začít pracovat s disketou, musíme zavést DOS do paměti počítače. Některé firemní programy (např. Synfile...) mají DOS již v sobě obsaženy, takže jej již nemusíme zavádět zvlášť. Ostatní programy, zejména v podobě zásuvných modulů, vyžadují zavedení DOSu. Pomoci DOSu lze připravovat k použití prázdné diskety, pořizovat archivní kopie důležitých souborů a mnoho dalších věcí.

Publikace kterou máte právě v rukou, obsahuje souhrn informací potřebných jak pro začátečníky, tak pro pokročilé. Obsahuje mnoho věcí, které možná nebudete zpočátku potřebovat, ale vrátíte se k nim později po zvládnutí základů. Část pro začátečníky popisuje základní funkce DOS 2.5 a poskytuje praktický návod, jak je používat. Uživateli se seznámí se základy práce s DOSem s ohledem na BASIC. Pro pokročilé je koncipována větší část publikace. Protože problematika disketové jednotky je velmi široká, není možné zahrnout do příručky podrobné vysvětlení všech dotčených oblastí. Proto jsou popisovány hlavně změny a doplňky, které se týkají používání disketové jednotky a předpokládá se, že si čtenář v případě potřeby doplní informace z doporučené literatury, ježíž seznam je na konci knihy. Největší důraz je kladen na diskový operační systém DOS 2.5, který je základním standardem a je dodáván výrobcem jako příslušenství jednotky. Dříve dodávaný systém DOS 3 má řadu nevýhod, není obecně používán a proto jej nepopisuji. Ovládnutí DOS 2.5 nám umožnuje porozumět i jiným systémům, kterých je celá řada a které jsou pro některé aplikace nezbytné. Nelze však popsat všechny existující DOSy. Zaměřím se jen na ty nejdůležitější. Jsou popsány ve vztahu k DOS 2.5 a podrobněji jsou probírány jen odlišnosti od něj.

### Struktura příručky:

Jak jsem již naznačil, je příručka koncipována jak pro začátečníka, tak pro pokročilého programátora. Pro začátečníky je určena kapitola, popisující disketové jednotky 1050 a XF551 a kapitola s "kuchařkou" pro základní operace se soubory v DOS 2.5. DOSy ostatní nejsou na začátečnické úrovni popisovány, protože jsou obvykle vhodné pro aplikace nedosažitelné začátečníkům, nebo jsou složitější a k jejich pochopení je třeba nejprve zvládnout základy DOS 2.5. Pro uživatele, kteří již tyto základy zvládli, jsou detailně

popisy všechny jeho funkce i s četnými přílohami. V příručce je dále zahrnut slovníček nejpoužívanějších termínů v angličtině, němčině i češtině se stručným vysvětlením jejich významu. Diskových operačních systémů je celá řada a není možné ani účelné je popisovat všechny. Systémy popsané v této knize byly voleny podle rozšířenosti a tak, aby představovaly určité typy, po jejichž zvládnutí by bylo možno pracovat se systémy ostatními. Při práci na knize bylo nutno překonat mnoho potíží, a to jak nedostatek českého odborného názvosloví (mnohé terminy neobsahuje například ani slovník výpočetní techniky SNTL, nebo jeho vysvětlení neodpovídají současné praxi, či problematice mikropočítačů), tak i nedostatek dokumentace a popisu systémů, ze kterých by bylo možno čerpat. Proto jsem musel vlastnosti a obsluhu systémů ověřovat a zkoumat přímo na počítači. Protože nemám k dispozici všechny existující konfigurace, potřebné pro ověřování různých funkcí, chtěl bych zde poděkovat pracovníkům pražského zastoupení firmy Atari za zapůjčení jednotky XF551. Všechny příklady a fakta uvedená v knize jsem prakticky vyzkoušel a ověřil na všech sestavách, které jsem měl k dispozici u sebe nebo u přátel.

V průběhu celé knihy se hovoří o disketách, floppy discích, discích a disketových jednotkách. Pokud jde o problematiku programovou, jsou výrazy disk, disketa, floppy disk a pružný disk míňena jako synonyma a označují disketovou jednotku s vloženou disketou jako celek. Tam, kde je třeba rozlišit fyzické zařízení od magnetického nosiče informace, se rozlijuje disketová jednotka a disketa. Disketová jednotka je také zkráceně nazývána jednotkou. Pevné disky (Winchestery, Hard disky) představují zvláštní problematiku. Ve spojení s osmibitovým Atari se v Československu patrně vůbec nevyskytuje a proto jim není věnována zvláštní pozornost. Pokud se hovoří o operačním systému, nebo je použita zkratka OS, je tím míněn operační systém v pevně naprogramované paměti ROM. Až na začáteční část jsem nevysvětloval podrobně všechny použité terminy, například významy systémových adres a podobně, protože byly již vícekrát popsány. Zde bych čtenáře odkázal na seznam doporučené literatury. Protože jsem nechtěl pokročilého čtenáře, kterých bude dříve čítat později většina, obtěžovat stále opakováním vysvětlováním pojmu, které by znepřehledňovalo tématiku, najde čtenář začátečník v vysvětlení ve zvláštním glosáři - slovníku na konci knihy. Doufám, že se tato příručka stane Vaším užitečným pomocníkem při práci s disketovou jednotkou.

## **2. Disketová jednotka a DOS pro začátečníky**

### **2.1. Zapojení a obsluha disketové jednotky**

Přinesli jste si domů disketovou jednotku. Pro její připojení k počítači je nezbytné mít:

- připojovací kabel sériové sběrnice
- napájecí zdroj
- systémovou disketu s DOSem

Přepravní polystyrénový obal uschovějte. Je třeba při přepravě disketové jednotky. Rovněž uschovějte kartonovou vložku, která je při transportu vložena do otvoru pro diskety a chrání citlivé čtecí a záznamové hlyvy proti otřesům. To je třeba zdůraznit zejména u jednotky XF551, která je citlivější.

#### **2.1.1. Připojení disketové jednotky**

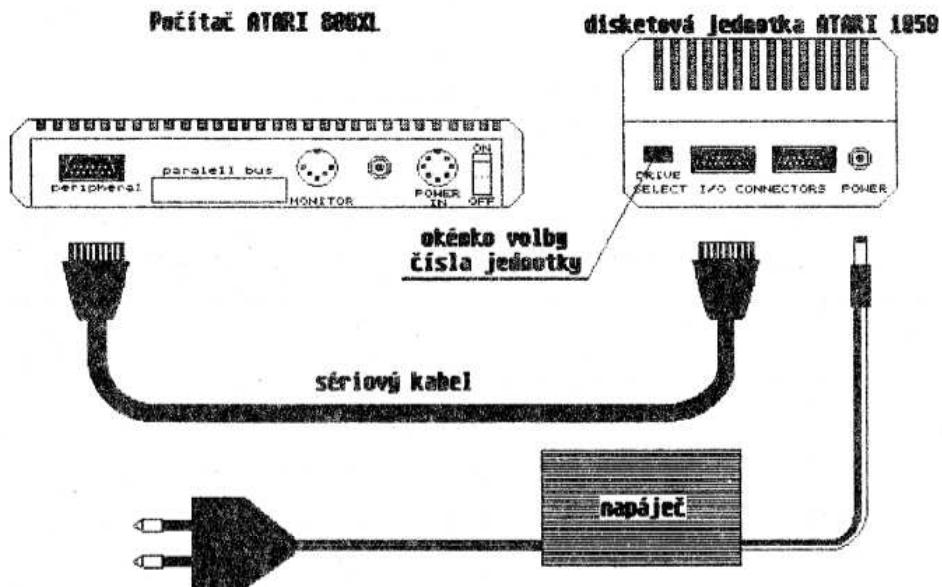
Při zapojování jednotky postupujeme takto:

1. Vypneme síťové vypínače všech přístrojů v sestavě počítače (kromě televizoru nebo monitoru).
2. Přesvědčíme se, zda je vypínač disketové jednotky v poloze OFF. Vypínač je u jednotky 1050 umístěn na předním panelu vlevo dole, u XF551 je na zadní straně přístroje.
3. Malý konektor sítového napáječe zasuneme do otvoru označeného jako "POWER IN" na zadní straně jednotky. Potom zapneme napáječ do sítové zásuvky.
4. Jeden konec kabelu sériové sběrnice zapojíme do zásuvky počítače, označené jako "PERIPHERAL", druhý konec zasuneme do jedné ze dvou zásuvek, označených "I/O CONNECTORS" na zadní straně jednotky.
5. Vyjměte z disketové jednotky ochranný karton a zapněte ji. Uslyšíte zavření a rozsvítí se příslušné kontrolky. U jednotky 1050 je to červená kontrolka "POWER", která svítí stále a je umístěna vedle vypínače. Další kontrolka "BUSY" je o něco výše a rozsvítí se asi na jednu vteřinu, po dobu po kterou se otírá motor jednotky. U jednotky XF551 je jen jedna kontrolka "BUSY" na předním panelu. Po zapnutí problikne jen velmi krátce, stejně tak velmi krátce zavří i motorek. U této jednotky se bez vizuální kontroly vypínače, který je na zadním panelu nepříliš přístupný, jen těžko poznává, zda je zapnuta. Vodítkem může být to, že při vypínání je zavření a rozsvícení kontrolky o něco málo delší. V každém případě při chybě nejprve zkонтrolujte, zda je jednotka zapnuta.

Nyní je systém připraven k vložení systémové diskety. Než budete pokračovat dále, doporučujeme přečíst si kapitolu 2.2.

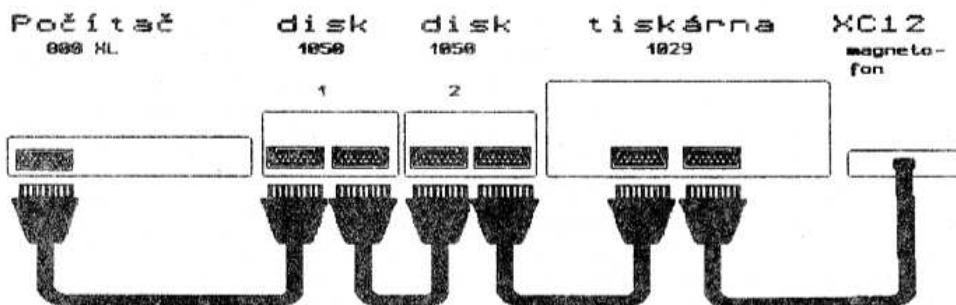
**Upozornění:** Disketová jednotka nesmí být umístěna těsně u televizoru. Televizor nebo monitor produkuje magnetické pole, které může narušit informace,

uložené na disketě. Naopak disketová jednotka produkuje vysokofrekvenční signál, který může rušit obraz televizoru.



Obr.1: Zapojení disketové jednotky

#### 2.1.2. Připojení více periferních zařízení



Obr.2: Propojení více periferních zařízení

K počítači Atari lze pomocí sériové sběrnice připojit více různých periferních zařízení. Kromě maximálně čtyř disketových jednotek lze připojit kazetový magnetofon, tiskárnu a u nás méně obvyklá zařízení jako sérioparalelní interface Atari 850, modem a podobně. Všechny tyto komponenty se zapojují za sebou do řetězu pomocí sériových propojovacích kabelů. Na pořadí přístrojů nezáleží, pouze magnetofony novějšího typu (XC12) nemají dvě zásuvky a musí se zapojovat na konec řetězu.

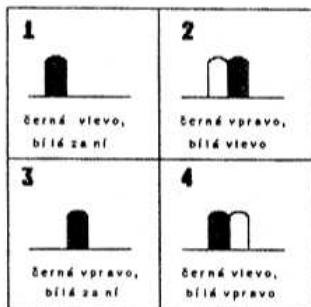
Na zadní straně každé disketové jednotky jsou dva konektory, stejně jako na zadní straně tiskárny 1029. Propojení se provede kabelem s prvním zařízením v řetězu. Druhé zařízení propojíme do druhé zásuvky "I/O CONNECTORS" prvního zařízení atd. podle obrázku.

### 2.1.3. Nastavení čísla jednotky

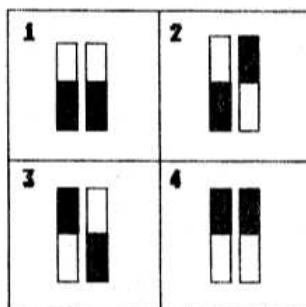
Chceme-li zapojit do řetězu více, než jednu disketovou jednotku, musíme pomocí identifikačních přepínačů nastavit její číslo. Při nastavování čísla vypneme napájení. Jednotku otočíme tak, abyhom viděli na zadní straně okénko s nápisem "DRIVE SELECT". U modelu 1050 je okénko nalevo, u XF551 napravo. U 1050 vidíme v okénku černou páčku, za ní je v zákrytu schována páčka bílá. U XF551 jsou miniaturní páčkové přepínače. Polohy páček a přepínačů pro nastavení jednotlivých čísel jsou na obrázku.

Je vhodné si každou jednotku označit, abyhom si je nepletli. V systému musí být vždy jedna jednotka nastavena na číslo 1, jinak nelze počítač nastartovat. Půjčíte-lisvou jednotku kamarádovi na provedení operací vyžadujících dvě jednotky, a on vám zapomene vrátit číslo na jedničku, nepodaří se Vám spustit DOS. Není důvod k panice, stačí opravit číslo jednotky na 1 a všechno zase funguje.

### **Nastavení čísla jednotky volbou přepínačů:**



**model 1050**



**model XF 551**

**Obr.3: Volba čísla disketové jednotky**

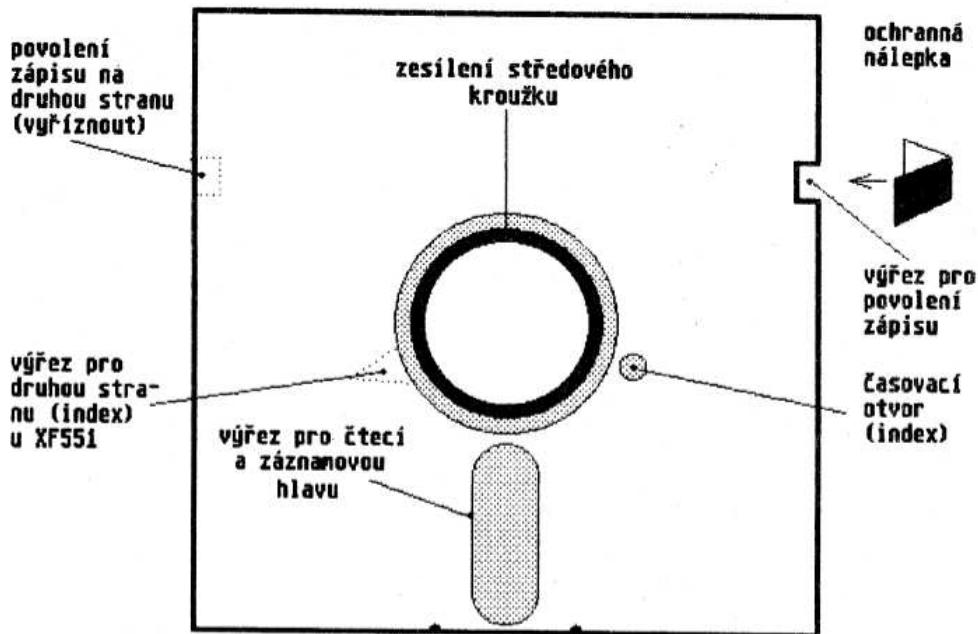
### 2.1.4. Pěče o diskety

Povrch diskety je pokryt citlivou magnetickou vrstvou, která uchovává data. Abyhom dosáhli dlouhou životnost uložených dat, je třeba s disketami správně zacházet. Disketa je uložena v černém pouzdře z umělé hmoty a i s ní je ukládána do papírové obálky. Kromě některých disket s firemními programy (např. i systémová disketa), mají všechny po straně obdélníkový výrez. Pokud je zlepíme neprůhlednou nálepkou (jsou dodávány s disketami), nelze na diskety zapisovat a data jsou chráněna proti náhodnému vymazání nebo přepsání. Další možnosti ochrany dat proti vymazání jsou uvedeny v následujících kapitolách. Pravidla pro správné zacházení s disketami jsou následující:

- Nezapínejte a nevypínejte disketovou jednotku, je-li v ní umístěna disketa. Nenechávejte disketu ve vypnuté jednotce.
- Disketu nemamáčejte a nemyjte. Je-li nutno disketu vyčistit od prachu, použijte měkký štětec, nebo prach vyfoukejte.
- Diskety neohýbejte. Disketa se musí v ochranném obalu volně otáčet. Vkládejte je a vydávejte opatrně.
- Diskety skladujte v papírových obálkách a ve svislé poloze.
- Nedávejte diskety k televizoru a k jiným elektrickým zařízením, včetně telefonu. Jejich magnetické pole by mohlo částečně poškodit uložená data.
- Nenechávejte diskety na přímém slunci. Chraňte je před vysokou teplotou.
- Protože se diskety v ochranném obalu otáčejí, může poškození obalu poškodit i povrch diskety.
- Nepište na diskety tužkou nebo propisovačkou. Ostrý hrot může poškodit jejich povrch. K popisování nálepky použijte fixy, nebo nálepku popište před nalepením.
- Disketové nálepky negumujte. Gumová drť je abrazivní a poškozuje magnetický povrch.
- Nedávejte na diskety kancelářské sponky.
- Nedotýkejte se magnetické vrstvy ve výrezech ochranného obalu. Otisky prstů jí mohou poškodit.

U disketové jednotky Atari 1050, která je jednostranná (t.j. má jen jednu čtecí/záznamovou hlavu a může číst a zapisovat jen na jednu stranu diskety), se vžilo používání obou stran diskety. Na druhé straně se vystříhne, nebo vyráží další otvor pro povolení zápisu. Jedna disketa tak slouží dvakrát. Tento postup disketám v podstatě přiliš nevadí, je však třeba dodržet určité zásady, aby se nesnížila životnost uložených dat. Chloupy vystříhané pouzdra jsou orientovány ve směru otáčení. Čistí a leští povrch diskety a usazuje se na nich nečistoty. Když se směr otáčení změní, vlákna se přesouvají na druhou stranu, přičemž se značná část nečistot dostane zpět na magnetický povrch. Tento proces se opakuje při každém otočení diskety.

Není to natolik vážné, aby se používání druhé strany diskety nedoporučilo, je však vhodné omezit otáčení disket na minimum. Proto je nejlépe používat opačných stran disket se kterými se často pracuje pro málo používaná data, jako archívy nebo sbírky her. U jednotky XF551 nemá otáčení disket smysl, protože tato jednotka je oboustranná. (K oboustrannému používání je však nutný jiný DOS, než DOS 2.5). Opačnou stranu diskety, nahranou na 1050 lze na XF551 přečíst, lze na ní i zapisovat nová data. Problémy jsou při formátování opačné strany diskety. Opačnou stranu je třeba buď formátovat na jednotce 1050, nebo vyfíznout v ochranné obálce další výřez pro indexový otvor, jak je naznačeno na obrázku 4. Výřez se samozřejmě vystříhne jen v ochranném obalu (z obou stran) tak, aby se při tom nepoškodil vlastní disk a jeho magnetická vrstva. Pozor na zmagetované nůžky!



Obr.4: Výřezy v obálce diskety a jich funkce.

## 2.2. Práce s DOS 2.5

### 2.2.1. Zavedení DOSu do paměti.

Zavedení DOSu pro použití s Atari Basicem se provede podle následujících pokynů (zavedení DOSu bez Atari Basicu je v dalším odstavci):

1. Vypněte počítač a disketovou jednotku. Ujistěte se, že v počítači není zasunut zásuvný modul (Cartridge) a v disketové jednotce není zasunuta ani disketa, ani ochranný karton.
2. Zapněte disketovou jednotku. Máte-li v systému více jednotek, použijte jednotku číslo 1. Jednotka završí a rozsvítí se kontrolka "BUSY". (Rozdíly mezi XF551 a 1050 viz 2.1.1 bod 5)
3. Když kontrolka zhasne, vložte do jednotky systémovou disketu DOS 2.5 nálepkou nahoru. Nastavte páčku do zavřené (svislé) polohy.
4. Zapněte počítač. Asi po vteřině se rozběhne motor a rozsvítí se kontrolka "BUSY". To je znakem, že se DOS nahrává do paměti. Jednotka občas vydává hřívivé zvuky. Pokud máte na televizoru nebo monitoru zapnuty zvuk, uslyšíte typický zvuk zavádění programu.
5. Až se na obrazovce objeví výzva **READY**, napište DOS a stiskněte RETURN.

### **2.2.2. DOS s Basicem a bez něj**

Atari Basic je do počítačů řady XL a XE vestavěn a aktivuje se automaticky. K aktivaci nedojde při zapnutí jen tehdy, když je zasunut programový modul, nebo když se při zapnutí počítače podrží klávesa OPTION. Přechod z Basicu do DOSu je velmi jednoduchý, jak jsme již viděli. Přechod z DOSu do Basicu je stejně jednoduchý a je popsán v odstavci "spouštění modulu z DOSu". Při přecházení z DOSu do Basicu a zpět je třeba si uvědomit, že jde o dva různé programy a pro jejich řízení je třeba různých povelů. Basic nerozumí povelu DOSu a DOS nerozumí povelu Basicu. Proto je nutno si vždy uvědomit, který program je momentálně "za ředitelským stolem". Basic poznáme podle výzvy READY, která se vypisuje po každé provedené instrukci a po každém stisknutí klávesy BREAK nebo RETURN. V DOSu se po stisknutí RETURN objevuje menu s hlavičkou.

### **2.2.3. Chyby při zavádění DOSu (BOOT ERROR)**

Načítání základního programu do počítače po jeho zapnutí se v angličtině nazývá BOOT, nebo BOOT PROCESS. Pokud nastane při tomto zavádění nějaká chyba, pokryje se obrazovka stále se opakujícím výpisem "BOOT ERROR", který chybu signalizuje. Příčinou chyby může být některá z následujících věcí:

- v jednotce není vložena disketa.
- na vložené disketě není nahraný DOS.
- disketa nebyla vložena správně.
- disketa je mechanicky poškozena. V tomto případě je nutno použít jinou disketu s nahraným DOSem.
- disketa nahraná ve dvojitě hustotě je vložena do jednotky 1050.

Kromě těchto nejobvyklejších příčin, které způsobují indikaci výpisem "BOOT ERROR", mohou nastat situace, kdy se DOS nezavede, ale neindikuje se při tom žádná chyba:

- počítač byl zapnut dříve, než disketová jednotka.
- disketová jednotka není správně připojena k počítači.
- síťový napáječ není připojen do zásuvky, nebo do jednotky, nebo jednotka není zapnuta.
- žádná připojená disketová jednotka nemá nastaveno číslo 1

V tomto případě se na obrazovce objeví buď SELFTEST, pokud jsme při zapnutí vyřadili BASIC, nebo se objeví výzva Basicu READY, ale každý pokus o komunikaci s disketovou jednotkou končí hlášením ERROR 130. Pokud se nepodařilo zavést DOS, a nenašla žádná z uvedených chyb, je pravděpodobně poškozen záznam na disketě. V tomto případě postupujte takto:

1. Vložte do jednotky systémovou disketu (originál, nebo pracovní kopii DOS 2.5) a zavedte DOS dříve popsaným způsobem.
2. Vyndejte disketu s DOSem a vložte nefungující disketu.
3. Všechny nepoškozené soubory z nefungující diskety překopírujte na jinou disketu (viz funkce "O", duplicate file).
4. funkci "D" zrušte všechny soubory a funkci "H" nahraje DOS.

Pokud disketa ani pak nefunguje, je třeba ji naformátovat (viz funkce "I") a bod 4 zopakovat.

#### **2.2.4. DOS menu**

Menu se na obrazovce objeví, pokud máme v počítači zavedený DOS. Používáme-li Basic, je třeba ještě použít povel DOS. Menu představuje seznam různých funkcí, které může DOS vykonávat. Pod seznamem je výzva "SELECT ITEM OR RETURN FOR MENU", která nás vyzývá k výběru funkce stisknutím klávesy s příslušným písmenem, po kterém je třeba ještě stisknout klávesu RETURN. DOS pak požaduje doplňující údaje, které potřebuje k provedení té které funkce. U každé funkce je tento dialog podrobně popsán.

```

DISK OPERATING SYSTEM TI VERSION 2.5
Copyright © 1983 ATARI CORP.

A. DISK DIRECTORY      I. FORMAT DISK
B. RUN CARTRIDGE       J. DUPLICATE DISK
C. COPY FILE           K. BINARY SAVE
D. DELETE FILE(S)     L. BINARY LOAD
E. RENAME FILE         M. RUN AT ADDRESS
F. LOCK FILE          N. CREATE MEM.SAV
G. UNLOCK FILE        O. DUPLICATE FILE
H. WRITE DOS FILES    P. FORMAT SINGLE

SELECT ITEM OR RETURN FOR MENU:
■

```

--- DOS menu ---

Na obrázku je menu, stručný popis jednotlivých funkcí je v následujícím odstavci. Funkce označené hvězdičkou mohou používat začátečníci a jsou v této kapitole popsány. Všechny funkce jsou detailně popsány v kapitole 4 pro pokročilé uživatele.

##### **A. DISK DIRECTORY - výpis obsahu diskety \***

Tato funkce umožňuje buď kompletní, nebo výběrový výpis jmen souborů nahraných na disketě. Udává kromě jmen souborů i informaci o jejich délce. Na konci výpisu je údaj, kolik volného místa na disketě ještě zbývá.

##### **B. RUN CARTRIDGE - přechod do modulu \***

(lze použít, jen když není vyfázen Atari Basic, nebo když je do otvoru pro programové moduly zasunut modul). Tato funkce umožní předat fízeň Basicu, nebo jinému zásuvnému modulu. Prakticky to znamená, že počítač od této chvíle přestává rozumět povelům DOSu a začne rozumět povělům modulu, například Basicu.

##### **C. COPY FILE - kopírování souboru**

Tato funkce se používá ke kopírování souboru dat na jinou disketu pomocí dvou disketových jednotek, nebo na tiskárnu či obrazovku. Tuto funkci lze také pořídit kopii souboru na tutéž disketu pod jiným jménem.

##### **D. DELETE FILE(S) - rušení souborů \***

Tuto funkci je možné rušit soubory a zvyšovat tak velikost volného prostoru.

**E. RENAME FILE - přejmenování souboru**

Tento funkci lze změnit jméno souboru.

**F. LOCK FILE - blokování (zamknutí) souboru**

Blokovaný soubor nelze měnit, přejmenovat, ani zrušit. Tuto funkci je vhodné chránit důležité soubory proti náhodnému přepsání nebo vymazání.

**G. UNLOCK FILE - uvolnění (odemknutí) souboru**

Opačná funkce, než F. Uvolňuje zablokovaný soubor, takže jej lze přepsat, přejmenovat a zrušit.

**H. WRITE DOS FILES - založení DOSu \***

Tato funkce vytvoří na vložené disketě nové soubory DOSu.

**I. FORMAT DISK - formátování diskety \***

Tento funkci si připravíme k použití prázdnou disketu. Formátování je nutné, než začneme na novou disketu zapisovat soubory. Formátováním se **vymaže** cokoliv, co bylo předtím na disketě eventuelně nahráno. Proto je nutno se ujistit, že na disketě nejsou žádné soubory, které bychom mohli ještě potřebovat, nebo že jde o zcela novou disketu. Formátováním můžeme opravit nefungující disketu, pokud není mechanicky poškozena.

**J. DUPLICATE DISK - kopírování diskety \***

Tato funkce slouží ke zhotovení přesné kopie diskety. Automaticky formátuje cílovou disketu a tedy i vymaže její původní obsah.

**K. BINARY SAVE - uchování obsahu části paměti na disketě.**

Používá se k uložení specifikovaného úseku paměti na disketu. Slouží k práci s programy ve strojovém kódu.

**L. BINARY LOAD - spuštění strojového programu**

Tento funkci lze program ve strojovém kódu zavést do paměti a spustit.

**M. RUN AT ADDRESS - skok na adresu**

Tento funkci lze předat fízení na hexadecimálně zadanou adresu v paměti. Slouží k práci s programy ve strojovém kódu.

**N. CREATE MEM.SAV - založení souboru MEM.SAV**

Tento funkci se na disketě vyhradí místo pro uložení obsahu paměti před zavedením souboru DUP.SYS. V některých případech je užitečné mít tento soubor založen. (u 130 XE a počítačů s dodatečně rozšířenou pamětí se zakládá automaticky na RAM-disku). Existence MEM.SAV jinde než v RAM-disku znamená zpomalení přechodu mezi DOSem a modulem. V těchto případech je možné MEM.SAV zrušit pomocí funkce "D".

**O. DUPLICATE FILE - duplikování souboru \***

Tato funkce umožňuje zkopirovat soubor pod stejným jménem na jinou disketu s použitím jedné disketové jednotky.

**P. FORMAT SINGLE - formátování v jednoduché hustotě (SD)\***

Tato funkce je téměř shodná s funkcí "I", ale formátuje disketu v jednoduché hustotě.

**2.2.5. Komunikace s DOSem**

Po napsání odpovědi na dotaz DOSu musíme stisknout RETURN, abychom mu předali svou odpověď. Pokud stiskneme RETURN samotný znamená to, že volíme standardní odpověď (viz standardy). Mnoho odpovědí je typu ano-ne. V tomto případě píšeme "Y", pokud chceme odpovědět "ano" a "N" pokud

chceme odpovědět "ne". Chybu při psaní lze opravit klávesou DELETE BACK-SPACE a napsáním správného znění. Celou odpověď lze vymazat současným stisknutím kláves SHIFT a DEL.B.-SPACE. (Podrobnosti k ovládání obrazovkového editoru jsou uvedeny například v příručce, která je dodávána s počítačem.)

### **2.2.6. A - prohlídka obsahu diskety**

Každá disketa, používaná k uchovávání souborů dat, programů apod., má tzv. adresář. Je to seznam souborů na disketě a informaci o nich. V adresáři je zapsáno jak se soubor jmenuje, kde je na disketě uložen (kde "bydlí"), jak je dlouhý a zda je povoleno jej měnit (viz funkce blokování a uvolnění souboru). Při popisu funkcí DOSu říkáme soubor každému celku, ať je již jeho význam jakýkoliv. Z pohledu DOSu je ihostejně, zda se jedná o data digitalizované fotografie, program, nebo soubor dat z databanky. Také sám DOS je na disketě uložen ve formě souborů. Informaci o tom, jaké soubory jsou na disketě uloženy nám podá funkce "A". Můžeme si ji vyzkoušet přímo na systémové disketě: stiskneme klávesu "A" a dvakrát RETURN. Na obrazovce se nám pod DOS menu objeví obsah systémové diskety. V prvním sloupci jsou názvy souborů, ve druhém typy souborů (extendery, koncovky). Třetí sloupec udává délku souboru v sektorech a poslední řádka udává velikost místa, které ještě na disketě zbyvá. Co je sektor, je vysvětleno v odstavci 2.2.9. o formátování disket. Soubory DOS.SYS a DUP.SYS vykonávají po zavedení do počítače standardní funkce DOSu. Vysvětlení o významu a používání souborů RAMDISK.COM, SETUP.COM, DISKFIX.COM a COPY32.COM je v kapitole 4 v odstavcích o používání RAM-disku a o služebních programech DOS 2.5.

A. DISK DIRECTORIES	I. FORMAT DISK
B. RUN CARTRIDGE	J. DUPLICATE DISK
C. COPY FILE	K. BINARY SAVE
D. DELETE FILE(S)	L. BINARY LOAD
E. RENAME FILE	M. RUN AT ADDRESS
F. LOCK FILE	N. CREATE MEM.SAV
G. UNLOCK FILE	O. DUPLICATE FILE
H. WRITE DOS FILES	P. FORMAT SINGLE

**SELECT ITEM OR RETURN FOR MENU :**

**A**

**DIRECTORY - SEARCH SPEC., LIST FILE ?**

DOS	SYS 037
DUP	SYS 047
RAMDISK	COM 089
SETUP	COM 078
COPY32	COM 056
DISKFIX	COM 057

**239 FREE SECTORS**

**SELECT ITEM OR RETURN FOR MENU :**

**A**

**Závěr: Vypis adresáře systémové diskety**

### **2.2.7. I : kopírování (duplikování) diskety**

Pomocí funkce DUPLICATE DISK lze pořídit přesnou kopii diskety tak, že se všechno z originální, zdrojové diskety zkopiuje na disketu cílovou. Přitom se cílová disketa automaticky naformátuje a smaže se veškerý její původní obsah. Pomocí této funkce si pořídte před veškerými dalšími pokusy pracovní kopii systémové diskety. Potom již můžete experimentovat bez obav. V případě smazání si ji můžete pořídit z uschovaného originálu znova. Při kopírování postupujte takto:

1. Máte na obrazovce DOS menu. Stiskněte "J" a RETURN.

Objeví se dotaz: DUP DISK-SOURCE,DEST DRIVES? , což znamená označení funkce kopírování diskety a dotaz na číslo zdrojové a cílové jednotky.

2. Většina uživatelů má jednotku jen jednu, proto odpoví:

1.1 a stiskne RETURN. Objeví se výzva INSERT SOURCE DISK, TYPE RETURN, neboli vlož zdrojový disk (originál) a stiskni RETURN.

3. Vložte do jednotky disketu, kterou chcete kopírovat (v tomto případě systémovou disketu DOS 2.5) a stiskněte RETURN. (Zejména majitelům 130 XE se může objevit žádost: TYPE "Y" IF OK TO USE PROGRAM AREA. CAUTION: A "Y" INVALIDATES MEM.SAV. Význam zprávy je popsán v kapitole 4 u popisu funkce "N". Nyní odpovězte "Y" a RETURN.) Jednotka začne načítat data, uložená na disketě. Potom se objeví výzva: INSERT DESTINATION DISK, TYPE RETURN, čili pokyn ke vložení cílové diskety. To je ta, na kterou chceme kopírovat.

4. Vyndejte z jednotky zdrojovou disketu (originál) a vložte disketu cílovou, (kopii) a stiskněte RETURN. DOS zapisuje na cílovou disketu data, která přečetl ze zdrojové diskety. Při prvním vložení cílovou disketu ještě před zapisováním naformátuje. Po zapsání všech dat se znova objeví výzva INSERT SOURCE DISK, TYPE RETURN, kterou již známe z bodu 2.

Kroky 3 a 4 se mohou opakovat vícekrát, záleží to na množství dat, zapsaných na disketu a na velikosti paměti, kterou má DOS k dispozici pro kopírování. Na konec kopírování se objeví základní výzva:

**SELECT ITEM OR RETURN FOR MENU :**

To znamená, že kopírování úspěšně skončilo a DOS očekává další povely. Vytvořenou disketu si zřetelně označte. Diskety s trvalejším obsahem je vhodnější si označit přímo na nálepce, než na obálce diskety. Obálky se při častější manipulaci snadno zamění. Při kopírování sledujte pečlivě výzvy ke vkládání zdrojové a cílové diskety a zkontrolujte si, zda je v jednotce opravdu ta disketa, o kterou DOS žádal, než "odpálíte" RETURN. Záměna disket je osudná a obvykle zničí obsah jak originálu, tak kopie. Proto je vhodné chránit originál proti přepsání ochrannou nálepkou. Blokování funkcí "F" v tomto případě nepomáhá.

Máte-li jednotky dvě, postupujte takto:

1. Viz bod 1 návodu pro jednu jednotku.
2. Napište 1,2 a RETURN. Objeví se výzva **INSERT BOTH DISKS, TYPE RETURN**, neboli vlož obě diskety, stiskni RETURN.
3. Zdrojovou disketu vložte do jednotky 1, prázdnou cílovou do jednotky 2 a stiskněte RETURN. (Zejména majitelům 130 XE se může objevit žádost: **TYPE "Y" IF OK TO USE PROGRAM AREA. CAUTION: A "Y" INVALIDATES MEM.SAV.** Význam zprávy je popsán v kapitole 4 u popisu funkce "N". Nyní odpovězte "Y" a RETURN.)
4. DOS kopíruje bez přerušování data z diskety v jednotce 1 na disketu v jednotce 2. Před kopírováním cílovou disketu naformátuje. Po skončení operace se objeví základní výzva:

#### **SELECT ITEM OR RETURN FOR MENU :**

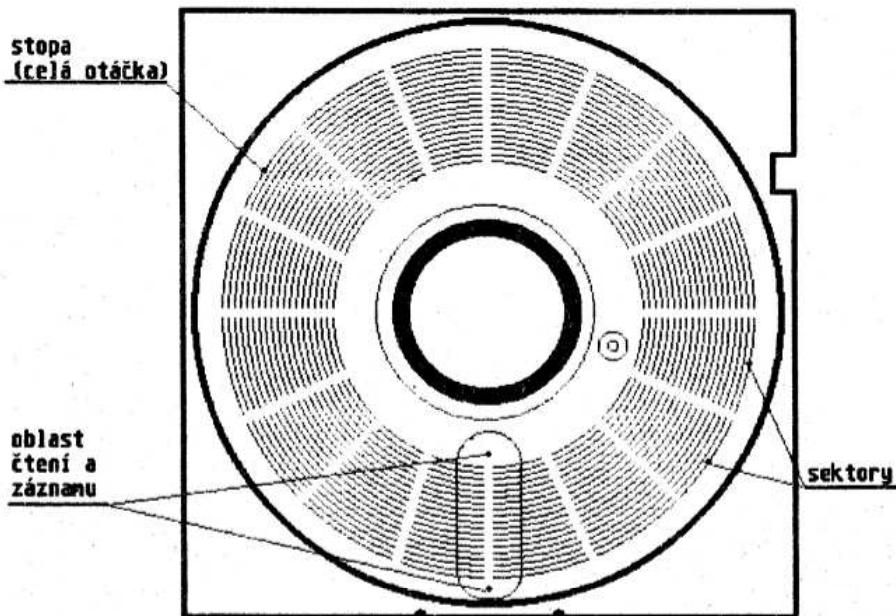
Kopírování diskety se někdy plete s kopírováním souborů funkcí "C". Rozdíl je v tom, že "J" kopíruje celou disketu bez ohledu na její obsah, kdežto "C" jen určené soubory.

#### **2.2.8. Používání ochranných nálepek**

Před kopírováním diskety, zvláště pokud obsahuje cenná a nepostradatelná data nebo programy, je vhodné chránit originální disketu proti přemazání zlepšením výřezu po straně diskety neprůsvitnou nálepou. Sada těchto nálepek je přibalena ke každé krabici disket a v nouzí lze použít i výstřížek z papírové samolepky. Přelepení výřezu má za následek, že čidlo v disketové jednotce zabrání jakémukoliv zápisu na disketu, nebo jejímu formátování. Tak je zajištěno, že eventuelní záměna disket při kopírování nemá osudné následky.

#### **2.2.9. I - Formátování diskety**

Aby bylo možno používat diskety zakoupené v obchodě, je nutno je pro ukládání dat připravit. Tento proces se jmenuje formátování, nebo také inicializace. Na prázdné disketě se při něm vytvoří jakési prázdné "kontejnery" na uskladnění dat. Tyto "kontejnery" se nazývají sektory a jsou uspořádány do řad (anglicky tracks). Do prázdných sektorů může počítac pomocí disketové jednotky zapisovat data a opětovně je vyhledávat a číst. Obrázek 7 ukazuje způsob ukládání dat. Povrch diskety je rozdělen na soustředné kružnice, zvané stopy (tracks), které se dále dělí na sektory. Sektor je nejmenší záznamovou jednotkou dat. Celkovou kapacitu diskety určuje hustota záznamu, zvolená při formátování. U DOS 2.5 můžeme pracovat v jednoduché, nebo rozšířené hustotě. Rozšířená hustota se často nepřesně označuje jako dvojitá hustota. Proto je třeba dát pozor, co je vlastně dvojitou hustotou méně. Ve spojitosti s jednotkou 1050, nebo s DOS 2.5 může jít jedině o hustotu rozšířenou. Skutečnou dvojitou hustotu, která je charakterizována délkou sektoru 256 byte lze použít jen u XF551 pomocí některých jiných diskových operačních systémů, než je DOS 2.5.



Obr. I: uspořádání záznamu na disketu

Menu DOS 2.5 nabízí oba možné druhy formátování. Funkce I formátuje disketu v rozšířené, funkce P v jednoduché hustotě. Průběh dialogu je při obou funkcích stejný:

1. Máte na obrazovce DOS menu. Stiskněte "I" a RETURN.  
Objeví se dotaz: WHICH DRIVE TO FORMAT?, neboli ve které jednotce se má disketa formátovat.
2. Napište číslo jednotky (obvykle 1) a stiskněte RETURN. Objeví se pokyn TYPE "Y" TO FORMAT DISK n, neboli písmenem "Y" (a klávesou RETURN) povolte formátování diskety v jednotce n, kde n je číslo, které jsme zadali v bodu 1. Tento bezpečnostní dotaz je poslední příležitost ke kontrole, zda je v zadané jednotce opravdu disketa, kterou jste chtěli naformátovat. Pamatujte, že formátování vymže všechny její předchozí obsah!!
3. Vložte do jednotky disketu, kterou chcete naformátovat a stiskněte RETURN. Kontrolka BUSY se rozsvítí a disketa se začne formátovat. Úspěšná operačce trvá cca půl minuty. Po jejím skončení se objeví základní výzva:

**SELECT ITEM OR RETURN FOR MENU :**

Nyní je disketa připravena k používání. Formátování je možno podle potřeby libovolně krát opakovat. Dělá se to v případech, kdy se záznam poškodí částečným vymazáním a podobně. Pokud se vyskytne při čtení chyba, například číslo 144, nebo 164 (viz seznam chyb), překopírujte si všechny "zdravé" soubory na jinou disketu. Chybnou disketu můžete po naformátování znova používat.

### 2.2.10. Pojmenovávání souborů

Abychom mohli pomocí DOSu pracovat se soubory dat, musí mít každý jednoznačné jméno, t.j. na jedné disketě se nemohou dva nebo více souborů jmenovat stejně. K úplnému určení souboru patří kromě jména ještě kód zařízení (např. diskové jednotky), na kterém je právě uložen. Kód zařízení a jméno souboru tvoří dohromady specifikaci souboru (filespec). Význam jejich jednotlivých složek je tento:

#### **Kód zařízení**

Kód zařízení je v našem příkladu "D1:". Reprezentuje jednotku, které se povel týká, a ve které je vložena disketa se souborem. "D" znamená disketovou jednotku, "1" je její číslo. Dvojtečka je povinná, označuje konec kódu zařízení. Soubor "D1:", uložený na disketě v disketové jednotce 1 by se označil "D1:D1". Kromě disketové jednotky jsou i další zařízení, jako je obrazovka (E:), tiskárna (P:) apod. Kazetový magnetofon má označení "C:". V menu DOS 2.5 s ním ale bohužel nelze pracovat. Uvedete-li jako parametr některé operace "C:", objeví se výpis OPTION NOT ALLOWED (volba není dovolena).

### **D1:PROGRAM1.BAS**

				kód zařízení
				číslo zařízení
				povinná dvojtečka
				jméno souboru
tečka je povinná jen při uvedení popisné části jména				tečka pro oddělení popisné části popisná část jména

**Obr.8: specifikace souboru**

#### **Standardy:**

Pro ulehčení práce v DOS 2.5 jsou připraveny standardy, které se použijí místo nezadané celé odpovědi, nebo její části. Příklad použití standardu je třeba při výpisu obsahu diskety funkcí "A", když po dotazu DIRECTORY - SEARCH SPEC,LIST FILE zadáme RETURN. Jako odpověď se dosadí standard "D1:.\*.,E:", to znamená výpis obsahu celé diskety na obrazovku. Pokud v odpovědi na dotaz v menu neuvedeme kód zařízení, předpokládá se "D1:". Pokud neuvedeme číslo jednotky, předpokládá se jednotka číslo 1. Zápis "1:" se doplní na "D1:".

#### **Jména souborů:**

Každý soubor musí mít jednoznačné jméno. Jméno souboru se skládá z vlastního jména a z nepovinné popisné části, tzv. extenderu nebo koncovky. Jméno se skládá z jednoho až osmi znaků. Pokud se používá koncovka, odděluje se tečkou. Znaky ve jméně mohou být jen písmena nebo číslice, jedinou výjimkou je tečka oddělující koncovku. Některé systémy nepovolují, aby jméno začínalo číslicí. Proto je vhodné tuto podmínu dodržovat i u DOS 2.5. Strukturu nejlépe ukáže příklad:

<u>dobře</u>	<u>špatně</u>
<b>PROGRAM.6J</b>	<b>PROG.6J.BAS</b>
<b>ACCT4324</b>	<b>ACCOUNT4321</b>
<b>DOPIS1</b>	<b># PROG21</b>

První jméno je špatně, protože tečka se smí ve jméně vyskytnout jen jednou. Druhé jméno je delší, než 8 znaků, ve třetím se vyskytuje speciální znak #. Koncovka slouží obvykle k rozlišení druhu souboru. Například programy psané v Atari Basicu se označují koncovkou "BAS" a podobně. Nemusíme však také používat koncovku žádnou.

#### Hvězdičková konvence - filtr pro výběr souborů

Pokud je třeba označit pro nějakou funkci místo jednoho souboru celou skupinu, například všechny programy s koncovkou "BAS", nemusíme vypisovat všechna jednotlivá jména. Můžeme použít filtr (wild card), který se řídí podle tzv. hvězdičkové konvence, mající původ u známého operačního systému UNIX. V této konvenci nahrazuje hvězdička libovolný počet libovolných znaků, otazník jeden libovolný znak. Nejlépe nám to vysvětlí praktická ukázka. Mějme na disketu soubory:

```
PROGRAM 1.BAS
PROGRAM 2.BAS
PROGRAM 3.BAS
PROGRAMX.BAS
PROGRAMY.LST
```

Filtr "PROGRAM?.\*", označí všechny uvedené soubory. Všechny mimo posledního nám vybere filtr "\*.BAS", nebo "PROG\*.BAS", nebo "PROGRAM?.BAS". Všechny soubory na kterékoliv disketě označíme filtrem "\*.\*".

#### 2.2.11. Přáce s Basicem

Pomocí Atari Basicu, nebo programovacího jazyku v zásuvném modulu můžeme psát vlastní programy. Abychom si je mohli ukládat na disketu, nebo je z ní načíst a používat jiné diskové povelů, musí být v paměti DOS. Kopírovat, přejmenovávat nebo mazat soubory lze v menu DOSu, do kterého přejdeme většinou povelom DOS. Po provedení potřebných operací se vrátíme zpět funkci RUN CARTRIDGE, která je v menu pod písmenem B. Dále popsané postupy předpokládají, že používáme vestavěný Atari Basic.

#### Z Basicu do DOSu a zpět

Je-li na obrazovce výzva READY, přejde se z Basicu do DOSu povelom DOS. Pro přechod z DOSu do Basicu (nebo jiného programu v modulu) se po výzvě:

**SELECT ITEM OR RETURN FOR MENU :**

napiše B a RETURN. Není-li Basic zapnut, objeví se zpráva NO CARTRIDGE. V tomto případě nezbývá, než počítač vypnout a zavést DOS s Basicem od začátku.

### **Ukládání a načítání programů v Basicu**

Atari Basic má vlastní instrukce pro načítání a ukládání programů. Zkuste si uložit a načíst zkušební program na neformátovanou pracovní disketu. Nejprve je třeba dosáhnout na obrazovce výzvy READY. Máme-li na obrazovce menu DOSu, napišeme B a RETURN. Nyní si můžeme napsat zkušební program:

```
10 PRINT "ZKUSEBNI PROGRAM"
20 END
```

Uložení programu na disketu se provede povelom Basicu:

```
SAVE"D:PROGRAM1.BAS"
```

Motor diskety se po napsání povelu rozběhne, program se nahrává. Nyní je program jak na disketě, tak v paměti. Vymažte jej povelom NEW. Když nyní vypíšeme obsah paměti povelom LIST, vypíše se jen READY. Program načteme do paměti zpět povelom:

```
LOAD"D:PROGRAM1.BAS"
```

#### **2.2.12. Kopírování souborů**

Funkcemi C a O můžeme kopírovat soubory z jedné diskety na jinou. Také můžeme vytvořit kopii souboru na téže disketě pod jiným jménem. POZOR, téměř funkci se nesmí kopírovat soubory DOS.SYS a DUP.SYS, pokud chcete na některou disketu přenést soubory DOSu, použijte funkci H.

#### **Založení cvičných souborů**

V předchozím odstavci jsme si na disketu uložili zkušební program PROGRAM1.BAS. Zaveděte DOS s Basicem, pokud jej již zaveden nemáte. Do disketové jednotky vložte datovou disketu s uloženým programem. Povelom LOAD program načtěte. Potom jej uložte povel:

```
SAVE"D:PROGRAM2.BAS"
SAVE"D:PROGRAM3.BAS"
SAVE"D:PROGRAM1.PPP"
```

#### **Kopírování souborů**

Při kopírování závisí postup na tom, zda máte k dispozici jednu, nebo více disketových jednotek.

#### **Z jedné diskety na jinou pomocí jedné disketové jednotky**

1. Máte na obrazovce DOS menu. Napište O a RETURN. Objeví se dotaz NAME OF FILE TO MOVE?, čili jméno kopírovaného souboru.
2. Napište jméno programu, například PROGRAM3.BAS a RETURN. (D: nemusíme na rozdíl od Basicu uvádět, DOS si kód zařízení doplní sám ze standardu). Objeví se výpis INSERT SOURCE DISK, TYPE RETURN, neboli vlož zdrojový disk (originál), stiskni RETURN.

3. Vložte do jednotky disketu, ze které chcete soubor kopírovat a stiskněte RETURN. (Na eventuelní žádost: TYPE "Y" IF OK TO USE PROGRAM AREA. CAUTION: A "Y" INVALIDATES MEM.SA Vodpovězte "Y" a RETURN. Význam zprávy je popsán v kapitole 4 u popisu funkce "N".) Jednotka začne načítat data, uložená na disketě. Potom se objeví výzva: INSERT DESTINATION DISK, TYPE RETURN, čili pokyn ke vložení cílové diskety. To je ta, na kterou chceme kopírovat.
4. Vyndejte z jednotky zdrojovou disketu (originál) a vložte disketu cílovou, (kopii) a stiskněte RETURN. DOS zapisuje na cílovou disketu program, který přečetl ze zdrojové diskety. Po zapsání dat se může znova objevit výzva INSERT SOURCE DISK, TYPE RETURN, kterou již známe z bodu 2.

Kroky 3 a 4 se mohou opakovat vícekrát, záleží to na velikosti souboru a velikosti paměti, kterou má DOS k dispozici. Na konec kopírování se objeví základní výzva:

#### **SELECT ITEM OR RETURN FOR MENU :**

To znamená, že kopírování úspěšně skončilo a DOS očekává další povely. Stejně jako u duplikování diskety je třeba pečlivě sledovat výzvy ke vkládání originálu a kopie. Záměna disket při kopírování může způsobit poškození jejich obsahu. Při kopírování je proto vhodné chránit obsah důležitých disket ochrannou nálepkou (viz odst. 2.1.4.)

#### Z jedné diskety na jinou pomocí dvou jednotek

1. Máte na obrazovce DOS menu. Napište C a RETURN. Objeví se dotaz COPY FROM-TO?, čili specifikaci odkud kam budeme kopírovat.
2. Napište úplnou specifikaci kopírovaného souboru.  
D1:PROGRAM3.BAS,D2:PROG.BAS
3. Zdrojovou disketu vložte do jednotky 1, cílovou do jednotky 2 a stiskněte RETURN. (Na eventuelní žádost: TYPE "Y" IF OK TO USE PROGRAM AREA. CAUTION: A "Y" INVALIDATES MEM.SA Vodpovězte "Y" a RETURN. Význam zprávy je popsán v kapitole 4 u popisu funkce "N".)

DOS kopíruje specifikovaný soubor z jednotky 1 na jednotku 2. Všimněte si, že se při tom mění i jméno souboru. Konec kopírování signalizuje základní výzva.

#### **Kopírování souboru na tutéž disketu**

Kopírování se provádí funkcí C v DOS menu, ať již máte jednu, nebo dvě jednotky. Postupujte stejně, jako při kopírování na jinou disketu při více jednotkách s tím rozdílem, že kódy zařízení budou stejné. V bodu 2 tedy odpovíte například:

D1:PROGRAM1.BAS,D1:PROGRAM.REZ

Program se zkopiřuje pod jiným jménem na tutéž disketu.

#### **Kopírování skupiny souborů pomocí filtru**

Když budeme kopírovat všechny cvičné soubory, můžeme je kopírovat jeden po druhém. To je ale zdlouhavé. Proto vyzkoušme rychlejší postup s použitím filtru. Například ve funkci O napišeme místo jména souboru filtr

"PROGRAM?.BAS". Postupně se překopírují první tři cvičné soubory, DOS pokaždé požaduje vkládání zdrojové a cílové diskety. O každém kopírováném souboru se vypíše zpráva.

**Upozornění:** Pokud se při kopírování souboru na cílové disketě již vyskytuje soubor stejného jména, bude nahrazen (přepsán) nově kopírovaným souborem. Důležité soubory lze chránit proti přepsání zablokováním funkcí F.

### **2.2.13. D- rušení souborů**

Někdy je potřeba zrušit nepotřebné soubory a uvolnit tak na disketě místo. Rušení souborů se v DOS 2.5 provádí funkcí D. Tuto akci je třeba pečlivě uvážit, protože se nedá tak snadno vztít zpět. Zrušený soubor můžeme za určitých okolností restaurovat programem DISKFIX.COM (viz 4.4.2.). Jméno zrušeného souboru se přestane objevovat ve výpisech adresáře. Funkci si můžeme vyzkoušet na souboru PROGRAM.REZ, který jsme vytvořili funkcí C:

1. V menu napište D a stiskněte RETURN. Objeví se výzva DELETE FILESPEC, která žádá o určení zrušeného souboru.
2. Napište D1:PROGRAM.REZ a stiskněte RETURN. DOS požádá o potvrzení výzvou TYPE "Y" TO DELETE... pod níž napiše jméno zrušeného souboru. To nám dává možnost zrušit funkci, pokud jsme soubor vybrali chybně, nebo zrušit jen některé soubory ze skupiny vybrané filtrem.
3. Napište "Y" a RETURN. Soubor se zruší.

Pokud vybíráme skupinu souborů filtrem, vypisuje se postupně název každého nalezeného souboru a rušení se potvrzuje pro každý soubor jednotlivě. Pokud v bodu 3 stiskneme jen RETURN, nebo N a RETURN, soubor se nezruší.

### **2.3. Typy disketových jednotek**

K počítačům Atari lze připojit disketové jednotky, vybavené konektory pro speciální sériovou sběrnici, kterou tyto osmibitové domácí počítače používají. Jde o jednotky vyráběné jak firmou Atari, tak i jinými výrobci. Kromě toho se vyrábějí různé doplňky k originálním jednotkám, které rozšiřují jejich možnosti.

Nejstarší jednotka firmy Atari má ozaření 810. Dodávala se k počítačům fady 400/800 a mohla používat jednostranně jednoduchou hustotu. S touto jednotkou se už těžko setkáme. Typický systém pro ni je DOS 2.0, který slouží stále jako základ pro kompatibilitu všech ostatních. Další jednotkou firmy Atari je zatím nejrozšířenější jednotka 1050. Může pracovat buď v modu 810, neboli v jednostranné jednoduché hustotě, nebo v jednosíranné rozšířené hustotě. V tomto novém modu se zvýšila kapacita disket z 88 na 127 kB. Pro jednotku 1050 byl původně vyvinut DOS 3, který však byl později pro nekompatibilitu s DOS 2.0 a ostatními systémy nahrazen DOS 2.5, který se stal pro jednotku 1050 základním diskovým operačním systémem. Jednotka 1050 je vybavena zařízením pro automatické rozpoznávání hustoty záznamu, proto se o přepínání mezi jednoduchou a rozšířenou hustotou nemusíme nikak starat.

Poslední v řadě disketových jednotek pro osmibitovou řadu Atari je jednotka XF551, která se objevila koncem roku 1987. Tato jednotka může kromě modů, kdy se chová naprostě shodně s modely 810 a 1050 pracovat v jedno- i oboustranné dvojitě hustotě, s maximální kapacitou 360 kB. Podle zpráv ze zahraničního tisku může také pracovat s téměř trojnásobnou rychlosťí přenosu dat mezi jednotkou a počítačem. V době, kdy je psána tato publikace, není ještě dokončen vývoj operačních systémů psaných speciálně pro XF551. Jediným nově vyvinutým systémem je zatím pouze BiboDOS, vytvořený v Compy Shopu. Ve vývoji je již od roku 1987 slibovaný ADOS firmy OSS a firma Atari údajně vyvíjí XE-DOS. V Československu je momentálně dostupno několik systémů, které mohou využívat oboustranné diskové jednotky. Je to BiboDOS, který umí využít kapacitu, ale ne rychlosť XF551 a kromě toho ovládá i různá rozšíření paměti, kompatibilně se 130XE. Neumí však zakládat podadresáře. Lépe lze kapacitu 360 kB využívat například pomocí systémů SpartaDOS verze 3.2 nebo MYDOS 4.3 B, použít lze i TOPDOS.

Jednotka XF551 je z důvodu dosažení příznivé ceny vybavena standardní mechanikou, která nemá na rozdíl od 1050 kontakt spojený s uzavírací pávkou. Také není vestavěn obvod pro automatické rozpoznávání jednoduché a rozšířené hustoty se sektorem 128 byte od dvojitě hustoty s délkou sektoru 256 byte při vložení diskety. Proto nemá XF551 možnost zjistit výměnu vložené diskety a tím ani automatické přepnutí formátu. Z toho vyplývají některé potíže při kopírování dat mezi disketami jednoduché nebo rozšířené hustoty a disketami v hustotě dvojitě. Hustota se musí nastavit čtením sektoru ze stopy 0 s výjimkou sektorů 1 až 3, které mají vždy délku 128 byte. Jednotka tedy čte sektor 4. Pokud se čtení nezdáří, přepne se na další ze čtyř možných formátů a pokus opakuje. Proto trvá velmi dlouho, než pozná, že není vložena disketa. Rovněž přepnutí z dvojitě do jednoduché hustoty trvá dlouho díky tomu, že se pro každou hustotu používá jiný způsob modulace.

Kromě jednotek firmy Atari lze pracovat i s jednotkami jiných výrobců, které byly v době uvedení na trh vždy o něco ve výhodě. Všechny tyto jednotky mohou pracovat ve dvojitě hustotě, některé z nich i oboustranně. Standardem je výrobek firmy PERCOM, jehož definici konfigurační tabulky převzali všichni ostatní výrobci konfigurovatelných jednotek, včetně firmy Atari u modelu XF551. Dalšími značkami jsou The One (Astra Systems), Indus GT, Rana, Trak a další. Tyto jednotky jsou v Evropě prakticky nedostupné.

Hardware doplňky k jednotce 1050 vyrábí několik menších firem. Všechny doplňky umožňují pracovat s dvojitou hustotou a ve spojení s příslušným software zvyšují rychlosť práce diskové jednotky. Rychlý modus se u každého doplňku ovládá jinak, a všechny se liší od XF551. To je důležité, pokud chceme používat doplněk s jiným než originálním operačním systémem. Některé z doplňků umožňují i kopírování firemních disket, chráněných proti kopírování.

### **US Doubler**

Vyrábí firma ICD, inc., 1220 Rock Street, Rockford, IL 61101, USA. Jde o destičku, která se zamontuje do jednotky 1050 místo dvou čipů. Umožňuje používání dvojitých hustoty a rychlenou komunikaci s disketovou jednotkou (UltraSpeed). Je určen pro používání se systémem SpartaDOS.

### **Speedy 1050**

Dodává Compy Shop, Gneisenaustr. 29, 4330 Mülheim/Ruhr, BRD, prodává se také u firmy AMV, Mariahilferstrasse 77 - 79, Wien, Österreich. Umožňuje práci ve dvojitě hustotě i rychlou komunikaci, ale řízení rychlého modu není kompatibilní s US Doublerem, ani s XF551. Má tři možné přenosové rychlosti. Lze jej používat s některým DOSem pro dvojitou hustotu. Jako originální diskový operační systém je pro Speedy určen jednak systém uložený v pamětech ROM Bibomonu, víceúčelového rozšíření hardware pro řadu 800XL, jednak BiboDOS verze 5.2.

### **Happy Enhancement**

Vyrábí Happy Computers Inc., PO Box 1268, Morgan Hill, CA 95037, USA. Lze jej také dostat v NSR v Compy Shopu, nebo v Rakousku u firmy AMV. Jde o soubor hardwarového doplňku pro jednotku 810 a 1050 s příslušným programovým vybavením. Kromě zrychleného přenosu data a dvojitě hustoty umožňuje pomocí zvláštních kopírovacích programů i kopírování chráněných firemních programů. Součástí dodávky je destička k zabudování do disketové jednotky, Warp Speed DOS a kopírovací programy.

### **1050 Duplicator**

Vyrábí Duplication Technologies, Inc., 99 Jericho Tpke, Suite 302 A, Jericho, NY 11750, USA. Jde o podobný doplněk, jako Happy Enhancement, ale nemá tak bohaté programové vybavení. Firma je poměrně nová a nelze spoléhat na zákaznickou podporu. Při některých aplikacích, jako je například kompilace assemblerovských programů s několika včleňovanými částmi, může "zbořit" celou disketu. V dodávce je destička k zabudování a dvě diskety s programy pro kopírování chráněných programů, DOS s vysokou rychlostí přenosu a různé další služební programy.

K většině uvedených doplňků lze říci, že výrazně zvýší užitné vlastnosti jednotky 1050, ale na druhé straně musíme počítat s problémy při některých aplikacích (spouštění proti kopírování chráněných her) a s chybami při výjimečných stavech. V tomto směru jsou negativní zkušenosti s Duplikátorem, problémy někdy dělá i Speedy a Happy. Při běžném používání se však potíže prakticky nevyskytují a zvýšení rychlosti práce bohatě vyváží možné ztráty způsobené sníženou kompatibilitou.

### **3. Základní principy DOSu**

Některé věci jsou společné pro všechny diskové operační systémy počítačů Atari. Řekneme si něco úvodem k podrobnému popisu DOS 2.5 a ostatních systémů.

#### **3.1. Práce DOSu a jeho umístění v paměti.**

Počítač si můžeme z hlediska funkce představit jako obrovský úřad naplněný inventářem, pracovními pomůckami, potrubní poštou, telefony atd. Tyto věci představují analogii hardware. Bez nich by chod úřadu možný, ale ony prostředky samotné nejsou schopny vykonat vůbec nic. K tomu je třeba ještě obsadit každou kancelář kvalifikovaným úředníkem. Tepřve v jeho rukou nabudou významu telefony, pořadače a potrubní pošta. A aby byla představa úplnější, různé skupiny úředníků hovoří různými jazyky a jsou specializovány na určitou problematiku. Proto potřebují ke vzájemné komunikaci ještě tlumočníky a jiné zprostředkovatele. Všichni jsou úzce specializovani a jsou ve svém obooru nesmírně výkonné. Úředníci se dělí na různé vrstvy a skupiny podle stupně styku s veřejností a podle oboru činnosti.

Nejnižší vrstva jsou specialisté na ovládání nástrojů a přístrojů. Jeden ovládá řkněme sešívání spisů sešíváčkou a ukládání do pořadačů. Druhý umí perfektně vyhledávat v telefonních seznamech a vytáčet čísla, zatímco třetí ovládá ukládání zásilek do potrubní pošty. O obsahu a významu spisů se kterými pracují, nevědí tito úředníci vůbec nic, ale ostatní by bez jejich služeb nemohli pracovat. Této skupině úředníků odpovídá v počítači nejnižší vrstva operačního systému, obslužné programy periferních zařízení, v odborné hantýrce nazývané handlery nebo drivers. Česky by se jim dalo říkat třeba obsluhovače. Tyto programy jsou schopny například rozpoznat, že jsme stiskli klávesu, jaká klávesa to byla, a poslat o tom zprávu kolegovi, obsluhujícímu obrazovku. Ten zařídí zobrazení znaku na obrazovce. Operační systém je v případě počítačů osmibitové fady Atari umístěn v trvale naprogramované paměti ROM. Naši úředníci - specialisté na nástroje a přístroje tedy v úřadě trvale bydlí a nemusí docházet nebo dojíždět. Jsou stále k dispozici.

Další kategorie úředníků neumí sice zacházet se základními prostředky, ale zprostředkovává dorozumění mezi nejnižší vrstvou a vrstvou nadřízenou. Řekne-li třeba jejich šéf, že mají dodat k přečtení spis XY, zjistí u správce seznamu dokumentů podle čísla spisu kde je umístěn, a vyžádají si u specialisty vyndání z archivu a zaslání potrubní poštou. Došly spis pak srovnají a donesou řéfovi. Této kategorii úředníků pak odpovídají zhřuba programovací jazyky. K ovládání prostředků používají služeb specializovaných programů operačního systému. Nemohou přímo hovořit s veřejností. Uživatel jiní musí naservírovat problém "po jejich", aby mu porozuměli a byli schopni jej řešit.

Nejvyšší vrstvu představují úředníci pro styk s veřejností, tedy prostředníci mezi uživatelem a programovacím jazykem. Této vrstvě například odpovídá program napsaný v Basicu, který slouží třeba jako databanka. Program zná význam dat, které vkládá uživatel a předává je nížším úřední-

kum ke zpracovani. Protože problematika "styku s veřejností" je velmi široká, je třeba velkého počtu takovýchto úředníků a musí se často měnit. Proto nemohou bydlet v úřadě, ale jsou to externí specialisté najímaní na určitý problém. U počítače jim odpovídají různé uživatelské programy, jako jsou textové a kalkulační programy, ale i hry apod., tedy vše s čím pracuje nespecializovaný uživatel.

Jak se do této hierarchie zařadí DOS? V operačním systému sídlí specialisti na zasílání dat na disketu a na jejich vybíráni. Musí však pro svou práci dostat přesný popis, kde data jsou a kolik jich je třeba vybrat, nebo uložit. Na manipulaci soubory to však nestačí. Protože počítač musí umět spolupracovat s různým počtem disketových jednotek různých typů, musí být povolán externí specialist - diskový operační systém. Ten umí požadavky na soubory, označené jménem převést do řeči obsluhovače diskety, tedy do čísel sektorů a vede si evidenci o tom, který sektor patří ke kterému souboru. Protože požadavek na manipulaci se soubory může přijít i od "veřejnosti" - uživatele, má DOS také svého úředníka pro styk s veřejností.

Povolávaný expert se jmenuje File Manager a u DOSu 2.5 a přibuzných je uskladněn v souboru DOS.SYS na disketu, odkud je příslušným způsobem povoláván do úřadu - počítače. Pokud jej zavedeme do paměti, zpracovává jak požadavky od uživatelských programů, tak od vlastního úředníka pro styk s veřejností. File Manager zůstává v paměti stále přítomen, je u všech DOSů trvale rezidentní částí. Přímý styk s uživatelem zprostředkovává druhá část, která je vlastně pro uživatele viditelným projevem existence DOSu. Tato část se aktivuje na základě požadavku na přechod do DOSu, tedy v programovacích jazycích obvykle povelom "DOS". Má dvě možné podoby. Buď se objevuje ve formě menu (DOS 2.0, 2.5, MYDOS, SMART DOS, TOPDOS a podobně), ve kterém si uživatel vybírá podle předloženého seznamu funkcí, nebo se realizuje jako tzv. příkazový procesor (Command processor). Uživatel předkládá své požadavky na práci v povelové řádce (OS/A+, Happy Dos). Objevují se také koncepty smíšené, kdy je možno pracovat i s menu, které vysvětlí naše záměry příkazovému procesoru, který je dále přetlumočí vhodnou formou organizátoru souborů. Organizátor (file manager) si vše vychází z záZNech vyhledá umístění souboru na disketu a další parametry, které předá specialistovi na komunikaci s disketou. Ten načte, nebo uloží příslušné sektory. Výsledky pak postupují po hierarchické stupnici zpět vzhůru. Část pro komunikaci s uživatelem bývá obvykle povolávána jen na vyžádání a je reprezentována souborem DUP.SYS. U DOSů povelových bývá rozdělena na dvě části, na příkazový procesor (command processor), se kterým komunikuje vyspělý uživatel, a z menu, které z méně vyspělého uživatele vytahne vhodné volenými dotazy co vlastně chce a přetlumočí jeho přání vykonavači povelů.

Napišeme-li například v Basicu povel LOAD "D:PROG.BAS", je postup vyzízení následující. Interpret Basicu si vyžádá na rutině operačního systému CIO načtení souboru a sdělí jí místo, kam má soubor uložit. Jedině on ví, že se bude jednat o program. Rutina CIO zjistí v tabulkách, že na disk má specialistu a předá požadavek organizátoru souborů. Organizátor si vyžádá od obsluhovače diskety obsah sektorů, na kterých je uložen adresář a podívá se do něj.

Když soubor nenajde, vrátí celou věc zpátky se záporným vyjádřením ERROR 170. Pokud soubor v adresáři najde, zjistí si kde začíná a požádá o první sektor souboru. Když jej dostane, uskladní si ho nejprve ve svém příručním skladu a po prohlédnutí ho uloží na začátek místa, které vyhradil Basic. Z přečteného sektoru zjistí, ve kterém dalším sektoru soubor pokračuje. Požádá o něj obsluhovač diskety a tak to jde až do konce souboru. Po předání posledního sektoru souboru do místa vyhrazeného Basicem pošle do CIO zprávu ERROR 136, která znamená nalezení konce souboru. CIO předá tuto zprávu Basicu. Basic ale ví, že konec souboru není z jeho hlediska chybou. Proto převeze soubor do své působnosti a zkontroluje zda se jedná o program v Atari Basicu ve správné tokenizované formě. Pokud je vše v pořádku, dá to uživateli najevu spokojeným READY. Pokud však nejsou data v pořádku, oznámí nám ERROR 21. To se například stane, když chceme povelem LOAD načíst program, uložený povelom LIST.

Jak už bylo řečeno není diskový operační systém v paměti počítače uložen trvale, to znamená že není uložen v paměti ROM, ale po zapnutí se vždy znovu zavádí do paměti RAM. Všechny DOSy se ukládají do oblasti od adresy \$700 hexadecimálně, neboli 1792 dekadicky. Od této adresy se ukládá rezidentní část systému. Velikost obsazené oblasti závisí na konkrétním typu DOSu a na jeho konfiguraci. Některé systémy jsou v paměti stále celé, jiné se dělí na část rezidentní a na část, která se zavádí jen při požadavku na přímou komunikaci s DOsem. První způsob má výhodu v rychlém přechodu mezi DOsem a aplikačním programem, druhý způsob nechází uživateli více prostoru v paměti, nebo poskytuje více funkcí při obsazení stejněho místa. U některých si dokonce lze mezi oběma variantami vybrat.

### 3.2. Studený start počítače

Studeným startem se nazývá základní inicializační pochod, který probíhá po zapnutí počítače. Kromě toho se pochod spustí po stisku klávesy RESET při Self Testu, nebo pokud byl závažným způsobem narušen obsah datové základny systému, nebo byl změněn stav zásuvného modulu (zasunut, vyndán, změněn), nebo pokud byl nastavením systémových proměnných WARMST a COLDST studený start vynucen programově. Tento pochod je velmi důležitý, neboť umožňuje zavádět do paměti základní programové vybavení mezi které patří i DOS. Podrobný popis dějů při studeném startu by přesáhl několik stránek. Omezíme jen na děje podstatné pro zavádění DOSu ve všech možných variantách. Zjednodušený průběh předpokládá práci s DOsem 2.5 a je znázorněn na obrázku 9.

Po zapnutí, nebo stisku klávesy RESET za podmínek pro studený start se nejprve provede inicializace operačního systému. Nastaví se základní obsah potřebných proměnných datové základny OS, vytvoří se tabulka obslužných programů periferii atd. Tyto akce nás jako uživatele nemusí zajímat. Potom se testuje, zda je stisknuta klávesa OPTION. Pokud ano, vypne se vestavěný modul BASICu, který má z programového hlediska stejnou povahu jako zásuvný modul. Dále se testuje, zda byla stisknuta klávesa START. Je-li tomu tak, provede OS pokus zavést do paměti z kazety program ve formátu BOOT

(viz 3.3.). Pokud nastane chyba, čtení je po výpisu BOOT ERROR přerušeno a počítač se nachází v ne definovaném stavu, ze kterého se lze dostat pouze opakováním studeného startu klávesou RESET. Zaváděné programy mohou být různé povahy. Může to být hra, obslužný program nějakého dodatečného připojovaného zařízení, zaváděč kazetového Turba a podobně. Pokud končí zavedený program instrukcí RTS a předá tak řízení zpět systému, pokračuje se po čárkováném spoji. Nyní se operační systém podívá, zda je aktivován modul. Může jím být vestavěný BASIC, nebo externí modul, zasunutý do příslušného otvoru. Není-li modul zjištěn, pokusí se systém zavést základní program z diskety číslo 1. Základním programem bývá obvykle DOS, ale může to být i hra nebo něco jiného. V našem schématu předpokládáme, že se jedná o DOS. Není-li v tomto okamžiku disketová jednotka č. 1 aktivní, t.j. zapnutá a schopná komunikace s počítačem, nic se nezavede a studený start končí Self Testem. Pokud se DOS úspěšně zavедl, dívá se, zda je na disketě soubor s názvem AUTORUN.SYS. Pokud ano, načte jej a odstartuje. Z toho vyplývá, že jménem AUTORUN.SYS lze pojmenovat jen strojový program ve formátu binárního segmentovaného souboru. Když takový program na disketě není, nebo když po vykonání své funkce předá řízení zpět DOSu, načte se soubor DUP.SYS, který realizuje DOS menu.

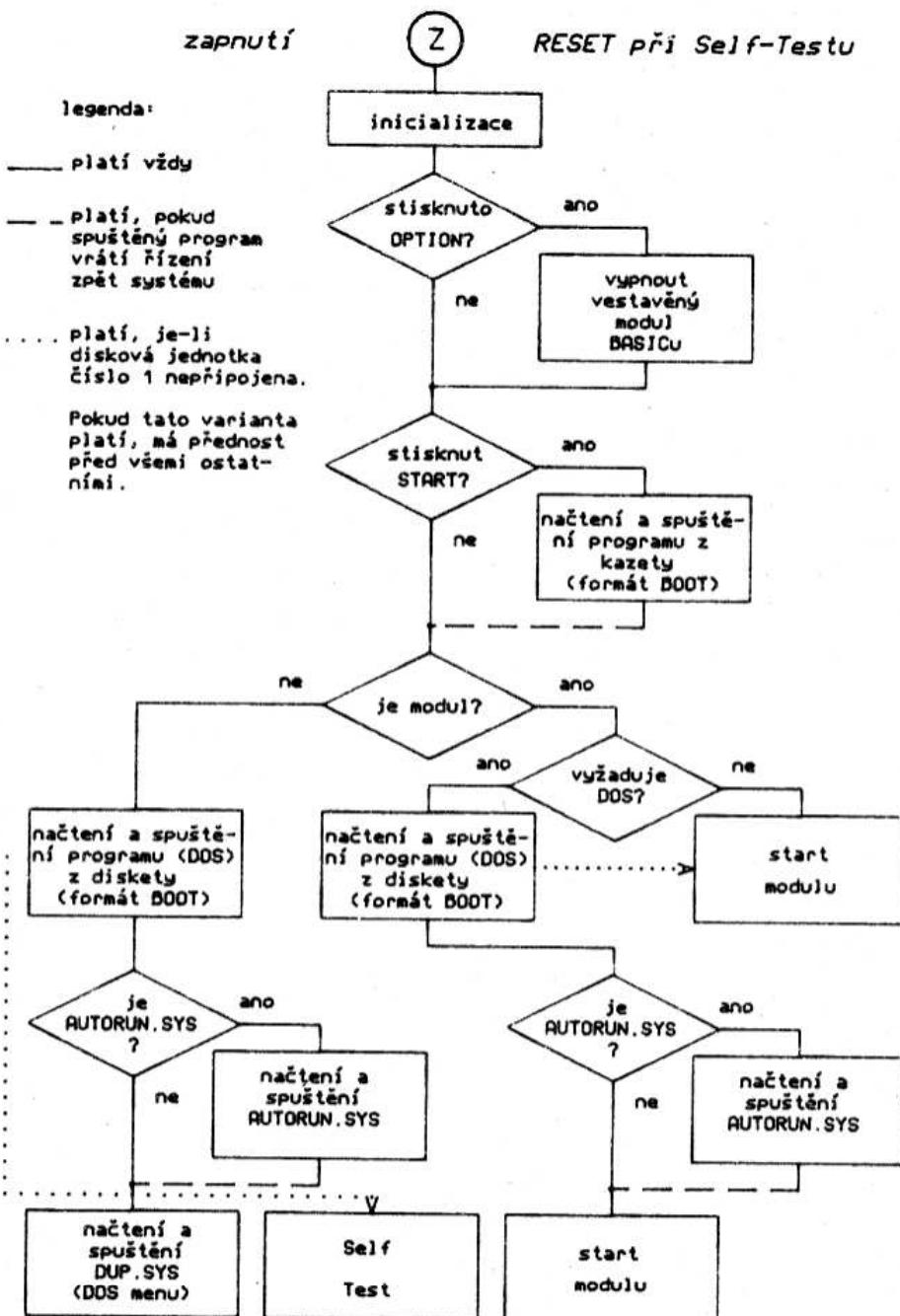
Pokud je modul zjištěn, pokračuje se podle hodnoty v CARTFG, která určuje mimo jiné, zda se k modulu bude zavádět nějaká disková podpora. Pokud ne (například u her), modul se nastartuje. Pokud ano, načte se z diskety DOS, (nebo to, co je na disketě ve formátu BOOT). DOS potom převezme řízení a zjistí, zda je na disketě přítomen soubor AUTORUN.SYS a pokud ano, načte jej a spustí. Pokud tento program vrátí systému řízení, nastartuje se modul. Není-li disketa č. 1 v okamžiku čtení aktivní, nastartuje se modul bez diskové podpory a každý pokus o čtení nebo zápis končí chybou ERROR 130.

### **3.3. Druhy a organizace diskových souborů**

Data se mohou na disketě ukládat různým způsobem. Nejjednodušší je organizace sekvenční, kdy se jednotky dat ukládají za sebou tak, jak přicházejí. Tato organizace dat je typická například pro kazetu a tiskárnu. Když potřebujeme nějakou informaci z prostředka sekvenčního souboru musíme jej projít celý od začátku abychom se k ní dostali. Zařízení, která umožňují pouze tento druh souboru se nazývají zařízení se sekvenčním přístupem.

Jiné možnosti skýtá zařízení s přímým přístupem, jehož typickým představitelem jsou diskové paměti jakéhokoliv typu. V kapitole o formátování jsme si řekli, že se plocha diskety dělí na stopy a tyto stopy na sektory. Nejjednodušší možnost organizace dat je stanovit, že soubor začíná v sektoru 1, pokračuje v sektoru 2 atd. I když je tento model souboru značně nevýhodný, protože nemůže efektivně využít kapacitu diskety, má své použití. Tento typ souboru se nazývá BOOT formát a díky jednoduchosti jeho čtení a zpracování se používá u programů zaváděných přímo operačním systémem.

Efektivní využití záznamové plochy diskety přináší formát tzv. linkovaného souboru, který je používán většinou diskových operačních systémů. Při tvorbě takových souborů se používá schopnost disketové jednotky číst a zapi-



obr. 9: Průběh studeného startu počítače

sovat na libovolném místě záznamového povrchu. Linkované soubory jsou nejobvyklejším druhem organizace souboru. Z hlediska uživatele jsou to vlastně rovněž soubory sekvenční, ale je možnost do nich přistupovat i přímo na kterékoliv místo, jak si ukážeme v kapitole o programových prostředcích BASICu.

Pro úplnost se zmíním o dalších dvou druzích organizace souboru, které se vyskytují u domácích mikropočítačů poměrně řidce. U souboru s přímou organizací je poloha jednotlivých jednotek dat - vět je určena hodnotou tzv. klíče. Klíč ve větě je určitá skupina bytů, která jednoznačně určuje větu, ve které je obsažena. V telefonním seznamu je například jméno a bydliště účastníka. Tento způsob umožňuje nejrychlejší možné vyhledávání vět, ale zachází nehospodárně s místem. (Příkladem takového souboru je slovník pro kontrolu textu u Atari Writeru). Proto se používá ještě kříženec sekvenčního a přímého souboru, soubor indexsekvenční. Věty jsou rozděleny do sekvenčně uspořádaných skupin. Hodnota klíče poslední věty každé skupiny je uložena ve zvláštním souboru klíčů. Při vyhledávání se projde nejprve sekvenčně soubor klíčů a po nalezení první skupiny jejíž klíč je větší, než klíč hledané věty, se začátek této věty vyhledá výpočtem jeho umístění. Poté se nalezená skupina projde sekvenčním způsobem.

### **3.3.1. Formát sekvenčního souboru (BOOT formát).**

Sekvenční soubor formátu BOOT je nutno použít pro programy, zaváděné přímo operačním systémem v ROM. Tvar souboru je následující:

byte:	-----	-----	(bez významu)
0	příznaky		
1	počet sektorů		
2	adresa, od které se		
3	bude zavádět do paměti		
4	inicializační		
5	adresa		
7	začátek vlastního programu		
	:		

Počet bytů v sektoru je 128 a data jsou uložena za sebou ve vzestupně číslovaných sektorech od sektoru 1 počínaje. Význam jednotlivých bytů:

- byte 0 - příznaky  
hodnota tohoto bytu se ukládá do DFLAGS (\$0240), ale jinak se nepoužívá.  
Má být nulová.

- byte 1 - počet sektorů

udává počet sektorů o délce 128 byte, které tvoří soubor typu BOOT a tím pádem také počet sektorů, které se budou číst. Toto číslo může nabývat hodnot 1 až 255 a 0 s tím, že 0 znamená 256.

- byte 2 a 3 - zaváděcí adresa

Tyto dva byty obsahují adresu (LO-HI), od které se začne ukládat do paměti načítaný soubor, počínaje bytem 0. (hlavička se tedy ukládá také)

- byte 4 a 5 - inicializační adresa

tyto dva byty obsahují adresu (LO-HI), na kterou se předá řízení (instrukci JSR) po zavedení souboru a kdykoli později při stisku klávesy RESET.

Zpracování takového souboru je celkem jednoduché a provádí ho po zapnutí počítače podprogram umístěný v ROM operačního systému. (Stejný podprogram jen s malou ohměnou zpracovává i soubory typu BOOT na kazetách. To je umožněno stejným formátem obou souborů.) Proces zavedení souboru typu BOOT má následující průběh:

1. Sektor 1 se načte do kazetového bufferu (\$0400)
2. Vybere se informace z prvních šesti bytů (0 až 5):  
Příznaky se uloží do DFLAGS (\$0240,1), počet sektorů k zavedení do DBSECT (\$0241,1), zaváděcí adresa do BOOTAD (\$0242,2) a inicializační adresa do DOSINI (\$0C,2).
3. Přečtený sektor se přenesekazetového bufferu na určenou zaváděcí adresu.
4. Zbývající sektory se načítají přímo na své místo v paměti.
5. Instrukci JSR na zaváděcí adresu + 6 předá OS řízení zavedenému programu. Ten může pokračovat v zavedení dalších částí, pokud se jedná o vícesegmentový program. Při návratu z programu (pokud k němu dojde) signalizuje Carry bit stavového registru procesoru výsledek operace (C nastaven - chyba, C vynulován - bez chyby).  
**Poznámka:** Po skončení zaváděcího procesu v bodu 5 se předá řízení na zaváděcí adresu + 6. Program může pokračovat v zavedení dalších částí, pokud se jedná o vícestupňové zavedení. Hodnota MEMLO (\$02 E7) má po zavedení programu ukazovat na první volnou adresu RAM za právě zavedeným programem. Lze toho dosáhnout například zařazením úseku (psáno pro Atari Assembler/Editor):

```

LDA    #END+1      ; dolní byte
STA    MEMLO
STA    APPMHI
LDA    #END+1/256   ; horní byte
STA    MEMLO+1
STA    APPMHI+1

```

Pokud má program po zavedení převzít řízení, musí být v tomto okamžiku nastavena hodnota DOSVEC (\$0A) tak, aby ukazoval na startovací a restartovací bod zavedeného programu. Pokud nemá zavedený program převzít řízení, zůstane hodnota DOSVEC beze změny.

```

LDA      #RESTRT      , dolní byte start. adresy
STA      DOSVEC
LDA      #RESTRT/256
STA      DOSVEC+1

```

6. JSR nepřímo přes vektor DOSINI na inicializační adresu programu. Program se má inicializovat a vrátit řízení.

Poznámka: OS spouští inicializaci při každém stisku klávesy RESET, probíhá při ní vždy i interní inicializace. Zavedený program se také může inicializovat až po provedení bodu 7.

7. JMP nepřímo přes DOSVEC předá řízení zavedenému programu.

Poznámka: Stisk klávesy RESET po zavedení programu způsobí opakování kroků 6 a 7.

Jako ilustraci principů popsaných výše uvádíme příklad programu pro zavádění během BOOT procesu při zapnutí počítače:

```

; tohle je začátek našeho programu
PST      -          $700          , nebo jiná adresa
          *          PST
; následuje hlavička BOOT souboru
.BYTE    0            ,příznak
.BYTE    PND-PST+1 27/128 ,počet sektorů
.WORD    PST          , zaváděcí adresa
.WORD    PINIT        , inicializační adresa
; začátek kódu pro pokračování zaváděcího procesu
LDA      #PND         , nastavení dolní hranice
STA      MEMLO        , uživatelské paměti
STA      APPMHI
LDA      #PND/256
STA      MEMLO+1
STA      APPMHI+1

LDA      #RESTRT       ,nastavení restartovacího
STA      DOSVEC        ,vektoru
LDA      #RESTRT/256
STA      DOSVEC+1

CLC          ,nastavení příznaku "bez chyb"
RTS

; vstupní bod programu pro inicializaci
PINIT RTS          , nedělá se nic
; následuje hlavní tělo programu
RESTRT -*

;     vlastní
;     program
;
;     ...
;     zde končí vlastní program
PND      -*

```

Soubory ve formátu BOOT nelze vytvořit běžnými prostředky DOSu. Pokud nepřesahuje délka programu 256 sektorů, lze jej na disketu ve formátu BOOT uložit pomocí jednoduchého přídavného programu v assembliere (psáno pro Assembler/Editor firmy Atari). Program předpokládá spojení s předchozím příkladem.

```

; Tento program zapisuje program uložený od adresy
; PST do adresy PND, které jsou definovány v před-
; chozim příkladu
SECSIZ = 128           ; délka sektoru
FLEN = PND-PST+SECSIZ-1/SECSIZ
; počet sektorů souboru
; nastavení řídicího bloku pro vyvolání disketového
; handleru
LDA    =FLEN          ; počet sektorů k zapsání
STA    COUNT
LDA    #1              ; disk 1
STA    DUNIT
LDA    #'W             ; zápis s verifikací
STA    DCOMND
LDA    =PST            ; začátek programu je adre-
STA    DBUFLO          ; sa bufferu
LDA    =PST/256
STA    DBUFHI
LDA    #1              ; počáteční sektor je 1
STA    DAUX1
LDA    #0
STA    DAUX2
; teď zapisujeme po jednotlivých sektorech
BOT1 JSR   DSKINV      ; zapiš sektor
BMI    DERR            ; chyba
LDA    DBUFLO          ; zvýšení adresy bufferu
CLC
ADC    =SECSIZ
STA    DBUFLO
LDA    DBUFLO+1
ADC    #0
STA    DBUFLO+1
INC    DAUX1            ; zvýšení čísla sektoru
BNE    BOT2
INC    DAUX2
BOT2 DEC   COUNT        ; ještě je co zapisovat?
BNE    BOT1
BRK
DERR BRK              ; konec, zapsáno
COUNT **              ; konec při chybě
**1                  ; počítadlo sektorů
.END

```

Každá disketa, na které je nainstalován DOS, je vlastně kombinací BOOT formátu a souborů v linkovaném formátu, který je popsán v kapitole 3.3.2. Sekvenční soubor obsahuje první tři sektory a obsahuje krátký program pro zavedení File Manageru ze souboru DOS.SYS. Protože program musí být krátký, nemůže hledat soubor DOS.SYS podle adresáře. Při nahrávání DOSu na disketu funkci H se kromě vytvoření DOS.SYS a DUP.SYS modifikuje zaváděcí program a zapisuje se do něj číslo sektoru, ve kterém začíná DOS.SYS. Proto nelze soubory DOSu instalovat na jinou disketu překopirováním.

### 3.3.2. Formát sektoru DOS (linkovaný formát)

Tento formát sektorů používá organizátor souborů (File Manager) pro zápis souborů pod řízením diskového operačního systému. Používá jej 99% všech souborů na disketách a umožňuje efektivní využití jejich kapacity. Struktura sektoru je takováto:

bit	7	6	5	4	3	2	1	0
	+-----+-----+-----+-----+-----+-----+							
		data						+0
	-							-
	+-----+-----+-----+-----+-----+-----+							
		číslo souboru	HI					+125
	-							-
	+-----+-----+-----+-----+-----+-----+							
		LO - číslo násled. sekt.						+126
	-							-
	+-----+-----+-----+-----+-----+-----+							
	S   počet byte v sekt.							+127
	-							-

Prvých 125 byte v sektoru se používá pro ukládání dat, poslední 3 byte pro záznam organizačních informací. Číslo souboru je pořadové číslo souboru k němuž sektor patří, v adresáři diskety. Používá se kontrola celistvosti souboru dat. Pokud číslo souboru v sektoru nesouhlasí s číslem souboru v adresáři, generuje se chyba 164, FILE NUMBER MISMATCH.

Číslo souboru je v bitech 2 až 7 bytu 125. Zbývající dva bity tvoří horní byte čísla sektoru, ve kterém soubor pokračuje. Toto číslo je dlouhé 10 byte. Z toho vyplývá, že maximální počet sektorů které může tento formát obsahnot, je 1024. DOS používá všechny adresovatelné sektory. Dalších 17 sektorů, které jsou v rozšířené hustotě k dispozici (celkem 1040) se nepoužívá. Některé sektory si DOS vyhrazuje pro svou administrativu. Maximální počet sektorů, které má k dispozici uživatel je 1010.

Bit S v bytu 127 indikuje, že jde o "krátký" sektor, čili sektor obsahující méně než 125 datových byte. U krátkého sektoru je S = 1. Zbytek bytu tvoří údaj o počtu platných datových bytů v sektoru. Z tohoto čísla lze určit, které byty v krátkém sektoru jsou platnou součástí souboru a které jsou nevýznamovou výplní. Krátký sektor se zpravidla vyskytuje jen na konci souboru, ale teoreticky může být i uprostřed.

### 3.4. Struktura adresáře diskety

Adresář diskety je umístěn na sektorech 360 až 368. Sektor 360 obsahuje tabulkou obsazení jednotlivých sektorů VTOC, sektory 361 až 368 slouží k záznamu informací o uložených sektorech. Přesný formát a obsah těchto sektorů se může sice u některých DOSů lišit, ale je to spíše výjimkou vzhledem ke snaze zachovat slučitelnost se základním standardem DOS 2.0 a 2.5. Z tohoto důvodu je tento odstavec zařazen do kapitoly o obecných principech diskových operačních systémů. U DOSů, které používají odlišný formát adresáře bude na tuto skutečnost zvláště upozorněno.

#### 3.4.1. Struktura VTOC

bit	7	6	5	4	3	2	1	0
	\$02, příznak DOS2							+0
	celkový počet sektorů	LO						+1
		---	počet	---				
		HI						+2
	počet volných sektorů	LO						+3
		---	volných	---				
		HI						+4
	vyhrazeno							+5
								až
								+9
	S S S				S			+10
	Ø	1	2	.	.	.	7	
	S S				S			+11 až +99 pro SD
	8	9	.	.	.	.	15	
								+11 až +137 pro ED
	a t d.							- 0 až 127 v s.1040

Tabulka VTOC je umístěna v sektoru 360. Uchovává informaci o tom, které sektory na disku jsou obsazeny a které volné. Při použití rozšířené hustoty je kromě toho další spolupracující tabulka VTOC v sektoru 1024. Tato specifika bývá u jiných DOSů, než DOS 2.5 řešena odlišně. Proto jsou záznamy pojízené s těmito systémy navzájem slučitelné jen v rozsahu jednoduché hustoty. Struktura tabulky je stejná jak u jednoduché hustoty (SD), tak u rozšířené (ED). Obě tabulky jsou používány tak, že byty 10 až 127 tabulky v sektoru 360 jsou shodné s byty 0 až 117 tabulky v sektoru 1040. Byty 118 až 127 v sektoru 1040 jsou jedinou informací o obsazení sektorů 944 až 1023, protože sektor má jen 128 byte a do sektoru 360, který má prvních 10 byte obsazeno jinými údaji, se tato informace již nevejdou. Systémy, které umožňují pracovat s více sektory než 1024, řeší potíebu větší VTOC různě, obvykle přibírají potíebný počet sektorů od sektoru 359 dolů (MYDOS).

### 3.4.2. Struktura věty adresáře souborů

Od sektoru 361 do 368 leží adresář souborů, mající kapacitu 64 vět. Každá věta obsahuje informace o jednom souboru a má délku 16 byte. Struktura věty je následující:

status souboru	+0
L O	+1
počet sektorů	- - -
H I	+2
číslo LO	+3
prvního sektoru	- - -
H I	+4
jméno souboru	+5
.	až
8 byte	+12
popisná část jména	+13
.	až
(extender)	+15

#### status souboru - byte 0

podává informaci o tom, v jakém stavu se daný soubor právě nachází. Obsah informace se liší podle toho, zda jde o disketu v jednoduché, nebo rozšířené hustotě, resp. zda soubor na disketu v rozšířené hustotě používá sektory z rozšířené části, t.j. sektory s číslem vyšším než 720. Při výpisu adresáře diskety funkci A v DOS 2.5 jsou tyto soubory označeny špičatými závorkami. Význam bitů:

bit	děk	hex	význam
7	128	\$80	soubor je zrušen (SD, ED)
6	64	\$40	pozice v adresáři je použita pro soubor (SD)
5	32	\$20	soubor je blokován proti zápisu (SD, ED)
4	2	\$02	bit je stále nastaven na 1, slouží k rozlišení formátu DOS 2.x
3	1	\$01	ve spojení s bitem 6 signalizuje, že je soubor otevřen pro zápis. (SD, ED)
			Je-li bit 6=0, znamená to že pozice je použita pro soubor zasahující do oblasti nad sektor Y20 (ED).

Je-li stavový byte souboru nula, znamená to že pozice adresáře nebyla ještě od naformátování diskety použita a to je také signálem konce adresáře. Po-

drobnější vysvětlení si zasluhuje signalizace stavu souboru v oblasti jednoduché a rozšířené hustoty. Platný nezablokováný soubor má status buď \$42, nebo \$03 pokud některým sektorem zasahuje do oblasti ED. Je-li nastaven bit 7, nemůže již být nastaven žádný další bit. To znamená, že zrušený soubor má vždy status \$80, ať již je umístěn kdekoli. Bit 2 a 5 platí stejně pro soubory v části SD i ED. Rozdíly jsou v bitech 0 a 6. Platný soubor v oblasti jednoduché hustoty má status \$42 (nebo \$62 je-li blokován), zatímco v oblasti rozšířené hustoty má status \$03 (\$23). Soubor otevřený pro zápis (a regulérně neuzávřený) má status \$43 bez ohledu na jeho umístění. Jestliže tedy máme soubor v oblasti rozšířené hustoty (ED) ve stavu \$03 a otevřeme jej pro zápis, dostane status \$43. Po úspěšném uzavření se status vrátí opět na \$03. Tento poněkud nejednoznačný a zamotaný význam bitů vznikl v historických souvislostech, když byl vypracováván na podkladě kompatibility s DOS 2.0 systém DOS 2.5.

#### **Počet sektorů - byte 1 a 2**

Udává v uspořádání dolní - horní byte délku souboru v sektorech. Tento údaj je jen informativní a objevuje se ve výpisu obsahu diskety. Při čtení se systém řídí počtem nalezených sektorů při čtení posloupnosti linkovaného formátu. Tento počet je zpravidla stejný jako údaj v adresáři, ale díky různým chybám mohou nastat situace, kdy se liší a výpis adresáře ukazuje falešné hodnoty. V tomto případě je nejlépe zkopirovat všechny soubory na nové naformátovanou disketu.

#### **Číslo prvního sektoru - byte 3 a 4.**

Číslo prvního sektoru souboru v uspořádání dolní - horní byte. Od tohoto sektoru začíná čtení souboru a každý další sektor se vypočte z linkovacích informací (viz 3.3.2.). Konec souboru se signalizuje tím, že jako číslo následujícího sektoru je nula. Takovýto sektor neexistuje protože čislování začíná od jedničky. Poslední sektor bývá obvykle krátký.

#### **Jméno souboru - byty 5 až 12.**

Jméno souboru tvoří 8 znaků v kódu Atascii. (viz 2.2.10)

#### **Popisná část jména - koncovka (extender) - byty 13 až 15.**

Poslední tři byty tvoří popisnou část jména souboru - koncovku.

### **3.5. Stavba segmentovaného binárního souboru**

Binárním souborem se rozumí soubor dat, určený k zavedení na určité místo paměti. Může se jednat buď o program ve strojovém kódu, což je nejčastější případ, ale i o obrázky, znakové sady nebo jiná data. Zde popsanou strukturu používají všechny existující diskové operační systémy k zavádění a spouštění strojových programů. U počítačů Atari se vyskytuje pouze tzv. absolutní programy, které mohou fungovat jen v tom místě paměti, pro které jsou určeny. Proto je při uschovávání takovýchto programů v podobě disketového souboru třeba spolu s vlastním obsahem paměti uložit i definici

adresové oblasti, do které kód patří. Segment je úsek kódu, který se ukládá v souvislé oblasti paměti. Jeho délku jednoznačně určuje jeho počáteční a koncová adresa. Není nutné, aby segment tvořil nějaký logický nebo významový celek. Délka segmentu je libovolná od 0 teoreticky do \$FFFF (65536) byte. Podrobnosti o stavbě a používání binárních segmentovaných souborů jsou blíže rozvedeny v 4.1.11.

#### 4. DOS 2.5

Disketový operační systém DOS 2.5 je produkt firmy Atari. Nahrazuje starší systém DOS 3, který se neujal. Je určen pro diskovou jednotku 1050 a počítače řady 130XE, zvětšenou paměť těchto modelů lze používat jako RAM-disk. Pracuje samozřejmě i na jiných modelech řady XE a XL. V této kapitole je podrobný popis funkcí DOSu 2.5. Přitom se předpokládají základní znalosti ve spouštění počítače s DOSem apod., které jsou popsány v kapitole 2 na začátečnické úrovni. Některé funkce se tu objevují znova. Tentokrát jsou ale popsány podrobně a do hloubky. Kapitola je rozdělena na část popisující menu, část věnovanou práci s disketovou jednotkou v Basicu a část příloh obsahující informace o používání RAM-disku, služebních programů a o přizpůsobování parametrů systému.

##### 4.1. Funkce menu DOS 2.5

Vyvolání menu bylo popsáno v kapitole 2, stejně tak způsob volby funkcí. Menu nabízí 16 funkcí:

**DISK OPERATING SYSTEM II VERSION 2.5  
COPYRIGHT 1984 ATARI CORP.**

<b>A. DISK DIRECTORY</b>	<b>I. FORMAT DISK</b>
<b>B. RUN CARTRIDGE</b>	<b>J. DUPLICATE DISK</b>
<b>C. COPY FILE</b>	<b>K. BINARY SAVE</b>
<b>D. DELETE FILE(S)</b>	<b>L. BINARY LOAD</b>
<b>E. RENAME FILE</b>	<b>M. RUN AT ADDRESS</b>
<b>F. LOCK FILE</b>	<b>N. CREATE MEM.SAV</b>
<b>G. UNLOCK FILE</b>	<b>O. DUPLICATE FILE</b>
<b>H. WRITE DOS FILES</b>	<b>P. FORMAT SINGLE</b>

**SELECT ITEM OR RETURN FOR MENU :**

■ Obr.10: menu DOS 2.5

##### 4.1.1. Výpis adresáře - DISK DIRECTORY

A

Adresář je oblast diskety, uchovávající informace o soubory jsou na ní umístěny. Funkcí A si z něj můžeme informace vypsat. Pro každý soubor se vypisuje jméno s koncovkou a délka v sektorech. Lze vypsat celý adresář, nebo jeho část, podle zadání filtru. Výstup může být na určené zařízení nebo do souboru. Při zobrazení výzvy SELECT ITEM OR RETURN FOR MENU stiskneme A a RETURN. Na obrazovce se objeví dotaz:

**DIRECTORY--SEARCH SPEC,LIST FILE?**

Tento dotaz znamená: výpis adresáře--zadej filtr a specifikuj zařízení (a soubor), kam bude výpis poslán. Standardní filtr je "D:.\*.", standardní výstup obrazovkový editor "E:". Příklad základního výpisu je v odstavci 2.2.6. Pří-

výpisu adresáře disket v rozšířené hustotě je třeba upozornit na jednu zvláštnost při udávání délky souboru. Celkový počet sektorů je při formátování v rozšířené hustotě 1010, ale číslo udávající počet sektorů je jen třímístné. Čerstvě naformátovaná disketa ukáže 999+ volných sektorů, to znamená více, než 999. Pokud by byl na disketě soubor delší, než 999 sektorů, bude jeho délka ve výpisu adresáře jen 999 sektorů. Soubor takové délky se však může vyskytnout jen velmi zřídka. Podívejme se na další informace, které nám výpis adresáře poskytuje:

```
*DOS      SYS 037
*DUP      SYS 042
FILE1    DAT 204
<FILE3   DAT>350
<FILE4   DAT>022
236 FREE SECTORS
```

Před prvními dvěma soubory je hvězdička která znamená, že jsou zamknuty (blokovány proti zápisu). Špičaté závorky znamenají, že disketa je formátována v rozšířené hustotě a takto označené soubory používají sektory s číslem vyšším, než 720. Tato skutečnost je podstatná při posuzování kompatibility s ostatními systémy. Většina systémů jiných firem je s DOS 2.5 slučitelná jen v oboru jednoduché hustoty, do sektoru číslo 720. Soubory ve špičatých závorkách proto nebudou ve výpisu adresáře jiným DOSem vidět a nepůjde s nimi v takovém DOSu pracovat. Povšimněme si, že označení nezávisí na velikosti souboru.

Další varianty výpisu adresáře se určí parametry SEARCH SPEC (filtr pro výběr souborů) a LIST FILE (zařízení, ev. soubor, na který se obsah vypíše). Tvorba filtru pro výběr souborů je popsána v odstavci 2.2.10. Filtr obvykle používáme při hledání nějakého určitého souboru nebo druhu souborů. Nejčastější použití má výtisk obsahu diskety na tiskárnu. Na příkladech si ukážeme některé možnosti.

Př. 1. výpis všech programů s koncovkou .BAS na obrazovku:

```
SELECT ITEM OR RETURN FOR MENU:
A      (RETURN)
DIRECTORY--SEARCH SPEC,LIST FILE?
*.BAS  (RETURN)
```

Př. 2. výpis obsahu celé diskety na tiskárnu:

```
SELECT ITEM OR RETURN FOR MENU:
A      (RETURN)
DIRECTORY--SEARCH SPEC,LIST FILE?
.P:
```

(Čárka na začátku znamená, že první parametr je vynechán)

Př. 3. uložení seznamu programů ve strojovém kódu na disketu v jednotce č. 2 do souboru SEZNAM na RAM-disk (číslo 8, vynechané D si DOS doplní):

```
SELECT ITEM OR RETURN FOR MENU:  
A      (RETURN)  
DIRECTORY--SEARCH SPEC,LIST FILE?  
*.COM,8:SEZNAM (RETURN)
```

(Programy ve strojovém kódu se často označují popisnou částí .COM)

Př. 4. Výpis obsahu diskety v jednotce číslo 2:

```
SELECT ITEM OR RETURN FOR MENU:  
A      (RETURN)  
DIRECTORY--SEARCH SPEC,LIST FILE?  
D2:.*
```

Př. 5. výpis všech souborů, které mají první znak jména A, třetí C a tříznakovou koncovkou začínající na .DA z RAM-disku na tiskárnu:

```
SELECT ITEM OR RETURN FOR MENU:  
A      (RETURN)  
DIRECTORY--SEARCH SPEC,LIST FILE?  
8:A?C*.DA?,P:
```

Př. 6. výpis všech souborů s třípísmenovým názvem bez koncovky z disku 1 na obrazovku:

```
SELECT ITEM OR RETURN FOR MENU:  
A      (RETURN)  
DIRECTORY--SEARCH SPEC,LIST FILE?  
???(RETURN)
```

Příklady ilustrují používání standardních parametrů na místě, kde celý nebo část parametru neuvedeme. Vynechaný kód zařízení v SEARCH SPEC se doplní na "D1:", vynechané číslo zařízení se doplní jedničkou. V parametru LIST FILE je standard "E:". Pokud se pokusíme vypsat obsah disku, který není ve formátu DOSu 2.5, mohou nastat tyto případy:

- jde-li o novou, nenaformátovanou disketu, nebo o disketu zapsanou ve dvojitě hustotě, ozývá se zhruba každých pět vteřin pípnutí a po nějaké době se vypíše hlášení o chybě ERROR 144. Disketu je třeba naformátovat funkcí I nebo P. Majitelé XF551 by se měli nejprve podívat, zda nejde o disketu ve dvojitě hustotě pomocí některého z DOSů, které ji umožňují.
- funkci A nelze přímo vypisovat adresou diskety v DOSu 3. Jde-li o tento případ, prostudujte popis programu COPY32.COM v odstavci o služebních programech DOSu 2.5.
- Jde-li o komerčně produkovanou hru, nebo jiný program ve formátu BOOT, uvidíme obvykle buď změl nesmyslných znaků, nebo někdy název programu a identifikaci výrobce. Obsah takovýchto disket nelze vypisovat protože jejich uspořádání neodpovídá pravidlům DOSu.

#### **4.1.2. Přechod do modulu - RUN CARTRIDGE**

**B**

Pokud jako odpověď na základní výzvu SELECT ITEM OR RETURN FOR MENU odpovíme B a (RETURN), DOS předá řízení programu v zásuvném modulu, nebo vestavěnému Atari Basicu. Je-li Atari Basic vypnut podržením klávesy OPTION při zapnutí počítače a není-li zasunut žádný modul, vypíše se hlášení "NO CARTRIDGE" a základní výzva. Je-li vše v pořádku, obrazovka se vymaže a objeví se ohlášení modulu. V případě Atari Basicu je to výzva READY, u Assembleru/Editoru výzva EDIT a podobně.

Upozornění: Pokud nemáte na disketu založen soubor MEM.SAV, je přechodem z modulu do DOSu jakýkoliv program který se nachází v paměti ztracen. U modelu 130XE se soubor MEM.SAV zakládá automaticky na RAM-disku D8. Příčinou ztráty je skutečnost, že paměťová oblast nad rezidentní částí DOSu používá modul i soubor DUP.SYS, který je při přechodu do DOSu načítán a který realizuje funkce diskového menu. Pokud pracujeme s modulem, je tato část paměti používána pro ukládání programů se kterými právě pracujeme. Toto sdílení paměťové oblasti umožňuje uživateli větší paměťový prostor.

Příklad:    **SELECT ITEM OR RETURN FOR MENU:**  
**B (RETURN)**

Pokud existuje soubor MEM.SAV, oblast paměti kterou by narušil DUP.SYS se při přechodu do DOSu uloží do tohoto souboru a teprve poté je načten a spuštěn DUP.SYS. Při přechodu do modulu funkcí B se nejprve obsah porušené části paměti obnoví. Poté se předá řízení modulu. Proto máme v paměti program tak, jak jsme jej před přechodem do DOSu opustili. Některé funkce menu však za určitých okolností poruší další oblast paměti, čímž se MEM.SAV zneplatní. Podrobnosti viz funkce N, C, O a J. Funkce B s variantou platnosti MEM.SAV předpokládá, že v jednotce 1 je stále tatáž disketa, na kterou byla uložena paměť při přechodu z modulu. U 130XE je MEM.SAV uložen na RAM-disku, který jak známo nelze vyměnit, takže předpoklad je splněn automaticky.

#### **4.1.3. Kopírování souborů - COPY FILE**

**C**

Funkce C se používá ke kopírování souborů z jednoho zařízení na jiné zařízení (z jedné diskové jednotky na druhou, do RAM-disku, na tiskárnu, na obrazovku), nebo na stejnou diskovou jednotku pod jiným jménem. Ke kopírování souboru z jedné diskety na jinou pomocí jedné jednotky použijte funkci "O".

Povel COPY FILE vyžaduje zadání dvou parametrů FROM a TO, tedy určení co chceme kopírovat a kam to chceme kopírovat. První parametr bývá specifikace souboru a může obsahovat i výběrový filtr. Filtr umožňuje velmi pohodlně kopírovat skupinu souborů z jedné jednotky na druhou. (viz příklad 6). Výjimkou jsou soubory s popisnou částí .SYS a soubor AUTO-RUN.SYS. Tyto soubory se nezkopírují, i když jsou filtrem vybrány a musí se kopírovat zvlášť jmenovitě.

Druhý parametr může obsahovat volbu "/A" (append), která umožňuje připojit první soubor z parametru FROM za druhý soubor, uvedený v parametru TO. Oba parametry musí obsahovat platnou specifikaci souboru, po-

kud se jedná o data z jiného zařízení, než disketové jednotky číslo 1, je nutno psát i kód zařízení. U sekvenčních zařízení (tiskárna, obrazovka) není nutno uvádět jméno souboru (viz příklady 3, 5, 6). Povel COPY FILE se také často používá k vytvoření kopie některého souboru na stejnou disketu pod jiným jménem, přičemž se jména mohou lišit buď úplně, nebo jen koncovkou. Použitím volby "/A" můžeme také skládat jednotlivé segmenty strojového programu do segmentovaného binárního souboru. Při spojování souborů volbou append se druhý soubor připojuje bezprostředně za první a využívá i dříve neobsazeného místa krátkého sektoru na konci prvního souboru (viz 3.3.2.). Proto může být vzniklý soubor kratší, než součet délek souborů, z nichž vznikl. Pokud chceme kopírovat a máme na disketě platný soubor MEM.SAV, objeví se nám po zadání a odeslání parametrů (RETURN)em dotaz:

**TYPE "Y" IF OK TO USE PROGRAM AREA.**

**CAUTION: A "Y" INVALIDATES MEM.SAV**

DOS se ptá, zda může jako paměť pro kopírování použít i oblast, která není uschována v souboru MEM.SAV. Pokud odpovíme ano-Y, kopírování bude probíhat normální rychlostí, ale soubor MEM.SAV již nebude platný a při následném přechodu do modulu bude obsah programové paměti ztracen. Odpovíme-li ne-N, DOS použije ke kopírování jen malou paměť v oblasti uschovávané do MEM.SAV. Tím je kopírování o mnoho pomalejší, ale obsah paměti a soubor MEM.SAV zůstane platný. Po návratu do modulu najdeme svůj program zachován.

#### *Upozornění:*

Nespojujte dohromady tokenizované programy Basicu (viz 4.2.1.), t.j. programy, uložené povelom SAVE. Každý tokenizovaný program má svou vlastní tabuľku symbolů a nače se jen první z nich. Spojovat lze programy, uložené povelom LIST, nebo strojové programy vytvořené některým překladačem assembleru. U programů Basicu uložených povelom LIST je nutno počítat s tím, že pokud jsou v obou programech fádky stejného čísla, fádky z druhého programu nahradí fádky z prvního.

Př.1: Kopie DOSEX.BAS z D1 na D2.

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY--FROM. TO?  
DOSEX.BAS,2:DOSEX.BAS (RETURN)
```

Př.2: Bezpečnostní kopie souboru na stejnou disketu:

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
DOSEX.BAS,DOSEX.BAK
```

Př.3: Výpis programu na obrazovku:

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
DOSEX.LST,E: (RETURN)
```

Př.4: Kopie souboru z obrazovky do diskového souboru TEMP.DAT. To, co napišeme na obrazovku se uloží. Zadávání dat ukončíme znakem (CTRL) - 3.

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
E:,TEMP.DAT (RETURN)
```

(obrazovka se vymaze a kurzor se objeví v levém horním rohu). Napište:

```
PETR (RETURN)  
PAVEL (RETURN)  
HONZA (RETURN)  
KONEC (RETURN)  
(CTRL) - 3
```

Př.5: Výpis programu PROG.LST na tiskárnu:

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
D:PROG.LST,P: (RETURN)
```

Poznámka: Tímto způsobem lze vypisovat jen programy, uložené povelom LIST. Tokenizované programy, ukládané povelom SAVE lze v čitelné formě vypsat jen pomocí Basicu (viz 4.2.1).

Př.6: Kopíruje všechny soubory z D1 na D2 vyjma souborů s koncovkou ".SYS":

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
*.*,D2: (RETURN)
```

(Povšimněte si, že ačkoliv jsou filtrem vybrány všechny soubory, nezkopíruje se systémové soubory DOS.SYS, DUP.SYS a soubor AUTORUN.SYS. AUTORUN.SYS se musí kopírovat samostatně. DOS.SYS a DUP.SYS se nekopírují, ale pořizují se funkci H)

Př.7: Připojení PROG2 k souboru PROG1 na D1.

```
SELECT ITEM OR RETURN FOR MENU:  
C (RETURN)  
COPY__FROM,TO?  
PROG2,PROG1/A
```

#### **4.1.4. Rušení souborů : DELETE FILES**

**D**

Tato funkce umožňuje rušení jednoho, nebo více souborů. Zrušený soubor je v adresáři označen jako zrušený (viz 3.4.2). Pro určení skupiny zrušených souborů lze použít filtr. Před rušením každého souboru se požaduje potvrzení akce, což nám možnost změnit minění, nebo ze skupiny vybrané filtrem zrušit jen některé soubory. Požadavek verifikace lze potlačit volbou "/N". Tím se rušení urychlí, ale nemáme šanci napravit eventuelní chybu. Soubor, který rušíme, nesmí být blokován proti zápisu, jinak se objeví chyba ERROR 167. Zrušíme-li některý soubor omylem, existuje cesta k nápravě v případě, že jsme mezičím na disketu nic nezapsali. Postup je uveden v odstavci 4.4.2. u služebního programu DISKFIX.COM.

Př.1: Rušení všech souborů začínajících na "REM" s popisnou částí ".BAS" s potvrzováním rušení každého souboru:

```
SELECT ITEM OR RETURN FOR MENU:  
D (RETURN)  
DELETE FILE SPEC  
D2:REM*.BAS (RETURN)  
TYPE "Y" TO DELETE ...  
REM1.BAS?  
Y (RETURN)  
REMBAA.BAS?  
Y (RETURN)
```

Př.2: Zrušení jednoho souboru z potvrzením:

```
SELECT ITEM OR RETURN FOR MENU:  
D (RETURN)  
DELETE FILE SPEC  
TEMP.DAT (RETURN)  
TYPE "Y" TO DELETE ...  
TEMP.DAT?  
Y (RETURN)
```

Př.3: Zrušení souboru bez potvrzení :

```
SELECT ITEM OR RETURN FOR MENU:  
D (RETURN)  
DELETE FILE SPEC  
DOSEX.BAS/N (RETURN)
```

Př.4: zrušení všech souborů na disketě (i s \*.SYS a AUTORUN.SYS):

```
SELECT ITEM OR RETURN FOR MENU:  
D (RETURN)  
DELETE FILE SPEC  
*.*/N
```

Potvrzování akce se provádí písmenem Y, pokud souhlasíme, nebo čímkoliv jiným pokud nesouhlasíme. Proto je nejkratší jako zápornou odpověď používat samotný (RETURN).

#### 4.1.5 . Přejmenování souborů - RENAME FILE

E

Tato funkce umožňuje změnit jméno jednoho nebo více souborů. Zadávají se dva parametry. OLD NAME čili staré jméno a NEW NAME neboli nové jméno. Parametr OLD NAME musí být úplné jméno (pokud vynecháme kód diskové jednotky, doplní se jako obvykle "D1:"). NEW NAME nesmí obsahovat kód zařízení, protože ten je již určen v OLD NAME a soubor zůstává na místě a jen změní jméno. Pokud se ve starém jméně použije filtr pro přejmenování skupiny souborů, musí být hvězdičky a otazníky v minimálně stejném počtu a na stejných místech i ve jméně novém. Nejnázornější bude příklad:

platné zadání: TEST,NEW

TEST.\*,NEW.\*

\*.DAT,\*.BAK

\*.??1,\*.??2

TEST3.DAT,FILE?.\*

TEST3.DAT,FILE\*.\*

neplatné zadání: TEST.\*,NEW

TEST?,NEW

TEST\*.??1,TEST3.??2

Poznamenejme, že ve filtru pro nové jméno může být hvězdiček a otazníků více, než ve jméně starém. V takovém případě se příslušné znaky kopírují beze změny ze starého jména. Tuto možnost ukazují poslední dvě platná zadání. Stejněho výsledku by se dosáhlo výrazem TEST3.DAT.FILE.DAT.

Pamatujme, že každé jméno na disketu musí být jednoznačné. Pokud přejmenováváme soubor jako jednotlivý, můžeme jako nové jméno zadat cokoliv, včetně jména již existujícího na stejně disketě. Pokud se takováto chyba stane, dostaneme se do písekerní situace, protože DOS při operaci se skupinou souborů vybere samozřejmě oba stejně pojmenované soubory. S oběma soubory současně pracuje také při rušení, blokování a odblokovávání skupiny (D, F, G). Pomoci si lze opětným přejmenováním. Pokud jsou na disketě dva, nebo více souborů stejného jména, je přejmenován první z nich. Kromě toho lze také přejmenovat soubory pomocí služebního programu DISKFIX.COM.

#### 4.1.6. Blokování (zamknutí) souboru - LOCK FILE

F

Touto funkcí můžeme zablokovat jednotlivé soubory i skupiny. Do blokování souboru nelze zapisovat, připojovat k němu jiné soubory, přejmenovat ho ani zrušit. Pokus o akci tohoto druhu končí chybou ERROR 167. Pomoci filtru lze zamknout skupinu souborů. Blokováný soubor je ve výpisu adresáře diskety označen hvězdičkou před jménem. (Tato hvězdička nemá nic společného s hvězdičkou ve filtru).

**Pozor:** Blokování nechrání soubor před vymazáním při formátování diskety!

Př.1: Blokujeme soubor DOS.SYS na jednotce číslo 1:

SELECT ITEM OR RETURN FOR MENU:

F (RETURN)

WHAT FILE TO LOCK?

DOS.SYS (RETURN)

Př.2: Zamknutí všech programů v Basicu - mají koncovku .BAS:  
 SELECT ITEM OR RETURN FOR MENU:  
 F (RETURN)  
 WHAT FILE TO LOCK?  
 D1:\*.BAS (RETURN)

Př.3: Blokování všech souborů začínajících na T na disku 1:  
 SELECT ITEM OR RETURN FOR MENU:  
 F (RETURN)  
 WHAT FILE TO LOCK?  
 T\*.\* (RETURN)

Př.4: Blokování všech souborů na RAM-disku D8:  
 SELECT ITEM OR RETURN FOR MENU:  
 F (RETURN)  
 WHAT FILE TO LOCK?  
 D8:.\* (RETURN)

#### **4.1.7. Uvolnění (odemknutí) souboru - UNLOCK FILE**

**G**

Tuto funkci můžeme uvolnit soubory, zamknuté funkcí F. Po provedení funkce zmizí hvězdička před jménem blokovaného souboru ve výpisu adresáře. I u této funkce lze použít filtr pro výběr uvolňovaných souborů.

Př.1: Odemkneme soubor DOS.SYS na jednotce číslo 1:  
 SELECT ITEM OR RETURN FOR MENU:  
 G (RETURN)  
 WHAT FILE TO UNLOCK?  
 DOS.SYS (RETURN)

Př.2: Uvolnění všech programů v Basicu - mají popisnou část .BAS:  
 SELECT ITEM OR RETURN FOR MENU:  
 G (RETURN)  
 WHAT FILE TO UNLOCK?  
 D1:\*.BAS (RETURN)

Př.3: Uvolnění všech souborů začínajících na T na disku 1:  
 SELECT ITEM OR RETURN FOR MENU:  
 G (RETURN)  
 WHAT FILE TO UNLOCK?  
 T\*.\* (RETURN)

#### **4.1.8. Instalace DOSu - WRITE DOS FILES**

**H**

Jak jsme již říkali v odstavci o kopírování souborů, nelze soubory DOS.SYS a DUP.SYS přenášet na jinou disketu kopírováním. Proto je k jejich vytvoření určena speciální funkce H. Tato funkce se používá jednak k založení DOSu na nové prázdné disketě, jednak ji lze uložit na disketu aktuální stav DOSu po modifikaci (viz 4.3.4, 4.3.5). Po zapsání obou souborů se vypíše základní

výzva SELECT ITEM OR RETURN FOR MENU. Disketa nesmí být chráněna nálepkou proti zápisu a pokud se soubory ukládají na disketu, na které již jsou soubory DOSu zapsány, nesmí být blokovány.

Př.:      SELECT ITEM OR RETURN FOR MENU:  
 H (RETURN)  
 DRIVE TO WRITE DOS FILES TO?  
 I (RETURN)  
 TYPE "Y" TO WRITE DOS FILES TO DPIPE I.  
 Y (RETURN)  
 WRITING NEW DOS FILES

Zápis souborů DOSu je předcházen bezpečnostním dotazem. Pokud odpovíme "Y", soubory se zapíší.

#### **4.1.9. Formátování - FORMAT DISK**

**I**

Tato funkce se používá k formátování diskety v rozšířené hustotě. Lze ji použít jak u jednotky 1050, tak u XF551. Podrobně je formátování probráno v odstavci 2.2.9. Prázdná disketa obsahuje 1010 volných sektorů, které se ve výpisu objevují jako 999+ sektorů.

**Výstraha:** Formátování diskety vymaže vždy všechny soubory, které se na disketě nacházely!

#### **4.1.10. Duplikování disket - DUPLICATE DISK**

**J**

Tato funkce se používá ke zhotovení přesné kopie diskety ve formátu DOSu. Nelze jí duplikovat diskety s programy ve formátu BOOT (viz 3.3.1.) a firemní originály, chráněné proti kopirování. Duplikacní proces probíhá sektor po sektoru a soubory, přenesené na cílovou disketu jsou i umístěny na stejných místech. Duplikování bylo velmi podrobně vysvětleno v odstavci 2.2.7.

#### **4.1.11. Ukládání úseků paměti - BINARY SAVE**

**K**

**Poznámka:** Tato funkce není určena pro začátečníky. K jejímu pochopení je třeba znát hexadecimální číselnou soustavu a mít určité znalosti o strojovém kódu a obsazení paměti počítače.

Funkci K lze uložit do souboru na disketu obsah vymezené části paměti. Soubor má standardní binární formát (viz 3.5.). Parametry pro tuto funkci jsou hexadecimální čísla, před které se v tomto případě nepíše znak dolar. Udáváme tyto hodnoty: START, END a nepovinné INIT a RUN. Pomocí hodnot START a END udáváme úsek paměti, který se má uložit, hodnoty INIT a RUN umožňují definovat rozbehové a inicializační adresy tak, že se uložený program po zavedení automaticky spustí. Uživatel má plnou odpovědnost za správnost zadaných parametrů.

Př. 1:      Do souboru BINFIL.OBJ uložíme obsah paměti 3C00 až 5BFF:

SELECT ITEM OR RETURN FOR MENU  
 K (RETURN)  
 SAVE--GIVE FILE,START,END(,INIT,RUN)  
 BINFIL.OBJ,3C00,5BFF (RETURN)

V tomto příkladu se nepoužívá ani INIT, ani RUN. Proto pravděpodobně nepůjde o program, ale o data, jako je znaková sada, obrázek a podobně. BINFIL.OBJ je jméno souboru, do něhož se data uloží.

Všechny binární segmentované soubory, ať již byly vytvořeny překladačem assembleru, nebo funkcí K, mají stejnou strukturu. Na začátku souboru je šestibytevá hlavička, jejíž příklad je vypsán v tabulce. Snadno v ní najdeme počáteční a konečnou adresu:

č. byte	dek	hex.	popis
1	255	FF	identifikační kód binárního souboru
2	255	FF	
3	0	0	počáteční adresa - dolní byte
4	60	3C	- horní byte
5	255	FF	koncová adresa - dolní byte
6	91	5B	- horní byte
dále obsahuje segment 8192 (dekadicky) byte dat.			

Dva nepovinné parametry RUN a INIT dávají možnost automatického spuštění strojového programu po zavedení do paměti. Soubor, který užívá tyto dvě adresy se také označuje jako "load-and-go", t.j. "načtení-a-rozběh". Soubor, který nemá žádnou z těchto dvou adres, se jen zavede, jde o typ "load". Program uložený tímto způsobem lze rozbahnout funkcí "M". Musíme k tomu ovšem znát adresu, na které začíná.

Všeobecně řečeno, RUN specifikuje bod programu, na který se má přejít poté, co je celý program načten do RAM, t.j. po nalezení konce souboru. Pokud jde o složený soubor (soubor, který se skládá ze segmentu dat a z dvou segmentů, definujících RUN a INIT, se nepokládá pro jednoduchost za složený), může být platná jen jedna adresa RUN. Pokud má více segmentů definovanou adresu RUN platí ta, která byla definována naposled.

Je-li definována inicializační adresa, předá se řízení na adresu na kterou ukazuje okamžitě po načtení definujícího segmentu a dotyčný úsek programu se spustí instrukcí JSR. To platí i u souboru složeného z několika segmentů typu "load-and-go", tedy když má každý definovánu svou vlastní inicializační adresu. V tomto případě se každý segment s definovanou INIT adresou spustí, jakmile se adresa načte. Každý segment se zavede a spustí dříve, než následující. Vykonání určené adresou v segmentu INIT se v každém případě provede dříve, než rozbeh určený startovací adresou RUN. Soubory vytvořené Assembler/Editorem, nebo jiným komplilátorem assembleru mohou používat pro automatický rozbeh také inicializační a startovací adresy. K tomu je třeba nastavit v příslušném místě adresový čítač na odpovídající adresu (RUN-\$2E0, INIT-\$2E2) a uložit do ní příslušnou hodnotu. Při stavbě automaticky spouštěných programů musíme mít na paměti několik faktů:

- INIT adresa se použije vždy dříve, než adresa RUN.
- Program, končící instrukcí RTS předá vždy řízení DOSu.
- Každý segment kódu, který se spustí adresou INIT musí končit instrukcí RTS, pokud se mají zavést i segmenty, následující za ním.

- Během zavádění složeného segmentovaného binárního souboru je otevřen kanál 1, který je ke čtení zaváděného programu používán. Segmenty spouštěná adresou INIT během zaváděcího procesu nemají proto kanál 1 k dispozici a nesmí jej nijak narušit, jinak se zaváděcí proces zhroutí.

Příklad kostry programu s definovanými adresami RUN a INIT:

```

,Fiktivní program začíná na adrese $2000
; nejprve deklarace konstant
RUN      - $02E0      ; rozbehová adresa
INIT     - $02E2      ; inicializační adresa
*- RUN      ; definice RUN může být kdekoli
.WORD START

        -- $2000      ; počáteční adresa programu
START    - *          ; zde začíná vlastní program
.

.

INIT1   LDA #0      ; začátek bloku pro inicializaci
        STA 559      ; může třeba vypínat obraz
.

RTS      ; konec bloku
*- INIT
.WORD INIT1
*- $5000      ; další inicializační blok
INIT2   LDX #$20      ; třeba zavírá kanál 2
        LDA #12
        STA $342,X
.

JSR $E456
RTS
*- INIT
.WORD INIT2
.END

```

Načítání a spuštění programu by probíhalo takto: Nejprve se definuje rozbehová adresa RUN. Ta se ale zatím jen uloží. Poté se načte tělo programu. Pro jeho rozbehnutí připravují podmínky segmenty INIT1 a INIT2. Nejprve se načte INIT1. Potom je definována jeho spouštěcí adresa jako INIT, takže načítání se přeruší a provede se podprogram INIT1. Protože na jeho konci je instrukce RTS, pokračuje načítání segmentem INIT2, který má rovněž definovanou adresu INIT a je tedy po jejím načtení spuštěn. INIT2 také končí instrukcí RTS. DOS chce tedy pokračovat v zavádění další části. Ta ale již neexistuje a je nařazen konec souboru. DOS proto předá řízení na adresu, definovanou jako RUN.

### Ukládání paměti s volitelnými parametry

Adresy RUN a INIT lze definovat nejen v programovém kódu, ale i pomocí funkce "K" v menu. Předpokládejme, že máme přeložený program v assembleru, který leží v paměti od \$4200 do \$4FFF a potřebuje mít před spuštěním naplněnu datovou oblast. Naplnění provede inicializační podprogram, který je v paměti od \$4000 do \$41FF. Budeme při tom předpokládat, že oba programy budou uloženy za sebou do souboru PROG.OBJ, že jde o vykonatelné programy, končící instrukcí RTS a že soubor je již v paměti načten.

```
Př.2:      SELECT ITEM OR RETURN FOR MENU
          K   (RETURN)
          SAVE--GIVE FILE,START,END(,INIT,RUN)
          PROG.OBJ,4000,4FFF,4000,4200 (RETURN)
```

Při načítání takto vytvořeného souboru do paměti funkci "L" bude průběh dějů takovýto:

1. Oblast paměti od \$4000 do \$4FFF se naplní programem.
2. Adresa INIT (\$4000) se uloží do paměti na adresy \$2E2, \$2E3.
3. Vykoná se inicializační program od \$4000 do \$41FF.
4. Adresa RUN (\$4200) se uloží na adresy \$2E0, \$2E1.
5. Spustí se hlavní program a poběží, dokud se neprovede instrukce RTS, nebo se nestiskne (RESET).

Fakud je soubor složen z více segmentů, je situace komplikovanější a záleží na pořadí, v jakém byly jednotlivé segmenty spojovány. Následující odstavec ukazuje různé případy, vzniklé připojováním souborů.

### Struktura složeného binárního souboru

Než začneme vysvětlovat příklady, podívejme se na strukturu složeného souboru. Je složen z několika jednoduchých binárních souborů uložených do hromadky. Soubory můžeme skládat dvojím způsobem, z něhož plynou dva možné formáty. První je vytváření funkci C - COPY FILE a vyznačuje se tím, že každá část má na začátku hlavičku binárního souboru. Tento formát nelze zavádět do paměti Assembler/Editorem, ale lze jej spouštět funkci L v DOSu. Druhý formát se vytváří postupným spojováním souborů funkci K. Tento formát má hlavičku binárního souboru jen na začátku a lze jej načítat jak Assemblerem, tak funkci L. V obou případech se funkce provádí s parametrem "/A". Tedy formát COPY má hlavičku na začátku každého segmentu, formát BINARY SAVE jen na začátku.

Sledujme průběh událostí při zavádění různě spojených souborů s různě definovanými adresami INIT a RUN. Pro lepší pochopení si představujte, že programový segment s adresami RUN a INIT tvorí jeden celek.

```
Př.3:      Předpokládejme tři soubory, každý má definovanou adresu RUN,
          žádný adresu INIT. Příklad ukazuje jeden z možných způsobů, jak tyto soubory založit (předpokládáme, že programy jsou v paměti a že existuje platný MEM.SAV):
```

```

SELECT ITEM OR RETURN FOR MENU
K (RETURN)
SAVE--GIVE FILE,START,END(,INIT,RUN)
PART1.OBJ,2000,21FF,,2000 (RETURN)

SELECT ITEM OR RETURN FOR MENU
K (RETURN)
SAVE--GIVE FILE,START,END(,INIT,RUN)
PART2.OBJ,2200,23FF,,2200 (RETURN)

atd.

```

Všechny tři soubory, vyvořené stejným způsobem jako soubor PART1 lze spojit funkcí COPY nebo funkcí "K", obě s parametrem "/A". Předpokládejme, že jsou spojeny za sebou v pořadí PART1, PART2, PART3. Při zavádění výsledného souboru bude sled událostí následující:

1. PART1 se načte, ale nespustí (nemá INIT)
2. Adresa RUN pro PART1 se uloží do \$2E0 a \$2E1.
3. PART2 se načte, ale nespustí (nemá INIT)
4. Adresa RUN pro PART2 se uloží do \$2E0 a \$2E1 a přepíše adresu pro PART1.
5. PART3 se načte, ale nespustí (nemá INIT)
6. Adresa RUN pro PART3 se uloží do \$2E0 a \$2E1 a přepíše adresu pro PART2.
7. Protože jsme na konci souboru, spustí se program od platné adresy RUN, což je adresa pro PART3, která se uložila naposled.

Př. 4: Mějme třisegmentový soubor (segmenty definující adresy chápeme jako součást programového segmentu) BIGFIL.OBJ. Předpokládejme, že se každý segment ukládá do jiné části paměti a že:

```

SEG1.OBJ má adresu INIT a nemá RUN
SEG2.OBJ nemá ani RUN, ani INIT
SEG3.OBJ má svou INIT adresu, a RUN pro SEG2.OBJ.
pořadí segmentů v souboru je SEG1, SEG3 a SEG2.

```

Soubory spojíme do BIGFIL.OBJ. Při jeho člení se děje toto:

1. Načte se SEG1.OBJ
2. SEG1.OBJ se spustí od své adresy INIT.
3. Načte se SEG2.OBJ
4. SEG3.OBJ se načte za SEG1.OBJ.
5. SEG3.OBJ se spustí od své adresy INIT
6. SEG3.OBJ se spustí od adresy RUN, definované v SEG3.OBJ.

Př. 5: Konverze existujícího souboru typu "load" na typ "load-and-go", t.j. přidání definice startovacích adres:

Tuto akci lze provést načtením souboru do paměti a uložením pod novým jménem funkci "K". To ale může způsobit problémy, pokud jsme zapomněli adresy, na kterých je soubor v paměti uložen, nebo když je složen z více segmentů. Pokud se segmenty překrývají, t.j. předchozí segment je po vy-

konání přemazán následujícím, je tento postup zcela vyloučen. Pokud se segmenty sice nepřekrývají, ale neleží v paměti vedle sebe, je výsledný soubor delší, než původní. Proto je lepší zvolit postup, který ukáže příklad 5, kde přidáme k programu rozbehovou adresu RUN 4000 hexadecimálně a umožníme tím automatické spuštění. O programu nemusíme kromě startovací adresy nic vědět.

V příkladu se používá určitý trik. K souboru přidáme jednobytový segment umístěný do adresové oblasti ROM operačního systému se stejnou rozbehovou adresou, jako náš soubor. Protože přidaný jednobytový segment je ukládán do oblasti ROM, kterou nelze přepsat, nemůže se zavedením nadbytečného bytu nic porušit. Definovaná adresa RUN u nového segmentu je stejná, jako u programu a je zavedena nakonec. Proto platí pro celý soubor.

```
SELECT ITEM OR RETURN FOR MENU
K (RETURN)
SAVE--GIVE FILE,START,END,,INIT,RUN)
LOADFIL.OBJ/A,FF00,FF00,,4000
```

#### 4.1.12. Načítání binárních souborů - BINARY LOAD

**L**

Tato funkce umožňuje zavést do paměti binární soubor a pokud má definovanou některou z rozbehových adres, spustí jej po zavedení. Z toho vyplývá, že se musí jednat o program ve strojovém kódu. Jinými slovy, funkcí "L" lze spouštět strojové programy, nebo je jen zavádět do paměti podle volby parametru. Používá se ke spouštění služebních programů DOSu 2.5, které jsou popsány v kapitole 4.4. Spouštět lze jak soubory, vytvořené funkcí K, tak soubory vytvořené překladačem assembleru. Pokud má zaváděný soubor definovanou spouštěcí adresu RUN, nebo inicializační adresu INIT, provádí se při načítání podle těchto adres inicializace a spuštění. Podrobnosti viz 4.1.11. a 3.5. Pokud za jméno souboru uvedeme volbu "/N", soubor se zavede do paměti, ale adresy INIT i RUN se ignorují a program se nespustí. Můžeme jej spustit ručně funkcí "M", nebo jej ponechat v paměti a dále s ním pracovat.

Př. 1: Příklad ukazuje načtení programu do paměti s ignorováním spouštěcí a inicializační adresy:

```
SELECT ITEM OR RETURN FOR MENU
L (RETURN)
LOAD FROM WHAT FILE?
MYFILE.OBJ/N (RETURN)
```

Použití funkce "L" bez volby "/N" ukazuje příklad 2. Předpokládáme, že soubor má startovací adresu, proto se po načtení spustí:

Př. 2:   SELECT ITEM OR RETURN FOR MENU  
 L (RETURN)  
 LOAD FROM WHAT FILE?  
 BINFIL.OBJ (RETURN)

Ve třetím příkladu se předpokládá, že soubor nemá definovanou ani adresu

RUN, ani INIT. Proto se nerozběhne ani když nepoužijeme volbu "/N".

Př.3:     SELECT ITEM OR RETURN FOR MENU  
 L (RETURN)  
 LOAD FROM WHAT FILE?  
 NORUN.OBJ (RETURN)  
 načtení programu  
 SELECT ITEM OR RETURN FOR MENU

#### 4.1.13. Skok na adresu - RUN AT ADDRESS

**M**

Poznámka: Tato funkce není určena pro začátečníky.

Funkci M můžeme spustit jakýkoliv strojový program, nacházející se v paměti počítače, například načtený funkcí L. K použití funkce M je třeba vědět, kde začíná program, který spouštíme. Kromě toho lze v některých případech touto funkci restartovat program, z něhož jsme přešli do DOSu.

Př. 1: Odstartujeme program, který jsme předtím zavedli funkcí L. Program nemá startovací adresu a začíná na \$3000:

SELECT ITEM OR RETURN FOR MENU  
 M (RETURN)  
 RUN FROM WHAT ADDRESS?  
 3000 (RETURN)

Př. 2: V Turbo Basicu jsme přešli povelom DOS do menu. Abychom nemuseli celý Turbo Basic (má přes 140 sektorů) načítat znova, používáme si skokem:

SELECT ITEM OR RETURN FOR MENU  
 M (RETURN)  
 RUN FROM WHAT ADDRESS?  
 2080 (RETURN)

Pozor: Tuto fintu lze použít jen tehdy, když máme platný soubor MEM.SAV (viz 4.1.14). Nejpraktičejší je na i30XE, nebo na modelech s rozšířenou pamětí.

Př. 3: Ukazuje způsob vynucení studeného startu počítače:

SELECT ITEM OR RETURN FOR MENU  
 M (RETURN)  
 RUN FROM WHAT ADDRESS?  
 E477 (RETURN)

(následuje studený start, ekvivalent vypnutí a zapnutí počítače)

V menu DOSu takto ušetříme počítač proudových rázů při zapínání, pokud potřebujeme "přeládovat".

#### 4.1.14. Založení MEM.SAV - CREATE MEM.SAV

**N**

Tato funkce založí soubor MEM.SAV, do kterého se uschovává obsah dolní části uživatelské paměti při každém přechodu do menu DOSu. Při požadavku na přechod do DOSu (obvykle povelom DOS) se nejprve uloží obsah části

paměti, která bude později přemazána programem DUP.SYS. Do MEM.SAV se uloží momentální obsah RAM, ať již se jedná o program Basicu, nebo o strojový program. Po vykonání potřebných akcí v menu se při přechodu zpět nejprve ze souboru MEM.SAV obnoví obsah paměti na stav před voláním DOSu. To však platí jen potud, pokud obsah MEM.SAV nezneplatníme.

**MEM.SAV** se zneplatní, pokud při funkcích C, O a] odpovíme "Y" na dotaz:

TYPE "Y" IF OK TO USE PROGRAM AREA.  
CAUTION: A "Y" INVALIDATES MEM.SAV

Založení MEM.SAV probíhá takto:

```
SELECT ITEM OR RETURN FOR MENU  
N (RETURN)  
TYPE "Y" TO CREATE MEM.SAV  
Y (RETURN)
```

Pokud zvolíte tuto funkci, a na disketě již soubor MEM.SAV je, vypíše se o tom oznámení **MEM.SAV FILE ALREADY EXISTS.**

#### **Význam souboru MEM.SAV**

Do tohoto souboru se uschovává obsah paměti RAM. Soubor má délku 45 sektorů a musí být na disketě v jednotce číslo 1. (pozor: u modelu 130XE se automaticky zakládá MEM.SAV v RAM-disku D8). Když je tento soubor na disketi přítomen, ukládá se do něj obsah té oblasti paměti, kterou přepisuje DUP.SYS. Tím se provede vlastně výměna obsahu paměti. Když potom předáme řízení zpět do modulu, obsah paměti se opět obnoví z MEM.SAV, neboli se obsah "přepne" zpátky. Pokud pracujete v Basicu a potřebujete přejít do menu, nemusíte předtím ukládat program na disk. Po návratu zpět z menu se obsah paměti obnoví a program je neporušen.

**Poznámka:** MEM.SAV je nejúčinnější u modelů s instalovaným rozšířením paměti, nebo u 130XE, pokud se používá RAM-disk. V tomto případě se při instalaci RAM-disku MEM.SAV automaticky zakládá. V ostatních případech je rychlejší MEM.SAV nepoužívat, a před přechodem do DOSu si program vždy uložit a po návratu zase načíst.

#### **MEM.SAV při práci v Basicu:**

Jste v Basicu, na obrazovce svítí READY, na disku máme program PROG.BAS.

1. Napište LOAD"D:PROG.BAS" (RETURN).
2. Program upravte, a spusťte povelom RUN (RETURN). Program funguje. Nyží chcete původní program přejmenovat a nechat jej jako rezervní archivní soubor.
3. Přejdete do menu povelom DOS a (RETURN). Obsah paměti se předtím uloží do MEM.SAV.
4. Přejmenujte program povelom E (RETURN). Na dotaz odpovězte PROG.BAS,PROG.OLD.

5. Povolení B (RETURN) se vrátíte do Basicu. Předtím se obnoví obsah paměti z MEM.SAV, proto je program neporušen. Můžete na něm dále pracovat.
6. Po skončení práce program uložte povelení SAVE"D:PROG.BAS".

Při práci bez MEM.SAV bychom museli postup modifikovat:

- 2a. SAVE"D:PROG.NEW" (RETURN)
3. DOS (RETURN)
- 5a. LOAD"D:PROG.NEW" (RETURN)

#### **MEM.SAV a assembler**

Soubor MEM.SAV umožňuje psát programy, nebo načítat binární data, která sdílejí s DUP.SYS uživatelskou paměť. To znamená, že lze psát programy, nebo načítat data do oblasti od MEMLO (která se pohybuje podle závislosti na počtu disketových jednotek a počtu souborů, které mohou být současně otevřeny) do MEMTOP, která je ovlivněna použitým grafickým modelem.

Předpokládejme, že máme binární soubor a chceme, aby se po načtení automaticky spouštěl. Rozběhová adresa je již v programu obsažena, takže nepotřebujeme funkci M, abychom program spustili. V tomto případě také není nutný MEM.SAV. Program se zavede a spustí. Pokud chceme program ukončit a vrátit se do DOSu, je nejlépe provést instrukci RTS. Pokud nás program nepřepsal v paměti DUP.SYS, objeví se okamžitě menu. Pokud je DUP.SYS přepsán, zavede se znova a přepíše předtím zavedený program.

*Pozor:* Pokud program piše do adres pod MEMLO, může se počítač po provedení instrukce RTS nacházet v nedefinovaném stavu. Pokud se to stane, nezbývá než jej vypnout a znova zapnout.

#### **MEM.SAV a zavádění binárních dat**

Zde se podíváme na okolnosti zavádění binárních souborů, které se nespouštějí v okamžiku zavedení, nebo které jsou potřeba jako data pro jiný program. Pokud se zmíněné soubory nepřekrývají s DUP.SYS, není MEM.SAV zapotřebí. Pokud se jakákoli část souboru překrývá byl i částečně s DUP.SYS, je MEM.SAV nezbytný. Při funkci L s použitím MEM.SAV se dělí následující akce:

1. Funkcí L zvolíme čtení binárního souboru.
2. Původní MEM.SAV se nahraje do paměti a přepíše DUP.SYS.
3. Zvolený soubor se zavede do paměti, přičemž může částečně, nebo i úplně přepsát oblast naplněnou předtím z MEM.SAV.
4. Taktéž modifikovaná oblast paměti se nahraje do MEM.SAV na disketu.
5. DUP.SYS se znova nahraje z diskety do paměti.
6. V menu zůstáváme, dokud nezvolíme funkce:

**RUN CARTRIDGE**, kdy se obsah paměti obnoví z MEM.SAV a řízení přejde na příslušný modul.

**RUN AT ADDRESS**, kdy se obsah paměti obnoví z MEM.SAV a řízení se předá na zadanou adresu

**BINARY LOAD**, kdy chceme zavést automaticky spouštěný soubor. V tomto případě, pokud nový soubor sice také přepisuje část DUP.SYS, ale nepřepisuje dříve zavedený soubor, bude nyní po dokončeném čtení v paměti jak původně zavedený soubor (z MEM.SAV), tak soubor nový. Pokud ale nový soubor vůbec nekoliduje s DUP.SYS, zavede se jen nový a obsah MEM.SAV se nenačte. Protože se nový program spustí automaticky, přiveze řízení do doby, než se provede instrukce RTS, která vrátí řízení DOSu.

**Poznámka:** Pokud chceme mít v paměti dva programy, z nichž jeden je celý umístěn v oblasti DUP.SYS a druhý celý mimo tuto oblast, je nutno spojit oba dohromady a zavést nový takto vzniklý soubor.

#### 4.1.15. Duplikování souborů : DUPLICATE FILE

Tato funkce umožňuje kopírovat soubory z jedné diskety na jinou s použitím jedné disketové jednotky, která musí být nastavena na číslo 1. Protože se ke kopírování používá jen jedna jednotka, je třeba na pokyny operačního systému vyměňovat zdrojovou a cílovou disketu. Pokud je kopírovaný soubor velký, je třeba vyměnit diskety i vícekrát. Podrobnosti viz 2.2.12. Místo jména lze zadat i filtr pro výběr skupiny souborů. V tomto případě se vypisuje jméno každého kopírovaného souboru a při každém je třeba minimálně jednou vyměnit diskety.

Příklad:

```

SELECT ITEM OR RETURN FOR MENU
O (RETURN)
NAME OF FILE TO MOVE?
PROG.BAS (RETURN)
INSERT SOURCE DISK, TYPE RETURN
(CRETURN)
INSERT DESTINATION DISK, TYPE RETURN
(CRETURN)
```

Příklad 2: Kopírujeme soubory začínající na TEST. Předpokládáme, že takové soubory jsou na disketě uloženy dva.

```

SELECT ITEM OR RETURN FOR MENU
O (RETURN)
NAME OF FILE TO MOVE?
TEST? (RETURN)
INSERT SOURCE DISK, TYPE RETURN
(CRETURN)

COPYING__D1:TEST!
INSERT DESTINATION DISK, TYPE RETURN
(CRETURN)

INSERT SOURCE DISK, TYPE RETURN
(CRETURN)
```

```

COPYING--D1:TEST2
INSERT DESTINATION DISK, TYPE RETURN
(RETURN)

INSERT SOURCE DISK, TYPE RETURN
(RETURN)

```

Př. 3: ukazuje kopírování všech souborů. DOS v tomto případě kopíruje všechny soubory kromě těch, které mají koncovku .SYS. V tomto příkladu předpokládáme, že jsou na disketě tři soubory, a sice TEST1, TEST2 a MEM.SAV.

poznámka: Založit MEM.SAV na nové disketě funkci N je rychlejší, než ho kopírovat.

```

SELECT ITEM OR RETURN FOR MENU
O (RETURN)
NAME OF FILE TO MOVE?
*.* (RETURN)
INSERT SOURCE DISK, TYPE RETURN
(RETURN)

COPYING--D1:MEM.SAV
INSERT DESTINATION DISK, TYPE RETURN
(RETURN)

INSERT SOURCE DISK, TYPE RETURN
(RETURN)

COPYING--D1:TEST1
INSERT DESTINATION DISK, TYPE RETURN
(RETURN)

INSERT SOURCE DISK, TYPE RETURN
(RETURN)

COPYING--D1:TEST2
INSERT SOURCE DISK, TYPE RETURN

```

Pozn.: Kopírování větších skupin souborů je efektivnější s využitím vhodných programových prostředků, jako Filecopy nebo Multifile Copy, u kterých není nutno vyměňovat diskety po každém souboru.

#### 4.1.16. Formátování SD - FORMAT SINGLE

P

Tuto funkci použijeme, pokud potřebujeme naformátovat disketu v jednoduché hustotě (SD), ať již kvůli přenosu souborů do jiného DOSu, nebo proto, že disketa nejde formátovat funkcí I. Průběh komunikace s počítačem je zcela shodný, jako u funkce "I". Pamatujte, že formátování diskety smaže cokoliv, co na ní bylo předtím nahráno.

## **4.2. Basic a DOS 2.5**

V části věnované Basicu předpokládáme znalost základů programování alespoň v rozsahu základní příručky. Nejprve probereme dva způsoby ukládání programů na disketu a povely, kterými lze obě formy dosahovat. Při popisu syntaxe povelů budeme používat běžné konvence. Text, který se musí doslově napsat je uveden velkými písmeny. Parametry, do nichž se dosazuje nějaká hodnota, ať již textová, nebo číselná, se psí malými písmeny. Část výrazu, uzavřená v hranatých závorkách [] se nemusí uvádět a lze ji vynechat. Část ve složených závorkách označuje seznam možností, z nichž musí být jedna uvedena. Jednotlivé možnosti jsou odděleny svislou čarou. Pokud má být uvedeno jedno z písmen A, B, nebo C, napíše se to takto: { A | B | C }.

### **4.2.1. Čtení a ukládání programů**

Ke čtení a ukládání programů na kazetu se používá téměř výlučně povel CLOAD a CSAVE. Při práci s disketovou jednotkou se používají jiné povelky, a to LOAD, SAVE, ENTER a LIST. Tyto dvojice povelů souvisí se způsobem kódování programu při ukládání. První dvojice vytváří a čte tzv. tokenizovaný soubor, druhá dvojice pracuje s formou textovou. Rozdíl mezi oběma formami lze snadno demonstrovat. Uložte si stejný program povelem SAVE a povellem LIST, pokaždé pod jiným jménem a zkopírujte si je na obrázku povellem C. Zatímco program uložený LISTem je normálně čitelný, program uložený SAVE je změl nesrozumitelných znaků. Patrně Vám neunikne, že druhý soubor je kratší. Co je smyslem těchto formátů si nyní řekneme.

#### **Textová forma uložení programu**

vytváří se povellem LIST. Jedná se o čitelnou formu programu. Jejimi základními rysy je možnost spojovat a zatížovat do sebe různé programy a možnost uložení a zpětným načtením vyčistit tabulku proměnných. Nevhodou je delší načítání a větší délka uloženého programu. Načítání programů v textové formě se provádí povellem ENTER.

#### **Tokenizovaná forma**

je nejúspornějším způsobem uložení programu. Je téměř shodná se způsobem uložení programu v paměti. Vytváří se již při ukládání programových řádků, ať již ručně z klávesnice, nebo při načítání textové formy povellem ENTER. Po uložení každé řádky a stisknutí (RETURN) proběhne tokenizační pochod, při němž se každé klíčové slovo Basicu (povel, deklarace) převede na číslo. Stejně tak se na číslo převeďe proměnná, nebo konstanta v parametru příkazu. Jméno proměnné se uloží do tabulky proměnných a v příkazu se nahradí číslem, které udává umístění jména v tabulce. Kromě toho se také uloží údaje o délce řádky a o její délce. Například řádka:

10 LET SOURADX-1:PRINT SOURADX

vypadá po tokenizačním procesu třeba takto:

0A 00 13 0F 06 80 2D 0E 40 01 00 00 00 00 14 13 20 80 16

Je zřejmé, že ve většině případů se tímto způsobem uložení šetří jak místo na záznamovém médiu, tak čas při čtení a ukládání programu, protože se již neprovádí žádná konverze. Nevýhodou je, že se každá proměnná, která byla jednou použita uloží do tabulky a zůstává v ní, i když se třeba zase z programu vymaže a nepoužívá se. Kromě toho je u Atari Basicu verze B chyba, při níž se vždy s programem ukládá 16 byte zbytečného "šumru", který ubírá místo v paměti i na disku. Oba tyto problémy lze řešit uložením programu v textové formě, vymazáním z paměti povelom NEW a opětovným načtením. Pro všechny následující povely platí, že ve specifikaci souboru je až na výslovně uvedené výjimky nutno povinně uvádět kód zařízení. Pokud u disketové jednotky nenapišeme číslo, doplní se číslo 1.

#### povel LOAD (LO.)

formát:           LOAD specifikace souboru  
příklad:          LOAD "D:PROGRAM.BAS"  
                      LOAD "D8:PROG2.BAS"

Tento povel slouží k načtení programu v tokenizované formě. Program musí být předtím uložen povelom SAVE. Po načtení do paměti lze program editovat, nebo jej spustit povelom RUN. Předchozí obsah paměti se vymaže, proto nelze spojovat dohromady programy uložené v tokenizované formě. Ve specifikaci lze použít filtr. V tom případě se načte první nalezený soubor. Povel LOAD má velmi nebezpečnou zkratku "LO.", která se dá bez velké námahy splést se zkratkou "L." poveli LIST. Záměna této zkratky způsobí, že místo načtení přepíšeme program momentálním obsahem paměti. Obvykle se tento omyl stává na začátku práce kdy je paměť prázdná. Místo programu tak máme prázdný soubor a můžeme začít tvrcit znova. Obranou je kromě opatrnosti blokování důležitých programů funkcí F v DOSu.

#### povel SAVE (S.)

formát:           SAVE specifikace souboru  
příklad:          SAVE "D:PROGRAM.BAS"

Ukládá program v paměti na disketu v tokenizované formě pod jménem, uvedeným ve specifikaci souboru. Povelom SAVE lze uložit i na kazetu. Filtr ve specifikaci souboru není povolen. Program se uloží celý.

#### povel ENTER (E.)

formát:           ENTER specifikace souboru  
příklad:          ENTER "D:PROGRAM.LST"

Tento povel načte program jména určeného specifikací souboru do paměti. Program musí být uložen v textové formě, tedy povelom LIST. Načítání probíhá pomaleji, protože se každá načtená řádku tokenizuje, jako by byla napsá-

na přes klávesnici a program se po načtení nachází v počátečním stavu. Proto lze takto opravovat "nabourané" programy, které sice mají text v pořádku, ale nefungují. Programy v textové formě lze povelem ENTER také spojovat. Přitom se nově načítané řádky začlení podle čísel mezi řádky již přítomné. Nově načtená řádka přepíše řádku stejného čísla, která je již v paměti. Vídeť, že zákonitosti jsou zcela stejné, jako při psaní programu na klávesnici. Tuto vlastnost lze použít třeba k mazání velkého počtu řádek v Atari Basicu, který nemá žádný jiný povel na mazání, než napsání samotného čísla řádky. Rekneme, že je třeba vymazat řádky 2000 až 2400. Zapíšeme si do souboru čísla 2000 až 2400 s přirůstkem po jedné, a povelem ENTER tento soubor načteme. Řádky v uvedeném rozsahu se vymažou.

#### povel LIST (L.)

formát: LIST [{ specifikace souboru[,číslo1[,číslo2]]} | číslo1[,číslo2]]]

- kde: - specifikace souboru je úplné jméno, včetně kódu zařízení. Filtr není povolen.
- číslo1 je číslo řádky, od které se bude program ukládat.
- číslo2 je číslo řádky, po kterou se bude ukládat.

příklady:

- LIST "D:PROGRAM.LST"
- LIST "P;"
- LIST "P:",10,100

Povel LIST je všem uživatelům dobře znám, zde však je v nových souvislostech. Při nejobvyklejším použití pro výpis na obrazovku se používá standardní specifikace souboru "E;". Pokud se neuvede číslo1 a číslo2, vypisuje se program celý. Výpis má textovou formu. List je jediný povel, kterým lze uložit jen část programu.

#### povel RUN

formát: RUN [specifikace souboru]

příklad: RUN "D:PROGRAM.BAS"

Povel RUN načte do paměti program v tokenizované formě, stejně jako povel LOAD, a po načtení jej spustí. Pokud není uvedena specifikace souboru, spustí program, který se právě nachází v paměti. Načtený program nahradí v paměti předchozí. Povelem RUN lze realizovat řetězení programů, například v podobě menu. Na základě volby uživatele se může vyvolat program, realizující zvolenou funkci. Vyvolávaný program musí být zcela nezávislý na volajícím programu, neboť jej při načítání zcela přemáže. Po spinění své funkce může opět vyvolat menü.

#### 4.2.2. Řídící bloky vstupu a výstupu - IOCB

Veškeré vstupní a výstupní operace s periferními zařízeními řídí takzvané řídící bloky vstupu a výstupu - Input/Output Control Blocks neboli IOCB.

Každý z osmi vstupních/výstupních kanálů je řízen jedním řídícím blokem. Řídící blok - IOCB je tabulka, do které je třeba před spuštěním operace vstupu/výstupu zaznamenat potřebné informace a parametry, jako je druh operace, umístění a délka paměti, která se ukládá nebo kam se čte, doplňkové řídící parametry AUX1, AUX2, apod. Při programování v Basicu za nás jeho interpret většinu parametrů vyplní sám, uživatel určuje jen některé. Z osmi kanálů máme k dispozici jen 7, protože kanál 0 je vyhrazen operačním systémem pro obrazovkový editor. Kromě toho Basic používá kanál 6 pro grafický výstup na obrazovku a kanál 7 pro operace při povelích LPRINT, LOAD, SAVE, CLOAD a pod. To sice neznamená, že by uživatel tento kanál nemohl použít na něco jiného, ale je třeba mít na zřeteli, že Basic ho při jakémkoliv chybě zavírá (viz CLOSE). Počáteční určení těchto tří kanálů ukazuje tabulka:

<u>kanál</u>	<u>zařízení</u>	<u>kdy je používán</u>
0	E:	stále
1	S:	
7	různé	povely LPRINT, LOAD, CLOAD SAVE apod.

Ostatní kanály, tedy 1 až 5 jsou volně použitelné. To bohatě stačí, protože DOS povoluje při standardní konfiguraci maximálně tři současně otevřené soubory a kromě toho můžeme mít otevřena ještě dvě nedisková zařízení.

*Pozor:* Basicovský program **nemůže** otevřít, nebo zavírat kanál 0!

Každý povel vstupu a výstupu v Basicu musí mít určen kanál, kterého se týká. Povely, které lze používat s DOS 2.5 jsou:

```
OPEN/CLOSE
INPUT/PRINT
PUT/GET
STATUS
XIO
```

#### 4.2.3. Povely pro ovlivňání a zavírání souborů                    OPEN/CLOSE

Než se začne pracovat s nějakým souborem je potřeba určit jeho jméno, na kterém zařízení se nachází, zda bude vstupovat nebo vystupovat a další parametry. Tomuto přiřazení se říká otevření souboru a provádí se povelom OPEN. Po skončení práce je zvláště u výstupních souborů nutné provést tzv. zavření souboru. Při tom se provedou operace které jsou nutné k tomu, aby byl soubor platný a použitelný. Vstupní soubory sice není nutné zavírat, ale patří k dobrému programátorskému stylu. Při práci se souborem má operační systém pro každý otevřený soubor čítač polohy, který se při otevření nastaví na začátek, nebo na konec, když jde o připojování nových dat do stávajícího souboru. Každá operace vstupu nebo výstupu ovlivňuje čítač tak, aby vždy ukazoval na místo bytu, který se bude při této operaci číst, nebo kam se bude při této operaci zapisovat. Na tyto změny nemá uživatel vliv. Záměrně lze hodnotu čítače měnit příkazy POINT a zjistit ji lze příkazem NOTE.

povel OPEN (O.)

formát:           OPEN #výraz1,výraz2,výraz3,string

kde:

- znak # je třeba uvést
- výraz1 je aritmetický výraz, nebo číslo určující číslo kanálu se kterým bude otvíraný soubor spojen
- výraz2 je aritmetický výraz nebo číslo, určující hodnotu AUX1. Tato proměnná určuje druh prováděné operace:
  - 4 - vstupní operace. Čítač souboru se nastaví na jeho začátek.
  - 6 - čtení adresáře diskety, pouze soubory z oblasti jednoduché hustoty (SD, DOS 2.0)
  - 7 - čtení adresáře diskety, soubory z oblasti rozšířené hustoty (ED, DOS 2.5)
  - 8 - výstup, nastaví čítač souboru na začátek
  - 9 - append, připojování dat na konec souboru. Čítač se nastaví na konec souboru.  
AUX2-9 znamená u E: nucené čtení z obrazovky bez pomocí klávesy (RETURN).
  - 12 - vstup i výstup. Tomuto režimu se říká UPDATE. Používá se hlavně u E:, ale umožňuje i aktualizaci disketového souboru přepisováním vět. Čítač se nastaví na začátek souboru.
- výraz2 určuje hodnotu AUX2. Její význam se mění podle druhu zařízení. Nejčastěji bývá nulová.
- string je stringová proměnná nebo konstanta, která obsahuje v okamžiku provádění povelení specifikaci souboru včetně jména zařízení. Filtr je povolen jen u vstupu, tj. AUX1-4. V tom případě se otevře první filtrem vybraný soubor. (specifikace souboru viz 2.2.10).

příklad:           OPEN #2,4,0,"D8:SOUBOR.DAT"

V příkladu se otvírá soubor "SOUBOR.DAT", uložený na RAM-disku D8. Soubor se bude číst přes kanál 2. Pokud se otvírá soubor pro zápis, a na disketu je již soubor stejného jména, bude starý soubor přepsán novým. Pokud je starý soubor zamknut, nastane chyba 167. Systém dovoluje otevřít pro stejný soubor a stejné zařízení i více kanálů současně. Ve většině případů to nemá žádný smysl a může tozpůsobit "pomatení" dat a různé jiné chyby. Jediný případ, kdy může mít smysl otevřít současně dva kanály pro tentýž soubor, je aktualizace disketového souboru přímým přepisováním vět. Oba kanály se otevřou s AUX1-12 a jeden se používá pro čtení a jeden pro zápis. Pokud by se pro čtení i zápis používal stejný kanál, bylo by nutno pro každý zápis opravovat čítač polohy v souboru. Jak bylo napsáno dříve, každá vstupní a výstupní operace aktualizuje hodnotu čítače tak, aby ukazovala za právě zpracované místo. Po otevření ukazuje čítač na začátek souboru. Nyní přečte-

me větu a chceme ji po nějaké změně obsahu zapsat na totéž místo. Po načtení věty ale čítač ukazuje na začátek druhé věty. Pokud tedy větu uložíme takto, přepíše nové znění první věty větu druhou, o kterou tím přijdeme. Proto je nutno buď použít pro zápis jiný kanál, než pro čtení, nebo každý kanál má samostatný čítač, nebo si poznamenat při čtení polohu věty povelom NOTE a před zápisem ji nastavit povelom POINT. Druhé metodě tím pádem stačí jen jeden kanál a je vhodnější tam, kde se nečte celý soubor. Pokud je kanál již otevřen, nové otevření se ignoruje a nastane chyba 129.

#### povel CLOSE (CL.)

formát:           CLOSE # výraz

kde hodnota výrazu určuje číslo kanálu, který se zavírá.

příklad:           CLOSE #2

Povelom CLOSE se uzavře určený kanál. Při tom se u výstupního souboru zapíše zbytek vyrovnávacích paměti a vytvoří se koncový sektor souboru, nebo v případě kazety se zapiší koncové bloky. Poslední datový blok, nebo poslední sektor souboru bývá krátký (viz 3.3.2). Status souboru na disketu se z \$43 změní na \$42, nebo na \$03 (viz 3.4.2). Zavírání souborů otevřených pro jakoukoliv formu výstupu je bezpodmínečně nutné, aby bylo možno s vytvořeným souborem dále pracovat. Vstupní soubory sice není nutno zavírat, patří to ale dobrému programátorskému stylu.

Poznámky: - příkaz END zavírá vždycky všechny kanály kromě kanálu 0  
                   - všechny otevřené kanály se zavírají při stisku klávesy RESET  
                   - kanál 7 je interpretem Basicu zavřen při jakékoli chybě.

#### 4.2.4. Čtení a zápis vět

INPUT/PRINT

Větou se rozumí jakákoli posloupnost bytů, ukončená znakem EOL (ATASCII 155, \$9B). Délka věty je omezena na 255 znaků. Na to musíme dát pozor při zápisu, protože příkaz PRINT nehlásí překročení délky. Ke čtení libovolně dlouhého proudu dat je nutno použít cyklu s příkazem GET. Povely INPUT a PRINT slouží ke všeobecnému vstupu a výstupu vět. Veškerá data jsou v textovém formátu, včetně čísel. Při vstupu se znakově zadaná čísla převádějí podle typu proměnné do které se ukládají buď do řestibytevého formátu čísel s pohyblivou řádovou čárkou, nebo zůstávají nezměněna, pokud jde o string. Při zápisu souborů, které se budou později číst příkazem INPUT se musí uživatel sám postarat o doplnění správných oddělovačů jednotlivých položek. Mezi údaje je nutno vložit čárky, aby se při vstupu mohlo rozpoznat, kde jednotlivé údaje končí.

**povel INPUT (I.)**

formát:            INPUT [ #iocb(.,.) ] (avar|svar) [ ,(avar|svar)...]

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- avar je aritmetická proměnná, výraz nebo konstanta
- svar je stringová proměnná nebo konstanta
- tečky znamenají, že se skupina může libovolněkrát opakovat

příklad:        100 INPUT #2; X,Y  
                  100 INPUT #1; N\$

Standardní číslo kanálu je 0, proto pokud číslo neuvedeme, bude se číst rádek z obrazovkového editoru "E:". Povel INPUT čte datové věty, ukončené znakem EOL. Maximální délka je 255 byte.

**povel PRINT (PR. nebo ?)**

formát:        {PRINT|?} [ #iocb(.,.) ] [ {string|exp}[[ .,){string|exp}].... ]]

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- string je stringová proměnná nebo konstanta
- exp je aritmetický výraz, proměnná nebo konstanta

příklad:        100 PRINT #2;X,Y  
                  100 ? #2; A\$  
                  100 ? C\$  
                  100 ? "X - ",X  
                  100 ? #2;X,Y

Pokud se neuvede kanál, směřuje výstup na standardní zařízení "E:". Důležitou roli hraje oddělující znaky dvojtečka a středník. Povšimněte si výsledků prvního a posledního příkladu. Čárka znamená vložení počtu mezer, odpovídajícímu nastavení tabelátoru. PRINT může zapisovat věty libovolně dlouhé a nicím nesignalizuje překročení délky 255, kterou může ještě zpracovat INPUT. Proto je na uživateli, aby si délku zapisované věty pohlidal, pokud chce soubor později příkazem INPUT čist.

### Komplexní příklad INPUT/PRINT

```

5 REM ** ZALOZIME DATOVY SOUBOR **
7 REM ** OTEVREME SOUBOR PRO VYSTUP **
10 OPEN #1, 8, 0, "D: SOUBOR.DAT"
20 DIM A$(40)
30 ? "NAPISTE VETU DLOUHOU MAXIMALNE 40 ZNAKU:"
40 INPUT A$
50 ? #1, A$
60 ? "TED NAPISTE TRI LIBOVOLNA CISLA, ODDELENA CARKOU"
70 INPUT A,B,C
75 REM * ZAPIS CISEL, JE NUTNO VLOZIT CARKY *
76 REM * PRO POZDEJSI CTENI *
80 ? #1,A,".",B,".",C
90 REM ** ZAVREME SOUBOR **
100 CLOSE #1
110 REM ** A TED TO PO SOBE PRECTEME **
111 REM ** OTEVREME SOUBOR PRO VSTUP **
120 OPEN #1, 4, 0, "D:SOUBOR.DAT"
130 INPUT #1,A$
140 ? A$
150 INPUT #1, A, B, C
160 ? A, B, C
170 REM ** A ZAVREME SOUBOR **
180 CLOSE #1
190 END

```

#### 4.2.5. Přímý přístup do souboru

#### NOTE/POINT

Normálně jsou soubory čteny a zapisovány sekvenčně (v logickém smyslu) od začátku do konce. Protože diskové paměti mají možnost číst a zapisovat data na kterékoliv místo záznamového povrchu, lze u diskových souborů číst a psát i jiným, než sekvenčním postupem. Pomoci příkazu NOTE a POINT lze jednak zjistit adresu libovolného místa souboru, jednak na ni čítač souboru kdykoli nastavit. Toho lze využívat ke zrychlení práce s velkými soubory dat, u nichž potřebujeme pracovat jen s některými větami, a u nichž můžeme polohu požadované věty určit. Nejlépe je metoda přímého přístupu vidět na příkladech.

#### povel NOTE (NO.)

formát:            NOTE #iocb, avar1, avar2

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- avar1 je proměnná, do níž funkce uloží číslo sektoru
- avar2 je proměnná, do níž funkce vloží číslo byte

příklad:        100 NOTE #1,A,B

Povel NOTE zjistí momentální nastavení čítače polohy v souboru, pro který je otevřen zadáný kanál. Čítač se skládá z čísla sektoru a čísla byte v sektoru, a ukazuje na byte, který se bude při příštím povale vstupu/výstupu zpracovávat. Proto musíme polohu věty zjišťovat před čtením nebo zápisem. Číslo sektoru může nabývat hodnot 1 až 719 u jednoduché a 1 až 1023 u rozšířené hustoty, číslo byte 0 až 124. Následující program nám ukáže, jak si můžeme poznámenat polohy jednotlivých vět v datovém souboru. Použití získaných informací je uvedeno u povelu POINT.

```

1 REM ** NOTEST - UKAZKA PRIKAZU NOTE **
2 REM PROGRAM CTE DATA Z KLAVESNICE A UKLADA
3 REM JE DO SOUBORU "D:RADKY.DAT". POLOHU JEDNOTLI-
4 REM VYCH VET POZNAMENAVA DO SOUBORU
5 REM "D:RADKY.IDX"
10 DIM A$(40)
20 OPEN #1, 8, 0, "D:RADKY.DAT"
30 OPEN #2, 8, 0, "D:RADKY.IDX"
40 REM ** PRECTEME RADKU Z KLAVESNICE **
50 INPUT A$
60 REM SAMOTNY RETURN - KONEC
70 IF LEN (A$)=0 THEN 300
80 NOTE #1,SEK,BYT
90 ? #1, A$
100 REM NYNI ULOZIME POLOHU ZAPSANE VETY
110 ? #2,SEK, ",",BYT
120 ? "SEKTOR - ",SEK,"BYTE - ",BYT
130 GOTO 50
200 REM INDIKACE KONCE SOUBORU
300 ? #2,0, ",",0
310 CLOSE #1 : CLOSE #2
320 END

```

Při běhu programu se vypisují polohy jednotlivých vět na obrazovku. I když na sebe většinou čísla sektorů navazuji, není to nikdy zaručeno. Obsazování sektorů při vytváření souborů dělá DOS tak, že hledá v tabulce VTOC volné sektory od nejnižších čísel a postupně je přiděluje zapisovanému souboru. Když je disketa prázdná, přiděluje sektory popořadě. Když se však potom některý soubor smaže, vznikají díry, které se DOS snaží při zápisu dalšího souboru zaplnit. Pokud je nový soubor delší, než neobsazená "díra", dostane nespojitou oblast. V extrémním případě nemusí žádný sektor souboru sousedit s jiným. Jediné co lze zaručit je, že čísla sektorů tvoří vzestupnou posloupnost. Je pochopitelné, že doba zpracování rozseparovaného souboru je delší.

## povel POINT (PO.)

formát: POINT #iocb, avar1, avar2

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- avar1 je proměnná nebo výraz, určující číslo sektoru
- avar2 je proměnná nebo výraz, určující číslo byte v sektoru

příklad: 100 POINT #1,A,B

Povel POINT doplňuje příkaz NOTE. Lze jím nastavit čítač souboru na zadанou hodnotu. Může se používat jen při čtení. Číslo byte a číslo sektoru je stejně, jako u povelu NOTE. Pokud se pokusíme nastavit čítač mimo otevřený soubor, nastane chyba 164. Ukázkový program čte soubor RADKY.DAT z předchozího příkladu a s použitím poloh vět ze souboru RADKY.IDX vypisuje věty v opačném pořadí.

pozn: I když jsou hodnoty sektoru a byte vstupní, nesmí to být konstanty. Atari Basic připouští jen proměnnou nebo výraz.

```
1 REM POINTEST - UKAZKA PRIKAZU POINT
2 REM PROGRAM PISE NA OBRAZOVKU VETY
3 REM ZE SOUBORU "D:RADKY.DAT" V OPACNEM
4 REM PORADI. POLOHU VET URCUJE ZE
5 REM SOUBORU "D:RADKY.IDX"
10 DIM B(20,1), A$(40)
20 REM ** OTEVRENI SOUBORU **
30 OPEN #1, 4, 0, "D:RADKY.DAT"
40 OPEN #2, 4, 0, "D:RADKY.IDX"
50 NACTENI POLOH VET DO MATICE
60 FOR I= 0 TO 20
70 INPUT #2,SEC,BYT
80 B(I,0)= SEC: B(I,1)= BYT
90 IF SEC=0 THEN LAST=I:I=20
100 NEXT I
110 REM * TISK SOUBORU V OPACNEM PORADI *
120 FOR I-LAST-1 TO 0 STEP -1
130 SEC=B(I,0):BYT=B(I,1)
140 POINT #1,SEC,BYT
150 ? "SEKTOR = ",SEC,"BYTE = ",BYT
160 INPUT #1,A$
170 ? A$
180 NEXT I
190 CLOSE #1:CLOSE #2
```

Při testování konce souboru stačí dotaz na číslo sektoru 0, protože takový sektor neexistuje. Na číslo byte již není třeba se ptát.

#### 4.2.6. Vstup a výstup po jednotlivých bytech PUT/GET

Povely PUT a GET umožňují načtení nebo zápis jednotlivých bytů bez ohledu na logickou strukturu souboru. Znak EOL zde nemá zvláštní význam.

##### povel PUT (PU.)

formát: PUT #iocb, aexp

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- exp je proměnná, výraz nebo konstanta v intervalu <0,255>

příklad: PUT #6,ASC("A")

Byte, jehož hodnota je určena hodnotou výrazu exp, se zapíše do souboru, pro nejž je otevřen kanál iocb.

##### povel GET (GE.)

formát: GET #iocb, avar

kde:

- iocb je výraz, proměnná nebo konstanta určující číslo kanálu, se kterým se bude pracovat
- exp je proměnná, do níž se uloží načtená hodnota v intervalu <0,255>

Ze souboru, pro který je otevřen kanál iocb se přečte byte a jeho hodnota se uloží do proměnné avar.

Použití poveliù GET a PUT je zejména při různých manipulacích se soubory. Ukázkový příklad slouží k přidání hlavičky segmentu binárního souboru k datům znakové sady, vytvořené například programem FONTMAKER. Chceme, aby se znaková sada zaváděla od adresy 2000 hexadecimálně. Znaková sada je v souboru CHARSET a má délku 128 \* 8 byte.

```

10 REM ADRTAM - PROGRAM PRO PRIPOJENI
20 REM ADRESY SEGMENTU K DATOVEMU
30 REM SOUBORU
40 OPEN #1, 4, 0, "D:CHARSET"
50 OPEN #2, 8, 0, "D:CHARSET.OBJ"
60 PUT #2, 255: PUT #2, 255:REM $FFFF
70 PUT #2, 0: PUT #2, 32:REM $2000
80 PUT #2, 255 : PUT #2, 36:REM $23FF
90 REM ** NYNI KOPIE SOUBORU **
100 FOR I=1 TO 1023
110 GET #1,A:PUT #2,A
120 NEXT I
130 CLOSE #1:CLOSE #2
140 END

```

#### **4.2.7. Status kanálu STATUS**

**povel STATUS (ST.)**

formát:            **STATUS #ioch, avar**

kde:

- ioch je výraz, proměnná nebo konstanta určující číslo kanálu, jehož stav zjišťujeme.
- avar je proměnná, do níž se status uloží.

příklad:

**100 STATUS #5, ERROR**

Povel STATUS umožňuje zjišťovat stav otevřeného souboru. Jeho použitelnost je mizivá, protože před zjištěním stavu je nutno soubor otevřít. Praktičtější je volat jej funkci XIO, kdy se můžeme ptát na stav souboru, aniž by bylo nutno jej otevřít. Proměnná povelu může nabývat těchto hodnot (viz též kap. 12):

chyby spojené se souborem:

- 160 neplatné číslo zařízení
- 161 příliš mnoho otevřených souborů
- 170 neplatné jméno, nebo nenalezený soubor
- 167 soubor blokován proti zápisu

chyby přenosu:

- 138 zařízení neodpovídá - nepřipojeno
- 139 zařízení odpovídá chybě
- 140 chyba sériové sběrnice
- 142 DATA OVERRUN
- 143 chybny kontrolní součet
- 144 sektor nečitelný / nezapsatelný

#### **4.2.8. Náhrada povelů DOS menu funkcí XIO**

Pomocí funkce XIO lze substituovat některé povely z menu DOSu. Jsou zvláště užitečné pro majitele počítačů bez RAM-disku, kdy není přechod mezi Basicem a DOSem nijak rychlý. Proto je výhodné některé operace jako blokování, uvolnění a podobně dělat funkci XIO bez přecházení do DOSu a zpět.

## povel XIO (X.)

formát: XIO povel, #ioch, aexp1, aexp2, string

kde:

- povel je číslo povelu, který se má funkci XIO vykonat
- ioch je výraz, proměnná nebo konstanta určující číslo kanálu, s nímž budeme pracovat
- aexp1 je výraz, proměnná nebo konstanta určující hodnotu AUX1
- aexp2 je výraz, proměnná nebo konstanta určující hodnotu AUX2
- string je stringová proměnná nebo konstanta obsahující specifikaci souboru

příklad: 100 XIO 13, #1,0,0,"D:TEST.BAS"

Číslo povelu určuje druh provedené operace, AUX1 a AUX2 jsou většinou nulové. Funkce XIO je všeobecný povel pro operace s periferním zařízením. Lze jím provádět i dříve popsané povely, ale u některých to nemá smysl, protože se musí psát vždycky specifikace souboru. Přehled funkcí s příklady je v tabulce:

číslo povelu	operace	příklad
13	STATUS	XIO 13,#1,0,0,"D:TEST.BAS"
32	RENAME	XIO 32,#1,0,0,"D:OLD,NEW"
33	DELETE	XIO 33,#1,0,0,"D:TEMP.BAS"
35	LOCK FILE	XIO 35,#1,0,0,"D:PROG.BAS"
36	UNLOCK	XIO 36,#1,0,0,"D:PROG.BAS"
253	FORMATS	XIO 253,#1,0,0,"D1:"
254	FORMATE	XIO 254,#1,0,0,"D1:"

Poznámka: U funkce 32 se nesmí uvadět dvakrát kód zafizení. FORMATS formátuje v jednoduché hustotě, FORMATE v rozšířené. LOCK je blokování, UNLOCK uvolnění souboru a RENAME přejmenování.

### 4.3. Různé informace

V tomto odstavci jsou nashromážděny různé drobné informace, které mohou být občas užitečné. Jsou zde uvedeny způsoby jak modifikovat různé funkce DOSu, opravovat poškozené soubory a podobné.

#### 4.3.1. Opravování poškozených souborů

Soubory mohou být poškozeny dvojím způsobem. Buď může být narušena věta adresáře, popisující dotyčný soubor (viz 3.4.2), nebo soubor sám. Pokud je narušen adresář, je oprava téměř nemožná. Vlastní soubor se může porušit buď náhodným zrušením, nebo pomícháním sektoriů, které vede k chybě 164, FILE NUMBER MISMATCH. Náhodně zrušený soubor lze vrátit do užívání

programem DISKFIX.COM (viz 4.4.2.). Soubor s pomíchanými sektory, nebo soubor postižený defektem diskety již nelze zachránit celý, je ale možné uchovat alespoň část od začátku do místa výskytu chyby pomocí triviálního programku:

```

10 OPEN #1,4,0,"D:SOUBOR1"
20 OPEN #2,8,0,"D:SOUBOR2"
30 TRAP 60
40 GET #1,A
50 PUT #2,A
60 GOTO 40
70 CLOSE #1
80 CLOSE #2
90 END

```

Program kopíruje SOUBOR1 na SOUBOR2, dokud nenastane jakákoli chyba. Potom oba soubory uzavře a kopie skončí. V SOUBORu2 je zdravá část SOUBORu1. Protože konec souboru je hlášen také jako chyba, lze takto kopírovat i bezchybný soubor.

#### **4.3.2. Soubor AUTORUN.SYS**

AUTORUN.SYS je smluvené jméno binárního segmentovaného souboru, který musí obsahovat vykonatelný program ve strojovém kódu, nebo binární data. Pokud je disketa se souborem AUTORUN.SYS umístěna při startu počítače v jednotce číslo 1, soubor se po natažení DOSu zavede a rozběhne, pokud má definovanou některou z rozběhových adres. Tento proces proběhne dříve, než se předá řízení uživateli. AUTORUN.SYS mohou být i data, program který se jen zavede a nerobí žádoucího, nebo program který se spustí okamžitě po načtení bez zásahu uživatele (viz též 3.2). Proto se AUTORUN.SYS hodí k výstavbě automaticky spouštěných uživatelských komplexů, automaticky instalovaných handlerů nových periferiích zařízení, nebo změněných znakových sad.

Následující příklad ukazuje použití programu se jménem AUTORUN.SYS ke změně priority DOS menu a modulu, to jest že se po spuštění dostaneme přímo do menu bez ohledu na to, zda je aktivován Basic nebo zasunut modul. Program normálně vraci řízení DOSu, respektive jeho inicializační rutině. Pokud program řízení nevraci, nebo pokud se před návratem stiskne RESET, je třeba inicializovat systém pomocí teplého startu. Dosáhne se toho nastavením dvou proměnných datové základny operačního systému, a sice COLDST (\$244) a BOOT?(\$09). COLDST se musí nastavit na 0, BOOT? na 1. Ukázkový program nejprve nastaví zmíněné hodnoty. Potom skokem na hodnotu uloženou v DOSVEC předá řízení DOSu, který vyvolá menu. Program je napsán pro zásuvný modul Atari Assembler/Editor. Může být také přeložen assemblerem EASMD, nebo MAC/65 firmy OSS. Uživatel, který nemá assembler k dispozici si může program vytvořit pomocí Basicu.

Assembler:           ; Rozběh DOSu před modulem

```

    COLDST  -      $244
    BOOT?   -      $09
    DOSVEC  -      $0A
    **      $3A98      ;za DUP.SYS
    DOSGO   LDX  #0
              STX  COLDST
              INX
              STX  BOOT?
              JMP  (DOSVEC)
    ; definice rozbaehové adresy
    **      $2E0
    .WORD DOSGO
    .END

```

Totéž v Basicu:

```

5 REM Rozbeh DOSu pred modulem
10 FOR J=0 TO 10
20 READ CODE
30 POKE 15000+J, CODE
40 NEXT J
50 DATA 162,0,142,68,2,232,134,9
60 DATA 108,10,0

```

Assemblerovský program se může kompilovat přímo na disk. Basicem vytvořený program se musí z paměti uložit na disk funkci K:

```

SELECT ITEM OR RETURN FOR MENU
K (RETURN)
SAVE--GIVE FILE,START,END(),INIT,RUN)
AUTORUN.SYS,3A98,3AA2,,3A98

```

#### 4.3.3. Obsazení paměti DOSem 2.5

Při programování ve strojovém kódu je důležité znát obsazení paměti, aby nám program nekolidoval se systémem a neničil jej. DOS 2.5 obsazuje paměť takto:

ADRESA		OBSAH
dekadicky hexadecimálně		
0	0	datová základna operačního systému (OS ROM), včetně bufferu kazety *
	47F	
480		Různé, používáno jazyky
6FF		(Atari Basic nepoužívá \$580 až \$6FF)
1792	700	Organizátor souborů - File Manager
6780	19CB	(DOS.SYS)
6781	19CC	Buffer jednotky 1 oblast Buffer jednotky 2 bufferů, Buffer jednotky 8 (RAM-disk) Sektorový buffer 1 vyhrazeno Sektorový buffer 2 pro DOS Sektorový buffer 3 2.5 Sektorový buffer 4
1D7C		Disk Utility Package   MEMLO **
3305		(DUP.SYS)
různé		oblast uživatelských programů
různé		MEMTOP *** (HIMEM)
různé		podle grafického modu a druhu, či přítomnosti modulu
		obrazová paměť
32767	1FFF, 9FFF nebo BFFF	
32768	8000 nebo A000	zásvuný modul 8 nebo 16K nebo uživatelská paměť, pokud
49151	BFFF BFFF	není modul
49152	C000	operační systém
65535	FFFF	ROM

\* Operační systém nepoužívá oblast \$80 až \$FF základní stránky.

\*\* Liší se podle počtu jednotek a sektorových bufferů

\*\*\* Záleží na zvoleném grafickém modu

Hodnotu MEMLO zjistíme na adresách \$2E7,\$2E8, nebo 743, 744 dekadicky  
Hodnotu MEMTOP zjistíme na adresách \$2E5,\$2E6 (741,742) dekadicky

#### 4.3 .4. Zrychlení zápisu dat a volba počtu jednotek

V základním stavu provádí DOS2.5 zápis dat s verifikací. To znamená, že se každý sektor po zapsání ještě kontrolně přečte. To sice zajišťuje správný zápis dat, ale zároveň se tím zápis zpomaluje. Zkušenosti ukazují, že chyby při zápisu se prakticky nevyskytují, zato zrušení verifikace zrychluje nikoli zanedbatelnou měrou práci s disketou. Verifikaci můžeme zapínat, nebo vypínat změnou obsahu adresy 1913 dekadicky. Pokud do této buňky zapísemme hodnotu \$57 (87 dekadicky), zapisuje se s verifikací. Hodnota \$50 (dek. 80) verifikaci vypne. Jde vlastně o přepisování kódu povetu pro zápis. Hodnotě 80 odpovídá "P", hodnotě 87 "W". Zrychlený zápis dosáhneme v Basicu příkazem:

POKE 1913,ASC("P")

Verifikaci obnovíme nastavením hodnoty zpět na 87:

**POKE 1913,ASC("W")**

Modifikovaný systém lze trvale uložit funkcí H v menu DOSu. Verifikaci lze volit také pomocí programu SETUP.COM (viz 4.4.3).

**Pozor:** uložení jiné hodnoty, než 80 nebo 87 způsobí zhroucení systému a může způsobit i ztrátu dat.

V DOSu 2.5 jsou standardně konfigurovány dvě disketové jednotky, číslo 1 a 2. Jiný počet můžeme nastavit zapsáním kódu na adresu 1802 dekadicky. Každá konfigurace odpovídá jeden z kódů tabulky:

konfigurované jednotky	kód
1	01
2	02
3	04
4	08
1+2	03
1+2+3	07
1+2+3+4	15

Stejně jako u přepínání verifikace se změny provedou jen v paměti. Pokud chceme, aby byly trvalé, musíme v menu DOSu uložit modifikovanou verzi na disketu funkci H v menu.

Model 130XE má vždy kromě fyzických jednotek konfigurován i RAM-disk jako jednotku č. 8. Teoreticky lze mít konfigurováno až 8 disketových jednotek a 8 současně otevřených souborů, ale firma Atari nevyrobí jednotku, na které by bylo možno nastavit vyšší číslo, než 4 (viz 2.1.3.). Osm současně otevřených souborů také vlastně nelze použít, protože téměř všechny programy potřebují kanál 0 pro editor obrazovky "E:". Další omezení vyplývá ze souboru DUP.SYS. Tento program se zavádí na určitou adresu a zvýšením počtu bufferů souborů nad standard dojde k jejich překrytí s DUP.SYS a nelze používat menu. Dalším omezením je to, že řada programů předpokládá určitou hodnotu MEMLO (\$2E7) a při jejím překročení nefunguje. Praktickým výsledkem těchto omezení je pět diskových jednotek (u 130XE 4 jednotky a RAM-disk) a 7 současně otevřených souborů u jazyků v zásuvném modulu a u programů, které se modifikují podle platné hodnoty MEMLO. Pokud potřebujete používat menu, neboli soubor DUP.SYS, platí omezení na maximálně 4 jednotky (opět včetně RAM-disku). Minimální počet současně otevřených souborů je 2, protože tento počet je nutný pro práci DUP.SYS. Obvyklá konfigurace je dvě jednotky a tři současně otevřené soubory. Při této konfiguraci odpovídá hodnota MEMLO standardní hodnotě původního DOS 2.0 a nejsou problémy s provozováním starších programů na 130XE s přidaným RAM-diskem.

#### 4.3.5. Funkce RAM-disku u 130 XE

Model 130 XE je vybaven 128K paměti, o 64K více, než starší modely. Tuto přídavnou pamět je možno využívat různým způsobem, ale pro uživatele disketové jednotky je nejjazdnejší RAM-disk, neboli virtuální disketová jednotka, simulovaná v přídavné paměti. Základními vlastnostmi RAM disku jsou na jedné straně vysoká rychlosť práce, na druhé straně malá kapacita (499 sektorů) a spolehlivost. Informace se z RAM-disku ztratí nejen vypnutím počítače, ale také studeným startem. DOS 2.5 bohužel nemá možnost inicializovat RAM-disk bez mazání, takže studeným startem se obsah ztratí také. Velkou výhodou je podstatně zrychlení přechodu mezi DOSem a programem a rychlá práce se souborem MEM.SAV. Kromě těchto specifik se RAM-disk chová jako skutečná disketová jednotka. Vedle 130 XE mohou pracovat s RAM-diskem i modely 800XL a XE, které mají dodatečně rozšířenou paměť za předpokladu, že přepínání jednotlivých bank paměti vychází ze 130XE. Tato rozšíření mají obvykle navíc 192K nebo 256K, ale s DOS2.5 lze z tohoto množství využít jen 64K.

#### Aktivace RAM-disku

Systémová disketa DOS 2.5 obsahuje soubor RAMDISK.COM, který při studeném startu počítače automaticky inicializuje v přídavné paměti RAM-disk. Pokud počítač nemá rozšířenou paměť, nedělá nic. RAM-disk se vytvoří jen při studeném startu. Spuštěním programu RAMDISK.COM z menu se RAM-disk neinicializuje. Inicializace probíhá takto:

- zobrazí se zpráva o inicializaci RAM-disku
- provede se ustavení přídavné paměti jako RAM-disk, RAM-disk se naformátuje
- do RAM-disku se zkopiují soubory DUP.SYS a MEM.SAV, které se používají místo stejnojmenných souborů na "D1:".

Pokud je třeba kapacitu RAM-disku zvýšit, můžeme postupovat dvojím způsobem:

- 1) Nastartovat systém s disketou, na které není DUP.SYS. Tento postup se hodí u uzavřených uživatelských systémů, protože neumožňuje přechod do menu DOSu. RAM-disk se při něm inicializuje, ale nezkopíruje se do něj žádné soubory.
- 2) Po normálním nastartování s DOSem a RAM-diskem docílime toho, aby se používaly soubory DUP.SYS a MEM.SAV z jednotky 1 a na RAM-disku D8 je zrušíme:
  - Změníme obsah adresy, určující systémovou jednotku. Je to adresa 5439, neboli \$153F a obsahuje hodnotu ATASCII kódu čísla systémové jednotky. V tomto případě ji nastavíme například takto:

POKE 5439,ASC("1")

- Na disku 8 zrušíme všechny soubory funkcí D s parametrem 8:.\*/\*N. Tím máme RAM-disk volný pro své účely.

## Přenos souborů mezi diskem a RAM-diskem

Ke kopirování normální diskety do RAM-disku nelze použít funkci J, protože RAM-disk má jen 499 sektorů. Proto je třeba kopirovat soubory individuálně funkcí C. Takto naplněný RAM-disk již lze zkopirovat na fyzickou disketu funkcí J s tím, že disketa má po provedení kopie kapacitu stejnou jako RAM-disk, t.j. 499 sektorů. Takovouto disketu s omezenou kapacitou již můžeme kopirovat zpět do RAM-disku funkcí J.

## Práce bez RAM-disku

Pokud nechceme, aby se při startu DOSu aktivoval RAM-disk, zrušíme, nebo přejmenujeme soubor RAMDISK.COM. Lepší je přejmenování, soubor zůstává a vrácením původního jména můžeme RAM-disk opět používat. RAM-disk se nesmí aktivovat, pokud potřebujeme přídavnou paměť pro jiné účely. Jiným důvodem nepoužívání RAM-disku mohou být problémy se spouštěním některých programů. Například Atari Assembler/Editor stažený ze zásuvného modulu na disk se s RAM-diskem nesnáší.

## 4.4. Služební programy DOS 2.5 (Utility)

Systémová disketa DOSu 2.5 obsahuje kromě souborů DOSu ještě takzvané služební programy, neboli utility, které mají koncovku .COM a slouží k následujícím účelům:

COPY32.COM umožňuje převádět programy z formátu DOS3 do DOS2.5.

DISKFIX.COM slouží k opravě a verifikaci disket ve formátu DOS2.5 a DOS 2.0 a kompatibilních, k opětné instalaci zrušených souborů a k přejmenovávání souborů podle čísel v adresáři.

SETUP.COM umožňuje měnit parametry DOSu a vytvořit program pro automatický start programu v Atari Basicu.

RAMDISK.COM viz 4.3.5.

### Spouštění utilit

Protože všechny služební programy jsou napsány v assembleru, spouštějí se v menu funkci L - BINARY LOAD. Jako odpověď na dotaz dáme jméno spouštěného programu.

### 4.4.1. Program COPY32.COM

Tento program funguje podobně jako kopírovací program, ale kopíruje soubory z diskety ve formátu DOS3 na disketu ve formátu DOS2.5. Po spuštění se program ptá, ve které jednotce bude disketa DOS3 (zdrojová) a ve které DOS2.5 (cílová). Pokud máte jen jednu jednotku, odpovíte v obou případech 1 a budete podle výzev programu vyměňovat diskety DOS3 a DOS2.5. Komunikace s programem vypadá takto:

**COPY 3 to 2.x**

**Copy files from a DOS3 disk to  
a DOS 2.5(or DOS 2.0S) disk.  
(Hit RETURN for drive \* to quit)  
On which drive (1-4) is DOS3 disk? 1  
On which drive (1-4) is DOS2.x disk? 1  
Place DOS 3 disk in drive 1  
Caution: You will be swapping disks.**

**Put a write protect tab  
on your DOS3 disk!**

**Push (START) when ready.**

Po oznámení názvu a účelu programu je instrukce k ukončení práce. Program končí tím, že místo čísla jednotky stiskneme samotný (RETURN). Potom se ptá na číslo jednotky, ve které je zdrojová a cílová disketa. Protože jsme na oba dotazy odpovíděli "1", je po výzvě ke vložení zdrojové diskety formátu DOS3 ještě upozornění, že budeme muset diskety vyměňovat a výzva k ochraně zdrojové diskety nálepkou. Pokud máme zdrojovou a cílovou disketu uvedenou jednoikách, žádá program o vložení obou disket. V každém případě stiskneme po splnění požadavků klávesu START. Program čte adresář diskety DOS3 a zobrazuje jména souborů s pořadovými čísly. Na obrazovku se jich vejde 16. Je-li na disketě souborů více, přejdeme klávesou (RETURN) na další řešináčku. Po vypsání všech stránek adresáře (aniž bychom některý soubor vybral) máme volbu Restart (spuštění programu od začátku), návrat do DOSu, nebo opakováný výpis adresáře. Konverzi některého souboru spustíme napsáním pořadového čísla v adresáři. Program požaduje potvrzení volby klávesou START.

Nyní program čte celý soubor, nebo tolik, kolik se mu vejde do paměti. Poté zapisuje na cílovou disketu zkonzervovaný soubor. Pokud pracujeme s jednou jednotkou, požádá předtím o vložení cílové diskety ve formátu DOS 2.5. Pokud nastane při konverzi chyba, zobrazí se její číslo a je požadována volba Restart klávesou (START), nebo návrat do DOSu klávesou (SELECT).

*Poznámka:* Pokud nemáte dvě jednotky, nemůžete konvertovat soubory větší, než 124 700 byte. To je o 300 byte méně, než je maximální velikost souboru u DOS2.5.

**4.4.2 . Program DISKFIX.COM**

Program DISKFIX.COM slouží k verifikaci a opravám disket tím, že ruší všechny nějak narušené soubory. Dále je možné jím restaurovat omylem zrušený soubor, pokud se mezikm na disketu nezapisovalo, a přejmenovat soubory podle čísel v adresáři. Program má základní menu s těmito možnostmi:

1. Change Drive \*
2. Unerase File
3. Verify Disk
4. Rename file by \*
5. Quit to DOS

**Volba se provádí stisknutím čísla funkce bez klávesy (RETURN). Funkce mají tento význam a použití:**

**Change Drive = - změna čísla jednotky**

Vybiráme číslo jednotky, kterého se budou týkat funkce 2 až 4. Na dotaz na číslo jednotky odpovíme číslem 1 až 8 (tedy lze pracovat i s RAM-diskem).

**Unerase File - obnovení zrušeného souboru**

Zrušený soubor není z diskety vymazán, je pouze označen jako zrušený v adresáři a sektory, které obsazoval jsou v tabulce VTOC označeny opět jako volné (viz 3.4.). Dokud se na disketu nezapíše jiný soubor, nejsou data přepsána. Proto je možný soubor bezprostředně po zrušení opět obnovit. Funkce obnovení může také navíc opravit status souboru, který byl otevřen pro výstup a nebyl uzavřen, pokud neobsahuje chybné odkazy na následující sektory. Soubor otevřený pro výstup a neuzávřený má status \$43, není ve výpisu adresáře vidět, ale zabírá na disku místo, které nelze využít. Největší šance na obnovu souboru jsou bezprostředně po zrušení, nebo neuzávření.

Po volbě funkce 2 se z diskety, kterou chceme opravovat, vložené do jednotky čísla nastaveného funkcí 1, vypisují názvy souborů spolu s jejich pořadovým číslem v adresáři. Na obrazovce se zobrazí 32 názvů najednou, t.j. polovina adresáře. Jsou označeny 0 až 31 a každý soubor má před sebou ještě označení typu. Mezera znamená platný soubor v pořádku. W znamená neuzávřený soubor, otevřený pro zápis a D soubor zrušený. Dosud nepoužité pozice adresáře jsou označeny jako (unused). Po výpisu se buď vybere číslo souboru, který chceme obnovit, nebo se zobrazí druhá polovina adresáře samotnou klávesou (RETURN). Obnovovaný soubor musí mít status W nebo D. Pokud se zadá číslo větší, než 63, nebo (RETURN) po zobrazení druhé poloviny adresáře, vypíše se zpráva "You didn't choose anything!" oznamující, že nebylo nic vybráno. Při volbě čísla, které odpovídá nepoužité pozici se vypíše "That file is unused", pokud se zvolí soubor, který nemá status W nebo D, je to opět okomentováno patřičnou zprávou. Ve všech těchto případech se po stisknutí (START) opakuje cyklus od začátku.

Je-li volba v pořádku, zobrazí se číselně vyjádřený status souboru (viz 3.4.2.), jméno, délka počáteční sektory, vše jak dekadicky, tak hexadecimálně. Status 80 znamená zrušený soubor, 43 neuzávřený. Program nyní žádá potvrzení volby. Stisk (Y) znamená obnovení souboru. Před vlastním obnovením souboru program kontroluje všechny soubory diskety (podrobnosti viz následující odstavec). Pokud se nenalezně chyba, kontroluje se i obnovovaný soubor a je-li v pořádku, zapíše se do adresáře věta o souboru zpět s kódem platného souboru. Po verifikaci posledního souboru se na disk zapíše aktualizovaná tabulka VTOC a jsme vyzváni klávesou (START) posunout běh věci opět do základního menu programu.

**Verify disk - verifikace diskety**

Tuto funkci se všeobecně kontrolují všechny platné soubory na disketě. Při vybrání této funkce nás program požádá o vložení ověřované diskety do nas-

tavené jednotky. Vykonání žádaného oznamíme klávesou (START). Program pak ověřuje platnost a kompaktnost každého souboru. Při tom kontroluje řetězení všech sektorů a vytvoří si tak novou tabulku obsazení sektorů VTOC a zapíše ji zpět na disk. Program DISKFIX.COM nesmíme poštít na žádnou disketu s ochranou proti kopírování, protože by nám ji pravděpodobně naboural. Při verifikaci se kontroluje shoda údajů adresáře se skutečnou délkou souborů a do nové VTOC se jako obsazené zapíší jen ty sektory, které jsou částí některého z platných souborů. Protože se původní VTOC při tom nepoužívá, lze takto opravit disketu s vymazanou tabulkou obsazení sektorů. Během verifikace se status souborů W mění na D (viz unerase) a vypíše se o tom zpráva. Pokud má soubor délku 0, jako v případě souboru se stejným počátečním sektorem jako soubor zapsaný později, nelze takový soubor obnovit a je signalizován zprávou "Bad file - deleting". V ostatních případech můžeme opravit alespoň část souboru. Každý sektor je kontrolován, zda patří do souboru, který na něj ukazuje (viz 3.3.2.). Pokud se zjistí špatný odkaz, tedy že následující sektor patří do jiného souboru, vypíše se zpráva "Bad link in file - truncating", soubor se před vadným sektorem "ušmikne" a obnoví se jen bezchybná část.

#### **Rename file by # - přejmenování souboru podle čísla**

Tato funkce je užitečná při přejmenování druhého ze dvou stejných jmen. Přitom nezáleží, zda je soubor blokován, t.j. lze na rozdíl od menu přejmenovávat i zamčené soubory. Přejmenování prvního ze dvou souborů se dá jednoduše provést funkcí E v menu DOSu. Z praktického hlediska je však rychlejší přejmenovat první soubor, potom druhý a pak vrátit prvnímu souboru původní název, než kvůli přejmenování natahovat DISKFIX.COM. Funkce se používá takto:

Po umístění příslušné diskety v aktivní jednotce se stiskne (START). Vypíše se očíslovaný obsah adresáře. Číslem zvolíme soubor, který chceme přejmenovat. Před provedením operace se stejně jako u funkce UNERASE vypíše hlavička souboru s dalšími údaji. Kromě toho jsme požádáni o nové jméno. Po ukončení akce žádá program stisknutí (START) a vrací se do svého základního menu.

#### **Quit to DOS - návrat do DOSu**

Vypíše se dotaz Return to DOS 2.5 (Y/N). Při N zůstaneme v programu DISKFIX.COM, při Y se dostaneme do menu DOSu.

#### **4.4.3. Program SETUP.COM**

Slouží k nastavování parametrů DOSu. Po nastartování se objeví menu:

#### **SETUP**

This program will work with and  
affect the diskette inserted in  
drive number 1 unless you use  
option 1 in the menu below!

**Choose an option:**

- 1. Change current drive number**
- 2. Change system configuration**
- 3. Set up an AUTORUN for Boot**
- 0. Quit - Return to DOS**

**Your choice (0,1,2, OR 3)?**

Podstatné funkce jsou 2 a 3. Volba se provede stisknutím příslušné číslice a (RETURN).

#### **Change Current Drive Number** - změna aktuální disketové jednotky

Funkce 2 a 3 pracují s aktuální jednotkou, jejíž číslo se touto funkcí volí. Číslo jednotky může být 1 až maximálně 4, podle počtu připojených jednotek.

#### **Change system configuration** - změna parametrů systému

Při vybrání této funkce se objeví podřízené menu:

##### **Change System Configuration**

###### **Current System Configuration:**

**Active Drives: 1 2**

**Up to three files open simultaneously.**

**Disk writes occur with verify**

**Do you want to change any part of  
that configuration (Y/N)?**

V menu je vypsána momentální konfigurace systému: aktivní jednotky 1 a 2, až 3 soubory otevřené současně, zápis na disk s verifikací. Na závěr je otázka, zda chceme změnit některou část konfigurace. Pokud odpovíme Y, máme možnost měnit zobrazené parametry jeden po druhém. Protože u DOS 2.5 spotřebuje každá aktivní jednotka 144 byte a každý současně otevřený soubor 128 byte, lze ubíráním zvětšovat uživatelskou paměť. Přidáváním jednotek a souborů získáme větší diskovou kapacitu a pružnost, ale stojí nás to zmenšenou pamětí (viz 4.3.4.). **Pozor:** když použijeme 4 aktivní jednotky a máme 130XE s RAM-diskem, nemůžeme v menu DOSu používat tyto funkce:

- C. COPY FILE**
- D. DELETE FILE**
- O. DUPLICATE FILE**

Za normálního stavu zapisuje DOS na disketu tak, že po sobě každý sektor znova čte, tj. zapisuje s verifikací. To sice zvyšuje bezpečnost uložení dat, ale podstatně zpomaluje zápis. Zde je další možnost, jak verifikaci zrušit (viz též 4.3.4.). Po změně všech parametrů se nová konfigurace vypíše a následu-

jí dotazy:

**Are you sure this configuration  
is what you want (Y/N)**

neboli zda vypsaná konfigurace je to, co jsme chtěli, a

**Current system configuration has  
been changed. Do you want to  
make these changes to the disk  
currently in drive n (Y/N)?**

oznamuje, že konfigurace systému v paměti byla změněna, a ptá se, zda se mají změny uložit na disketu v jednotce n, kde n je číslo aktuální jednotky zvolené funkcí 1. Pokud změny neuložíme, bude změněn jen DOS v paměti. Uložit jej však můžeme i později funkci H. Změny nelze uložit, když na disketu není DOS.SYS, nebo je blokován proti zápisu, nebo když má disketa nálepku.

**Set Up An Autorun for Boot - volba automaticky spouštěného programu**  
 Tuto funkcí lze vytvořit program, který při zapnutí počítače automaticky spustí program v Atari Basicu. Spouštěcí program je vytvořen pod názvem AUTORUN.SYS (viz 3.2.) a může kromě toho načíst a aktivovat řídící programy pro interfejs Atari 850. Druhá funkce nemá praktický význam, protože interfejs 850 se u nás neprodává a spojení po telefonní síti pomocí modemu se zatím nerozvinulo. Po zvolení bodu 3 v základním menu programu se objeví podřízené menu:

**Make an AUTORUN.SYS program file**

**When the disk currently in drive  
number 1 is next booted, what  
do you want to happen?**

1. The RS232 (R:) drivers for the  
Atari 850 Interface module  
are loaded and made active.
2. A BASIC program will automatically  
load and RUN
3. Both actions (1. and 2. above)  
will occur.
0. None-quit to main menu.

**Your choice (0, 1, 2, OR 3)?**

Program se ptá, co se má stát po příštém zapnutí počítače, když bude v jednotce 1 vložena disketa, se kterou se nyní bude pracovat. Při zvolení funkce 1 nejsou třeba žádné parametry a soubor AUTORUN.SYS se rovnou zapíše. Při funkcích 2 a 3 se rozvine dialog, při kterém sdělíme název programu, který má Atari BASIC automaticky načíst a spustit. Při tom se nepíše specifikace zařízení, ale jen jméno s platnou koncovkou. Program musí být v tokenizovaném tvaru, tj. uložen povelom SAVE.

Please enter the name of the BASIC  
Program that you wish to have  
automatically RUN when this disk  
is next booted.

Do NOT enter the drive specifier  
(i.e., do not use D:, D1:, etc)  
but DO use the proper extension  
(e.g., .BAS, .SAV etc.) if you  
SAVEd it with an extension.

REMEMBER: The Basic program that you  
wish to 'AUTORUN' in this way MUST  
be SAVEd on the same disk which  
receives this AUTORUN.SYS program  
file!

Now enter your BASIC program's name here:

>

Program nejprve vysvětluje, že se do jména programu nepíše kód zařízení (předpokládá se 'D1:') ale koncovka se musí uvést podle skutečného jména programu. Dále je zde upozornění, že Basicovský program musí být na stejně disketě, na kterou byl uložen vygenerovaný AUTORUN.SYS. Pokud by se pravidlo nedodrželo, vypsala by se při spouštění počítače chybová zpráva. Disketa musí kromě toho obsahovat DOS.SYS.

## 5. Happy DOS II+/D

Happy DOS byl otištěn jako listing v časopise Happy Computer číslo 3/86. Je dílem tehdy 21-ti letého Stefana Dorndorfa, kterého známe i z MikroDosů pro spouštění her. Happy DOS byl vytvořen prakticky paralelně s DOS 2.5, z čehož vyplývají různé odchylky. Je představitelem ryze povelového systému a jeho největšími přednostmi jsou:

- malý rozsah (zabírá málo paměti i místa na disku)
- okamžitý přechod mezi DOSem a programem, neboť je v paměti stále celý
- RAM-disk i pro modely s 64 kB paměti
- možnost instalace příkazového souboru pro dávkovou práci
- vestavěný monitor
- filtry pro výběr souborů ve zkrácené podobě a s širší volbou
- rozšířená a přidané funkce XIO pro přístup na disk

Kromě těchto výhod jsou tu i nevýhody. Některé z nich vyplývají z odlišného řešení od DOS 2.5, některé z výstavby systému jako takového. Mezi hlavní nevýhody patří:

- nemožnost instalace RAM-disku v rozšířené paměti u 130XE
- RAM-disk pro Basic umožňuje uložit jen jeden soubor a načíst jen soubor naposledy uložený.
- nemožnost vypsat povelom DOSu adresář na tiskárnu
- nemá ekvivalent funkce J DOSu 2.5

Z uvedených rysů vyplývá nevhodnější určení Happy DOSu pro programování v Basicu, assembleru, nebo jiném programovacím jazyku v zásuvném modulu, nebo pro konstrukci uživatelských systémů pomocí příkazových souborů na modelech bez rozšířené paměti, tj. 800XL, 800XE a 65XE. Okamžitý přechod mezi DOSem a modulem šelší významnou měrou čas a také obsluha DOSu je po zvládnutí povelů rychlejší, než obsluha pomocí menu v DOS 2.5. Písmeno D v názvu Happy DOSu má znamenat Double Density, neboli dvojitou hustotu. Toto označení je matoucí, protože se ve skutečnosti jedná o hustotu rozšířenou s 1040-ti sektory po 128 bytech, zatímco dvojitá hustota je charakterizována délkou sektoru 256 byte. Při programování ve strojovém kódu jsou tu některé drobné zvláštnosti okolo startu programů a zacházení s adresou DOSINI. Uživatelská paměť je volná již od \$IDBD a celý Happy DOS zabírá na disketu 37 sektorů, disketa formátovaná v rozšířené hustotě Happy DOSem má 1026 volných sektorů (proti 1010-ti u DOS 2.5). Hlavička Happy DOSu vypadá takto:

Happy-Computer DOS II+/D V:4.5M  
COPYRIGHT 1985 by Stefan Dorndorf

DI:■

## 5.1. Zvláštní rysy Happy DOSu

Happy DOS má některé specifické rysy, které se u jiných systémů nevyskytují. Jedná se zejména o rozšíření výběrových znaků souborového filtru a možnosti volby účinku klávesy (RESET).

### 5.1.1. Konvence jmen a konstrukce filtru

Konvence pro jména se od konvence DOSu 2.5 liší jen v jednom detailu. Místo dvojtečky lze pro označení kódu zařízení (viz 2.2.10.) použít i středník. Středník je na stejně klávese jako dvojtečka, takže ušetříme mačkání klávesy SHIFT. Zápis "D:PROGRAM" je stejný, jako "D;PROGRAM".

Větší doplňky jsou u konstrukce filtru souborů a umožňují jednak větší zkrácení filtru, jednak dodatečný výběr z filtrem definované skupiny.

značka	význam
-	*.* nebo *.*/
:	:
/n	výběr n-tého souboru z definované skupiny, n=1 až 9
/V	zápis s verifikací (viz 4.3.4. a 4.4.3.)
/A	připojení (append)

příklady:

zápis	význam
D,PRO-	D:PRO*.*
D,*.BAS/2	druhý program s koncovkou .BAS
D,-3	třetí soubor na disketu

### 5.1.2. Řízení zápisu

Zápis probíhá u Happy DOSu standardně bez verifikace. Pokud chceme zapsat důležitý soubor s verifikací, musíme přidat za jeho jméno parametr "/V". Připojování souboru za jiný soubor se provádí přidáním parametru "/A" stejně jako u DOS 2.5. Spojované soubory Happy DOSu mají poněkud jiný formát a nelze je připojovat za soubory pořízené DOsem 2.5. Pokud je takovéto spojení potřeba, je nutno soubor překopírovat na disketu Happy DOSu povelom COP a teprve poté připojovat další soubor. Zápis do n-tého souboru určené skupiny se provede přidáním "/n".

### 5.1.3. Odlišnosti VTOC a adresáře

Happy DOS má na rozdíl od DOS 2.5 tabulkou VTOC rozloženou do sektorů 360 a 1023, neboli \$168 a \$3FF. Proto nelze zapisovat do oblasti rozšířené hustoty křížem, tj. Happy DOsem na disketu DOS 2.5 a naopak. Takový zápis by způsobil totální zmátek. Adresář má stejnou stavbu, jako standard. Jediný rozdíl je v slavovém byte souboru zasahujícího nad sektor T20. U Happy DOSu jsou tyto soubory označovány stejně, jako soubory z oblasti jednoduché hustoty (viz 4.1.1.) a to je důvodem, proč čte DOS 2.5 z diskety Happy DOSu

všechny soubory. Tato vlastnost může způsobit selhání při čtení diskety DOsem, neovládajícím rozšířenou hustotu. Příkladem je OS/A+ a DOS XL firmy OSS.

#### 5.1.4. Zvláštní funkce kláves

Happy DOS má některé funkce řízeny zvláštními kombinacemi kláves. Dají se jimi volit, nebo naopak přerušovat různé akce.

kombinace kláves	význam
ESC	při studeném startu potlačí příkazový soubor JOB
SHIFT-BREAK	okamžité přerušení operace s disketou
OPTION-RESET	přechod do DOSu (vykonavače povelů)
RESET	přechod do modulu (pokud je)

#### 5.2. Příkazy pro řízení Happy DOSu

Happy DOS je typickým příkladem rye povelového diskového operačního systému, jehož komunikaci s uživatelem řídí příkazový procesor (viz 3.1.). To znamená, že uživatel nedostává na výběru žádné možnosti, ale musí znát příkazy procesoru. Tento způsob se může zdát obtížný, ale je tomu tak jen zpočátku, než si zapamatujeme potřebné příkazy. Po jejich ovládnutí je pak práce rychlejší, než s výběrovým menu. Příkazy se píší po vypsání základní výzvy "D1:" a popisujeme ve stejném pořadí, jako jsou popsány jejich ekvivalenty v DOS 2.5.

##### 5.2.1. Výpis adresáře

**DIR**

formát povelu:      DIR\* [specifikace]

kde:

- "\*" je povinná mezera
- specifikace je filtr pro výpis adresáře. Standardní hodnota je D1:\*,\*

Povelenem DIR vypíšeme na obrazovku obsah adresáře diskety. Výpis má prakticky stejný formát jako u DOS 2.5 s tím rozdílem, že se neoznačují soubory používající sektory z oblasti rozšířené hustoty. Pokud vypisujeme Happy DOsem obsah diskety nahrané DOsem 2.5, nevypíší se soubory z oblasti rozšířené hustoty. Naopak DOS 2.5 zobrazí všechny soubory z diskety Happy DOSu. Výpis adresáře nelze směrovat jinam, než na obrazovku. Pokud je třeba adresář vytisknout, je třeba použít buď DOS 2.5, nebo program napsaný například v Basicu:

```

10DIM A$(40):OPEN #1,6,0,"D1:*,*"
20TRAP 100
30INPUT #1,A$
40LPRINT A$
50GOTO 30

```

Pro výpis adresáře je možno použít filtr se všemi specialitami, které Happy DOS nabízí.

### **5.2.2. Přechod do modulu**

**CAR**

formát povelu:      CAR x [ povelová řádka ]

kde:

- "x" je povinná mezera
- povelová řádka je řádka do 40-ti znaků

Povel CAR způsobí předání řízení zasunutému modulu, nebo vestavěnému Basicu. Stejný účinek jako CAR má stisknutí klávesy (RESET). Protože se Happy DOS nachází v paměti stále celý, probíhá přechod DOS <-> modul okamžitě, a pokud nepoužijeme v DOSu povel COP, LOA, nebo vyvolání programu, zůstává obsah uživatelské paměti nezměněn. Přechod z modulu do DOSu lze kromě standardního povelu (tfeba DOS v Basicu) provést stisknutím OPTION+RESET. Za klíčovým slovem CAR lze uvést až 40-ti znakovou povelovou řádku, která se předá modulu. Například lze tak přímo z DOSu nastartovat program:

D1:CAR ? "A jedem!":RUN "D:ZEMEPIS.BAS"

### **5.2.3. Kopírování souborů**

**COP**

formát povelu:      COP x [ zař.1 ] jméno1 [ , zař.2 [: jméno2 ] ]

kde:

- zař.1 je zdrojové a zař.2 cílové zařízení. Není-li uvedeno předpokládá se "D1". Kazeta se jako vstup označuje "#C" kvůli rozlišení od disketového jména začínajícího písmenem C. Pokud je kazeta jako jméno2, příše se již normálně. Jako zař.1 nelze použít "K:" ani "E:".
- jméno1 se musí vždy uvést
- jméno2 se uvést nemusí. V tom případě se převezme jménem1.
- "x" je povinná mezera

Kopírování povelom COP odpovídá zhruba funkci C a O v DOSu 2.5. Navíc je možno v režimu dlouhých meziblokových mezér pracovat i s kazetou. Při kopírování se vždy po načtení příslušné porce zdrojového souboru práce přeruší s výpisem -> DESTINATION. Stisknutím klávesy (RETURN) se spustí zápis do cílového souboru. To umožňuje kopírovat soubory z jedné diskety najinou s použitím jedné jednotky. Na rozdíl od DOSu 2.5 při tom lze měnit jméno souboru. Pokud by se zdrojový soubor nepřekopíroval najednou, nebo při kopírování skupiny souborů, je vložení zdrojové diskety vyžádáno výpisem -> SOURCE. Pokud je uvedeno cílové jméno, musí se uvést i kód zařízení. Poněkud nezvyklé je obhospodaření kazety. Je-li kazeta jako vstup, musí

se před ní psát znak \*, aby se odlišila od disketového jména, začínajícího na C. Pokud je jako výstup, piše se normálně. U výstupního zařízení se nemusí psát dvojtečka, pokud za ním není uvedeno jméno2. Pokud není uvedeno zař.2 (a tím pádem ani jméno2), kopíruje se na "D1". Nejlépe to budou ilustrovat příklady:

povel	kopíruje se
COP HELPCOM	D1:HELP.COM -> D1:HELP.COM
COP TEXT,E	D1:TEXT-> E:
COP PROG,D2	D1:PROG-> D2:PROG
COP PRG,C	D1:PRG-> C:
COP *C,D:PRG	C:> D1:PRG
COP CPROG,CPROG2	D1:CPROG-> C:, chybný zápis, neuvezeno zař.2. jméno je chápáno jako zařízení.

Při kopirování je možno používat jak filtr pro výběr souboru, tak parametry pro řízení zápisu /A, /V nebo /n (viz 5.1.1. a 5.1.2).

#### 5.2.4. Rušení souborů

**DEL**

formát povelu: **DEL** \* [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "\*" je povinná mezera

Při rušení souborů je třeba rozvaha, protože co napíšeme to se stane bez přiležitosti korigovat eventuelní chybu.

#### 5.2.5. Přejmenování souboru

**REN**

formát povelu: **REN** \* [zař.] jméno1, jméno2

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno1 je úplné staré jméno nebo filtr, jméno2 je nové jméno nebo filtr. Pro oba filtry platí stejná pravidla, jako u DOS 2.5.
- "\*" je povinná mezera

#### 5.2.6. Blokování a uvolňování souborů

**LOC, UNL**

formát povelu: **LOC** \* [zař.] jméno

**UNL** \* [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "\*" je povinná mezera

Blokování se provede povelom LOC, odblokování povelom UNL. Povely odpovídají funkcím F a G v DOSu 2.5.

### 5.2.7. Instalace Happy DOSu

**IN\*\***

formát povelu:      **IN#**

Složí k vytvoření souboru DOS.SYS. Ekvivalentně lze tento soubor vytvořit i v Basicu povely OPEN #1,8,0,"DOS.SYS":CLOSE#1.

### 5.2.8. Formátování diskety

**FO\*\*, FD\*\*, CL\*\***

formát povelu:      **FO# [\*/název]**      pro SD  
**FD# [. /název]**      pro ED  
**CL# [/název]**      pro zrušení všech souborů

kde:

- \* je povinná mezera
- název je název diskety, který se vypisuje jako hlavička adresáfe
- SD je jednoduchá a ED rozšířená hustota

Při formátování jsou dva povely. K nim je přidán příbužný povel CL#. FO# vytvoří 707 volných sektorů, FD# 1026. Při formátování lze zadat jako parametr jméno diskety, které se zapíše jako první soubor do adresáfe. Jméno se zobrazuje inverzně a zabírá jedno místo v adresáfi. Při použití jména může proto být na disketě jen 63 souborů. Povel CL# diskuť neformátuje, ale zruší všechny soubory a zapíše nový prázdný adresáf se jménem diskety, pokud je zadáno. Pokud se jako jméno diskety uvede "()", je výsledkem holá neformátovaná disketa bez adresáfe.

### 5.2.9. Uložení úseku paměti

**SAV**

formát povelu:      **SAV\* [zař.] jméno.adr1.adr2**

kde:

- jméno je jméno souboru, do kterého se úsek paměti uloží
- adr1 je počáteční, adr2 koncová adresa úseku
- "\*" je povinná mezera

Povel SAV ukládá do zvoleného souboru určený úsek paměti. Úsek lze později načíst povelom LOA. Na rozdíl od DOSu 2.5 zde nelze definovat rozběhové a inicializační adresy.

**5.2.10. Načtení a spuštění binárního souboru****LOA**

formát povelu:      { LOA \* [ zař. ] jméno }  
                         jméno

kde:

- zař. je kód vstupního zařízení
- jméno je jméno souboru
- "\*" je povinná mezera

Načtení binárního souboru se provede buď povelom LOA, nebo přímo uvedením názvu souboru. Soubor musí mít strukturu binárního segmentovaného souboru (viz 3.5. a 4.1.11) a pokud má definovány příslušné spouštěcí adresy, je po načtení příslušné části spuštěn. Rozdíl mezi povelom LOA a přímým uvedením jména je v zacházení s popisnou částí. Není-li uvedena, předpokládá se u LOA, že extender není. U přímo uvedeného jména bez extenderu se předpokládá popisná část .COM. Proto je v tomto případě jméno bez popisné části označeno tečkou za jménem. Napišeme-li "PROGRAM", hledá se "D1:PROGRAM.COM", zatímco "PROGRAM." znamená "D:PROGRAM".

**5.2.11. Skok na adresu****RUN**

formát povelu:      RUN \* [ adresa ]

kde:

- adresa je hexadecimálně zadaná adresa, na kterou skáčeme
- "\*" je povinná mezera

Povel RUN předá řízení na udanou adresu. Pokud se adresa neudá, skočí se na adresu, kterou byl naposledy nastartován program při povelu LOA, nebo se zopakuje naposledy provedený povel RUN. Vynucení studeného startu, které je uvedeno jako příklad u funkce M v DOS 2.5, se v Happy DOSu napiše jako RUN E477, návrat do Turbo Basicu RUN 2080.

**5.2.12. Změna aktuální disketové jednotky****/n**

formát povelu:      /n

kde:

- n je číslo (1 až 4) disketové jednotky, které se budou týkat povely.

V základním stavu se všechny akce týkají jednotky č.1. Pokud máme jednotek více, můžeme tímto povelom systému sdělit, že se bude pracovat s jednotkou číslo n. Aktuální jednotka je vykonavačem povelů vždy vypisována ve formě výzvy k zadávání povelu, jako např. "D1:".

### 5.2.13. Monitor

V Happy DOSu je k dispozici monitor, pomocí něhož lze vypisovat paměť a do jisté míry i ladit strojové programy. Monitor umožňuje výpis paměti, modifikaci paměti a výpis obsahu registrů.

#### Výpis paměti

formát povelu: > adresa

kde:

- adresa je šestnáctibitová adresa v hexadecimálním tvaru, levostrianné nuly se musí psát

Tímto povellem se vypíše hexadecimální hodnota osmi byte od uvedené adresy počínaje. Vypsánou řádku je možno editovat a klávesou (RETURN) zapsat změněné hodnoty do paměti.

#### Změna obsahu paměti

formát povelu: > adresa × hod [ × hod ....]

kde:

- adresa je šestnáctibitová adresa v hexadecimálním tvaru, levostrianné nuly se musí psát
- hod je nová hodnota adresovaného bytu.
- "×" je povinná mezera

Tímto povellem lze měnit obsah zadlého úseku paměti. Na jednu řádku lze napsat až 8 hodnot, neboli lze změnit až 8 byte paměti od zadlé adresy počínaje.

#### Výpis obsahu registrů

formát povelu: >R

Povel vypíše obsah registrů procesoru při nalezení poslední instrukce BRK. To lze použít jako prostředek k ladění strojových programů, i když lepší možnosti skýtají různé debuggery, které jsou součástí vývojových systémů pro assembler. Každopádně se touto možností zabrání zhroucení programu při nalezení instrukce BRK. Instrukci BRK odpovídá kódová hodnota nula, proto může k nalezení instrukce dojít i při zabloudění programu do prázdné části paměti. U Happy DOSu nedojde ke zhroucení systému, ale k přechodu do povelového procesoru DOSu a k výpisu obsahu registrů.

**5.2.14. Uložení příkazového souboru****JOB**formát povelu:      **JOB\*[ { povel } / ]**

kde:

- povel je buď lomítko, nebo povelová řádka
- "\*" je povinná mezera

Příkaz JOB bez parametru vypíše momentálně platný příkazový soubor. Povel může být buď některý z povelů Happy DOSu i s parametry, nebo lomítko. Lomítko znamená prázdný povel. Takto definovaný příkazový soubor se uloží v sektoru 1 a provede se vždy při studeném startu. Provedení lze potlačit podřízením klávesy ESCape během startu počítače.

**5.3. Nové povelы Basicu**

Happy DOS má možnosti, rozšiřující použitelnost Basicu ve vztahu k práci s disketovou jednotkou. Kromě všech funkcí DOSu 2.5 má navíc rozšířený význam povelů POSITION, NOTE, POINT a funkce XIO, které implementují nové povelы Happy DOSu.

**5.3.1. Nové povelы XIO**

Nové funkce, vyvolávané přes XIO umožňují operace, pro které by jinak bylo třeba podprogramů ve strojovém kódu. Jedná se o tyto funkce:

číslo	význam
31	práce s určeným sektorem
34	zrušení všech souborů
39	načtení (ev. spuštění) programu
255	formátování v rozšířené hustotě

Pro funkci 31 je třeba stanovit parametry poněkud nezvyklým způsobem, a to povelom POSITION, jehož vedlejší účinek je posun kurzoru na pozici 0,0 aby se předešlo chybě 141. S tím je třeba počítat při výstupu na obrazovku.

**Práce se sektorem**

Čtení, zápis a testování stavu libovolného sektoru bez ohledu na příslušnost k některému souboru se provede funkci XIO 31:

formát povelu:      **POSITION s,n:XIO 31,\*1,lo,hi,"D[m]:x"**

kde:

- s je číslo prvního zpracovávaného sektoru (0 až 1010)
- n je počet sektorů
- lo je dolní byte adresy bufferu
- hi je horní byte adresy bufferu
- m je číslo jednotky
- x určuje druh prováděné operace:  
R - čtení, W - zápis, S - status

Parametr "s" udává začátek zpracovávané oblasti číslem prvního sektoru. Počet sektorů udává parametr "n". Povel XIO musí následovat bezprostředně za povelem POSITION. "lo" a "hi" určuje začátek paměťové oblasti, ze které se bude číst, nebo do které se bude zapisovat. Jednoznakový string "x" udává druh provedené operace.

Příklad: Načteme do obrazové paměti sektory 361 až 364 z disku 1.

POSITION 361,4:XIO 31.=1.PEEK(88),PEEK(89),"D1,R"

#### **Mazání všech souborů diskety**

Tento povel odpovídá povelu CL# v DOSu. Zruší všechny soubory a založí nový prázdný adresář.

formát povelu:      XIO 34.=1,0,0."D[ n ] [ ;/\*jméno]"

kde:

- n je číslo diskové jednotky
- jméno je jméno diskety
- "x" je povinná mezera

#### **Načtení strojového programu**

Povel načítá soubor ve formátu binárního segmentovaného souboru (viz 3.5. a 4.1.11.).

formát povelu:      XIO 39.=1,f,0."D[ n ] : jméno"

kde:

- n je číslo diskové jednotky
- f řídi spouštění programu
- jméno souboru

Parametr "f" určuje, zda se načtený program nastartuje. Je-li f=0, program se jen načte, rozbehové adresy se ignorují. Je-li f <> 0, rozbehové adresy se uplatní.

#### **Formátování diskety**

Odpovídá povelu FD#. Může být připojeno jméno diskety.

formát povelu:      XIO 255.=1,0,0."D[n] [ ;/jméno]"

kde:

- n je číslo diskové jednotky
- jméno je jméno diskety

Tento povel vytvoří 1024 volných sektorů.

#### **5.3.2. Čtení a zápis úseků paměti, status disku**

Jedná se o modifikované povely NOTE a POINT. NOTE slouží ke čtení, POINT k zápisu. Pro oba povely je třeba specifikovat další nutný parametr povelem POSITION. Při čtení lze zjistit i počet přečtených byte.

## Čtení a zápis úseku paměti

formát povelu: POSITION a.255:POINT \*1,l,x

kde:

- a je počáteční adresa ukládané oblasti
- 255 slouží k rozlišení od normálního POINT
- l je délka oblasti, max. 32767
- x je druh operace: 4 čtení, 8 zápis

Po provedené operaci je kurzor nastaven na 0.0 aby se předešlo chybě 141. Basic nehlásí chybu syntaxe jen tehdy, když jsou l a x proměnné. Při použití konstant se brání hlášením syntaktické chyby.

## Zjištění počtu přečtených byte

formát povelu: POSITION 0.0:NOTE \*1,l,x

kde:

- l je počet čtených, nebo zapsaných byte
- x je druh operace: 4 čtení, 8 zápis

POSITION 0.0 slouží k rozlišení od normálního NOTE. Proměnná l obsahuje po provedení povelu počet byte. Operace funguje jak při čtení, tak při zápisu, ale význam má jen při čtení, kdy nastane před načtením požadovaného počtu bytů konec souboru, čili chyba 136 a chceme vědět skutečný počet načtených dat. Stejně jako u POINT musí být l i x proměnné, t.j. ani x nesmí být číslo. Příklad ukazuje načtení souboru do stringu a vypsání na obrazovku:

```
10 DIM A$(2000+1)
20 OPEN =1,4,0,"D;PROGRAM"
30 TRAP 50:POSITION ADR (A$),255
40 L=20000:X=4:POINT *1,L,X
50 NOTE *1,L,X:CLOSE #1
60 A$(L+1)+"!":A$(L+1)+"":?A$
```

## Zjištění druhu chyby

Pokud nastane na nějakém kanálu chyba, zjistí se její číslo povellem:

STATUS =n,avar

kde:

- n je číslo kanálu
  - avar je proměnná, do které je uloženo číslo chyby.
- Pokud žádná chyba nenastala, je uloženo číslo 1.

## 5.4. Různé informace

### AUTORUN.SYS

Konvence jména AUTORUN.SYS u Happy DOSu neplatí. Automatický start jakéhokoliv programu lze uskutečnit pomocí příkazového souboru. Strojové programy nesmí měnit vektor DOSINI. Pokud má program převzít řízení a po stisku klávesy RESET opět nastartovat, je třeba DOSINI obejít a dosáhnout inicializace jiným způsobem. Inicializační adresu je třeba uložit do CASINI (\$02) a k použití tohoto vektoru přinutit systém nastavením bitu 2 proměnné BOOT? (\$09). Tím předstíráme, že byl program zaveden z kazety a systém pak inicializuje podle kazetového inicializačního vektoru.

### Obsazení paměti

Happy DOS leží v paměti v oblasti \$700 až \$1DBD. Tabulka ukazuje některé adresy pro řízení jeho parametrů:

adresa	název	obsah	popis
\$709 1801	SECCOD	2	počet sektorových bufferů
\$70A 1802	DRV COD	1	počet bufferů disk.jednotek
\$70B 1803	TIMCOD	2	TIMEOUT (sekundy)
\$70C 1804	BUFCOD		adresa bufferu
\$70E 1806	DOSCOD	0	příznak pro DOS.SYS
\$70F 1807	STSCOD	4	první sektor DOS.SYS
\$801 1809	TRYCOD	3	počet opakování diskového povelu
\$802 1810	ADRCOD	\$880	poč.adresa DOS.SYS
\$804 1812	VBL COD	1	příznak pro rychlý auto-repeat.

Adresa 1801 určuje maximální počet současně otevřených souborů, 1802 počet připojených disketových jednotek. Je-li 1812 nenulová, zařadí Happy DOS do VBI rutinu pro zrychlený autorepeat klávesnice.

### RAM-disk pro Basic

Happy DOS instaluje pro potřeby Basicu velmi jednoduché virtuální sekvenční periferní zařízení "M:", simulované v úseku paměti \$5000 až \$7F00. V popisu se "M:" nazývá RAM-disk, ale jeho funkce a vlastnosti se blíží více kazetě, protože nepoužívá jména a soubor v něm může být uložen jen jeden. Proti přepsání je v Basicu chráněn. Pokud by program překročil vymezenou délku a měl zasáhnout do oblasti RAM-disku, ohláší Basic chybu 147. Pokud uložíme do "M:" soubor a potom další, bude v něm jen ten naposledy uložený, protože přemazal soubor původní. Možnosti použití RAM-disku jsou zejména při dočasném uložení programů, nebo pracovních dat. Tabulkou proměnných lze redukovat postupem LIST "M":NEW a ENTER "M". Programy lze ukládat povellem SAVE "M" a načítat pomocí LOAD "M". Programy lze spouštět povellem RUN "M". Nemožnost uložit více souborů dost omezuje použitelnost tohoto pseudo RAM-disku.

### **Nová chybová hlášení**

**ERROR 163 - pokus o zápis na disketu chráněnou proti zápisu**

**ERROR 166 - neplatný POINT. Buď je číslo byte přípiš veliké, nebo je povol POINT použit na kanál, otevřený pro čtení**

**ERROR 171 - chybný zaváděcí soubor. Povolení XIO 39 byl učiněn pokus načíst soubor neodpovídající formátu binárního segmentovaného souboru.**

**ERROR 172 - není formát Happy DOSu. Tato chyba se vyskytuje při pokusu připojit funkci Append soubor ve formátu DOS 2.5, nebo zapsat povolení JOB příkazový soubor na disketu, která není formátována Happy DOSem.**

### **Tabulka povelů a jejich ekvivalenty v DOS 2.5**

<u>DOS 2.5</u>	<u>Happy DOS</u>
A	DIR adresář
E	REN přejmenování
D	DEL zrušení
G	UNL od blokování
M	RUN skok na adresu
O/C	COP kopírování
B	CAR modul
I	FD# formátování ED
P	FO# formátování SD
H	IN# nahrání DOSu

## 6. OS/A+

OS/A+ je diskový operační systém firmy OSS - Optimized systems software. S touto firmou se programátor setkává velmi často. Pod ochrannou známkou Precision Software Tools se prodávají nepostradatelné prostředky pro vytváření programů. Vedle řady assemblerů (EASMD, MAC/65 diskový, MAC/65 modul), Basiců (Basic/A+, Basic XL, Basic XE) dodává ještě jazyk Action! a C/65 a nakonec i diskové operační systémy. OS/A+ je nejstarší z nich. Je to systém povelovéhotypu, ovládaný vykonavačem povelů (command processor) ve stylu známého CP/M. Protože OS/A+ byl vytvořen v roce 1981 je jasné, že byl určen pro diskovou jednotku 810 s jednoduchou hustotou. To velmi snižuje použití pro jednotku 1050, kde se pochopitelně nechceme bez podstatného důvodu omezovat na jednoduchou hustotu. Takové důvody mohou být tři. OS/A+ má tzv. prioritu před modulem. To znamená, že pokud modul vyžaduje diskovou podporu, OS/A+ nevrátí po načtení fázi operačnímu systému a ohláší se sám hlavičkou:

```
OSS OS/A+ - ATARI version 1.2
COPYRIGHT (c) 1981 OSS
```

D1:

Přitom lze obsah modulu uložit jako soubor na disk a přidáním patřičného programového obložení z něj udělat program spouštěný z disku. Pokud modul diskovou podporu nevyžaduje, nemůžeme uspat bez hardwarového čtení (viz 3.2.).

Druhým důvodem pro použití jinak zastaralého OS/A+ jsou externí povelové, které jsou realizovány programy s popisnou částí .COM, zejména pak disková verze vynikajícího makroassembleru MAC/65. MAC/65 je sice dodáván s novějším systémem DOS XL, ale s OS/A+ funguje naštěstí také. DOS XL je podstatně delší, proto je pro vývoj programů pro použití v DOS 2.5 vhodnější používat MAC/65 s OS/A+. Třetím možným důvodem je možnost tvorby sekvencí povelů pro dávkové zpracování. Nevýhodou je kromě omezení na jednoduchou hustotu nemožnost kopírovat soubor z jedné diskety na jinou pomocí jediné jednotky, "zatuhnutí" když se pokusíme přejít do neexistujícího modulu a nemožnost vypsat obsah adresáře na tiskárnu. Všechny tyto funkce splní nejlépe Sparta DOS. OS/A+ však musíme použít u programů, které svými paměťovými nároky s témito systémy kolídují. S DOS 2.5 je plně kompatibilní v oboru jednoduché hustoty. OS/A+ je v paměti stále celý a přechod mezi aplikáčním programem a vykonavačem povelů DOSu je okamžitý. Pro uchování uživatelské paměti platí prakticky totéž, co u Happy DOSu. Stisknutím klávesy RESET se dostaneme vždy do vykonavače povelů. OS/A+ provádí zápis na disk bez verifikace v rychlém režimu.

### 6.1. Interní povelové OS/A+

Interní povelové provádí OS/A+ a nezasahuje do uživatelské paměti. Po návratu do aplikáčního programu zůstává paměť nezměněna. Výjimkou je sa-

moží jejmě povel LOA, který je v tomto případě ekvivalentní externímu povelu.

**Výpis adresáře:** DIR $\times$  [specifikace]

kde:

- "\*" je povinná mezera
- specifikace je filtr pro výpis adresáře. Standardní hodnota je D1: $\ast$ \*

Obsah adresáře lze vypsat jen na obrazovku.

**Přechod do modulu:** CAR

Pokud neexistuje modul, to znamená není aktivován vestavěný Basic, ani není zasunut žádný modul, počítač "zatuhne".

**Rušení souborů:** ERA $\times$  [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "\*" je povinná mezera

**Přejmenování souboru:**

REN $\times$  [zař.] jméno1 $\times$ jméno2

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno1 je úplné staré jméno nebo filtr, jméno2 je nové jméno nebo filtr. Pro oba filtry platí stejná pravidla, jako u DOS 2.5.
- "\*" je povinná mezera

**Blokování a odblokování souborů:**

PRO $\times$  [zař.] jméno

UNP $\times$  [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "\*" je povinná mezera

**Uložení úseku paměti:**

SAV $\times$  [zař.] jméno  $\times$  adr1  $\times$  adr2

kde:

- jméno je jméno souboru, do kterého se úsek paměti uloží
- adr1 je počáteční, adr2 koncová adresa úseku
- "\*" je povinná mezera

**Načtení a spuštění binárního souboru:**

{ LOA\* [ zař. ] jméno }  
jméno

**kde:**

- zař. je kód vstupního zařízení
- jméno je jméno souboru
- "\*" je povinná mezera

Načtení binárního souboru se provede buď povelom LOA, nebo uvedením názvu souboru. Soubor musí mít strukturu binárního segmentovaného souboru (viz 3.5. a 4.1.11) a pokud má definovány příslušné spouštěcí adresy, je po načtení příslušné části spuštěn. Rozdíl mezi povelom LOA a přímým uvedením jména je v zacházení s koncovkou a ve způsobu spuštění programu. Není-li uvedena, předpokládá se u LOA, že koncovka není. U přímo uvedeného jména bez koncovky se předpokládá popisná část .COM. Proto chceme-li spustit program s jménem bez ní, musíme za jménem napsat ještě tečku. Napíšeme-li "PROGRAM", hledá se "D1:PROGRAM.COM", zatímco "PROGRAM," znamená "D:PROGRAM". Načítání s přímým uvedením jména umí spustit jak standardní binární segmentovaný soubor, tak program ve speciálním formátu externího povelu. Při načítání s klíčovým slovem LOA musí být program ve standardním i varu.

**Skok na adresu: RUN\* [ adresa ]****kde:**

- adresa je hexadecimálně zadáná adresa, na kterou skáčeme
- "\*" je povinná mezera

Povel RUN bez uvedené adresy skočí buď na začátek naposledy nastartovaného programu (povelom LOA nebo jako externí povel), nebo na adresu, zadanou předchozím povelom RUN podle toho, co se vykonalo naposled.

**6.2. Externí povely**

Externí povely vykonávají programy ve zvláštním formátu s koncovkou .COM. Tyto programy nemají definovány žádné startovací, ani inicializační adresy a spouštějí se na začátku prvního segmentu. Proto je nelze spustit pomocí DOS 2.5. V povelové řádce lze témto programům předávat parametry pro řízení jejich činnosti. Stejný formát externích povelů používá i DOS XL a Sparta DOS a externí povely lze do jisté míry zaměňovat. Uživatel může psát také své vlastní externí povely, přičemž lze použít i standardní formát binárně segmentovaného souboru (viz 3.5. a 4.1.11) s definovanými inicializačními a rozbehovými adresami. Externí povel se spustí uvedením jména bez koncovky jako odpověď na základní výzvu povelového procesoru. Je samozřejmé, že povel lze použít jen tehdy, když je na disketu přítomen program, který jej reálizuje.

### Povel COPY

Je realizován souborem COPY.COM a slouží ke kopírování souborů obecně s tím, že nemá možnost zařadit pauzu mezi čtením originálu a zápisem kopie. Proto nelze pomocí tohoto povelu kopírovat soubor z jedné diskety na jinou pomocí jedné jednotky. Protože však je OS/A+ směrem nahoru plně kompatibilní s DOS 2.5, lze bez problému pro tuto operaci použít funkci O. Nepřijemné je jen to, že je třeba opakováným studeným startem nainstalovat DOS 2.5 a po provedené operaci opět zpátky OS/A+.

COPY { [zař.1] jméno1 | zař.1} × { [zař.2] jméno2 | zař.2}

kde:

- zař.1 je zdrojové a zař.2 cílové zařízení. Není-li uvedeno předpokládá se "D1". Jako zař.1 nelze použít "C:", ale je možné "E:" a "K:", "P:".
- jméno1 se musí uvést u souborů na disku
- jméno2 se musí uvést u souborů na disku
- "\*" je povinná mezera

### Povel DUPDSK

Povel je realizován programem DUPDSK.COM a je řízen dialogově. Umožňuje kopírovat diskety i na jedné jednotce. Lze jím kopírovat diskety v jednoduché hustotě.

### DUPDSK

Pokud je cílová disketa již formátovaná, lze formátování vynechat a ušetřit tak něco času. Výměna zdrojové a cílové diskety je řízena pokyny podobně, jako funkce C nebo O u DOS 2.5. Na konci kopírování je učiněn dotaz, zda chceme kopírovat další disketu. Pokud ne, stisk klávesy RETURN způsobí nové zavedení systému studeným startem.

### Povel INIT

Povel je realizován programem INIT.COM, který je řízen dialogově. Menu nabízí možnost pouze formátovat, formátovat a zapsat DOS.SYS, jen zapsat DOS.SYS a návrat do vykonavače poveli OS/A+.

### INIT

Povel INIT v sobě slučuje funkce P a H DOSu 2.5 v různých kombinacích.

### Povel HELP

Povel je realizován programem HELP.COM a vypisuje základní povely OS/A+.

### HELP

### Povel EASMD

Povel EASMD je realizován programem EASMD.COM. Jde o vývojový systém pro Assembler, ve kterém je integrován textový editor, překladač a ladící program. Funkčně je shodný s Assemblerem/Editorem firmy Atari v zásuvném

modulu, má ale navíc direktivu .INCLUDE, která umožnuje mít text rozsáhlého programu rozdělen na několik editovatelných částí. Při komplikaci se direktivou .INCLUDE řídí začleňování jednotlivých částí do sebe. Podrobnosti práce s EASMD přesahují rámec této příručky. Více informací lze najít v jiných publikacích Atari klubu Praha a sice v referenční příručce MAC/65 a překladu knihy Marka Chasina Programování počítačů Atari.

### EASMD

Assembler EASMD je již značně starý a byl překonán řadou makroassemblerů MAC/65. Má však jednu vlastnost, pro kterou stojí za to jej uchovávat. Je relokativní, to znamená dovede se přemístit na volné místo paměti a je tak vlastně jediným systémem, pomocí nějž lze studovat obsah zasunutého modulu. Další vlastností je, že vytvořený segmentovaný kód je dělen na segmenty podle nastavení čítače adres. Výsledný zkompilovaný program má tolik segmentů, kolikrát byl čítač příkazem "\*" změněn. Tím se vytváří nej-kompaktnější možný kód s rychlým zaváděním. Na druhou stranu je však komplikace velmi pomalá a EASMD se proto hodí jen na menší programy.

### Povely BASIC300 a odvozené

Povely jsou vyvoláním různých verzí Basicu A+. Číslo za klíčovým slovem znamená číslo verze, třeba BASIC304 vyvolá Basic A+ verze 3.04.

**BASICnnn x [jméno]**

kde:

- nnn je číslo verze bez tečky.
- jméno je jméno programu, který se má načíst a spustit

Známy jsou verze 3.00 až 3.05. Basic A+ je pravděpodobně předchůdcem známých Basiců XL a XE. Oproti Atari Basicu obsahuje navíc diskové povely, podporu grafiky hráčů a střel Palyer-Missile Graphics a další. Pro zajímavost výčet klíčových slov:

REM, DATA, LIST, ENTER, LET, IF, FOR, NEXT, GOTO, GO TO, GOSUB, TRAP, BYE, CONT, CLOSE, CLR, DEG, DIM, WHILE, ENDWHILE, TRACEOFF, TRACE, ELSE, ENDIF, END, NEW, OPEN, LOAD, SAVE, STATUS, NOTE, POINT, XIO, ON, POKE, DPOKE, PRINT, RAD, READ, RESTORE, RETURN, RUN, STOP, POP, GET, PUT, LOMEM, DEL, RPUT, RGET, BPUT, BGET, TAB, CP, DOS, ERASE, PEEK, DPEEK, PROTECT, UNPROTECT, DIR, REN, MOVE, COLOR, GRAPHICS, PLOT, POSITION, DRAWTO, SETCOLOR, LOCATE, SOUND, LPRINT, CSAVE, CLOAD, MISSILE, PMCLR, PMCOLOR, PMGRAPHICS, PMMOVE, PMWIDTH, SET, LVAR, L

### 6.3. Tvořba povelového souboru pro dávkové zpracování

OS/A+ má možnost přijímat povely nejen z klávesnice, ale i ze souboru se smluvným jménem STARTUP.EXC. Jde o textový soubor, obsahující různé povely interní i externí, které již byly probrány. Kromě toho může obsa-

hovat i speciální povely, které mohou být jen v povelovém souboru. Soubor STARTUP.EXC se provádí po studeném startu a spuštění OS/A+. Povely se provádí jeden po druhém. Pokud je povelem vyvolán program, převezme řízení. Po jeho skončení pokračuje sekvence dalším povelem. Takovému druhu práce se říká dávkový režim, nebo dávkové zpracování. Povely se provádějí podle povelového souboru v dálce a uživatel do nich nemůže zasahovat. Dávkové zpracování je účelné buď při opakovaném provádění určité posloupnosti povelů, nebo pro automatické nastavení a spuštění nějakého uživatelského systému s maximálně jednoduchou obsluhou. Všechny potřebné operace se uloží do STARTUP.EXC a uživatel se nemusí o nic starat. Jen zapne počítáč.

#### **Speciální povely pro dávkové zpracování**

V souboru profižení dávky STARTUP.EXC se mohou uvádět libovolné externí i interní povely OS/A+. Kromě toho jsou povely, které se mohou vyskytovat výhradně v dálce povelů. Před každým povelem se píše číslo řádky.

#### **Poznámka**

**číslo REM [ text ]**

kde:

- číslo je číslo řádku
- text je libovolná poznámka

Poznámky se vypisují na obrazovku a slouží jako vysvětlující texty, nebo pokyny pro obsluhu.

#### **Potlačení výpisu**

**číslo NOSCR**

Povel potlačí veškeré výpisy na obrazovku při provádění povelového souboru.

#### **Obnovení výpisu**

**číslo SCR**

Obnoví výpisy na obrazovku.

Příklad povelové sekvence:

```
100 REM +++ Příklad sekvence pro OS/A+
110 NOSCR
120 PRO *.ASM
130 EASMD
140 UNP *.ASM
150 SCR
160 DIR
170 END
```

Při vyvolání EASMD se provádění dávky zastaví a pokračuje řádkou 140, jakmile se v EASMD poprvé zadá CP nebo DOS.

#### **6.4. Vazba programu na OS/A:**

Program spouštěný jako externí povel má možnost si z povelové řádky přečíst parametry a i jinak spolupracovat s OS/A+. K tomu je třeba znát umístění pointerů a řídících tabulek. Některé důležité adresy jsou v následující tabulce (adresy a pointery mají délku dvou bytů. Slova, indexy jsou jedno-bytové).

; pointer umístění command processoru		
CPALOC	-	\$0A
; indexy funkcí a hodnot CP, dosažitelné nepřímo přes ; CPALOC, t.j. (CPALOC).Y		
CPGNFN	-	3
; čti další jméno souboru		
CPDFDV	-	7
; aktuální disk		
CPBUFP	-	\$0A
; index dalšího znaku v povelo- ; lovém bufferu		
CPEXFL	-	\$0B
; Execute Flag		
CPEXFN	-	\$0C
; jméno příkazového souboru		
CPEXNP	-	\$1C
; sektor a byte v přík. soub.		
CPFNM	-	\$21
; buffer jména souboru		
RUNLOC	-	\$3D
; zav./rozb. adresa CP/A		
CPCMDB	-	\$3F
; povelový buffer		
CPCMDGO	-	\$F3
; hodnoty příznaku Execute Flag		
EXCYES	-	\$80
; povел probíhá		
EXCSCR	-	\$40
; opisování povelů na obraz.		
EXCNEW	-	\$10
; modus execute start up		
EXCSUP	-	\$20
; příznak provedení stud.startu		

Na základě hodnot CPBUFP a CPCMDB v tabulce může program načítat parametry a podle jejich hodnot řídit svou práci. Podle stejného principu je realizována spolupráce externích povelů a DOSu u DOS XL a Sparta DOS. Adresa RUNLOC je používána u všech externích povelů. Nejsou-li definovány startovací a inicializační adresy, použije se RUNLOC i jako startovací adresa.

## 7. DOS XL

DOS XL je další z diskových operačních systémů firmy OSS a zdokonaluje předcházející systém OS/A+. Na rozdíl od OS/A+ je DOS XL systém smíšeného typu. Kromě povelového procesoru má i menu, které je alternativou pro méně vyspělé uživatele. Zda se po nastartování DOS XL objeví výzva modulu, menu, nebo výzva vykonavače povelů, záleží na více okolnostech, které budou dálé popsány. DOS XL pracuje buď s jednoduchou, nebo dvojitou hustotou. Bohužel ani on neumí hustotu rozšířenou, ale s jednotkou XF551, nebo jinou umožňující dvojitou hustotu by mělo jít pracovat vedvojité hustotě, která má největší kapacitu. Menu je ve stylu DOS3, povelový procesor DOS XL odpovídá procesoru OS/A+. Navíc jsou tu další externí povelty týkající se dvojité hustoty a povely, jejichž funkci lze bez uživatelské dokumentace těžko rozluštit. Interní povelty DOS XL jsou stejné, jako u OS/A+, ale mají rozšířeny některé funkce. Cenou za nové funkce je značné prodloužení systému. Protože řada starších programů předpokládá MEMLO na \$1F00, má DOS XL dvě varianty. Normální varianta má MEMLO na \$2C00 pokud je aktivní menu, a \$2200 pokud není. DOS XL je v paměti v celku. U modelů XL a XE lze použít druhou variantu, kdy je DOS XL roztržen na dvě části. První část je v obvyklém místě, druhá je schována do RAM pod ROM operačního systému od \$C000. MEMLO je v tomto případě \$1B00, nebo bez menu \$1100. V každém případě je DOS XL v paměti celý (včetně menu, pokud je aktivní). Okamžitý přechod mezi DOS XL a aplikačním programem je zachován, pravidla pro uchování obsahu uživatelské paměti při této přechodech jsou stejná, jako u Happy DOSu a OS/A+. Povely pro DOS XL lze psát i malými písmeny. Pokud není nějaký zvláštní důvod k používání DOS XL, doporučuji používat Sparta DOS, který má stejně omezuje podmínky, jako DOS XL (nelze provozovat s programy, zasahujícími do RAM nad \$C000), ale je podstatně dokonalejší.

### 7.1. Start DOS XL a volba varianty uložení v paměti.

Průběh startu DOS XL odpovídá obecnému schématu, uvedenému v kapitole 3.2. Popsaný průběh doplňuje detaily o činnosti větve "načtení a spuštění DOS.SYS" v případě DOS XL. Pokud neexistuje modul, načte se a nastartuje DOS XL, který testuje existenci AUTORUN.SYS. Pokud existuje, provede se. Potom se testuje existence STARTUP.EXC. Pokud existuje provedou se povely v souboru obsažené a řízení skončí ve vykonavači povelů DOS XL. Pokud neexistuje, testuje se existence MENU.COM. Pokud existuje, skončí řízení v menu, pokud neexistuje, aktivuje se vykonavač DOS XL.

Pokud existuje modul a vyžaduje DOS, načte se a spustí DOS XL, který nejprve testuje existenci AUTORUN.SYS a pokud existuje, provede se. Poté se zkoumá existence STARTUP.EXC. Pokud existuje, provede se a řízení se předá příkazovému procesoru DOS XL. Pokud neexistuje, načte se MENU.COM, pokud existuje předá se řízení modulu. Přechod z modulu do DOSu vede buď do menu, pokud existovalo, nebo do DOS XL, pokud MENU.COM na disketě nebylo. Tento spletitý průběh lze znázornit ve stylu jazyka Pascal. Při tom používáme fiktivní proceduru exist(jméno), která vrací hodnotu "pravda", pokud na

disketě existuje soubor "jméno". Pojmem DOS XL se bude rozumět příkazový procesor, příkaz "jméno" znamená provedení příslušného souboru. Průběh tedy lze popsat takto:

```

if exist(modul) then
begin
    if vyžaduje DOS then
begin
    start DOSu;
    if exist(AUTORUN.SYS) then proved AUTORUN.SYS,
    if exist(STARTUP.EXC) then
begin
        STARTUP.EXC,
        DOS XL;
end else
begin
    if exist(MENU.COM) then načíst MENU.COM
    start modulu;
end
end else start modulu;
end else
begin
    start DOSu;
    if exist(AUTORUN.SYS) then AUTORUN.SYS;
    if exist(STARTUP.EXC) then
begin
        STARTUP.EXC,
        DOS XL;
end else
    if exist(MENU.COM) then MENU.COM else DOS XL;
end;

```

AUTORUN.SYS musí být binární segmentovaný soubor ve standardním tvaru a mohl být proveden, musí být vybaven příslušnými startovacími a rozberovými adresami. STARTUP.EXC je povelový soubor, který může obsahovat interní i externí povely DOS XL a speciální povely pro dávkové zpracování. DOS XL znamená vykonavač povelu, načtení MENU.COM znamená že se program načte do paměti, ale aktivuje se až po přechodu z modulu do DOSu. Výraz MENU.COM znamená, že se aktivuje menu.

DOS XL je na disketě uložen v souboru DOS.SYS. Kromě toho jsou na distribuční disketě soubory DOSXL.SUP a DOSXL.XL. Standardně se DOS XL instaluje v první variantě umístění vcelku s vysokým MEMLO. Druhou variantu zvolíme tak, že soubor DOSXL.XL přejmenujeme na DOSXL.SYS. Po novém nastartování například povelem RUN v DOS XL, nebo G v menu na adresu E477 se nainstaluje druhá varianta. Kromě nízké hodnoty MEMLO ji poznáme podle hlavičky vykonavače povelu DOS XL, který má hlavičku s číslem verze 2.3X, zatímco první varianta má v hlavičce 2.3.

## 7.2. Menu

Menu se aktivuje, pokud je na disketu soubor MENU.COM a není STARTUP.EXC. Není-li aktivní zásuvný modul, dostaneme se do menu po nastartování DOSu. Je-li modul aktivní, dostaneme se do menu povelom pro přechod do DOSu "DOS", nebo někdy "CP" (MAC/65, BASIC A+ apod.). Menu je v paměti rezidentní a spotřebujecca 2kB paměti. Lze jej opustit volbou "Q", ale MEMLO se tím již nesníží. Pokud potřebujeme maximum paměti, musíme menu "zneškodnit" již při startu DOSu pomocí souboru STARTUP.EXC, ve kterém stačí uvést třeba i jen komentář, nebo soubor MENU.COM zrušit. Menu vypadá takto:

```
DOS XL MENU      version 2.30
copyright (c) 1983 OSS, Inc.
```

<b>F</b> iles on Disk	<b>P</b> rotect Files
<b>T</b> o Cartridge	<b>U</b> nprotect Files
<b>C</b> opy Files	<b>R</b> ename Files
<b>D</b> uplicate Disk	<b>S</b> ave Binary
<b>E</b> rase Files	<b>L</b> oad Binary
<b>I</b> ntialize Disk	<b>G</b> o To Address
<b>X</b> tended Command	<b>Q</b> uit to DOS XL

Enter your selection.

Většinu funkcí známe z DOS 2.5, i když pod jinými písmeny. Požadovanou funkci zvolíme písmenem, které je zobrazeno na začátku každého řádku invertzně. Každou funkci menu lze na rozdíl od DOS 2.5 zrušit klávesou ESC. Menu klade dotazy na parametry a pomocí nich připraví povelovou řádku, kterou předá vykonavači povelů DOS XL. Proto se menu hodí zejména pro začátečníky a uživatele neobeznámené s povely pro DOS XL. Je samozřejmé, že práce s menu je pomalejší, než s povelovou řádkou. Nyní popíšeme jednotlivé položky nabídky z menu:

### **F**iles on Disk

Výpis adresáře, odpovídá funkci A v DOS 2.5 a povelu DIR. Menu se táže: File Spec:, na kterou odpovíme stejně jako v DOS 2.5. Standard je "D1:.\*.\*.E\*", takže pro výpis celého adresáře na obrazovku stačí zadat (RETURN).

### **T**o Cartridge

Aktivace modulu, odpovídá B a CAR.

### **C**opy File

Kopírování souborů. Odpovídá funkcím C a O DOSu 2.5 a externímu povelu COPY v DOS XL. Menu se ptá:

**From File:** vstupní soubor, originál  
**To File:** výstupní soubor, kopie  
**Single Drive?** Y nebo N,  
**Insert disk(s) to be copied**  
**and hit return when ready**

**Copy xxxxxxxxxxxx**  
**to yyyyyyyyyy?■ Y nebo N**

Na dotaz From file zadáme specifikaci souboru a/nebo zařízení originálu (může být filtr). To File se nemusí psát, pokud kopirujeme soubor na jinou disketu pod stejným jménem. Do parametru To File se nejprve přebere celý From File a přepíše se zadáním z klávesnice. To co nenapišeme, zůstane ze vstupu. Dotaz Single Drive znamená volbu, zda se má kopirování pokaždé mezi čtením originálu a zápisem kopie zastavit s výzvou ke vložení příslušného disku. Tento dotaz se objevuje i při kopirování z E: na P:, kde disky nejsou. Poté se načte program COPY.COM a požaduje připravit média ke kopirování. provedení akce se oznámi klávesou (RETURN). Dále se vypíše platný obsah parametrů From File a To File s dotazem, zda kopirovat. Y kopii spustí, N nebo cokoliv jiného zruší. Funkcí C lze pracovat i s kazetou v startstopním režimu - dlouhé meziblokové mezery. Na závěr se napíše copied, nebo not copied podle výsledku operace a po klávesce (RETURN) se vrátíme do menu.

#### **Duplicate Disk**

Odpovídá funkci J DOSu 2.5 a externím povelům DUPDSK a DUPDBL. Po zadovězení dotazu, zda jde o dvojitou hustotu (Y/N) se vyvolá příslušný kopirovací program, který se dotáže na číslo jednotky originálu a kopie a pokud jsou stejná, říká si o výměnu původního disku a kopie.

#### **Erase Files**

Rušení souborů. Na dotaz Filespec to erase napišeme jméno nebo filtr. Poté musíme potvrdit operaci písmenem Y. Vybraný soubor, nebo skupina se bez výjimky zruší, rušení jednotlivých souborů se nepotvrzuje jednotlivě jako v DOS 2.5 a tak je třeba opatrnosti. DOS XL nemá prostředek k restauraci omylem zrušených souborů.

#### **Initialize Disk**

Nahrávají funkce P a H DOS 2.5. Realizuje se externím povelem INIT, který řídí další práci pomocí dialogu s uživatelem.

#### **Extended Command**

Tuto funkcí lze spustit externí povel, který není zařazen do menu, například INITDBL pro inicializaci diskety v dvojitě hustotě apod.

#### **Protect Files**

Blokování souborů - odpovídá F v DOSu 2.5 a povelu PRO.

**Unprotect Files**

Uvolnění souborů - G a UNP.

**Rename File**

Přejmenování souboru - E a REN

**Save Binary**

Uložení části paměti do souboru - K a SAV. Zadává se počáteční a koncová adresa.

**Load Binary**

Načtení a ev. spuštění programu. Odpovídá povelu LOA a L v DOS 2.5.

**Go To Address**

Skok na adresu. Odpovídá RUN a M v DOS 2.5.

**Quit to DOS XL**

Ukončení práce menu a přechod do povelového procesoru DOS XL.

**7.3. Povelový procesor (command processor) DOS XL.**

Povelový procesor se aktivuje vždy, když je na disketě STARTUP.EXC, nebo pokud se do něj přejde z menu povelem Q. Je velmi podobný procesoru u OS/A+, ale některé interní a většina externích povelů je rozšířena nebo změněna. Vysvětlení co jsou externí a interní povelů je v 6.1. a 6.2.

**7.3.1. Interní povelové**

**Výpis adresáře:** DIR\* [specifikace]

kde:

- "\*" je povinná mezera
- specifikace je filtr pro výpis adresáře. Standardní hodnota je Dl:\*.\*

Povel DIR je u DOS XL ekvivalentní funkci A u DOS 2.5. Pokud chceme uložit výpis adresáře jako soubor na disk, musíme napsat i filtr pro výběr, t.j. DIR \*.\*.D:jméno. Výpis na tiskárnu: DIR \*.\*.P:

**Přechod do modulu: CAR**

Zde byla opravena chyba OS/A+. Pokud neexistuje modul, to znamená není aktivován vestavěný Basic, ani není zasunut žádný modul, vypíše se zpráva NO CARTRIDGE.

**Rušení souborů:** ERA × [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "x" je povinná mezera

**Přejmenování souboru:**

REN × [zař.] jméno1 × jméno2

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno1 je úplné staré jméno nebo filtr, jméno2 je nové jméno nebo filtr. Pro oba filtry platí stejná pravidla, jako u DOS 2.5.
- "x" je povinná mezera

**Blokování a uvolnění souborů:**

PRO × [zař.] jméno

UNP × [zař.] jméno

kde:

- zař. je označení diskové jednotky (standard D1:)
- jméno je úplné jméno nebo filtr
- "x" je povinná mezera

**Uložení úseku paměti:**

SAV × [zař.] jméno × adr1 × adr2

kde:

- jméno je jméno souboru, do kterého se úsek paměti uloží
- adr1 je počáteční, adr2 koncová adresa úseku
- "x" je povinná mezera

**Načtení a spuštění binárního souboru:**

{ LOA × [zař.] jméno  
jméno }

kde:

- zař. je kód vstupního zařízení
- jméno je jméno souboru
- "x" je povinná mezera

Načtení binárního souboru se provede buď povelom LOA, nebo uvedením názvu souboru. Soubor musí mít strukturu binárního segmentovaného souboru (viz 3.5. a 4.1.11) a pokud má definovány příslušné spuštěcí adresy, je po načtení příslušné části spuštěn. Rozdíl mezi povelom LOA a přímým uvedením jména je v zacházení s koncovkou a ve způsobu spuštění programu. Není-li uvedena, předpokládá se u LOA, že koncovka není. U přímo uvedeného jména bez koncovky se předpokládá popisná část .COM. Proto chceme-li

spustit program s jménem bez ní, musíme za jménem napsat ještě tečku. Napíšeme-li "PROGRAM", hledá se "D1:PROGRAM.COM", zatímco "PROGRAM." znamená "D:PROGRAM". Načítání s přímým uvedením jména umí spustit jak standardní binární segmentovaný soubor, tak program ve speciálním formátu externího povelu. Při načítání s klíčovým slovem LOA musí být program ve standardním tvaru.

**Skok na adresu:** RUN x [adresa]

kde:

- adresa je hexadecimálně zadaná adresa, na kterou skáčeme
- "x" je povinná mezera

Povel RUN bez uvedené adresy skočí buď na začátek naposledy nastartovaného programu (povelem LOA nebo jako externí povel), nebo na adresu, zadanou předchozím povelem RUN podle toho, co se vykonalo naposled.

### 7.3.2. Externí povel

Externí povel mají stejné vlastnosti a strukturu, jako u OS/A+, kde jsou také obecně popsány (viz 6.2).

#### Povel COPY

Je realizován souborem COPY.COM a slouží ke kopírování souborů obecně s tím, že má možnost zařadit pauzu mezi čtením originálu a zápisem kopie. Proto lze povelom COPY kopírovat cokoliv na cokoliv včetně kazety v modu dlouhých mezibílkových mezér.

formát povelu: COPY x [zař.1|[zař.1]jméno1] x [zař.2|[zař.2]jméno2]  
[x - [q][f][s][w]]

kde:

- zař.1 je zdrojové a zař.2 cílové zařízení. Není-li uvedeno předpokládá se "D1".
- jméno1 se musí uvést u souboru na disku
- jméno2 a zař.2 se nemusí uvádět. V tom případě se vynechaná část přebírá ze zař.1 a jména1
- "x" je povinná mezera
- "-" je povinné minus, označující parametry

Parametry q, s, w, f řídí průběh kopírovacího procesu. Parametr "q", neboli query (dotaz) způsobí, že se před kopírováním vypíše specifikace originálu i kopie s dotazem, zda se má kopírovat. Odpověď "Y" spustí, cokoliv jiného zruší kopii. Parametr "s" znamená switch, neboli vyměňování originálu a kopie. Mezi každým čtením a zápisem se program zastaví a požaduje vložit patřičný disk. Za zmínu stojí, že se tento parametr používá zcela tupě, protože vyžaduje výměnu disků i když nekopírujeme na disk, nebo když se pracuje s různými jednotkami. Parametrem "w" se po načtení programu

COPY.COM a před spuštěním kopie počítač zastaví a požaduje vložení originálu a kopie. Po stisku klávesy (RETURN) se začne kopirovat. "f" znamená force. Program COPY přepíše již existující soubor novým jen tehdy, když je uveden tento parametr. Pokud uveden není, skončí činnost s výpisem "File already exists" neboť soubor již existuje a nezkopíruje nic.

#### **Povel DUPDSK a DUPDBL**

Povely jsou realizovány programy DUPDSK.COM a DUPDBL.COM. Jsou řízeny dialogově. Umožňují kopírovat diskety i na jedné jednotce. DUPDSK pracuje s jednoduchou, DUPDBL s dvojitou hustotou.

formát povelu: DUPDSK nebo DUPDBL

Pokud je cílová disketa již formátovaná, lze formátování vynechat a ušetřit tak něco času. Výměna zdrojové a cílové diskety je řízena pokyny podobně, jako funkce C nebo O u DOS 2.5. Na konec kopírování je učiněn dotaz, zda chceme kopírovat další disketu. Pokud ne, stisk klávesy RETURN způsobí nové zavedení systému studeným startem.

#### **Povel INIT a INITDBL**

Povel je realizován dialogově řízenými programy INIT.COM a INITDBL.COM. V INIT nabízí menu možnost jen formátovat, formátovat a zapsat DOS.SYS, jen zapsat DOS.SYS a návrat do povelového interpretu DOS XL. DUPDBL žádné menu nemá, formátuje v dvojitě hustotě jen jedním možným způsobem. Z toho vyplývá, že DOS XL nemůže být spouštěn z diskety ve dvojitě hustotě.

formát povelu: INIT nebo INITDBL

Povel INIT v sobě sloučuje funkce P a H DOSu 2.5 v různých kombinacích.

#### **Povel MENU**

Povel je uskutečněn programem MENU.COM. O funkcí menu jsme se již zmínili v 7.1. Povel MENU slouží k přechodu z vykonavače povelů DOS XL do menu, ať už jsme se do povelového procesoru DOS XL dostali jakýmkoliv způsobem.

formát povelu: MENU

#### **Povely VERIFY a NOVERIFY**

Povely slouží k zapnutí nebo vypnutí verifikace zápisu, t.j. kontrolního čtení každého sektoru bezprostředně po jeho zapsání.

formát povelu: VERIFY  
NOVERIFY

**Povel CONFIG**

Slouží jednak ke zjišťování, jednak k nastavování konfigurace programovatelných disketových jednotek. Bohužel skladbu parametrů lze bez dokumentace jen velmi těžko rozluštit. Proto doporučujeme pro práci s dvojitou hustotou Sparta DOS, který může být nasazen za stejných podmínek jako DOS XL a práci s různě formátovanými disketami zvládá bez problému.

**povel DO**

Slouží ke spuštění programů a to jak systémových ve zvláštním formátu externích povelů, tak segmentovaných binárních souborů. Jeho účel a použití je nejasné, protože programy lze spouštět interněm povelom LOA nebo uvedením jména programu.

**povel RS232**

Slouží k načtení handleru zařízení R: z interfejsového modulu Atari 850.

**povel RS232FIX**

Slouží k opravě chybových stavů při sériové komunikaci přes port RS232.

**povel BUG65**

Slouží kladění a testování strojových programů. Popis přesahuje rámec této příručky.

**povel MAC65**

Spuštění makroassembleru MAC/65 v disketové verzi. Popis MAC/65 byl publikován v překladu referenční příručky, vydaném Atari klubem Praha.

**7.4. Tvorba povelového souboru pro dávkové zpracování**

DOS XL má sejně jako OS/A+ možnost přijímat povelы nejen z klávesnice, ale i ze souboru se smluvným jménem STARTUP.EXC. Popis je uveden u OS/A+ v odstavci 6.3., proto je zde již nebudeme opakovat.

**7.5. Vazba programu na DOS XL**

Program spuštěný jako externí povel má možnost si z povelové řádky přečíst parametry a i jinak spolupracovat s DOS XL. K tomu je třeba znát umístění pointerů a fidscích tabulek. Popis této vazby je v 6.4.

**7.6. Možnost RAM-disku**

RAM-disk zajímá zcela samozřejmě všechny majitele počítačů s rozšířenou pamětí. DOS XL sice nemá tuto možnost zabudovanou, ale handler RAM-disku lze při startu načíst. Jeden z více publikovaných programů pro instalaci RAM-disku je v časopise Happy Computer číslo 11, ročník 1985, strana 121, kde je RAM-disk pro modely 800XL, využívající paměť pod ROM operačního systému. Tento způsob není slučitelný s variantou "X" DOS XL. Logika programu je však v časopise dobře popsána a lze jej upravit i pro rozšířenou paměť.

## 8. Bibo DOS

Bibo DOS je první z univerzálních systémů vhodných pro jednotky 1050 i XF551. Pochází na rozdíl od všech dosud popisovaných systémů z NSR a dodává jej firma COMPY-SHOP, známá svou řízení vylepšení a doplňků osmibitových počítačů Atari a jejich diskových jednotek. Bibo DOS byl původně napsán pro Speedy 1050 (verze 5.4 N a S). Firma upravila svůj systém také pro jednotku XF551. Řada upravených verzí začíná označením 6.0 N z února 1988, nejnovější verze dorazivší do našich konců má číslo 6.4 N. Autorem je Erwin Reuß, kterého známe z nejpopulárnějšího programu v našich klubech - US Copy. Bibo DOS na nás sice mluví německy, ale zachovává styl DOS 2.5 a je na obsluhu stejně nenáročný. To dává pravděpodobnost, že bude přes určité nedostatky ze systémů pro oboustrannou dvojitou hustotu nejpopulárnější. Bibo DOS je fízen pomocí menu. Jeho hlavní rysy udává tabulka:

- ovládání pomocí menu
- po volbě funkce se nemačká RETURN
- nepoužívá standardní editor -> chyby nelze opravit přepsáním, chybnou řádku musíme napsat znova
- formáty záznamu: SD, ED, DD, DS/DD
- rezidentní RAM-disk 64, 128 nebo 256 Kb s volbou čísla jednotky, programátorskou úpravou lze přizpůsobit i jiným typům
- vyšší MEMLO - \$2193 pro základní konfiguraci jako u DOS 2.5
- funkce obnovy zrušeného scuboru
- generování automaticky prováděných povelů při startu
- některé funkce mají jiné označení, než u DOS 2.5
- možnost klávesového bufferu
- nemá funkci duplikování disku
- nevyužívá rychlého modu u XF551
- má neměnnou tabulku vektorů pro služby systému
- kombinací kláves TAB-CTRL-SHIFT vybudí studený start
- používá RAM pod CS-ROM
- zatím neexistuje pro XF551 verze S, využívající rychlý režim
- pracuje plnohodnotně s kazetou ve standardním režimu

Jednoduchost Bibo DOSu je připomíná oboustrannou jednotkou XF551 nevhodnou, protože za určitých okolností nemůže využít její kapacitu. Na disketu se dá uložit maximálně 128 souborů, a to při kapacitě 360 Kb nemusí stačit.

Pod hlavičkou Bibo DOSu je informační řádka, která ukazuje momentální konfiguraci systému. Nejprve jsou uvedeny aktívni jednotky. V seznamu jsou zobrazeny inverzně. Další je informace o počtu souborových bufferů, který určuje maximální počet současně otevřených souborů. Dále je momentální konfigurace RAM-disku. Pokud není RAM-disk instalován, je zde "--", v opačném případě číslo jednotky a je-li RAM-disk rezidentní, je za číslem jednotky písmeno R. Poslední informace v řádce je jeho velikost v kB. RAM-disk je vždy vytvořen v dvojitě hustotě s délkou sektoru 256 byte. U Bibo DOS nesmí začínat jméno souboru číslici. Bibo DOS je na disketu uložen v souborech BDOS.SYS (rezidentní část) a BDUP.SYS (menu) u verze 5.x, nebo

XDOS.SYS a XDUP.SYS u verze 6.x. Díky vyššímu MEMIO ale hlavně díky používání RAM pod OS-ROM (schovává se tam BDUP.SYS během práce modulu nebo programu), s Bibo DOS některé programy nechodi vůbec, nebo jen částečně (např. Turbo Basic, SpeedScript, Čapek atd.). Bez problému jsou systémy v modulech. Verze, která je momentálně k dispozici nevyužívá rychlého modu XF551. V blízké budoucnosti se však dá očekávat rychlá verze, která by mohla mít označení S, protože pro své modifikované jednotky 1050 Compy Shop dodává i rychlé verze Bibo DOSu.

```
-----  

<< BIBODOS >> Version: 6.4RN  

(p) E.Reuss (c) 06/88 COMPY SHOP  

-----  

D:12345678     BUf:2    RD:8R    RS:128k  

-----  

1-8: Directory Dn   ^i_8: Dir/Spezial  

A: Disk Inhalt      H: Dos schreiben  

B: Zum Basic/Modul I: Formatieren  

C: Datei kopieren  J: Zurueckholen  

D: Datei loeschen   K: Binaer Save  

E: Namen aendern   L: Binear Load  

F: Datei sichern   M: Start ab Adr  

G: Datei freigeben N: Startup Edit  

-----  

Kommando: ■
```

#### menu Bibo DOS 6.4

##### 8.1. Menu Bibo DOS

###### **Výpis adresáře - Directory Dn, Dir Spezial, Disk Inhalt**

Bibo DOS má tři prostředky pro výpis adresáře diskety. Uvedením čísla jednotky se vypíše celý adresář na obrazovku. Tento způsob je velmi rychlý. Pokud spolu s číslem jednotky stiskneme ještě (SHIFT), vypisuje se také jména zrušených a neužavených výstupních souborů. Přitom se v prvním sloupci výpisu kromě obvyklé mezery nebo hvězdičky objevuje ve výpisu adresáře před jménem znak "-" u zrušených a "?" u neužavených výstupních souborů. Potřebujeme-li vypsat adresář na tiskárnu, nebo hledáme určitý soubor nebo skupinu pomocí filtru, musíme použít funkci A a na dotaz Dateiname odpovíme

{jméno|filtr} [/P] [/W]

Standardní odpověď je "D:.\*.\*", výpis na obrazovku. Uložení obsahu adresáře do souboru na disk není na rozdíl od DOS 2.5 možné. Výpis adresáře má tvar zcela shodný s výpisem v DOS 2.5, včetně označení oblasti rozšířené hustoty ostrými závorkami. V záhlaví výpisu se navíc uvádí hustota, ve které je disketa nahrána. Jde buď o jednoduchou (SINGLE), rozšířenou (MEDIUM), dvojitou (DOUBLE) nebo dvojitou oboustrannou (DS/DD), která se někdy také označuje terminem QUAD. Parametrem "/P" přepneme výstup adresáře zobra-

zovky na tiskárnou, "/W" způsobí výpis na obrazovce do dvou sloupců vedle sebe. Účinek tohoto parametru trvá i u následujících výpisů adresáře spuštěných číslem jednotky.

#### **Přechod do modulu - Zum Basic/Modul**

Předávání řízení z menu DOSu do vestavěného Basicu, nebo zasunutého modulu. Přechod je okamžitý tam i zpět díky tomu, že se BDUP.SYS během aktivity modulu uloží do RAM pod OS-ROM (od \$E400 výše). Z toho vyplývá, že program nesmí do uvedené oblasti zasahovat. Pokud v menu nepoužijeme povel C je uživatelská paměť po návratu do modulu neporušena, i když není RAM-disk konfigurován. Stejně jako u ostatních DOSů se poškodi uživatelská paměť kopirováním.

#### **Kopírování souborů - Datei kopieren**

Funkce C v sobě slučuje funkce O a C DOSu 2.5. Řídi se zadáním souborů ke kopírování. Uvede-li se jen originál, nebo u originálu a kopie stejné zařízení, předpokládá se kopie pod stejným nebo jiným jménem na jinou disketu a systém požaduje vložení zdrojového (Original) a cílového (Ziel) disku. Standardní filtr je \*.\* a při kopírování celého obsahu disket stačí uvádět jen číslo jednotky. Proto nesmí v Bibo DOS začinat jméno číslicí. Přepínač /Q uvedený za specifikací filtru dává u každého souboru možnost určit, zda se má kopírovat. Standardně se nabízí  (ano). Nechceme-li soubor kopírovat, napišeme . Je-li uvedeno místo filtru jméno, nemá tento přepínač účinek. Další přepínač "/V" kopíruje na cílovou disketu jen soubory, které na ní ještě nejsou. Na rozdíl od DOS 2.5 lze kopírovat z kazety i na kazetu. Standardně je záZNAM s krátkými meziblokovými mezerami. Potřebujeme-li pracovat s dlouhými mezerami (start-stopní režim), uvedeme přepínač "/L".

Příklady:

Kopie všech souborů z jednotky 1 na jednotku 2 (2 může být i RAM-disk) podle dodatečného výběru:

```
Kommando: C
Datei kopieren, (nach):
1,2 /Q (RETURN)
```

Všechny soubory, (pozor - včetně BDOS.SYS (XDOS.SYS), na kopii ale nebude fungovat!) se zkopiují z disku 1 na disk 2. Soubory, které na disku 2 již existují, se přepíší pokud nejsou blokovány. Kopie všech souborů chybějících na cílové disketě pomocí jedné jednotky:

```
Kommando: C
Datei kopieren, (nach):
1/V (RETURN)
```

Kopírování se mezi každým členem a zápisem zastaví a systém požaduje vkládat originál (Original einlegen (RETURN)), nebo kopii (Ziel einlegen (RETURN)). Provedení požadavku se oznamuje klávesou (RETURN). Kopírování souborů mezi disketami v různých hustotách je diskutováno v odstavci 8.4. Ko-

pirování na kazetu s krátkými meziblokovými mezerami:

Kommando: C  
 Datei kopieren, (nach):  
 PROG.COM,C: (RETURN)

Totéž s dlouhými mezerami:

Kommando: C  
 Datei kopieren, (nach):  
 PROG.COM,C:/L (RETURN)

#### Rušení souborů - Datei loeschen

Obdoba funkce D v DOS 2.5. Liší se kratším průběhem dotazování, zda zrušit nebo nezrušit soubor. Předpokládaná odpověď je "N" (Nein). Pokud soubor zrušítechceme, zadáme "J" (Ja - ano). Musíme si uvědomit, že v Bibo DOSu konverzujeme německy, a že navýklá standardní odpověď "Y" zde znamená totéž, co "N". Verifikacirušení každého vybraného souboru potlačíme parametrem "/N", který se píše bez mezery hned za jméno nebo filtr. Zrušený soubor lze obnovit funkcí J, pokud se mezitím nepokodil zapsáním jiných dat.

Příklady: Rušení všech programů v Basicu (koncovka .BAS):

Kommando: D  
 Loesche - Dateiname:  
 \*.BAS (RETURN)  
 Loesche --> D1:XX.BAS (H)

N přepíšeme na J a soubor se zruší. Klávesa (RETURN), nebo cokoliv jiného kromě "J" soubor zachová. Rušení téhož bez kontrolního dotazu se provede takto:

Kommando: D  
 Loesche - Dateiname:  
 \*.BAS/N (RETURN)

Parametr /N se píše bez mezery. Bibo DOS zruší bez dalších dotazů celou vybranou skupinu.

#### Přejmenování souboru - Namen ændern

Funguje stejně, jako v DOSu 2.5, pouze výzvy k zadání parametrů jsou německy:

Kommando: E  
 Alter, neuer Dateiname:  
 2:BDUP.SYS,\*.COM (RETURN)

#### Blokování (zamknutí) souboru - Datei sichern

Provádí se stejně, jak v DOS 2.5:

Kommando: F  
 Sichern - Dateiname:  
 \*.SYS (RETURN)

**Uvolnění souboru - Datei freigeben**

Rovněž stejně, jako v DOS 2.5:

Komando: G  
 Freigeben - Dateiname:  
 Z:BDUP.COM (RETURN)

**Instalace DOSu - Dos schreiben**

Instalace se poněkud liší od podobné funkce DOS 2.5, je v ní vlastně zahrnutá funkce programu SETUP.COM. Pokud potřebujeme jinou konfiguraci, než nabízí instalace, máme možnost změnit vlastním programem tabulkou parametrů a potom tepřive funkcí H uložit změněný BiboDOS na disketu. Přitom již nevolíme možnost další změny konfigurace. Dialog probíhá takto:

Komando: H  
 Dos schreiben LW :

První dotaz je na číslo jednotky, na kterou se má DOS zapsat. Standardně se nabízí 1, ale pokud inverzní jedničku přepíšeme, vyvoří se DOS na jiné jednotce. (RETURN) se nemačká.

BDUP schreiben J/N :

Volíme, zda se zapíše transientní složka Bibo DOSu BDUP.SYS, realizující menu. Standardně ano (Ja). BDUP nemusíme zapisovat pokud jen měníme konfiguraci Bibo DOSu na disketě. Nesmíme jej zapsat, pokud máme napsán svůj vlastní BDUP.SYS.

Konfigurieren J/N :

Sdělujeme systému, zda si přejeme změnit jeho konfiguraci. Standard je ne a pokud jej zvolíme, přeskočí se následujících 7 dotazů, ve kterých se stejně jako u jiných funkcí nabízí inverzně vyspaný standard, který můžeme buď potvrdit klávesou (RETURN), nebo změnit přepsáním na jinou hodnotu:

Anz. der Laufwerke :  2  
 Anz. der Buffer :  2  
 Ramdisk aktiv J/N :   
 Ramdisk Nummer :  3  
 Ramd. resident J/N :   
 Groesse der Ramdisk  
 1-64k 2-128k 3-256k:  3  
 Tastaturbuffer J/N :

První dotaz určuje počet konfigurovaných diskových jednotek. RAM-disk se do tohoto počtu nezahrnuje. Druhý řádek udává maximální počet současně otevřených souborů. Ve třetím řádku řekneme, zda chceme RAM-disk (samořejmě jen tehdy, když máme 130 XE, nebo model s dodatečně rozšířenou pamětí). Číslo, pod kterým se budeme na RAM-disk odkazovat můžeme na rozdíl od DOSu 2.5 volit z volných čísel jednotek. To znamená, že RAM-disk nemůže mít stejně číslo, jako některá z konfigurovaných fyzických jednotek.

Dále určujeme, zda je RAM-disk rezidentní. Rezidentní znamená, že jeho obsah zůstane zachován do té doby, než se počítač vypne. Studený start obsah rezidentního RAM-disku nepoškodí. Zvolíme-li RAM-disk nerezidentní, má stejné vlastnosti jako u DOS 2.5. Velikost RAM-disku můžeme volit ze tří možností. 256k a méně lze volit u modelů počítače s pamětí 320k, jaké prodává Compy Shop. Návod na toto rozšíření byl například publikován v Atari magazinu 2 a 3/87, české zpracování ve zpravodaji Atari klubu Praha č. 6/87. Velikost 128k je určena pro počítače s pamětí 192k, ale musíme ji použít i pro rozšíření na 256k (RAMBO XL, Newell Industries), pokud si neuděláme RAM-disk na míru změnou parametrické tabulky Bibo DOSu. 64k je jediná možná velikost pro standardní 130 XE, ale mohou ji použít i všechny ostatní rozšířené modely s řízením přepínání banků paměti registrém PORTB (\$D301). Při konfiguraci RAM-disku nás Bibo DOS klidně nechá napsat cokoliv, třeba velikost 256k i u běžné 130 XE a dokud je RAM-disk prázdný, ukazuje počet volných sektoriů jakoby byl skutečně tak veliký. Konec euforie nastane když do něj zapíšeme více, než 64k dat. Potom nastane zhroucení, provázené ztrátou, nebo zkomolemním stávajícího obsahu. RAM-disk se instaluje vždy ve dvojitě hustotě.

Poslední dotaz je na buffer klávesnice. Normálně máčíme systém vyrovnavací paměť jen na jeden znak. Buffer, který instaluje Bibo DOS pojme znaků více, až tedy psát "do zásoby" i když je počítač momentálně zaměstnán něčím jiným. Některé programy však s instalovaným bufferem nechodi. Jedním z nich je například ladící program DDT z modulu MAC/65. Pokud chceme s takovým programem pracovat, musíme změnit standardní volbu na "N". Závěrem se Bibo DOS ptá:

**Dos schreiben J/N :**

(RETURN) způsobí zápis nově konfigurovaného DOSu na disketu, N funkci zruší.

#### **Formátování disku - Formatieren**

Funkce formátování v sobě obsahuje funkce I a P DOSu 2.5 a navíc může formátovat ve dvojitě hustotě jednostranně i oboustranně. Jednoduchá hustota se zde označuje S - single, rozšířená M - medium, dvojitá D - double a oboustraná dvojitá XF - DS/DD (také Quad).

**Kommando: I**

**Formatiere Disk Nr. :**

**Density S/M/D/XF :**

**Formatiere D1 J/N:**

Příklad ukazuje standardní formatování, čili disk 1, jednoduchou hustotu a na bezpečnostní dotaz se nabízí odpověď "N". Pokud tedy mechanicky odmačkáme (RETURN) na všechny odpovědi, nic se nestane. Standardní volby ani nebude mnoho používat. U neupravené jednotky 1050 budeme nejspíše nastavovat rozšířenou hustotu "M", u XF551 oboustrannou dvojitou "X". Na poslední dotaz odpovíme "J" a formátování se spustí. RAM-disk se bez ohledu na nastavení parametrů formátuje v dvojitě hustotě. U verze 5.4 je místo volby XF volba C, t.j. zapsání nového prázdného adresáře a VTOC.

### **Formáty 1050 a XF55.**

U jednotky 1050 lze používat jednoduchou a rozšířenou hustotu. Rozšířená hustota má VTOC řešenou zcela shodně s DOS 2.5. U jednotky XF55 lze navíc použít další dva formáty, dvojitou hustotu a obostrannou dvojitou hustotu. Dvojitá hustota používá naprostostejnou logiku, jako jednoduchá tím rozdílem, že sektory jsou dlouhé 256 byte a tím pádem je i délka VTOC a adresáře dvojnásobná. Kapacita diskety je 180 KB, počet souborů max. 128, což je dosažující. Tento formát je též nazyván Bibo DOS formát a je používán i u Speedy 1050. Poslední formát je oboustranná dvojitá hustota (Quad, XF, DS/DD). Při realizaci tohoto formátu při dodržení potřebné míry slučitelnosti s předchozími formáty se naráží na různé potíže. Především je počet sektorů 1440 (1 až 720 na jedné, 721 až 1440 na druhé straně, přepínání stran je automatické. Jednotka se tedy vlastně jeví jakoby měla jednu stranu a 80 stop). Pro vyjádření adres sektorů v linkovaném formátu je třeba o jeden bit více než dřív. Protože kontrola příslušnosti k souboru není nezbytně nutná, je potlačena a uvolněný bit je používán pro číslo sektoru. Tak je možno pokrýt linkovaným formátem všechny 1440 sektorů, ale za cenu potlačení chyby 164. Na formátu Quad už nelze odhalit chybě propojené sektory. Tato vlastnost má vliv zejména na následující funkci J.

### **Obnovení zrušeného souboru - Zurueckholen**

Tato velmi užitečná funkce je v DOS 2.5 umožněna zvláštním služebním programem DISKFIX.COM. Bibo DOS ji má podstatně praktičtěji přímo v hlavním menu. Okolnosti, za kterých lze zrušený soubor obnovit, jsou již popsány v odstavci 4.4.2. Pokud se na disketu již psalo, mohou nastat tři možnosti. Buď je již místo v adresáři přepsáno novým názvem souboru, nebo záznam o zrušeném souboru v adresáři zůstal, ale jsou přepsána data, nebo máme štěstí a zápis se dat obnovovaného souboru nedotkl. První případ hlásí ERROR - #170 (soubor nenalezen), druhý ERROR - #174 (poškozená data). Ve třetím případě obnovení úspěšně proběhne.

Kommando: J

Zurueckholen - Dateiname:

2:\*.COM (RETURN)

**Pozor:** U disket ve formátu Quad je třeba zvláštní opatrnosti. Protože se zde nekontroluje příslušnost sektoru k číslu souboru v adresáři, je záruka bezchybného obnovení zrušeného souboru jen v případě, že mezi zrušením a obnovováním nebylo na disketu zapisováno. V opačném případě budou k souboru připojeny sektory z jiných souborů, aniž by o tom byla vydána zpráva. Tím nastane totální guláš, protože některé sektory pak patří ke dvěma souborům současně a co se stane v případě zrušení nebo modifikace jednoho z nich si čtenář jistě dovede představit.

### **Ukládání úseků paměti - Binaer Save**

Tato funkce je zcela identická s funkcí K v DOS 2.5.

### **Načítání binárních souborů - Binaer Load**

Tato funkce je zcela identická s funkcí L v DOS 2.5.

### **Skok na adresu - Start ab Adr.**

Tato funkce je identická s funkcí M v DOS 2.5. Navíc má zadáním hvězdičky místo adresy možnost skočit na startovací adresu naposledy spouštěného programu.

### **Tvorba startovací sekvence - Startup Edit**

Možnosti obsažené v DOS 2.5 v programu SETUP.COM jsou u Bibo DOS verze 5.4 a 6.4 značně rozšířeny. Původní Autorun Gener byl nahrazen primitivním editorem, kterým lze sestavit celou sekvenci povelů, které se mohou provést při startu DOSu, nebo povelem v menu. V těchto sekvencích mohou být nejen funkce menu, ale i povely pro zásuvný modul. Kromě toho lze definovat jako primitivní makroinstrukce spojení jména programu (strojového) a klávesy, kterou se program z menu spustí. Po aktivaci funkce N se obrazovka vymaze. Kužor je vlevo nahore a my můžeme začít psát povely. Volba funkce menu se označuje inverzním písmenem, za kterým následují příslušné parametry. Po spuštění programu nebo zásuvného modulu mu lze do následujících rádek zadávat povely jakoby z klávesnice. Kromě funkci menu lze použít povel X pro nastavení barev. Při tvorbě sekvence povelů resp. její editaci platí stejná pravidla, jako při kopírování z obrazovky. To znamená, že rádku odeslanou Returnem již nelze měnit. Eventuální chybu zjištěnou po odeslání již nelze opravit, jedině klávesou Break editor opustit a po novém vyvolání napsat celou sekvenci znova. Maximální délka sekvence je 256 byte a z menu ji lze spustit znakem "^" neboli (SHIFT)-(\*) . Pokud nechceme aby se uložená sekvence při startu DOSu provedla, podřízme klávesu ESC.

### **Povely Startup Editoru**

se piší inverzními znaky. Všechny ostatní se piší normálně. Použit lze tyto povely:

**C** příkaz (přechod do modulu)

kde příkaz je libovoijný povel zásuvného modulu. Příklady:

**CRUN "D:PROGRAM.BAS"**

nebo **ELO."D:TEST.BAS":LIST**

**C** specifikace (kopírování)

kde specifikace je stejná jako u povelu C v menu. Tento povel je možno opakovat v sekvenci i vícekrát a je většinou výhodné použít přepínač "/V". Příklad:

**C\*.BAS,8/V**

**L** jméno souboru (načítání do paměti)

kde jméno označuje binární soubor. Nemá-li se spustit, uvedeme přepínač "/N".

Příklad: **LHELP.DAT/N**

**LAMAC.COM**

**X** aa bb cc dd (naplnění barvových registrů)

kde aa až dd jsou hexadecimální hodnoty které se naplní do stínových registrů barev COLOR0 až COLOR4 (\$2C5 až \$2C8, 709 až 712). Příklad:

**X 0A 02 00 00**

**I** až **B** jméno souboru (definice makropovelů)

kde jméno souboru označuje binární soubor zaveditelný funkcí L. Kombinaci

klávesy OPTION s definovanou číslicí se spustí program zadaného jména.  
Příklad:

**1**ZNAKY.DAT/N  
**2**EDITOR.COM  
**3**FILECOPY.COM  
... atd.

#### Ukončení sekvence:

Sekvenci ukončíme kombinací (CTRL)-(3).

## 8.2. Bibo DOS pro programátory

V tomto odstavci se budeme zabývat možnostmi, které poskytuje Bibo DOS programátorům jak rozšířením funkcí CIO, tak pomocí své tabulky parametrů a vektorů. Dále si vysvětlíme způsob jakým se pracuje s jednotkou XF551, zejména s ohledem na přepínání hustoty záznamu.

### 8.2.1. Doplňky funkcí CIO volaných z Basicu

BiboDOS má některé parametry standardních I/O funkcí změněny, některé jsou přidány.

OPEN má změněný význam dvou hodnot AUX2:

- čtení adresáře bez omezení na jednoduchou hustotu
- 7 - čtení adresáře i se zrušenými a neuzavřenými soubory (DIR SPEZIAL)

XIO má přidány některé funkce. Formát a parametry jsou stejně jako u DOS 2.5 nebo obdobné. Nově přidána je funkce 34 - UNDELETE, která obnovuje zrušené soubory (viz Zurückholen) a má stejné parametry jako funkce 33 - DELETE.

Formátování je možné jen funkcí 254, hustota a formát se volí hodnotami AUX1 a AUX2.

- AUX1 = 0 jednoduchá hustota (SD)
    - 1 dvojitá hustota (DD - jen XF551 nebo upravená 1050)
    - 2 rozšířená hustota (ED)
  - AUX2 = 33 oboustranná dvojitá hustota (DS/DD - jen u verze 6.x a XF551)
    - 128 Clear - zapsání prázdného adresáře.
- RAM-disk se formátuje jen do konfigurované velikosti.

### 8.2.2. Tabulka parametrů

Tabulka se načítá při startu počítače z diskety a ukládá se do stránky 7 podobně, jako u DOS 2.5. Tabulka je však rozšířena o parametry týkající se dvojitých hustot a podobně. Tabulka parametrů je podobně jako tabulka vektorů u všech verzí Bibo DOSu stejná a představuje základ pro kompatibilitu všech verzí. COMPY-SHOP garantuje, že se tabulka nebude měnit, možné je jen rozšířování. Nyní následuje popis jednotlivých buněk tabulky včetně jejich významu. Zkratkou DUP se v dalším textu bude rozumět menu, realizované souborem BDUP.SYS u verzí 5.x a XDUP.SYS u verzí 6.x.

**\$71E 1822 DUPDEN -** momentální hustota  
 Tuto adresu používá DUP ke zprostředkování kapacity momentálně používané diskety. Obsahuje stavový byte jednotky, se kterou bylo naposledy komunikováno. Mohou zde být tyto hodnoty:

bit 7 • 1 rozšířená hustota (Medium)  
 bit 5 • 1 dvojitá hustota (Double)

Jsou-li oba bity nulové, jedná se o jednoduchou hustotu. Tuto adresu nesmí uživatel měnit.

**\$71F 1823 SECLEN -** délka sektoru  
 Zde je uvedena délka (datové části) sektoru naposledy oslovené diskety:

\$7D • 125 byte  
 \$FD • 253 byte

První délka se používá pro jednoduchou a rozšířenou hustotu (Single a Medium), druhá pro dvojitou a oboustrannou dvojitou (Double a Quad). Tuto adresu nesmíme měnit.

**\$720 1824 DRIVES -** počet fyzických disk. jednotek  
 Zde se udává, pro kolik disketových jednotek se mají vyhradit buffery. Teoreticky je přípustných 8, ale na dostupných jednotkách nelze nastavit větší číslo než 4. RAM-disk se do tohoto počtu nepočítá. Obsah buňky DRIVES je orientován bitově. Bit 0 odpovídá jednotce 1, bit 1 jednotce 2 atd. Proto může mít Bibo DOS na rozdíl od DOS 2.5 konfigurovánu třeba jednotku 1 a 3. Pokud se hodnota v DRIVES změní, je nutno aktuální konfiguraci aktivovat vyvoláním DOS-Init (\$709). Pozor, tím se změní dolní hranice uživatelské paměti a části programu (Basic) v ní uložené mohou být porušeny!

**\$721 1825 BUFFERS -** počet bufferů pro soubory  
 Tato adresa udává, kolik smí být současně otevřeno souborů. Musí tu být nejméně 1, pro bezpečnou funkci DUP musí být nejméně 2. Nejvyšší počet je 6. Stejně jako u DRIVES musí být změněná konfigurace aktivována vyvoláním DOS-Init (\$709). Pozor, mění dolní hranici uživatelské paměti!

**\$722 1826 RDRIVE -** číslo RAM-disku  
 Zde je uvedeno, pod kterým číslem jednotky je instalován RAM-disk. Toto číslo se nesmí shodovat s číslem fyzicky připojené jednotky. Záporná hodnota způsobí deaktivaci RAM-disku. Změnu v této buňce je nutno aktivovat následným vyvoláním DOS-Init (\$709). Změna čísla nemění hranici uživatelské paměti, deaktivace nebo opětná aktivace RAM-disku ano.

**\$723 1827 RESID -** příznak rezidentního RAM-disku  
 Příznak určuje, zda má být současný obsah RAM-disku zachován při studeném startu. \$FF - 255 znamená vymazání obsahu RAM-disku, hodnota 0 říká, že RAM-disk je rezidentní. Tato adresa se testuje jen při startování DOSu a má význam jen u počítačů s rozšířenou pamětí když je RAM-disk aktivní.

**\$724 1828 MAXBANK -** maximální počet banků paměti

Zde jsou možné tři hodnoty:

\$04 pro	64 KB RAM-disk (130 XE se 128 KB)
\$08 pro	128 KB RAM-disk (XL/XE se 192 KB)
\$10 pro	256 KB RAM-disk (XL/XE s 320 KB)

Pozor!! Driver RAM-disku netestuje skutečnou velikost paměti počítače! Je-li ohlášen větší RAM-disk než paměť, dojde po překročení skutečné kapacity ke zničení jeho obsahu. Pokud sem napišeme jinou hodnotu (např. \$0C pro RAM-disk 192 KB) neze RAM-disk formátovat na více, než 64 KB a údaj o velikosti RAM-disku v menu zmizí. RAM-disk lze konfigurovat jen částečně. Pro maximální využití RAM-disků typu RAMBO XL a Newell Industries je nutno upravit tabulku banků RDTABL v souladu s fízením těchto rozšíření paměti. Velikost RAM-disku ale přizpůsobit nelze. Jediná možnost je nechat velikost buď omezenou na 128 KB, kdy je provoz zcela bezpečný, ale nelze využít celé jeho kapacity, nebo volit velikost 256 KB, kdy si uživatel musí sám hledat aby velikost obsazené části nepřekročila skutečnou velikost rozšířené paměti. U RAM-disku 192 KB nesmí velikost volného místa klesnout pod 304 sektory.

**\$734 1829 RDTABL -** tabuľka banků paměti

Tabuľka o délce 16 byte obsahuje hodnoty, které je třeba zapsat do PORTB (\$D301), aby se zapnul příslušný bank paměti. Konkrétní hodnoty závisí na bitech, kterými se to které rozšíření paměti řídí. RAM-disky z COMPY-SHOPU se řídí bity 2, 3, 6 a 7, RAMBO XL bity 2, 3, 5 a 6. V každém případě lze tabuľku bez úpravy použít pro standardní 130 XE a pro rozšíření paměti z COMPY-SHOPU. Podmínkou použitelnosti RAM-disku s Bibo DOSem je, aby byl fízen registrem PORTB a banky paměti musí být přepínány v oblasti \$4000 až \$7FFF. Bibo DOS má standardně v tabulce tyto hodnoty:

\$725	EC	E8	E4	E0	- blok 1
\$729	6C	68	64	60	- blok 3
\$72D	AC	A8	A4	A0	- blok 2
\$731	2C	28	24	20	- blok 4

Při RAM-disku velikosti 64 KB se používá blok 1, při 128 KB bloky 1 a 2, při 256 KB všechny čtyři. Jak již bylo uvedeno v předchozím odstavci, nelze použít tři banky. Bloky 2 a 3 jsou prchozeny, je to podle slov dokumentace výhodnější pro práci handleru RAM-disku. Přizpůsobení tabulky jinému rozšíření paměti je třeba provést podle dokumentace k němu náležející.

**\$735 1845 RUNFLG -** příznak nastartování programu

V závislosti na hodnotě tohoto příznaku se program načtený funkcí FLOADR (\$70F) nastartuje. Je-li příznak nulový, program se inicializuje a odstartuje. Hodnota \$FF znamená, že se program pouze zavede.

**\$736 1846 DUPFLG -** příznak stavu DUP

Příznak oznamuje, zda je DUP uschován v RAM pod operačním systémem v oblasti \$E400 až \$F600. Je-li zde \$FF, je DUP schován. Oznamená, že DUP je v paměti od adresy \$2400. Tuto adresu smí měnit jen DUP.

**\$737 1847 DEXIST -** příznak načtení DUP z disku  
 Hodnota příznaku říká, zda byl z diskety úspěšně načten DUP. 0 znamená, že DUP ještě nebyl načten. Při hodnotě \$FF byl DUP úspěšně načten a může být odstartován na adresu \$2400. Tento příznak nesmí být měněn.

**\$738 1848 SETDRV -** celkový počet diskových jednotek  
 Informace o tom, které jednotky jsou Bibo DOSu hlášeny. Není-li přítomen RAM-disk, odpovídá počet hodnotě DRIVES (\$720). Existuje-li RAM-disk, je do této buňky rovněž zanesen. Hodnotatéto adresy se bere v úvahu při stanovení počtu bufferů diskových jednotek během rutiny DOS-Init. Jinak nemá její změna na funkci DOSu žádný vliv.

**\$739 1849 KBFLG -** příznak buferu klávesnice  
 Příznak řídí vytváření bufferu klávesnice. Hodnota 0 znamená buffer aktivní, \$FF buffer vyřadi. Aktivovaný buffer zakazuje klávesový interrupt (\$208, \$209) a mění vektor VBI (\$222, \$223). To může být příčinou nefunkčnosti některých programů. V takovém případě je nutno buffer vyřadit a zkoušit program znovu.

### 8.2.3. Tabulka vektorů

Tabulka služeb Bibo DOSu obsahuje skokové vektory ukazující na adresy rutin realizujících různé služby. Tvůrci ujišťují, že se tabulka nebude užádáných budoucích verzí měnit, a že se může stát zárukou kompatibility programů používajících Bibo DOS. Používání těchto služeb vede ke kratším programům, než při přístupu pomocí CIO. Cenou za toto zkrácení je však omezení na provoz výhradně pod Bibo DOSem. Je otázka, zda má smysl za těchto okolností služeb Bibo DOSu využívat, protože jeho práce s formátem QUAD jednotky XF551 je kompromisem který není bez chyb a problémů. Sparta DOS je v tomto směru podstatně dokonalejší hlavně díky tomu, že není založen dědičtivým formátem DOS 2.0. Hlavní výhoda zde mají uživatelé Speedy 1050, kteří mohou ve svých programech používat rychlou rutinu SIO.

**\$709 1801 JMP RESINI -** inicializace Bibo DOSu

Na tuto adresu se skáče (instrukcí JSR), když je potřeba obnovit handlerovou tabulku DOSu, nebo inicializovat jeho vysokorychlostní rutinu. Inicializace je třeba k zaregistrování některých změn v tabulce parametrů.

**\$70C 1804 JMP TSTRDS -** člení a zápis sektorů RAM-disku, u S-verzi skok na rychlou rutinu

U normálních verzí Bibo DOSu je zde adresa SIO vektoru operačního systému (\$E459). Při použití rychlé verze (např. pro Speedy 1050, v budoucnu doufeme bude i rychlá verze pro XF551) je zde adresa rychlé rutiny SIO a programy ji mohou používat. Tabulka parametrů rychlé rutiny je stejná, jako u volání standardního SIO. Pokud odpovídá číslo jednotky v DUNIT (\$301) hodnotě v RDRIVE (\$722), lze touto rutinou čist a zapisovat i RAM-disk. Adresa bufferu přitom nesmí ležet v prostoru přepinané paměti \$4000 až \$FFFF.

**\$70F 1807 JMP FLOADR -** čtecí rutina Bibo DOSu

Programy v estrojovém kódu, resp. binárně segmentované soubory (zpravidla)

la koncovka .COM, .EXE, atd.) se touto rutinou načtou a v závislosti na hodnotě RUNFLG (\$735) se odstartují. Jako parametr se udává adresa jména souboru:

```
LDA    #FILENAME&255 ,dolní byte
LDY    #FILENAME/256 ,horní byte
JMP    $70F
FILENAME .BYTE "FILECOPY.COM",$9B
```

Před načtením programu se DUP uklidí pod operační systém a program má k dispozici celou paměť. Po ukončení programu se DUP vrátí zpět na své místo. Je pochopitelné, že program nesmí zasahovat do RAM pod systém, jinak nebude návrat do DOSu možný.

#### **\$712 1810 JMP SAVDAT - univerzální rutina SIO**

Rutina může být použita k ukládání programů. Před vyvoláním CIO se DUP uklidí pod operační systém, po dokončení je opět vrácen zpět. Kanál 1 musí být již otevřen a po dokončení operace opět zavřen.

Příklad:

```
LDX    #$10      ,kanál 1
LDA    #3         ,OPEN
STA    $342,X
LDA    #NAME&255 ,adresa jména souboru
STA    $344,X
LDA    #NAME/256
STA    $345,X
LDA    #8         ,otevřít pro zápis
STA    $34A,X
JSR    $E456      ,vyvolání CIO
LDA    #$B        ,PUT Byte
STA    $342,X
LDA    #ADRESA&255 ,adr. ukládané oblasti
STA    $344,X
LDA    #ADRESA/256
STA    $345,X
LDA    #DELKA&255 ,počet ukládaných bytů
STA    $348,X
LDA    #DELKA/256
STA    $349,X
JSR    SAVDAT ,rutina na $712
LDX    #$10      ,SAVDAT mění X-registr
LDA    #$C        ,CLOSE
STA    $342,X
JSR    $E456      , volání CIO
.....
atd.
```

#### **\$715 1813 JMP GOCART - skok do modulu nebo Basicu**

DUP je schován pod operační systém. Tento skok používá DUP Bibo DOSu.

\$718 1816 JMP RUNADR - skok na zadanou adresu

Také tento skok používá DUP Bibo DOSu. Před skokem se DUP uklidí pod operační systém. Adresa, na kterou se má přejít se předává v registrech X a Y.

LDX	#START&255	,dolní byte
LDY	#START/256	,horní byte
JSR	RUNADR	
.....		
atd.		

Adresy FLOADR (\$70F), SAVDAT (\$712), GOCART (\$715) a RUNADR (\$718) může vyvolávat jen DUP, který je těmito povely krátkodobě uschován do RAM pod operačním systémem a po dokončení práce opět vrácen zpět. Při psaní vlastního DUP je nutno dodržet tyto konvence: DUP může zabírat prostor mezi \$2400 až \$3FFF (tato oblast se schovává), startovací adresa musí být \$2400. Veškeré přístupy do DOSu měly být prováděny pouze přes CIO, nebo vektorovou tabulku.

### 8.2.3. Programování XF551

Jak již bylo řečeno v kapitole 2, nemá XF551 možnost rozpoznat automaticky výměnu diskety. Proto ani nemůže rozpoznat změnu hustoty a formátu. Tento skutečnostem je třeba přizpůsobit metody práce. Především je nutno při každém otvírání souborů nejprve testovat hustotu vložené diskety. K tomu se pokoušíme číst sektor, přičemž se jednotka pokouší rozpoznat hustotu vložené diskety a konfigurovat se podle ní. Tento postup je důvodem, proč trvá zjištění, že není vložena disketa, tak dlouho.

Konkrétní postup spočívá ve čtení stopy 0. Nejprve je třeba určit délku sektoru. Toho nelze dosáhnout čtením sektorů 1 až 3, protože z těch jednotka vydává jen 128 byte, až jsou jakkoliv dlouhé. Proto se čte sektor 4 a protože jeho délku zatím neznáme, je v tomto případě nastavena délka nulová. SIO sice hlásí chybu kontrolního součtu, ale tuto chybu ignorujeme. Důležité je, že jednotka sektor přečte a nastaví se podle hustoty vložené diskety. Teprve potom se může přečíst konfigurační blok a určit hustota záznamu. Mezi čtením sektoru a konfiguračního bloku musí být vložena zpožďovací smyčka, aby měla jednotka čas přečíst zbyvající data sektoru 4. Celý tento koloběh probíhá u Bibo DOSu při každém povelu OPEN a tím se práce s XF551 zdržuje. Proto má Bibo DOS ještě další časovou smyčku, protože předpokládá, že se v krátké době po otevření souboru disketa nevymění a po tuto dobu vyněchává test hustoty.

Bohužel se hustota nefestuje vždy a zcela důsledně, takže při kopirování souborů mezi disketami v jednoduché a dvojitě hustotě (vesmyslu délky sektoru) dochází při některých situacích k problémům. V tomto případě lze doporučit Sparta DOS, který je pro podobné případy vybaven lépe.

### **Konfigurační blok**

obsahuje informace o nastavení disketové jednotky, lze jej číst i psát. Konfigurování disketové jednotky je možné prakticky pouze v assembleru pomocí rutiny SIO. Čtení momentální konfigurace se provede povelem \$4E, zápis povelem \$4F (IN, OUT). Blok má délku 12 byte a jejich význam je v tabulce:

byte 1:	počet stop	(40)
byte 2:	Step Rate	(1)
byte 3:	poč. sekt. na stopu - horní byte	(0)
byte 4:	dolní byte	(18/26)
byte 5:	počet hlav	(0/1)
byte 6:	hustota	(0-SD/4-DD)
byte 7:	délka sektoru-horní	(0/1)
byte 8:	délka sektoru-dolní	(128/0)
byte 9:	on line	(255)
byte 10:	???	(0)
byte 11:	???	(0)
byte 12:	???	(65, "A")

Konfigurování jednotky je nutné pro jednoduchou, dvojitou a oboustrannou dvojitou hustotu, které používají stejný formátovací povel \$21. Pro rozšířenou hustotu je povel \$22 a konfigurační blok nemá na formát žádný vliv. Povolený \$21 se disketa naformátuje podle údajů konfiguračního bloku. Nebyla-li jednotka ještě konfigurována, formátuje se v jednoduché hustotě. K volbě konfigurace jsou potřebné pouze tyto údaje:

byte 4:	0 pro SD s DD, 1 pro Quad (DS\DD)
byte 5:	0 pro SD, 4 pro DD a Quad
byte 6-7:	délka sektoru, 0+128 pro SD, 1+0 pro DD a Quad

Je-li v bytu 4 hodnota 1, formátuje se bez ohledu na ostatní byty v hustotě Quad. Příklad čtení konfiguračního bloku:

START	LDA	#\$31	,disk
	STA	\$300	
	LDA	#1	,číslo jednotky
	STA	\$301	
	LDA	#\$4E	,povel-čtení KB
	STA	\$302	
	LDA	#\$40	,čtení
	STA	\$303	
	LDA	#BUFFER&255	
	STA	\$304	
	LDA	#BUFFER/256	
	STA	\$305	
	LDA	#6	,timeout
	STA	\$306	
	LDA	#12	,délka - dolní
	STA	\$308	
	LDA	#0	,délka - horní
	STA	\$309	
	JSR	\$E459	,SIO

```

        BPL      NOERR
        JMP      ERROR   ,ošetření chyby
NOERR    RTS
BUFFER   ** *+12      ,vyhrazení 12-ti byte

```

K nastavení konfigurace se může použít stejný podprogram, ale jako povel se použije \$4F místo \$4E a do buňky \$303 se dá hodnota \$80. Buffer pro jednoduchou hustotu by vypadal takto:

```
BUFFER .BYTE $28,1.0,$12,0.0,0,$80,$FF,0.0,0
```

12. byte má na formát vliv, ale jeho logiku nelze přesně vystopovat.

### 8.3. Služební programy pro Bibo DOS

Kromě funkcí menu má Bibo DOS ještě dva užitečné služební programy. První z nich slouží k převodu souborů z formátu DOS3, druhý je univerzální kopírovací program. Oba se spouštějí funkcí L.

#### 8.3.1. Program CONV32D.COM

Slouží ke konverzi souborů z formátu DOS 3 do formátu DOS 2.x. Menu programu má na výběr 4 funkce:

```

*** DOS III --> DOS II Converter ***
(c) Bibosoft 7/87 ----- SD-1 / DD-1

1 Set Source -----Drive
2 Set Destination----Drive
3 Copy_File
4 Exit

?■

```

Funkcemi 1 a 2 volíme číslo vstupní (Source Disk) a výstupní (Destination Disk) jednotky. Před volbou kopírování (funkce 3) vložíme disketu DOS 3. Vypíše se její obsah a příslušným písmenem vybereme soubor ke konverzi, nebo se klávesou (RETURN) vrátíme do menu. Funkce 4 nás vrátí do modulu, nebo do menu Bibo DOSu.

**Pozor:** Program nemůže pracovat s RAM-diskem! Proto nesmí být RAM-disk nastaven na číslo jednotky 1 nebo 2.

#### 8.3.2. Program FILECOPY.COM

Slouží k pohodlnému kopírování libovolně vybraných skupin souborů pomocí jedné i dvou jednotek, včetně RAM-disku. V první obrazovce se volí číslo vstupní jednotky klávesou OPTION a výstupní jednotky klávesou SELECT. "F" volí formátování cílové diskety z těchto možností: NO, SNG, MED, DUP, CLR. Jak vidíme, chybí tu DS/DD. CLR znamená zrušení všech souborů na kopii před kopírováním vybrané skupiny. Klávesa START způsobí načtení a přechod do druhé obrazovky, ve které je v levé polovině zobrazena stránka adresáře s velkým kurzorem a vpravo funkční klávesy. Pomocí kláves pro pohyb kurzoru nahoru a dolů jej můžeme nastavovat na jednotlivé soubory. Je-li adresář delší než jedna stránka, po dojetí kurzoru na kraj okénka začíná

tolovat. Výběr souborů pro kopírování se provádí nastavením kurzoru a mezerníkem, nebo klávesou (RETURN). Zrušení výběru je možné opakovaným stiskem mezerníku nebo (RETURN). Kromě toho můžeme vybrat všechny soubory klávesou "A". Klávesou START spustíme kopírování vybraných souborů, klávesou OPTION se dostaneme zpět do první obrazovky. Po stisku klávesy START se dostaneme do třetí obrazovky. Zde nám v levém okénku ukazuje kurzor na soubor, který se právě kopíruje a barva pozadí určuje fázi kopírování. Při čtení originálu je modrá, při zápisu kopie oranžová. Pokud jsou originál a kopie různé jednotky, kopíruje se bez přerušení. Pokud se pracuje na jedné jednotce, vyzývá spodní okénko ke vkládání příslušné diskety, přičemž je požadavek ještě dále určen barvou pozadí: originál-modrá, kopie-oranžová. Kopírování lze přerušit klávesou BREAK.

#### 8.4. Kompatibilita a DOS 2.5 a přechod na dvojitou hustotu.

Bibo DOS má formát ukládání souborů a organizaci adresáře odvozenou od DOS 2.5. Zachovává i status souborů z oblasti rozšířené hustoty. Proto je v oboru záznamu s délkou sektoru 128 byte plně kompatibilní s DOS 2.5. Při práci s dvojitou hustotou jedno- i oboustrannou pracuje stejně s tím rozdílem, že délka sektoru je dvojnásobná a tím je dána maximální kapacita disket 360k a 128 souborů. Pokud máme disketu v DS/DD, můžeme dojít k situaci, kdy sice jsou ještě volné sektory, ale v adresáři již není volné místo. Zatímco při SD, ED nebo i DD je tato situace zřídka, při DS/DD může nastat dosti snadno. Problém lze řešit přechodem na SpartaDOS, nebo MYDOS, které umožňují uložit libovolný počet souborů pomocí podadresářů, nebo se smířit s nedostačným využitím diskety.

Jednotky s dvojitou hustotou již existují delší dobu, ale teprve nahrazením modelu 1050 jednotkou XF551 nastávají podmínky pro její hromadné používání a tím také větší potřeba přechodu mezi hustotami s délkou sektoru 128 (jednoduchá, rozšířená) a 256 byte (dvojitá, DS/DD). Pro zjednodušení budeme v tomto odstavci shinnovat první dvě do pojmu jednoduchá a druhé dvě do pojmu dvojitá. Problém přechodu mezi jednoduchou a dvojitou hustotou spočívá v nutnosti přepínání jednotky do příslušného režimu. Bibo DOS je do jisté míry schopen přepínat se automaticky podle vložené diskety (viz 8.2.3.), ale v některých situacích je třeba mu trochu "pomoci". Problémy nejsou při výpisu adresáře. Jednotka i systém rozpozná formát vložené diskety a přizpůsobí se. Při práci s více jednotkami není kopírování mezi jednoduchou a dvojitou hustotou problém. Stačí si prohlédnout obsah originálu i kopie (což se stejně dělá pro informaci) a kopírovat jak pomocí funkce C, tak programem FILECOPY. Kopírování SD->DD přes RAM-disk jde také dobře, ale než spustíme kopii z RAM-disku na disketu v DD, musíme přepnout jednotku na dvojitou hustotu prohlédnutím jejího obsahu. Uživatel jedné jednotky bez RAM-disku má tento přechod méně pohodlný, ale také možný. Funkce C přikopírování jednotlivých souborů přepíná hustotu podle diskety, která je v jednotce právě vložena. Kopírování pomocí FILECOPY však v tomto případě selhává (BDOS 6.0). Problém je způsoben tím, že Bibo DOS není při testování hustoty vždy důsledný. Bibo DOS lze bez problému nainstalovat na disketu v DD a rovněž bez problému ho z ní spouštět.

## **9. Stromová struktura - podadresáře**

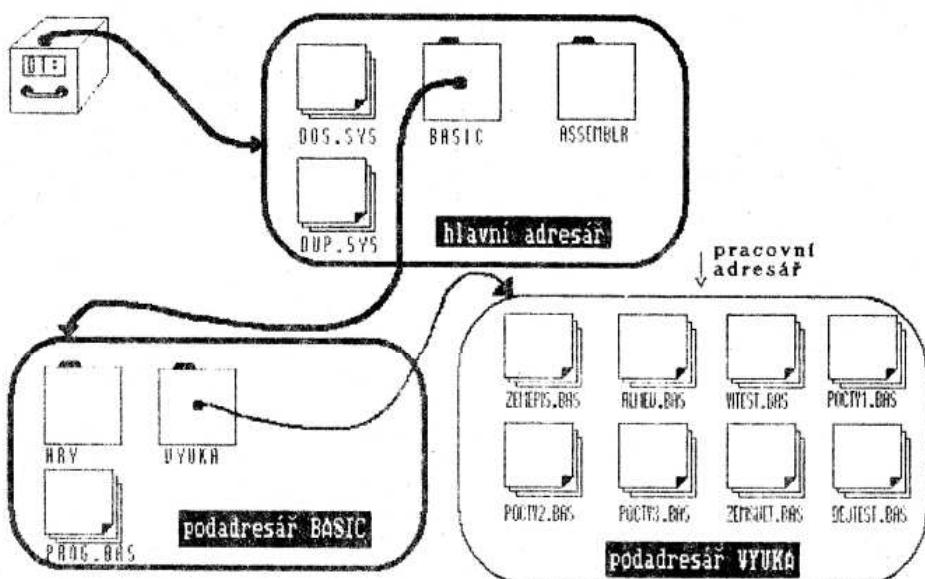
U Bibc DOSu jsme narazili na problém, který se musí řešit u všech velkokapacitních disků. Velikost adresáře je omezená a při ukládání souborů malého objemu narazíme brzy na jeho hranice. Na disku sice ještě místo zbyvá, ale v adresáři již není místo pro další soubory. Zvětšování adresáře není efektivní, protože by velký adresář v mnoha případech zabíral zbytečně místo. Proto se ustálilo používání tzv. podadresářů (subdirectory).

### **9.1. Podadresáře a jejich postavení v systému souborů**

Disketa má hlavní adresář (main, nebo root directory), ve kterém jsou kromě souborů na jaké jsme zvyklí, ještě zvláštní soubory, zvané podadresáře. Každý takový podadresář si můžeme představit jako další disketu, která je "vložena" na místě souboru. Nejlépe se souborová struktura s podadresáři dá vysvětlit na základě analogie s kartotékou. Každý si jistě dovede představit kartotéční zásuvku. Tato zásuvka představuje disketu. Má v sobě uložen na smluveném místě seznam dokumentů, které obsahuje. Pokud jsou v zásuvce jen dokumenty, odpovídá to organizaci, na kterou jsme zvyklí. Co když ale počet dokumentů vzroste a jejich názvy se na cedulku nevejdou? Řešení je jednoduché. Související dokumenty dáme do obálky, kterou vhodně pojmenujeme a název napíšeme do seznamu na čelní stěně šuplíku. Do obálky vložíme seznam dokumentů, které jsou uvnitř. Tím se na jedné straně rozrosté počet dokumentů, které se do šuplíku vejdu, na druhé straně se tím ztěžuje přehled o tom, jaké dokumenty vlastně v šuplíku jsou protože v něm mohou být kromě dokumentů i obálky, které je potřeba otevřít a přečíst si seznam. Při naplnění seznamu obálky do ní lze vložit další obálku. Tím vzniká víceúrovňová tzv. stromová struktura. Kromě toho, že se takto dá uskladnit neomezený počet dokumentů až do úplného zaplnění šuplíku, je v některých případech výhodné sdružit skupinu dokumentů do společné obálky. S obálkou lze totiž manipulovat buď jako s celkem, nebo i s jednotlivými dokumenty, které v ní jsou. Pokud například pracuje na jednom disku více uživatelů, mohou mít v hlavním adresáři společně užívané a systémové programy. Každý uživatel má v hlavním adresáři založenu vlastní obálku se soukromými soubory. Tímto uspořádáním má uživatel přístup ke svým a k společným souborům a cizí soubory ho nematou. Jiné výhodné použití je když máme rozpracovaných víc věcí. Můžeme si dát každou do jedné obálky, kterou si vždy na začátku práce nakopírujeme do RAM-disku. Naskytá se otázka, jak se v tomto systému vlastně určí umístění dokumentu. Zásuvku označme třeba D1, v ní je mimo jiné obálka s názvem BASIC, ve které jsou programy Atari Basicu rozdělené do dalších obalek VYUKA a HRY. V první z nich je program ZEMEPIS.BAS. Chceme-li takový program kopírovat, musíme určit přesně jeho umístění, neboť tzv. cestu (path) k němu. V našem případě to je D1-BASIC-VYUKA-ZEMEPIS.BAS, což je značně dlouhé. Proto se zavádí pojem aktuálního, nebo pracovního adresáře.

## 9.2. Pracovní adresář

Aby se nemusela cesta k souboru psát vždy celá, lze definovat pracovní adresář a polohu souboru určovat vzhledem k němu. Při spuštění systému je vždy pracovní adresář totožný s hlavním adresářem. Když chceme pracovat s obálkou BASIC, nastavíme na ni ukazatel pracovního adresáře a výše zmíněný program určíme jako VYUKA-ZEMEPIS.BAS. Pokud definujeme jako pracovní adresář obálku D1-BASIC-VYUKA, stačí k určení již jen jméno souboru. Situaci nám dokreslí obrázek:



obr. 10: Struktura diskety s podadresáři

Každé okénko nám zde ukazuje obsah nadefinovaného objektu. Výpis adresáře diskety D1 ukáže obsah hlavního adresáře. V ném vidíme kromě souborů DOS.SYS a DUP.SYS ještě podadresáře BASIC a ASSEMBLR, které svými jmény naznačují, co je jejich obsahem. Výpis podadresáře BASIC však ukáže, že nejsme ještě na konci. V tomto podadresáři jsou kromě souboru PROG.BAS ještě schovaný další dva podadresáře HRV a VYUKA, do kterých jsou zřejmě podle zaměření uloženy programy. I podíváme se do podadresáře VYUKA a konečně vidíme hledaný program ZEMEPIS.BAS. Z popisu jsou jasné zřejmě výhody i nevýhody stromové hierarchické struktury souborů. Protože je přehled o souborech víceúrovnovou strukturou dosti ztížen, má většina systémů prostředky k analýze celého disku, které vypisují všechny do sebe vnořené podadresáře a v nich evidované soubory. Pro každý podadresář

platí, že jména souborů a podadresářů v něm evidovaných musí být jedinečná. Z toho ovšem plyne, že v různých podadresářích se mohou různé soubory jmenovat stejně. Jsou totiž rozlišeny cestou, čili jmény nadřízených adresářů. Přesto se používání stejných jmen pro různé soubory v různých adresářích nedoporučuje, protože se tím dá způsobit totální chaos a uživatel takové diskety je ve značném nebezpečí hospitalizace v nervovém sanatoriu.

Při označování souborů můžeme použít absolutní adresace vzhledem k hlavnímu adresáři. Při tomto způsobu je nutno uvést celou cestu od začátku až k souboru. Nevhodou je dlouhé zadání, výhodou je, že si nemusíme pamatovat, nebo zjišťovat, který adresář je momentálně nastaven jako pracovní. Další alternativou je relativní adresování vzhledem k pracovnímu adresáři. Při tomto druhu adresace uvádime cestu od pracovního adresáře k souboru a musíme vždy vědět, kde je pracovní adresář momentálně nastaven. Relativní adresování je v případě výukových programů v Basicu výhodné, protože pracovní adresář je nastaven na D1-BASIC-VYUKA a soubory v něm se označují již jen jménem tak, jak jsme zvyklí. Teď si ale představme, že si do některého programu chceme zařadit strojový podprogram PODPR.OBJ, který máme v obálce ASSEMBLR. Jak se k němu dostaneme? Při absolutní adresaci je to jednoduché: D1-ASSEMBLR-PODPR.OBJ. Při relativní adresaci musíme použít zpáteční cesty po stromové struktuře, což některé systémy povoluje, jiné nikoliv. Předpokládejme, že takový návrat je povolen a v popisu cesty je označíme "<". Relativní cesta k našemu podprogramu bude <-<-ASSEMBLR-PODPR.OBJ. Od pracovního adresáře musíme projít nahoru do obálky BASIC, z ní ještě nahoru do hlavního adresáře, potom dolů do podadresáře ASSEMBLR, kde máme hledaný soubor. Je vidět, že v tomto případě je absolutní adresace výhodnější. Jinak tomu bude, když chceme natáhnout program z vedlejšího podadresáře HRY. Absolutní označení je D1-BASIC-HRY-jméno, kdežto relativní je <-HRY-jméno. Teď je výhodnější druhý způsob. Při práci je důležité si nastavit pracovní adresář tak, aby cesta k nejpoužívanějším soubcům byla co nejkratší a to je také jeho smyslem.

Stromová struktura souborů neznamená, že bychom museli používat podadresáře za každou cenu. Při práci na 20MB winchesteru se bez nich neobejdeme, ale na disketu 127kB můžeme klidně pracovat na úrovni hlavního adresáře a nekomplikovat si situaci. Podadresáře budeme používat jen tam, kde to bude potřebné, nebo výhodné. Nyní, když jsme si problematiku hierarchické stromové struktury vysvětlili, můžeme se pustit do výkladu dvou univerzálních diskových operačních systémů, které mohou obsluhovat diskové paměti až do kapacit 16MB. Tyto systémy jsou MYDOS a SpartaDOS.

## 10. MYDOS 4.3B

MYDOS verze 4.3B napsal Charles Marslett ze společnosti WORDMARK SYSTEMS, USA. Tento systém vychází z DOS 2.0 a rozšiřuje jeho možnosti. Může ovládat jak oboustranné disketové jednotky, tak i virtuální RAM-disk realizované v rozšířené paměti osmibitových Atari. Komunikace systému s uživatelem je řízena pomocí menu, které je podobné menu DOS 2.5, i když má ovládání spíše podobné Biho DOSu. Obsazení paměti MYDOSem je téměř stejné, jako u DOS 2.5. MEMLO je při jedné aktivní jednotce, třech současně otevřených souborech a RAM-disku \$1F04. Všechny funkce CIO, i volání DOSu v Basicu zůstaly zachovány. Změněn byl formát výpisu adresáře, takže programy závislé na jeho přesném formátu mají potíže. Z diskových jednotek a dalších zařízení lze pomocí MYDOSu pracovat kromě obou výrobků Atari s jednotkami PERCOM, kontrolerem disk/tiskárna/RS232 ATR8000, oboustrannými disky ASTRA, INDUS GT a pevnými disky Compco a Supra. Maximální počet disků je 8, nutný je alespoň jeden.

### 10.1. Funkce menu MYDOS 4.3B

V menu MYDOSu je 18 funkcí volených příslušnými písmeny. Na rozdíl od DOS 2.5 se po písmenu nemačká (RETURN), ale funkce se hned rozběhne. Menu nepoužívá standardní editor obrazovky, takže chybně napsané parametry nelze opravit, je třeba psát znovu. V menu je pod hlavičkou MYDOSu s označením verze a firmy stavová řádka, která informuje o konfiguraci systému:

**MYDOS 4.3B -- copyright 1986, WORDMARK**

**DISKS: 1S- 2D- 3H 8R**

**D: - D1:**

<b>1-8.Dir of D1:-D8:</b>	<b>*. Dir of D:</b>
A. Disk Directory	K. Save Memory
B. Run Cartridge	L. Load Memory
C. Copy File(s)	M. Run at Address
D. Delete File(s)	N. Load MEM.SAV
E. Rename File(s)	O. Change Config
F. Lock File(s)	P. Set Density
G. Unlock File(s)	Q. Make Directory
H. Write DOSFiles	R. Pick Directory
I. Initialize Disk	S. Set Ramdisk *
J. Duplicate Disk	V. Set Verify Flag

**Select Item (RETURN for menu):** ■

V našem příkladu ukazuje stavová řádka, že disk 1 má jednoduchou hustotu (S) jednostrannou (-), disk 2 dvojitou (D) oboustrannou (-), disk 3 je velkokapacitní pevný disk a disk 8 je RAM-disk. U konfigurovatelné disketové jednotky ukazuje řádka nikoli její možnosti, ale momentální nastavení. Jednotku

můžeme nastavit funkcemi O a P v menu. Kromě toho může MYDOS automaticky přepínat z dvojitě hustoty na jednoduchou podle vložené diskety. Obrázeně to však bohužel u XF551 automaticky nejde a kopírování z jednoduché hustoty na dvojitou pomocí jediné jednotky je problematické. Třetí řádka ukazuje momentální nastavení pracovního adresáře, který se u MYDOSu označuje jako "D:" bez čísla jednotky. Menu dostaneme vždy, když na základní výzvu stiskneme klávesu (RETURN).

#### **Pravidla pro zadávání jmen a parametrů**

Místo "D:" se dosadí to, co je právě nastaveno jako pracovní adresář. Nastavují se funkci R. Některé povely vyžadují potvrzení akce, aby nedošlo k náhodnému smazání dat. Jsou to funkce "I" a "J". Jakoukoli operaci můžeme přerušit bez poškození stávajících souborů klávesou RESET nebo BREAK. Funkce D, E, F a G požadují explicitní určení jmen souborů. Specifikace souborů, nebo disketových jednotek má stejná pravidla. Pokud se zadá jen číslo jednotky, předpokládají se všechny soubory, nebo první soubor pokud se operace týká jen jednoho. Jednotka se dá specifikovat buď dvoječkou ":" nebo číslem s nebo i bez následující dvoječky, nebo písmenem "D" následovaným nepovinným číslem a povinnou dvoječkou. Platná jména jsou například D1:Test.obj, 1:TEST.ASM, nebo D2TEST(-D1:D2TEST), ale i 2README.TXT(-D2:README.TXT). Jméno samotné musí být buď úplné, nebo může obsahovat filtr podle běžných pravidel. Jméno u MYDOSu rozlišuje malá a velká písmena (TEST <> Test <> test, zejména musíme dát pozor u filtrů) a smí začínat malým nebo velkým písmenem, podtržením "-", zavináčem "@" či znakem "#". Mezi zbývajícími znaky mohou být navíc číslice.

Kromě hlavního adresáře (může obsahovat až 64 u SD a ED nebo 128 u DD souborů, nebo podadresářů) může každá disketa MYDOSu obsahovat podadresáře, v nichž může být v každém dalších 64 (128) souborů nebo podadresářů. Cesta k souboru uloženému v některém podadresáři se vyjadřuje u MYDOSu oddělením jednotlivých úrovní dvoječkami. Příklad souboru ZEMEPIS.BAS z kapitoly 9 bude vypadat takto:

**D1:BASIC:VYUKA:ZEMEPIS.BAS**

Každý podadresář je ve svém nadřízeném adresáři uložen jako soubor o délce 8 sektorů.

Pokud je funkci požadováno více jmen souborů, lze první vynechat napsáním mezery nebo čárky jako prvního znaku, poslední lze vynechat ukončením řádky čárkou. Jsou-li uvedena obě jména, mohou být oddělena jak čárkou, tak mezery. Některé povely lze modifikovat přepínačem "/x", kde "x" je písmeno. Tyto přepínače jsou povětšinou popsány u funkcí, které ovlivňují, zde jen několik příkladů. Přepínač "/A" je použitelný u funkce C a A, a způsobuje připojení souboru za konec jiného. Přepínač "/N" potlačí formátování při funkcích I a J, u funkcí D, F a G potlačí kontrolní dotazy. Přepínač "/X" zastaví kopírování po každém čtení a zápisu, takže lze vyměňovat diskety při práci s jednou jednotkou. Výměnu lze provádět i když se kopíruje na různé jednotky. Je to proto, že existují jednotky, které jsou rozděleny na více logických jednotek.

Také to umožnuje nahrát výpis adresáře na jinou disketu, než ze které se četl. "/X" se předpokládá automaticky, když se u funkce C zadá jen jedno jméno. Přepínač "/S" počítá kopírování souborů, jejichž koncovka začíná na S (systémové soubory).

### Výpis adresáře

### A. 1-8. \*

Pro výpis adresáře má MYDOS tři možnosti, které produkuji výpis ve stejném formátu. Formát je poněkud odlišný od formátu používaného u DOS 2.5. Rozdíl je ve sloupci před jmény souborů, kde může být kromě hvězdičky znamenající blokování souboru také dvojčárka označující podadresář. Ostré závorky MYDOS nepoužívá. Druhý rozdíl je v počtu číslic při uvádění délky sektoru. MYDOS používá čtyřmístné číslo. Při výpisu na obrazovku se adresář vypisuje do dvou sloupců a dává tak lepší přehled je-li na disketě více souborů. Pokud se adresář posílá na tiskárnu, neboda souboru, má jednosloupcový formát. Nyní se podívejme na jednotlivé způsoby výpisu:

#### číslo disku

Odpovíme-li na základní výzvu MYDOSu číslem disku, vypíše se na obrazovku obsah hlavního adresáře.

#### \* (hvězdička)

Vypíše na obrazovku obsah pracovního adresáře, neboli toho, co je specifikováno ve třetí řádce menu.

### A

Úplný povel, umožňující různé varianty. Po stisku A se vypíše dotaz "Files to list, Destination?". Syntaxe odpovědi je následující:

[{jméno1|cesta1:jméno1|cesta1}] [{zař|jméno2|cesta2:jméno2}] [/A] [/X]

- kde:
- jméno1 je jméno nebo filtr pro vyhledání v pracovním adresáři, nebo v adresáři určeném cestou1.
  - cesta1 je cesta, definující vybraný podadresář. Začíná-li cesta kódem zařízení (číslo jednotky, nebo D, číslo jednotky dvojčárka, jde o cestu absolutní, jinak se cesta chápe jako relativní vzhledem pracovnímu adresáři).
  - zař je sekvenční zařízení E:, P:, S:, C: (v modu dlouhých mezí)
  - jméno 2 je jméno souboru v pracovním adresáři, nebo v adresáři určeném cestou2, do kterého se uloží výpis adresáře.
  - cesta2 určuje adresář, ve kterém je soubor pro zápis
  - /A výpis adresáře se bude k výstupnímu souboru připojovat.
  - /X mezi čtení adresáře a zápisem do souboru je pauza s požadavkem vložení zdrojové nebo cílové diskety. Praktický význam tohoto přepínače je sporadický, protože se diskety musí vyměňovat dvakrát při každém řádku výpisu.

Při výpisu adresáře a při práci s povely musíme dát pozor na to, co je momentálně nastaveno jako pracovní adresář, protože ten se dosazuje jako standard. Zvláště si musíme uvědomit, že může být nastaven i na jinou jednotku, než D1. Obecně platí, že neuvede-li se číslo jednotky, určuje se soubor vždycky vzhledem k pracovnímu adresáři. Pokud chceme vypsat podadresář, resp. jeho obsah, nesmíme zapomenout ukončit cestu dvojtečkou. Jde totiž o to, že každý podadresář je zároveň souborem ve svém nadřízeném adresáři. Rozdíl si nejlépe ukážeme na příkladu, kdy předpokládáme strukturu souborů z kapitoly 9. Stiskneme A a na dotaz odpovíme "1BASIC". Vypíše se informace ":BASIC 0008", která říká že soubor BASIC je podadresář a zabírá 8 sektorů. Odpovíme-li "1BASIC:", vypíše se obsah podadresáře BASIC, tedy informace o souboru PROG.BAS a podadresářích HRY a VYUKA.

#### **Přechod do modulu - Run Cartridge**

**B**

Funguje stejně, jako v DOS 2.5. Řízení se předá po natažení MEM.SAV (pokud byl poslední povel binárního čtení "N", nebo žádný), nebo přímo (pokud bylo poslední binární čtení povelem "L").

#### **Kopírování souborů - Copy File(s)**

**C**

Povel slouží ke kopírování jednoho, nebo celé skupiny souborů dat. Po stisknutí C jsou požadovány dvě specifikace souborů, určující co se kopíruje a kam. Buď se udá plně (absolutně), nebo částečně (relativně k pracovnímu adresáři) specifikované diskové jméno, nebo kód zařízení (E:, P:, C: apod.). Specifikace výstupu může obsahovat určení skupiny jen tehdy, když je jméno určeno vstupní specifikací (E:, D:A\*.\* nelze!). Udá-li se jen vstupní specifikace, předpokládá se výměna disků (odpovídá funkci O v DOS 2.5). Pomocí jedné jednotky nelze kopírovat z diskety formátované v jednoduché hustotě (SD) na disketu ve dvojitě hustotě (DD). Za specifikace souborů lze psát přepínač "/A", význam je jako u DOS 2.5. Je-li originál skupina souborů a kopie jediný soubor, bude v kopii jen poslední soubor skupiny, pokud se neudá /A. Povel C u MYDOSu používá vždy celou dostupnou paměť a tím zneplatní MEM.SAV. Po funkcích C a J nelze restartovat žádný program.

#### **Rušení souborů - Delete File(s)**

**D**

Rušení probíhá stejně, jako u DOS 2.5. Je pouze třeba upozornit, že adresář lze zrušit jen neobsahuje-li žádné soubory. Přepínač "/N" potlačí potvrzování každého rušení. Naobovenírušeného souboru nemá MYDOS žádný prostředek a programy jiného systému (DISKFIX.COM) nelze u MYDOSu použít kvůli odlišnému formátu VTOC. Specifikace rušeného souboru (souborů) může být buď absolutní (začíná číslem nebo označením disku), nebo relativní vzhledem k pracovnímu adresáři.

#### **Přejmenování souborů - Rename File(s)**

**E**

Specifikuje se cesta (absolutní či relativní) a staré a nové jméno. Neuvede-li se cesta, předpokládá se pracovní adresář. Přejmenovávat lze i podadresáře. Pokud přejmenujeme podadresář, který je právě pracovní, jeho jméno se změní ve výpisu nadřízeného adresáře, ale ne ve třetí řádce menu. Přesto však

výpis adresáře pomocí hvězdičky funguje správně i nadále.

#### Blokování souborů - Lock File(s)

**F**

Jestejně, jako u DOS 2.5 s tím rozdílem, že ve specifikaci souborů je eventuálně třeba uvádět cestu k souborům. U každého souboru se vypisuje ověřující dotaz, který lze potlačit přepínačem "/N".

#### Uvoľnení souborů - Unlock File(s)

**G**

Jako 10.1.7., ale jde o inverzní pochod k funkci F.

#### Instalace DOSu - Write DOS Files

**H**

Ukládá do tvaru potřebného ke spuštění DOSu na disk momentální konfiguraci MYDOSu. Pokud se MYDOS nainstaluje na disketu ve dvojitě hustotě, lze jej sice spustit, ale nefunguje instalace RAM-disku.

#### Formátování diskety - Initialize Disk

**I**

Funkce umožňuje formátovat disketu pro použití s MYDOSem, nebo z formátované diskety odstranit záznamy o všech souborech. Je-li za číslem jednotky uveden přepínač "/N", formátování se přeskočí a zapíše se jen prázdný hlavní adresář. Je to rychlejší, než rušení všech souborů funkci D. Hustotu určuje jednak nastavení systému, jednak odpověď na dotaz "Type 'Y' (or 'A') to format Dn:". Při nastavení na SD se při odpovědi "Y" vytvoří 707 volných sektorů (jednoduchá hustota), při odpovědi "A" 1023 (rozšířená hustota). Odpověď N zruší formátování. Formátování RAM-disku nainstaluje počet sektorů podle jeho velikosti bez ohledu na odpověď "Y" nebo "A".

#### Duplikování diskety - Duplicate Disk

**J**

Kopíruje vše z jedné diskety najinou. Rozsah kopírované oblasti je určen počátečním a koncovým číslem sektoru. Pokud se rozsah neuvede, kopíruji se jen sektory, označené v tabulce VTOC jako použité. VTOC může být typu DOS 2.0, nebo MYDOS, nebo kompatibilního DOSu. Je zřejmé, že diskety DOS 2.5 nelze takto plnohodnotně duplikovat (viz 3.4.). Specifikace rozsahu se dělá přidáním dvou čísel oddělených pomlčkou a společně uzavřených v závorkách na konci specifikace disků. Přepínač "/N" potlačí formátování cílové diskety.

Příklad 1: Kopírujeme z jednotky 1 na jednotku 3 bez formátování cílové diskety sektory 19 až 54. Odpověď na dotaz "Source, Destination (sectors)?" bude "1.3/N(19-54)". Kopie celé diskety v jednoduché hustotě s formátováním cílové diskety: 1.2(1-720). Tímto způsobem je třeba kopírovat diskety nekompatibilní s MYDOSem. Formátování při funkci J nekontroluje výskyt chyb. Funkce J používá celou dostupnou paměť a ruší platnost MEM.SAV. Pokud záleží na kvalitě kopie, je spolehlivější naformátovat cílovou disketu funkci I, která výskyt vadných sektorů kontroluje a kopirování provést s přepínačem "/N".

Příklad 2: Kopie disku 1 na disk 2 bez formátování cílové diskety:

1/N,2 nebo 1,2/N

Příklad 3: Kopie prvních dvou stop z disku 1 na disk 2 bez formátování. Disketa je v jednoduché hustotě:

1/N,2(1-36) nebo 1,2/N(1-36)

#### Uložení obsahu paměti na disk - Save Memory to Disk **K**

Uložení definované části paměti na disk ve formě binárního segmentovaného souboru. Parametry a funkce jsou stejné, jako u DOS 2.5. Pokud je aktivní MEM.SAV, načte se uchovaná oblast paměti před uložením specifikovaného úseku.

#### Načtení binárních souborů - Load from file

**L**

Načtení binárního segmentovaného souboru. Funkce je shodná s funkcí L DOSu 2.5. Při načítání se provádí inicializace a rozbeh podle definic inicializačních a rozbehových adres, pokud není zadán přepínač "/N". Při této funkci se ruší funkce souboru MEM.SAV.

#### Skok na adresu - Run at Address **M**

Povel má stejný tvar a význam, jako u DOS 2.5. Pokud je aktivní MEM.SAV, načte se jeho obsah do paměti před provedením skoku.

#### Načtení MEM.SAV - Load MEM.SAV

**N**

Tento povel se liší od stejně označeného povelu v DOS 2.5. Při něm je načten binární segmentovaný soubor, jakou funkce L, ale před načtením a spuštěním (podle definic rozbehových a inicializačních adres) se aktivuje soubor MEM.SAV. Při přechodu ze spuštěného programu zpět do DOSu se oblast paměti opět uchová v MEM.SAV.

#### Konfigurace systému a disketových jednotek -

System and drive configuration

**O**

#### Konfigurace systému:

Funkce O poskytuje rozsáhlé možnosti nastavování konfigurace jak konfigurovatelných diskových jednotek, tak i celého systému. Nastavování systému se od nastavování disketové jednotky rozlišuje zadáním samotného (RETURN) místo čísla jednotky. Nastavování systému probereme nejdříve.

Dialog s MYDOSem probíhá takto:

Select Item (**RETURN** for menu):  
Drive number or **RETURN**: (**RETURN**)

Nezadáním čísla jednotky sdělíme, že budeme nastavovat parametry celého systému. MYDOS pokračuje v dotazech:

Verify WRITEs? {Y | N}

Odpověď "Y" zapíná zápis s verifikací, který je pomalejší, ale poskytuje

zvýšenou spolehlivostí při zápisu dat. Odpověď "N" verifikaci vypíná. Po napsání odpovědi se nemačká (RETURN).

**Number of File Buffers? { číslo|(RETURN)}**

udává nejvyšší počet současně otevřených souborů. (RETURN)=standard=3.

**RAM disk present? { Y|N }**

odpověď "N" požádá instalaci RAM-disku a čtyři následující otázky:

**Size(K)? { velikost|(RETURN)|(RETURN)}**

(RETURN) volí standardní velikost 64K pro 130 XE. U počítačů s instalovaným rozšířením paměti se odpoví podle typu rozšíření a velikosti paměti, která je pro RAM-disk k dispozici. Velikost musí být násobkem 16-ti K, maximum je 1024 K. U rozšíření firmy Newell Industries a Rambo XL firmy ICD je velikost 192 K, u rozšíření 320K z Compy Shopu je pro RAM-disk 256K. Velikost RAM-disku můžeme udat o 64K menší, pokud chceme vyhradit paměť pro BASIC XE.

**Control Address(HEX)? { adresa (RETURN)|(RETURN)}**

Samočný (RETURN) volí standardní adresu \$D301 pro 130 XE a kompatibilní RAM-disk. RAM-disk Axlon pro staré modely 800 má řídící adresu \$CFFF.

**Page sequence? { (RETURN)|číslo|(RETURN)|specifikace|(RETURN)}**

kde: -(RETURN) znamená standardní sekvenci pro 130 XE.

-číslo (RETURN) volí jednu z předpřipravených posloupností stránek (banků) paměti:

- 0 - pro 130 XE
- 1 - pro rozšíření Newell Industries, nebo kompatibilní 192 K
- 2 - 130 XE kompatibilní 128K
- 5 - pro Axlon RAM power

předdefinované sekvence volí vlastně počátek posloupnosti stránek v následující tabulce:

č. sekv.	masky stránek	maska OR
0	E3, E7, EB, EF	1C
1	83, 87, 8B, 8F	1C
2	C3, C7, CB, CF	1C
3	E3, E7, EB, EF	1C
4	A3, A7, AB, AF	1C
5	00, 01, 02, 03	FF
6	04, 05, 06, 07	FF

(pozn.: Předdefinovaná posloupnost se nehodí pro RAMBO XL firmy ICD, pro něj je třeba vypsat posloupnost explicitně)

-specifikace (RETURN) určuje podrobné posloupnost stránek paměti u RAM-disků, pro které není sekvence předpřipravena, nebo když chceme vyhradit část rozšířené paměti pro jiné účely. Specifikace určuje masky, které se použijí pro výpočet hodnot řídicí adresy při přepínání stránek normální a rozšířené paměti. Hodnota řídicího registru se vypočte nejprve operací OR mezi jejím obsahem a maskou. Výsledek se potom upraví operací AND s maskou pro příslušnou stránku a zapíše se zpět. Počet stránek musí odpovídat velikosti RAM-disku, uvedené v předchozí odpovědi.

Příklad 1 - RAMBO XL, 192K:

Page sequence: A3,A7,AB,AF,C3,C7,CB,CF,  
E3,E7,EB,EF,1C (RETURN)

Příklad 2 - Newell Industries 192K, stránky E3 až EF jsou vyhrazeny pro BASIC XE, pro RAM-disk je 128 K:

Page sequence: 1 nebo  
83,87,8B,8F,C3,C7,CB,CF,1C

Poslední odpověď přiřadíme RAM-disku číslo jednotky:

RAM disk drive no? { číslo 1 až 9 }

Při konfigurování systému nelze provedené změny zrušit ani klávesami BREAK nebo RESET. (RETURN) obvykle volí předdefinovaný standard. Pokud je konfigurační proces z jakéhokoliv důvodu přerušen, je třeba systém znova nastartovat, nebo změnu konfigurace správně dokončit.

#### Konfigurace diskových jednotek:

Konfigurace disketové jednotky se aktivuje zadáním jejího čísla:

Select Item (RETURN for menu):  
Drive number or RETURN: číslo (RETURN)  
Exclude drive? { Y | N }

První dotaz zjišťuje, zda se má jednotka ze systému vyřadit. Odpovíme-li "Y", je vyřazení provedeno a konfigurace končí. Odpovíme-li "N", následuje dotaz

Is drive configurable? { Y | N }

Při záporné odpovědi "N" nastavování končí. Odpovíme-li "Y", pokračuje se dotazem

High capacity drive? { Y | N }

Odpověď "Y" je vyhrazena pouze pro pevný disk (winchester, hard disk). Při jeho konfiguraci je nutno zjistit z manuálu odpovědi na následující otázky na počet sektorů atd. Pokud se odpoví "N", ptá se systém dále:

Double sided drive? { Y | N }

U jednotky 1050 se odpovídá "N", neboť je jednostranná. Jednotka XF551 je oboustranná. Pokud chceme použít kapacitu 360K, odpovíme "Y". Můžeme ji

však konfigurovat i jako jednostrannou. Další dotaz zjišťuje počet stop na jedné straně diskety

**Tracks/Side? { počet stop}**

Jak u 1050, tak u XF551 se odpovídá číslem 40. U jiných jednotek je třeba prostudovat manuál. Poslední odpověď udáváme rychlosť přechodu čtečí hlavy z jedné stopy na druhou pomocí kódu

**Step Rate? {kód}**

Nejrychlejší je 0, nejpomalejší 3. Správnou hodnotu je nutno vyzkoušet, nebo určit podle manuálu. U XF551 bývá hodnota 1. O obou posledních odpovědích něco podrobněji.

Přípustný počet stop na stranu je určen možnostmi příslušné jednotky. MYDOS připouští kromě standardní hodnoty 40 stop ještě 35 a 80 u jednotek 5 1/4" a 77 u jednotky 8". Hodnota Step rate je uvedena tabulkou:

hodnota	disk 8"	disk 5 1/4"
0	3mm/stopu	6mm/stopu
1	6mm/stopu	12mm/stopu
2	10mm/stopu	20mm/stopu
3	15mm/stopu	30mm/stopu

**Nastavení hustoty - Set Density**

**P**

MYDOS si u většiny povelů určuje hustotu diskety podle dat, která jsou na ní zapsaná. Povel Set Density se používá k přepnutí hustoty záznamu před formátováním povelom "I", nebo k přepnutí u jednotek, které se nemohou přepínat automaticky (například Indus). Protože MYDOS přepíná hustotu podle vložené diskety při povelech, které čtou disketu nebo spouštějí program (B, L, N), nemá nastavení pomocí P účinek pokud se mezi "P" a "I" dělá některý z nich.

**Založení podadresáře - Make Directory**

**O**

Při formátování diskety se založí hlavní adresář (ROOT DIRECTORY), který může obsahovat až 64 (128) souborů nebo podadresářů, které mohou každý obsahovat dalších 64 (128) souborů nebo dalších podadresářů. Podrobnější vysvětlení problematiky a významu podadresářů je v kapitole 9. Podadresář je možno založit povelom Q. MYDOS se ptá:

**Full directory name?**

V odpovědi máme dvě možnosti, buď uvedeme cestu relativní, nebo cestu absolutní. U absolutní cesty začínáme cestu k podadresáři od hlavního adresáře, u relativní cesty od pracovního adresáře, tedy toho, co je nastaveno jako D: (viz kapitola 9 a funkce R). Podadresář zabírá 8 sektorů a může být v poveloch uváděn jako podadresář s dvojtečkou na konci, nebo zrušen povelom D. Zrušení však je možné jen tehdy, když v podadresáři nejsou obsaženy žádné soubory. Soubory lze vyřadit také povelom D. Nejlépe se zakládání

podadresáře a různé možnosti, které máme k dispozici ukáže na příkladech ukazujících založení struktury popsané v kapitole 9. Ve zkratce řečeno se v hlavním adresáři založí podadresář BASIC a ASSEMBLR. V podadresáři BASIC založíme podadresář HRY a VYUKA. Ukážeme si při tom závislost relativní adresace na nastavení pracovního adresáře.

Příklad 1: Založení podadresáře BASIC v hlavním adresáři na disku 1.

```
Select Item (RETURN for menu):Q
Full directory name?
BASIC (RETURN)
```

V tomto případě jsme použili relativní cestu, která je téměř shodná s cestou absolutní, protože pracovním adresářem je hlavní adresář.

Příklad 2: Za stejného stavu jako v příkladu 1 založte podadresář ASSEMBLR za použití absolutní cesty.

```
Select Item (RETURN for menu):Q
Full directory name?
1:ASSEMBLR (RETURN)
```

Je vidět, že v těchto případech je relativní a absolutní cesta téměř shodná, protože pracovní adresář je nastaven na hlavní adresář diskety. V dalších příkladech se s pracovním adresářem posuneme na nově vytvořený podadresář BASIC a uvidíme, jak se rozdíl zvětší.

Příklad 3: Pracovní adresář je D1:BASIC. Založte v něm podadresář HRY s použitím relativní cesty.

```
Select Item (RETURN for menu):Q
Full directory name?
HRY (RETURN)
```

Jednoduché, že? Založení adresáře pomocí absolutní cesty, ukáže další příklad.

Příklad 4: Pracovní adresář je D1:BASIC. Založte v něm podadresář VYUKA pomocí absolutní cesty.

```
Select Item (RETURN for menu):Q
Full directory name?
1:BASIC:VYUKA (RETURN)
```

Absolutní cesta je tím delší, čím více podadresářů do sebe vnořujeme. Proto je relativní cesta obvykle kratší a výhodnější. V některých případech však nelze relativní cestu použít. Je to v případě, kdy chceme založit podadresář jako součást některého nadřízeného nebo paralelního adresáře.

Příklad 5: Pracovní adresář je D1:BASIC. V podadresáři ASSEMBLR, který je v hlavním adresáři, založte podadresář SUBRUT.

Select Item (**RETURN** for menu):Q  
Full directory name?  
**D1:ASSEMBLR:SUBRUT** (**RETURN**)

V tomto případě nelze relativní cestu použít, neboť MYDOS nemá prostředek k vyjádření zpětného postupu do nadřízeného adresáře.

### Nastavení pracovního adresáře - Pick Directory

**R**

Pomocí této funkce můžeme nastavovat pracovní adresář, neboli určit co se při zpracování povetu dosadí za výraz "D:". To je velmi praktické, neboť si můžeme například nakopírovat potřebné soubory do RAM-disku a po přehození pracovního adresáře na RAM-disk pracovat na něm jako na "D:". Totéž platí pro podadresáře. Chceme-li pracovat na výukovém programu v Basicu, nastavíme si pracovní adresář na "D1:BASIC:VYUKA" a potom stačí uvádět všude jen "D:jméno". Cestu si MYDOS doplní sám. Práce je tedy stejná, jakoby na disketě byly jen soubory podadresáře "D1:BASIC:VYUKA", nebo jinak řečeno jakoby se tento podadresář stal pro tuto chvíli hlavním adresářem. Stejně jako u funkce Q lze pracovní adresář nastavovat relativně, nebo absolutně. Rozdíly viz též kapitola 9. Pokud se v jednotce disketa vymění, je nutno nastavit pracovní adresář znova, pokud se nejedná o hlavní adresář.

### 10.2. Nové a změněné funkce manažera souborů volané přes CIO

MYDOS podporuje všechny funkce CIO (Central Input - Output system), které má DOS 2.0 s některými doplňky a změnami. Modifikovaný jsou funkce OPEN (funkční kód 3) a FORMAT (funkční kód 254). Dále jsou nově přidány funkce pro práci s podadresáři MAKE DIRECTORY (funkční kód 34), SET DIRECTORY (funkční kód 41) a funkce LOAD MEMORY (funkční kód 39). Funkce CIO jsou z Basicu přístupné pomocí volání XIO nn, kde n je funkční kód.

#### Funkce OPEN, funkční kód 3

Funkce OPEN u ATARI DOSu 2 nepoužívá (u diskety) parametr AUX2. U MYDOSu, pokud je AUX1 = 8, tedy soubor je otevřen pro zápis, může mít AUX2 významy podle tabulky:

bit	význam
1	je-li bit=1, je soubor založen ve formátu DOS 1, pokud není disk ve dvojitě hustotě.
2	je-li bit=1, má soubor formát MYDOS 4
6	je-li bit=1, je soubor založen jako blokován proti zápisu

Nastavení bitu 1 způsobí založení souboru ve tvaru používaného nejstarším

diskovým operačním systémem firmy Atari DOS 1. Tuto funkci dnes již málo využijeme. Tento formát nelze použít na disketě se sektory délky 256 byte, proto je u disket ve dvojitě hustotě tento bit ignorován. Další řídicí bit je bit 2. Tímto bitem lze vyvložit soubor ve formátu MYDOS 4, který nelze (snadno) číst pomocí systémů ATARI nebo OSS. Soubory tohoto formátu mohou používat sektory vyššího čísla, než 1023. Nastavení bitu 6 způsobí, že založený soubor má nastaven příznak blokování proti zápisu. Funkce OPEN používá adresu bufferu jako pointer na místo uložení jména souboru, které je ukončeno jakýmkoli znakem mimo 0 až 9, A až Z, a až z, :, ?, \*. Zde je třeba poznamenat, že délka jména nemůže být omezena pevnou délkou, neboť může obsahovat ještě cestu, neboli určení podadresáře. Doporučeným terminátem je bud znak NULL (\$00), nebo EOL (\$9B).

Rozdíl oproti DOS 2 a 2.5 je u souborů otevřených pro připojení (append). MYDOS nenechává při připojování částečně zaplněné sektory. To má dva následky. Jednak otevření selže, pokud soubor nelze připojit těsně, jednak se velikost spojovaného souboru kopírováním na jinou disketu nemůže měnit.

#### **Funkce GET RECORD, funkční kód 5**

Zůstává beze změny. Odpovídá povelu INPUT v Basicu.

#### **Funkce GET CHARACTERS, funkční kód 7**

Zůstává beze změny. Odpovídá Basicovskému povelu GET v případě čtení jednoho bytu. Čtení řetězce delšího, než 256 bytů je rychlejší, než samostatné čtení několika kratších.

#### **Funkce PUT RECORD, funkční kód 9**

Zůstává beze změny. Odpovídá povelu PRINT

#### **Funkce PUT CHARACTERS, funkční kód 11**

Beze změny. V případě zápisu jednoho znaku odpovídá povelu PUT.

#### **Funkce CLOSE, funkční kód 12**

Beze změny.

#### **Funkce READ STATUS, funkční kód 13**

Povel STATUS se na rozdíl od DOS 2.5 posilá na neotevřený kanál, parametrem je jméno souboru, na které ukazuje adresa bufferu. Chybové hlášení dostaneme když soubor na disku není, nebo když je blokován. V ostatních případech je funkce ukončena úspěšně.

#### **Funkce RENAME, funkční kód 32**

Jako parametr je předáván znakový řetězec, který obsahuje určení přejmenovaného souboru, nebo souborů. Potom následuje jeden neplatný znak (viz 10.2.1.) a nové jméno, které nesmí obsahovat ani specifikaci jednotky, ani cestu.

Příklad: Jako neplatný znak je použita čárka.

"D2:TEST:PGMS:A.OUT,ZCPY"

Pokud se přejmenovává některý podadresář, udává se jen část cesty vedoucí k němu. Chceme-li v předchozím příkladu přejmenovat podadresář PGMS na PROGS, použijeme řetězec "D2:TEST:PGMS,PROGS". Tím se cesta k souboru z "D2:TEST:PGMS:ZCPY" změní na "D2:TEST:PROGS:ZCPY". Filtr je povolen jen za poslední dvojtečkou ve jméně.

#### Funkce DELETE FILE, funkční kód 33

Kromě rušení souborů lze touto funkci rušit i podadresáře, pokud jsou prázdné. Rušený soubor nesmí být blokován. Podobně jako u DOSu ATARI nelze zrušené soubory snadno obnovit, ani pokud nejsou přepsány.

#### Funkce MAKE DIRECTORY, funkční kód 34

Nová funkce odpovídající funkci Q v menu. Slouží k zakládání podadresářů v hlavním adresáři, nebo podadresáři. Nastavení AUX1 a AUX2 je jako u OPEN pro výstup. Založení podadresáře neovlivňuje nastavení pracovního adresáře.

#### Funkce LOCK FILE, funkční kód 35

Beze změny.

#### Funkce UNLOCK FILE, funkční kód 36

Beze změny.

#### Funkce POINT, funkční kód 37

Nastavení ukazatele pozice v souboru. Předává se 3-bytová disková adresa (první 2 byty sektor, třetí je číslo byte) ve 12-tém až 14-tém bytu IOCB.

#### Funkce NOTE, funkční kód 38

Čtení ukazatele pozice v souboru. Na rozdíl od DOS 2.5 předává tříbytovou adresu, jako u funkce 37.

#### Funkce LOAD, funkční kód 39

Funkce LOAD umožňuje načíst a eventuálně spustit binární segmentovaný soubor. Funkční kód je 39. Načítání a spouštění je řízeno hodnotou AUX1.

AUX1	význam
4	prováděj se adresy RUN i INIT
5	provádí se jen RUN
6	provádí se jen INIT
7	INIT i RUN se ignorují

Jakákoliv jiná hodnota AUX1 způsobí chybové hlášení a neudělá nic.

### Funkce SET DIRECTORY, funkční kód 41

Funkce nastavuje pracovní adresář. V bufferu se předpokládá jméno existujícího podadresáře.

### Funkce FORMAT, funkční kód 254

Formátování u DOS 2.0 neumožňuje žádné volitelné možnosti, jediný formát je jednoduchá hustota. MYDOS umožňuje více různorodých formátů, jejichž výběr je dán hodnotami parametrů AUX1 a AUX2. Bit 7 AUX1 určuje, zda se bude formátovat fyzicky celý povrch diskety. Je-li 1, formátování je vynecháno a je opouze založen prázdný adresář. Bity 6 - Ø AUX1 a celý parametr AUX2 určují počet sektorů, který má být na disku založen. Jsou-li všechny tyto bity (všech 15) nulové, je počet sektorů určen konfigurací disketové jednotky.

### 10.3. Struktura a organizace disku ve formátu MYDOS.

MYDOS 4.2 používá první tři sektory pro zaváděcí program a základní diskové informace. Zaváděcí program je zde jen v případě, že je na disketu DOS.SYS a DUP.SYS. Sektor \$168 (a další v pořadí \$167, \$166 ... atd podle potřeby) obsahuje tabulku VTOC a některé informace o formátu diskety. Sektor \$169 až \$170 obsahují hlavní adresář. Pokud je disketa formátována jako jednostranná jednoduché hustoty se 719 sektory, je přesně stejná, jako u DOSu 2 nebo DOSu 2.5. (Nesmí samozřejmě obsahovat podadresáře). Základní formát jednoduché hustoty MYDOSu se liší jen v tom, že sektor 720 není ponechán mimo jeho dosah a formátovaná disketa má 708 volných sektorů. Pokud je zvolen velkokapacitní formát, rozdíl se zvětšuje s tím, jak roste směrem dolů tabulka VTOC. Z toho také vyplývá, že kompatibilita mezi MYDOSem a DOS 2.5 je pouze v oblasti jednoduché hustoty za předpokladu, že se nepoužívají podadresáře. Adresář disku ve velkokapacitním formátu lze DOSem 2.5 číst, ale data v souborech lze dosáhnout pouze na prvních 1023 sektorech a pokud je vyfazena kontrola čísel souborů (viz kapitola 3.3.2.). MYDOS může celkově zpřístupnit disk s 65535 sektory o 256 bytech, což odpovídá asi 16Mb. Kompatibilita se dále snižuje u disket ve dvojitě hustotě, protože různí výrobci používají poněkud odlišná schémata.

Struktura diskety v dvojitě hustotě je však vždy taková, že první tři sektory jsou krátké (128 b) a všechny ostatní jsou dlouhé (256 b). To je důvodem, proč takovéto diskety často nelze kopírovat pomocí programů jako US-COPY. Velikost sektoru můžeme zjistit povelom STATUS. Pokud je bit 5 stavového slova nastaven do jedničky, je sektor dlouhý.

### 10.4. Obsazení paměti

MYDOS obsazuje oblast paměti od \$700 do \$1E6C, menu leží v oblasti \$1E6D až \$3721. Dále se při funkci menu používají adresy \$D4 až \$E3 a volají rutiny matematické ROM (jen DUP.SYS). Nejdůležitější adresy, jejichž obsah řídí parametry MYDOSu jsou povětšinou stejné, jakou u DOS 2.5. Maximální počet současně otevřených souborů je na adrese \$709 (1801). Může zde být 0 až 16, standard je 3, což je minimum požadované pro DUP.SYS. Každý soubor navíc zvětšuje

rezidentní část MYDOSu o 1024 bytů. Adresa řídící druh zápisu PUT nebo WRITE, neboli bez a s verifikací je \$779 (1913). Volba je stejná jako u DOS 2.5 (viz 4.3.4). Zápis bez ověřování se docílí zapsáním "P" (\$50, 80), s ověřováním pomocí "W" (\$57, 87) na tuto adresu.

#### 10.5. Změny parametrů a instalace RAM-disku.

Driver RAM-disku je v MYDOSu již začleněn a pokud jej při konfiguraci povélem O aktivujeme, je funkční při každém natažení MYDOSu. To je rozdíl oproti DOS 2.5, kde se RAM-disk instaluje programem RAMDISK.COM, který jej vždy naformátuje a původní obsah je ztracen. Kromě toho se RAM-disk u DOS 2.5 dá nainstalovat jen při zapnutí počítače, později už ne. MYDOS instaluje RAM-disk vždy (pokud ho má v konfiguraci, viz 10.1.16) a jeho inicializace spočívá jen v naformátování a nakopírování souborů DUP.SYS a MEM.SAV, eventuálně dalších. Inicializaci je možno si přizpůsobit podle vlastních potřeb. Na systémové disketě je k tomu účelu zdrojový text inicializačního programu s názvem RAMBOOT.MAC. Text je bohatě komentován a ani pro programátora, který není v assembleru hvězdou, by neměl být problém vytvořit si modifikaci podle přání. Vytvořený text lze zkompilovat některým z překladačů assembleru. Na systémové disketě je pro tento účel překladač A65, který se po spuštění ptá na jméno zdrojového souboru, kam uložit výsledný produkt a kam použít listing. Také lze použít komplikátor AMAC. Pro použití některého ze sérií komplikátorů Assembler/Editor, EASMD, MAC/65 je třeba zdrojový text poněkud upravit díky odlišným direktivám. Pokud se má inicializátor RAM-disku spouštět automaticky, musí se pojmenovat AUTORUN.SYS. Pokud se má kromě něj spouštět ještě nějaký další program, musí se s inicializátorem spojit pomocí volby APPEND funkce C, respektive připojit za inicializátor RAM-disku. Inicializace nesmí být automatická v případě, že chceme mít RAM-disk rezidentní, to znamená aby se jeho obsah uchoval i během studeného startu, pokud se nevypne počítač. Tehdy je nutno spouštět inicializaci ručně funkcí L po zapnutí počítače. Při dalším studeném startu již inicializaci nespouštíme a obsah RAM-disku zůstává zachován. U MYDOSu verze 4.3B je nová verze inicializačního programu RAMBOOT3, která při inicializaci nakopíruje do RAM-disku obsah podadresy "D1:RAMDISK:".

Změna parametrů MYDOSu je možná dvěma způsoby. Prvním je funkce O v menu (viz 10.1.16), druhým přepis obsahu řídicích adres příkazy POKE v Basicu, nebo některým z monitorů nebo ladicích systémů. Veškeré takto provedené změny jsou provedeny pouze v paměti. Pokud je chceme zachovat trvale na disku, je nutno použít funkci H v menu.

Nakonec několik poznámek ke dvojitě hustotě. MYDOS lze nainstalovat i na disketu nahranou ve dvojitě hustotě. Je schopen se z ní úspěšně spouštět pokud jednotka umí rozpozнат hustotu vložené diskety a přepnout se automaticky. Na jednotce XF551 lze MYDOS spustit i z diskety ve dvojitě hustotě s výjimkou automatické inicializace RAM-disku. Pokud ji potřebujeme používat, musíme MYDOS natahovat z diskety s krátkými sektory 128 b. MYDOS je schopen přepínat se automaticky podle formátu vložené diskety, pokud má tuto schopnost připojená jednotka. U XF551 se přepínání dáje při některé ze

čtecích operací. Protože před zápisem vloženou disketu již nekontroluje, není s ní pomocí MYDOSu možné kopírovat pomocí jedné jednotky soubory z diskety s krátkým sektorem 128 b na disketu s dlouhými sektory 256 b. Takovéto kopírování je možné jen pomocí další jednotky, nebo RAM-disku.

#### 10.6. Komunikace s diskem přes SIO

Disketové jednotky podporované MYDOSem jsou připojovány na sériovou sběrnici, i když velkokapacitní disky a pevný disk (winchester) lze připojit i na paralelní systémovou sběrnici. Část operačního systému počítače (OS - ROM), která zprostředkovává fyzický kontakt se sériovou sběrnici se nazývá SIO. MYDOS používá tuto fyzickou řevedení komunikace pro předávání povelů do a z jednotky. Některé povelby byly definovány firmou Atari pro jednotky 810 a systémy pro práci s jinými jednotkami používají tuto sadu rozšířenou o některé povelby. MYDOS může v omezené míře pracovat s kteroukoliv jednotkou která používá celý standard 810, ale pro použití všech možností MYDOSu je třeba aby jednotka podporovala celou rozšířenou sadu povelů. Jako další funkce nutná pro automatické nastavování hustoty záznamu je třeba, aby jednotka automaticky rozpoznala hustotu záznamu podle vložené diskety, pokud je první operaci čtení sektoru 1. Je to nutné pro nařízení systému z obou možných hustot. Minimální množina funkcí pro práci s MYDOSem je v tabulce.

zař. čís.	povel	směr	poč. bytů	byty AUX	funkce	
\$31	č.j.	\$21	vstup	128/256	1-720	formátování
-	..	\$50	výstup	128/256	1-720	PUT (zápis bez ověř.)
-	..	\$52	vstup	128/256	1-720	čtení
-	..	\$53	vstup	4	1-720	status
-	..	\$57	výstup	128/256	1-720	WRITE (zápis s ověř.)

Počet bytů v sektoru je vždy 128 pro jednoduchou hustotu a pro první 3 sektory při dvojitě hustotě. Funkce format se nikdy nevclá s AUX byty mimo rozsah 1 až 720. Předpokládá se, že první byte předaný povelom STATUS indikuje v bitu 5 velikost sektoru (1=256, 0=128). Byty AUX slouží k předávání čísla sektoru v rozsahu 1 až 720 u diskety kompatibilní s DOS 2.0 nebo jednoduchou hustotou DOS 2.5, nebo v rozsahu 1 až 65535 u velkokapacitních disků. Pro dynamickou konfiguraci konfigurovatelných disket slouží další dva povelby.

zař. čís.	povel	směr	poč. bytů	byty AUX	funkce	
\$31	č.j.	\$4E	vstup	12	1-720	čtení konfigurace
..	..	\$4F	výstup	12	1-720	zápis konfigurace

Tyto povelby se používají pro nastavování jednotek, které byly při startu systému rozpoznány jako konfigurovatelné. Pokud jednotka tyto funkce nezná, je třeba ji definovat jako nekonfigurovatelnou (Atari 1050, Indus GT). Dynamická konfigurace se také používá při funkci P v menu například pro formátování diskety. U jednotky Indus je třeba přepnout povelom P, potom

přepínačem shodně přepnout jednotku a pak teprve formátovat povelem I.) Konfigurační blok má následující strukturu.

- byte 0: Počet stop na jedné straně (standard 40)
- byte 1: Step Rate podle Western Digital (0 až 3)
- byte 2: nula (horní byte počtu sektorů na stopu)
- byte 3: počet sektorů/stopu (standard 18 nebo 26 pro ED)
- byte 4: kód počtu stran (0-jednostranná, 1-oboustranná)
- byte 5: kód typu disku:
  - bit 2: 0-jednoduchá hustota, 1-dvojitá
  - bit 1: 0- 5 1/4", 1- 8"
- byte 6: horní byte počtu bytů v sektoru (pro 1050 = 0)
- byte 7: dolní " .. - (pro 1050 = 128)
- byte 8: řízení operací
  - bit 7: 1-40-ti stopová operace na 80-ti stopové jednotce
  - bit 6: vždy 1 (indikuje přítomnost jednotky)
  - bit 1: 1- sektory 1,2 a 3 mají plnou velikost
  - bit 0: 1- číslování sektorů od nuly (0-17 místo 1-18)
- byty 9 až 11 MYDOS nepoužívá a po přečtení bloku je zapisuje zpět v tom tvaru, v jakém je přečetl.

Změny v těchto povelech jsou při konfiguraci pevného disku. Konfigurování pevného disku je složitá záležitost a konfigurace je obvykle vestavěna do kontroléru, nebo zapsána na "magické místo" disku. Pro československého uživatele naštěstí tyto starosti sotva přicházejí v úvahu.

## 11. Sparta DOS verze 3.2

Sparta DOS, nebo jak jej označuje výrobce SpartaDOS Construction Set, je součástí obsáhlé soupravy softwarových i hardwarových doplňků a rozšíření pro osmibitové počítače Atari, kterou dodává firma ICD, Inc., 1220 Rock Street, Suite 310, Rockford, IL 61101-1437, U.S.A. Sparta DOS je nejobsáhlejší diskový operační systém, jaký byl zatím napsán pro počítače Atari osmibitové řady. Jeho koncepcie a řízení zahrnuje všechno, co má správný operační systém mít a nemusí se stydět ani vedle CP/M nebo MS-DOS. S těmito celosvětově rozšířenými systémy má hodně společného a nenašlo by se mnoho funkcí, které v něm oproti nim chybí. Jsou tu podadresáře, možnost přesměrování vstupu i výstupu, možnost dávkového zpracování, označování souborů datem a časem vzniku, podpora všech možných rozšíření a specialit od RAM-disku přes široké spektrum diskových jednotek a hodiny reálného času až po pevný disk do 16 Mbyte. Sparta DOS ovládá bez problému všechny existující formáty záznamu na diskety a umí je automaticky přepínat. Nelze jej zmást výměnou disket různých formátů a umí i pomocí jediné jednotky kopírovat soubory mezi disketami nahranými v různé hustotě. Mezi velké přednosti patří i možnost operativního zapínání a vypínání vestavěného Basicu bez studeného startu. SpartaDOS pracuje s diskovými jednotkami maximální možnou rychlostí a neužene jej ani Speedy 1050 z Compy Shopu, které čte celou stopu najednou. Také daleko rychleji přepisuje a ruší existující soubory. Je jasné, že tak "chytrý" systém také musí i při vši optimalizaci zabírat pořádný kus paměti. Is tím se však tvůrci SpartaDOSu vypořádali. Větší část systému schovali do paměti RAM v adresovém prostoru ROM operačního systému, kde leží vlastně schována pod ním a podle potřeby se přepíná mezi OS a rutinami DOSu. Je to také snad jediná nevýhoda Sparta DOSu, protože pod ním nemohou běžet programy, činící si nárok na paměť pod OS-ROM. Jedná se zejména o populární Turbo Basic, nebo Čapek pozdější verze. Protože však Sparta DOS pochází z roku 1985, je spíše nevýhodou téhoto programů, že s ním nepočítaly. Není třeba vysvětlovat, že takový systém s mnoha možnostmi má také samozřejmě složitější ovládání, než třeba DOS 2.5 a využití všech jeho možností bude vždy vyhrazeno spíše pro vyspělé uživatele a programátory. Na druhou stranu umožňuje díky možnosti dávkové práce připravit uživatelské celky, které pak může podle vypisovaných pokynů obsluhovat i úplný začátečník.

Protože k vyspělému systému patří i podobná a přehledná dokumentace, kterou firma také dodává, budeme se při popisu dřížet struktury původního manuálu. Povely SpartaDOSu jsou sdruženy ve skupinách podle účelu a oblasti použití. Protože se předpokládá znalost jiných diskových systémů, nebudeme se již znova zabývat známou problematikou.

## 11.1. Všeobecný přehled SpartaDOSu

### Všeobecná terminologie

V popisu a cvládání SpartaDOS u jsou používány následující konvence:

- ♦ klávesa ESC přeruší většinu externích povelů. Povely jako DUMP se přeruší klávesou BREAK.
- ♦ (RETURN) znamená stisknutí klávesy RETURN. Kromě zvláštních případů se (RETURN) používá k ukončení zadávání povelu vždy, proto nebude u povelu již dále zdůrazňován.
- ♦ Uvozovky nebo apostrofy označují v textu začátek a konec povelu, nebo parametru, který se má zadat. Je-li třeba v textu napsáno "D3:INIT(RETURN)", napíše se D3:INIT bez uvozovek a stiskne se RETURN.
- ♦ Mnohé povely vyžadují zadání adres, nebo offsetů. Tyto hodnoty se zadávají vždy jako hexadecimální čísla.
- ♦ Některé povely mají omezenou platnost na určitý formát diskety, nebo určitou verzi SpartaDOSu. Zabýváme se sice jen verzí 3.2, ale je třeba vědět, že existují hlavní verze 1, 2 a 3, každá s více podverzemi. Značí se to výrazem n.x, kde n je číslo hlavní verze a x číslo její modifikace. Při tom se používají dva terminy. CP verze n.x označuje vykonavač povelů (Command Processor) příslušné verze, diskety verze n.x označují formát disket používaný příslušnou verzí. Některé povely nefungují na disketách založených staršími verzemi. Povel XINIT generuje diskety verze 2.x, povely INIT a FORMAT diskety verze 1.x! Diskety Atari DOS 2 jsou diskety v obvyklém formátu jednoduché hustoty vytvořené povelom AINIT nebo příslušným povelom DOS 2.5 apod. Povely týkající se podadresářů a podobně na těchto disketách neúčinkují. I když SpartaDOS používá odlišný formát disket, může od verze 2.x čist, spouštět programy a zapisovat i diskety Atari DOS 2 v jednoduché i dvojitě hustotě. SpartaDOS verze 3.2 může navíc čist a spouštět programy i z disket DOS 2.5 v rozšířené hustotě.
- ♦ Při syntaktických definicích a popisech povelů budeme používat následující označení:

- |       |  |
|-------|--|
| jméno | jméno souboru bez koncovky o délce 1 až 8 znaků.   |
| .ext  | koncovka   |
| cesta | úplný popis postupu od pracovního adresáře k adresáři ve kterém je dotyčný soubor. Neobsahuje jméno souboru! |

Dále se používají obvyklá označení volitelných nebo nepovinných částí pomocí hranatých a složených závorek a svislá čára oddělující alternativní položky. Pro popis disket ve formátu SpartaDOSu se často používají následující terminy.

### Jmenovky (Volume Names)

zapisuje SpartaDOS na diskety při formátování. Každé disketě je třeba přiřídit jednoznačné jméno jako "Hry\_1" nebo "000243". Jmenovka může mít 1

až 8 znaků a obsahovat kterýkoli z 256-ti možných. Diskety verze 1.x musí mít jedinečná jména, jinak mohou nastat vážné havárie.

SpartaDOS používá zcela jiný systém sektorových vyrovnávacích pamětí, než DOS 2.5. Kdykoliv je sektor čten, SpartaDOS kontroluje, zda jej již nemá v paměti. Pokud se v jednotce vymění disketa, nemá systém jinou možnost na to přijít, než kontrolou obsahu určitého sektoru s daty uloženými v paměti. Proto se pokaždé při otevírání souboru čte jmenovka a porovnává se se jmenovkou přečtenou předtím. Pokud se jmenovka změnila, SpartaDOS si ji uloží a vymaze všechny sektorové vyrovnávací paměti. Pokud by stará a nová disketa měla stejnou jmenovku, nemohl by systém poznat, že došlo ke změně a nová disketa by byla ohrožena chybným zápisem. Proto je od verze 2.x kontrola vylepšena ještě o zápis dvou náhodných čísel, takže u disket formátu 2.x již toto nebezpečí nehrozí.

#### **Adresáře (Directories)**

mohou obsahovat až 128 souborů. Hlavní adresář se jmenuje MAIN a mohou v něm být zakládány podadresáře. Pro podadresáře platí totéž, co pro soubory s tím rozdílem, že mají ve výpisu adresáře místo délky souboru označení "<DIR>" a pro jejich založení a zrušení jsou ve SpartaDOSu zvláštní povely. V hlavním adresáři a podadresářích mohou být zakládány další podadresáře. Jediným omezením je prostor na disku. Cesta se používá k označení způsobu, jak se dostat z jednoho adresáře do jiného. (viz kap. 9.)

#### **Pracovní adresář (Current Directory)**

je adresář, ve kterém se právě nacházíme. Pokud se v povelu zadá jméno souboru bez udání cesty, hledá se soubor v pracovním adresáři. Pracovní adresář se určuje povelom CWD.

#### **Hlavní adresář (MAIN Directory)**

nelze vymazat. SpartaDOS nastaví hlavní adresář jako pracovní pokaždé při vložení nové diskety, nebo po stisku klávesy RESET. Soubory otevírané pro čtení se hledají nejprve v pracovním, potom v hlavním adresáři. Proto je výhodné ukládat v hlavním adresáři všechny obecně používané soubory. Tento proces je interní záležitost SpartaDOSu, nikoli interpretu povelů. Pokynem pro jeho spuštění je otevření souboru v modu pro čtení, takže proces postupného hledání souboru funguje nejen pro externí povely a dávkové soubory, ale i v Basicu nebo jiném uživatelském programu.

#### **Podadresáře (Subdirectories)**

SpartaDOS používá stromovou strukturu. Hlavní adresář můžeme považovat za kmen, podadresáře za větve z nichž mohou odbočovat další větve. Při popisu cesty od jednoho adresáře ke druhému se používají znaky ">" pro posun do podadresáře nižší úrovně a "<" pro posun do nadřízeného adresáře. Znak ">" na začátku cesty způsobí, že cesta začíná od hlavního adresáře, neboli jde o absolutní cestu. Znak ">" se používá jako oddělovač jednotlivých názvů podadresářů ve specifikaci cesty. (viz též kapitola 9.)

### Povelový procesor - interpret povelů CP (Command Processor)

provádí interní povely, externí povely a povelové soubory dátového zpracování. Obecná syntaxe externího povelu je tato:

[Dn:] [cesta >] jméno [parametry] (RETURN)

Oba druhy povelů lze u CP verze 3.2 psát libovoině malými i velkými písmeny. Pro povelové soubory dátového zpracování je vyhrazena koncovka .BAT a spouští se napsáním "-jméno (RETURN)". Koncovka se nepíše, pokud se dodržují standardy (viz 11.2.).

### Handlery a drivery

SpartaDOS má jako součást systémové diskety několik handlerů (obsluhovaců zařízení), které se po zavedení do paměti stávají rezidentní, například RS232, AT\_RS232, RD atd.

### Volby při formátování

SpartaDOS má při formátování možnost volit z mnoha formátů, ale lze volit jen takové, které odpovídají možnostem disketové jednotky. Lze sice například formátovat jednotku 1050 jako osmdesátistopovou a těšit se z údaje 1400 volných sektorů ve výpisu adresáře, ale při kopírování na takovouto disketu nás brzy vrátí chyba 144 zpátky k realitě. Po zapsání formátu se jednotka automaticky přepíná kdykoliv se na ni má psát, nebo z ní číst.

## 11.2. Syntaxe SpartaDOSu

### Soubory

Protože SpartaDOS podporuje podadresáře, je třeba definovat který soubor chceme číst nebo zapisovat. Určení souboru se skládá ze dvou složek, z cesty a jména, které složeny dohromady určují umístění souboru.

### Konvence pro jména

Úplné jméno se skládá ze jména a popisné části ("jméno [.ext]"). Pravidla protivorbující jsou stejná, jako u DOS 2.5 s několika rozdíly. Lze používat znak podtržení ("\_") a jméno může začínat i číslicí. Podtržení se na začátku jména kvůli přehlednosti nedoporučuje, i když je přípustné. Kterýkoliv z nepřípustných znaků uvedených ve jméně způsobí, že je jím jméno ukončeno. Tak například "MJ28/j88.POS" bude SpartaDOS chápat jako "MJ28" a zbytek bude ignorovat.

SpartaDOS používá určité vyrazené koncovky jako standardy pro určité druhy souborů. Pokud tyto standardy používáme, není třeba psát koncovku. Standardy pro SpartaDOS jsou v tabulce.

- .COM - povelový soubor externího povelu (typ load-and-go)
- .BAT - povelový soubor dávky
- .DOS - modul SpartaDOSu pro INIT a XINIT

## Filtr

Konstrukce výběrového filtru je u SpartaDOSu stejná, jako u DOS 2.5. Pokud se v povelu neudá jméno, platí jako standard "\*.\*". SpartaDOS povoluje používání filtru při čtení souboru. Filtem se nesmí určovat soubor pro zápis.

## Jména adresářů

Jsou konstruována stejně jako jména souborů a mohou i obsahovat popisnou část, i když většina služebních programů SpartaDOSu popisnou část u jména adresáře nerozlišuje. Při výpisu adresáře ve formátu SpartaDOSu se koncovky objevují. V odkazech na adresáře lze používat i filtr podle stejných pravidel, jaká platí pro jména souborů s výjimkou povelu CREDIR, kde je nutno uvést celé jméno a filtr se nesmí použít.

## Cesta

SpartaDOS může mít na disketu celý systém do sebe vnořených adresářů a cesta se používá k popisu přechodu z jednoho adresáře k jinému (viz též kap. 9). Cesta je pro naše účely seznam adresářů, kterými je třeba projít do cílového adresáře. Znak ">" slouží jako oddělovač jednotlivých jmen adresářů při přechodu do nižší úrovně. Pokud cesta nezačíná v hlavním adresáři, můžeme přechod do nadřízeného adresáře označit znakem "<". Těchto znaků může být i více za sebou podle počtu úrovní, které je třeba projít směrem vzhůru, tedy ke hlavnímu adresáři. V některých případech je výhodné použít absolutní cestu, začínající od hlavního adresáře. Dosáhneme toho uvedením znaku ">" na začátku cesty. Pro účely vysvětlení dalších příkladů v manuálu se pro jméno adresáře bude používat termín "jménoA". Všeobecná syntaxe cesty je

[>] [jménoA > .. >jménoA]

kde volitelný znak ">" na začátku indikuje začátek cesty v hlavním adresáři a každé jméno adresáře "jménoA" nás posunuje o jeden adresář po cestě. ".." znamená libovolný počet opakování. Cesta začínající přechodem nahoru do nadřízeného adresáře má syntaxi

< [<..<] [jménoA] ..>jménoA]

kde každý znak "<" posunuje o jeden adresář zpátky. Předpokládejme na disketu systém souborů a adresářů z kapitoly 9. Výpisy adresářů by mohly vypadat třeba takto (data pořízení a délky souborů jsou vymyšleny a nejsou podstatné):

Úroveň 1: Volume: RAM-DISK  
Directory: MAIN

X32D	DOS	12890	10-07-85	3:50p
RD	COM	1585	1-15-86	4:25p
BASIC	<DIR>	10-07-85	5:30p	
ASSEMBLER	<DIR>	10-07-85	5:30p	
1391 FREE SECTORS				

úroveň 2:      Volume: RAM-DISK  
                   Directory: BASIC

HRY	<DIR>	10-07-85	5:31 p
VYUKA	<DIR>	10-07-85	5:31 p
PROG	BAS	890	10-07-85 3:51 p
1391 FREE SECTORS			

Volume: RAM-DISK  
                   Directory: ASSEMBLR

1391 FREE SECTORS

úroveň 3:      Volume: RAM-DISK  
                   Directory: HRY

JUMPER	BAS	890	10-07-85 3:51 p
1388 FREE SECTORS			

Volume: RAM-DISK  
                   Directory: VYUKA

ZEMEP	BAS	890	10-07-85 3:51 p
TEST	BAS	1292	10-07-85 3:51 p
MATEM	BAS	890	10-07-85 3:51 p
1388 FREE SECTORS			

Nyní si ukážeme popis absolutní a relativní cesty pro různé případy.

adresář	cesta absolutní	cesta relativní
pracovní	cílový	
MAIN	VYUKA	>BASIC>VYUKA
		BASIC>VYUKA
VYUKA	MAIN	>
		<<
HY	ASSEMBLR	>ASSEMBLR
		<<ASSEMBLR
ASSEMBLR	VYUKA	>BASIC>VYUKA
		<BASIC>VYUKA

Znak "<" pro postup nahoru lze uvádět jen na začátku cesty. Cesta označuje jen dotyčný adresář, nikoliv soubory v něm.

#### Typy povelů

SpartaDOS má dva typy povelů, *interní* a *externí*. Interní povely jsou plně prováděny povelovým procesorem. Jsou to povely jako DIR, ERASE apod. Většina interních povelů neporušuje uživatelskou paměť (např. pro Basic). Jsou zde však dvě výjimky: COPY používá programovou oblast jako buffer, BUFS mění její dolní hranici. Oba povely způsobují ztrátu obsahu uživatelské paměti a studený start zásuvného modulu. BuFs se u verze 3.2 nepoužívá.

Externí povely musí být při každém použití načteny z disku a v každém případě nižší obsah uživatelské paměti. Jsou to povely jako TREE (výpis

všech adresářů) nebo XINIT ( inicializace diskety) atd. externí povely spolu-pracují s povelovým procesorem pomocí tabulky, ve které jim CP předává parametry.

#### **Standardní jednotka (Default Drive)**

je jednotka, která se dosadí jako jméno zařízení, pokud se zařízení neuvede ve jméně souboru. Standardní jednotku používá pouze interpret povelů! Změna standardní jednotky se provede jednoduše. Jako povely se vypíše pouze kód nové jednotky. V příkladu je odpověď uživatele napsána kurzívou:

```
D1:D3:(RETURN)
D3:DIR (RETURN)
```

Písmeno D a dvojtečka jsou povinné. Uživatel nejprve změnil standardní jednotku na "D3:", potom s použitím standardů vypsal její hlavní adresář.

#### **Formáty výpisu adresáře**

SpartaDOS má dva možné tvary výpisu adresáře. Základní formát je ve stylu MS-DOS nebo CP/M a je ve světě osmibitových Atari unikátní. U každého souboru je kromě názvu a koncovky udána délka v bytech a datum a čas vzniku. Je samozřejmé, že poslední dva údaje mají smysl pokud je čas a datum nastaveno při začátku práce, nebo je-li instalován modul s hodinami reálného času R-TIME. Softwarový čas se u SpartaDOSu značně zpožděuje, protože je odvozen od americké obrazové frekvence 60Hz. Nicméně i tak může časový údaj pomoci rozlišit různá verze a rozmotat spletenecky několika-násobně přepisovaných nebo aktualizovaných souborů. Příklad takového výpisu:

```
Volume: POKUSY
Directory: MAIN
```

X32D	DOS	12890	10-07-85	3:50p
RD	COM	1585	1-13-86	4:25p
BASIC	<DIR>	10-07-85	5:30p	
ASSEMBLER	<DIR>	10-07-85	5:30p	
1391 FREE SECTORS				

Datum je udáno v americké formě mm-dd-yy, čas je ve 12-ti hodinovém systému. Písmeno "a" za ním značí dopoledne, "p" odpoledne. V tomto druhu výpisu není vidět, zda soubor je nebo není blokován proti zápisu.

Druhý možný tvar výpisu téhož adresáře je ve tvaru známém z DOS 2.5. Pod-adresáře jsou zde označeny inverzním DIR místo popisné části. Pokud by tedy měl některý z nich popisnou část, nebude při tomto druhu výpisu vidět. Jinak je tento výpis zcela shodný s výpisem DOS 2.5, ale pokud je počet sektorů (délky souboru, nebo počet volných) větší než 999, vypisuje se bez první číslice.

```

X32D      DOS 103
RD        COM 014
* BASIC    DIR 002
ASSEMBLR  DIR 002
391 FREE SECTORS

```

Hvězdičky a ostré závorky mají tentýž význam jako u DOS 2.5.

### **11.3. Popis povelů SpartaDOS**

Povelysou rozděleny do kapitol podle okruhu působnosti. U každého je uvedeno zda je interní nebo externí, a podmínky použití.

#### **11.3.1. Základní povely**

##### **DIR a DIRS**

Zobrazí výpis zadaného adresáře. DIR zobrazuje adresář v základní podobě s výpisem jmenovky, názvem adresáře, velikostí souboru v bytech, datem a časem. DIRS zobrazuje adresář ve stylu DOS 2.5 (u CP 2.x ve stylu DOS 2).

**Syntaxe:**     DIR [Dn:][cesta]>[jméno [.ext]]               nebo  
                 DIRS [Dn:][cesta]>[jméno [.ext]]

**Typ a omezení:** DIR i DIRS jsou interní povely

##### **Poznámky:**

Není-li zadáno jméno souboru, vypisují se všechny. Není-li zadána cesta, vypisuje se pracovní adresář. DIR aplikovaný na diskety formátu Atari dává výpis ve tvaru DIRS. Tisk adresáře na tiskárně viz povel PRINT.

##### **Příklady:**     DIR

Zobrazí ve formátu SpartaDOSu celý obsah pracovního adresáře standardní jednotky.

DIR D2:MODEM>XM\*.\*

Zobrazí z adresáře MODEM na disku 2 všechny soubory začínající na XM s libovolnou popisnou částí.

##### **CAR**

Slouží k přechodu z DOSu do zásuvného modulu.

##### **Syntaxe:**     CAR

**Typ a omezení:** Interní povel

##### **Poznámky:**

Protože SpartaDOS je rezidentní, program vytvořený pomocí modulu v paměti zůstává neporušen při přechodu z DOSu do modulu, pokud se při práci v CP nepoužil externí povel, nebo povel COPY nebo BUFS (je jen pro CP 1.x).

SpartaDOS verze 3.2 předává po nastartování řízení modulu. Pokud chceme aby byl po nastartování aktivní DOS, musíme použít povelové sekvence START-UP.BAT, ve které musí být alespoň jeden povrel, třeba i jen rádce komentáře.

### **BASIC**

slouží k zapínání a vypínání vestavěného BASICu na počítačích řady XL/XE.

**Syntaxe:**      **BASIC { ON|OFF }**

**Typ a omezení:** Interní povrel

### **Poznámky**

Při normálním nastartování počítače řady XL/XE bez zasunutého modulu se aktivuje vestavěný BASIC. Povel BASIC jej může za provozu vypínat a zapínat. Tento povrel způsobí RESET, neboli teplý start, ale může být v dávkovém povelovém souboru kdekoliv.

### **11.3.2. Inicializace diskety**

SpartaDOS umožňuje ve svých formátovacích povelech využití téměř jakékoli diskových jednotek od starých 810 přes 1050 a výrobků jiných firem jako PERCOM, INDUS, ASTRA až po XF551. Pomocí interfejsu ATR8000 firmy SWP je možné používat téměř jakoukoli jednotku. Formátovací povely nabízejí na výběr menu různých formátů a jsou stavěny tak, že je lze v budoucnu rozšiřovat.

### **INIT a XINIT**

jsou hlavní formátovací a instalacní povely SpartaDOSu.

**Syntaxe:**      **INIT**  
                     **XINIT**

**Typ a omezení:** Oba povely jsou interní.

INIT vytváří diskety formátu 1.x, XINIT formátu 2.x.

### **Povely:**

Oba povely čtou z disku moduly DOSu s koncovkou .DOS a podle výběru v menu se příslušná verze SpartaDOSu instaluje. INIT se prakticky nepoužívá.

**Příklady:**      **XINIT**

Program nejprve ukáže menu všech verzí SpartaDOSu obsažených na disketě a jako poslední položku "N" pro formátování bez instalace DOSu. Po vybrání se dotyčná verze načte do paměti. Následuje dotaz na číslo formátované jednotky. XINIT akceptuje čísla 1 až 8. Poté se objeví menu ve kterém vybíráme počet stop a stran jednotky. U 1050 volíme bod 1 (40 trk/SS), u XF551 lze volit 1 nebo 5 (40 trk/DS). U jiných jednotek je třeba určit správný formát

podle manuálu. Další krok je volba záznamové hustoty. U XF551 můžeme volit ze všech třech možností, u 1050 jen 1 nebo 3. Potom zadáme jmenovku diskety a poslední dotaz je na použití zvláštní organizace sektorů pro rychlý režim (UltraSpeed Sector Skew). Tuto volbu lze použít jen když je v jednotce nainstalován US Doubler. Ve všech ostatních případech je nutno odpovědět záporně. Pak již zbývá jen vložit do vybrané jednotky formátovanou disketu a stisknout (RETURN).

Z verzí SpartaDOSu je třeba se zmínit kromě verze X32D ještě o dvou verzích SpartaDOSu 2.x. Vlastnosti jednotlivých verzí jsou popsány v chronologickém pořadí.

Verze 2.x jsou podstatně vylepšeny oproti verzím 1.x. Mohou běžet jen na modelech XL/XE. Obě verze jsou téměř stejné, liší se pouze prioritou modulu a zpracováním AUTORUN.SYS při startu systému. Verze 2.x má již zahrnutou podporu rychlé komunikace s modulem UltraSpeed i s jinými zrychlenými systémy. Umí také číst a zapisovat diskety Atari DOS 2.

#### **XD23B.DOS**

Typ XD SpartaDOSu verze 2.x. Umí při startu obhospodařit STARTUP.BAT a prioritu při spuštění má DOS.

#### **XC23B.DOS**

Typ XC. Reaguje při startu na soubor AUTORUN.SYS a prioritu řízení má modul. Před načtením a provedením AUTORUN.SYS se zobrazí úvodní hlášení. Pokud se ihned po něm stiskne RESET nebo BREAK, zpracování AUTORUN.SYS se přeruší.

#### **X32D.DOS**

Vylepšená verze oproti verzím 2.x, upravená pro kompatibilitu s modulem Basic XE firmy OSS. Oproti verzím 2.x má následující vylepšení:

- ◆ Lepší podporu data a času (interní povely TD, DATE, TIME)
- ◆ Vnitřní interfejs pro modul R-TIME 8
- ◆ interní interfejs pro softwarové hodiny
- ◆ interní povel KEY a 32-znakový buffer klávesnice
- ◆ automatický minibufferový systém pro rychlejší jednobytové operace PUT a GET
- ◆ nové vektory pro podporu strojových programů
- ◆ Po stisku RESETu se řízení vraci tam, kde bylo předtím (DOS-DOS, modul-modul)
- ◆ podporuje STARTUP.BAT i AUTORUN.SYS
- ◆ povely BASIC ON/OFF mohou být i uvnitř povelového souboru, nejen na konci.
- ◆ bere se v úvahu řízení hlasitých nebo tichých I/O operací
- ◆ podporuje interfejs pro pevný disk SUPRA
- ◆ všechny povely se mohou psát malými i velkými písmeny
- ◆ čte v plném rozsahu formát rozšířené hustoty DOSu 2.5

**AINIT**

povel pro vytváření disket ve formátu Atari DOS 2. Slouží pro kompatibilitu, neboť SpartaDOS nelze z takovéto diskety spustit, ani jej na ni nainstalovat.

**Syntaxe:** AINIT [Dn:]

**Typ a omezení:** interní pod CP verze 2.x a vyšší

**Poznámky:**

Vytváří formát kompatibilní s DOS 2. Hustota závisí na konfiguraci jednotky, jak je běžné u implementaci DOS 2. Povel je možno vyvolat také přes XIO 254.

**Příklad:** AINIT

objeví se výpis:

FORMAT: Are you sure? Y/N

po odpovědi "Y" se disketa naformátuje ve stylu Atari DOS 2.

**BOOT**

určuje, který soubor se bude číst během boot procesu z diskety verze 2.x a vyšší.

**Syntaxe:** BOOT [Dn:][cesta>] jméno [.ext]

**Typ a omezení:** interní pod CP verze 2.x a vyšší

**Poznámky:**

Zaváděč DOSu je v prvních třech sektorech každé diskety tvaru 2.x a může zavádět a spouštět soubory stejného stylu, jako externí povedly. Normálně se zavádí DOS, ale může se zavádět cokoliv, pokud to nepřemáže paměť zaváděče (\$2E00 až \$3180). Tímto způsobem lze i nainstalovat DOS bez formátování. Na disketu formátu 2.x nakopírujeme příslušný modul DOSu a povelem BOOT jej instalujeme. Pokud se má načíst něco jiného bez DOSu, vytvoří se povel XINIT s volbou No DOS, zkopiuje se na ni program a povelom BOOT se jeho jméno oznamí zaváděči.

**Příklady:** BOOT STAR.BIN

Při zapnutí počítače se načte a ihned spustí STAR.BIN.

**Instalace RAM-disku**

Povedly instalují RAM-disk u modelů s rozšířenou pamětí. U CP verze 2.x může být instalováno až 8 disků, včetně RAM-disku. Pro instalaci je potřebné, aby byl zvolen odpovídající program pro odpovídající hardware.

**Syntaxe:** RDBASIC Dn: (XL/XE, interní Basic zapnutý)  
 RD Dn: [/N|E|NE]  
 RD260 Dn: [/N]

**Typ a omezení:** externí u všech verzí CP,  
 povol musí najít odpovídající hardware.

**Poznámky:**

RDBASIC je použitelný u verzí CP 2.x a vyšší u modelů XL/XE a vyžaduje aby byl interní Basic zapnut.

Oba další povely podporují hardwarové úpravy na 800XL a 130XE k vytvoření velkých RAM-disků. Oba povely automaticky RAM-disk formátují, pokud se nezadalo "/N". Délka sektoru je 128 byteů.

RD.COM podporuje standardní 130XE, 130XE s dalšími 64Kb navíc podle úpravy Rona Bolinga a rozšířenou paměť RAMBO XL firmy ICD pro 800XL, která dává RAM-disk 192 Kb. Parametr "/E" rezervuje první oblast 64Kb pro programy používající rozšířenou paměť 130XE, jako je třeba BASIC XE. Povel "RD D2: /E" instaluje na 800XL s RAMBO XL RAM-disk 128Kb.

RD260.COM je určen pouze pro 800XL s rozšířením na 256Kb podle časopisu BYTE (září 1985, od Clause Bucholze). Tento RAM-disk má banky 32Kb v oblasti \$0 až \$7FFF.

Parametr "/N" způsobí instalaci RAM-disku bez formátování. Tím se umožní zachovat i po studeném startu data, která byla v RAM-disku předtím. Není snad třeba říct, že jen když nebylo vypnuto napájení počítače. Proto je vhodné mít po ruce nějaké "bofítko" pro vynucení studeného startu v případě "zatuhnutí" programu. Je-li RAM-disk instalován je možno používat všechny standardní diskové povely, jakoby se jednalo o skutečnou diskovou jednotku. Lze jej konfigurovat i formátovat.

### 11.3.3. Podadresáře

Podadresáře jsou důležitým prvkem Sparta DOSu, zejména při práci s velkokapacitními disketami. Bez nich by vůbec nebylo možné používat pevný disk. Blížší vysvětlení účelu, použití a vlastností podadresářů je v kap. 9, 11.1 a 11.2.

#### ?DIR

Povely slouží k ukázání (absolutní) cesty k pracovnímu adresáři s specifikované jednotky.

**Syntaxe:** ?DIR [Dn:]

**Typ a omezení:** Interní pod CP verze 2.x a vyšší.

**Poznámky:**

Povel ukazuje absolutní cestu k pracovnímu adresáři zadané jednotky. Nezádá-li se jednotka, ukáže cestu k pracovnímu adresáři standardní jednotky. Absolutní cesta je cesta od hlavního adresáře k pracovnímu adresáři.

**CREDIR**

Zakládá podadresář na zadané jednotce v zadaném adresáři.

**Syntaxe:** CREDIR [Dn:] cesta

**Typ a omezení:** Interní u všech verzí CP.

**Poznámky:**

Podadresář, který se zakládá je poslední jméno v cestě. Cesta může být relativní (od pracovního adresáře k zakládanému), nebo absolutní. Všechna jména v cestě kromě posledního musí existovat.

**Příklady:** CREDIR D2:UTILITY

Založí na disku 2 v hlavním adresáři podadresář "UTILITY".

CREDIR GAMES>ARCADE

Založí v adresáři GAMES na standardní jednotce podadresář ARCADE. Podadresář GAMES musí existovat a být v pracovním adresáři.

**DELDIR**

zrušení prázdného podadresáře.

**Syntaxe:** DELDIR [Dn:] cesta

**Typ a omezení:** Interní u všech verzí CP

**Poznámky:**

Rušený adresář musí být prázdný, tj. při jeho výpisu se nesmí objevit žádný soubor. Ve specifikované cestě se jedná o poslední jméno. Hlavní adresář (MAIN) nelze zrušit.

**Příklad:** DELDIR GAMES>ARCADE

Zrušení podadresáře, založeného v příkladu 2 předchozího povelu. Není-li podadresář ARCADE prázdný, vypíše se chybové hlášení.

**CWD**

nastavení nebo změna pracovního adresáře na určené jednotce.

**Syntaxe:** CWD [Dn:] cesta

**Typ a omezení:** Interní pod všemi verzemi CP.

**Poznámky:**

V pracovním adresáři se hledají soubory, u kterých se neuvedla cesta. Je ta-

ké základem pro relativní cestu. Každá jednotka má svůj vlastní pracovní adresář nezávislý na ostatních jednotkách.

Pokud je soubor otevřen pro čtení, hledá se nejprve v pracovním adresáři. Pokud v něm není, hledá se v hlavním adresáři. Proto lze povelové soubory ".COM" nechat v hlavním adresáři a používat je i z podadresářů.

Při inicializaci DOSu se jako pracovní nastavuje hlavní adresář. Inicializace se provádí vždy při stisku klávesy RESET a některé programy ji dělají při svém spouštění.

Název adresáře je vypsán v hlavičce výpisu povelom DIR. Posun do nadřízeného adresáře se v popisu cesty signalizuje znakem "<".

**Příklady:** CWD <

Přechod do nadřízeného adresáře.

CWD D3:GAMES>ARCADE

Pracovním adresářem bude ARCADE, který je podadresářem adresáře GAMES na disku 3.

CWD >

Přechod do hlavního adresáře.

TREE

povel pro vypsání všech existujících adresářů na disketě, nebo v určeném adresáři. Může též vypsat všechny v nich obsažené soubory.

**Syntaxe:** TREE [Dn:] [cesta] [/F]

**Typ a omezení:** Externí u všech verzí CP.

**Poznámky:**

Povel TREE ukazuje všechny podadresáře, resp. všechny cesty, nalezené na disketě počínaje pracovním adresářem, pokud nebylo zadáním cesty v parametrech určeno jinak. Začíná-li se hlavním adresářem, ukáže všechny cesty na disketě. Začíná-li se podadresářem, ukáže všechny cesty v něm obsažené. Pokud se použije parametr "/F", vypíší se u každého podadresáře všechny v něm obsažené soubory v abecedním pořadí.

**Příklady:** TREE >

Výpis všech cest od hlavního adresáře počínaje, bez ohledu na momentální nastavení pracovního adresáře.

TREE D1:MODEM /F

Vypíše všechny podadresáře od adresáře MODEM počnaje. U každého podadresáře vypíše cestu a abecední seznam souborů v něm obsažených. V tomto případě reprezentuje údaj "Dn:" pracovní adresář.

#### **11.3.4. Kopírování**

Tato kapitola popisuje různé kopirovací možnosti SpartaDOSu se zřetelem na CP verze 2.x a vyšší. Stejný typ kopirovacího programu jako XCOPY a SCPOPY je součástí programu MENU.

##### **COPY**

je univerzální kopirovací prostředek mezi různými zařízeními. Může kopirovat jeden i více souborů a může během kopirování měnit jejich jména.

COPY může kopirovat soubory i na tutéž disketu, ale kopie musí mít buď jiné jméno, nebo se musí kopirovat do jiného adresáře. Tento povel nemůže kopirovat soubory z jedné diskety na jinou pomocí jediné jednotky, protože nemá možnost zastavit kopirovací proces mezi jednotlivými průchody a umožnit tak výměnu disket. Je-li taková kopie třeba, musí se použít XCOPY, DUPDSK nebo kopirovat pomocí MENU.

CP verze 2.x a vyšší umožňuje připojovat soubor k jinému pomocí přepínače "/A". Kopirováním lze přenášet data mezi kterýmkoliv systémovými zařízeními jako editor obrazovky, tiskárna, klávesnice apod.

##### **Syntaxe:**

**COPY D[n]:[cesta]>[jméno [.ext]]\* [D[n]:][cesta]>[jméno [.ext]]\* [/A]**  
nebo

**COPY [D[n]:][cesta]>jméno [.ext]\* [D[n]:][cesta]>[jméno [.ext]]\* [/A]**  
nebo

**COPY zař. 1 \* zař. 2**

kde                    - "\*" je povinná mezera

Všechny tři možnosti lze libovolně kombinovat tak, že se může z jednoho modelu vzít specifikace originálu, z druhého specifikace kopie.

**Typ a omezení:** Ve všech verzích CP interní,  
ale ničí uživatelskou paměť

##### **Poznámky:**

Povel COPY je jediný interní povel, který ničí obsah uživatelské paměti. Proto při práci se zásuvným modulem nebo jiným programovací jazykem je nutno před použitím povelu COPY nejprve uložit zpracovávaný program. Pro dočasné uchovávání se dobré hodí RAM-disk. První specifikovaný soubor je originál. Neudá-li se jméno, předpokládá se filtr "\*.\*". Zdrojové zařízení se v tomto případě musí udat. Pokud se však jméno udá, může se zařízení vyněchat. V tomto případě je výstupštandardní diskovou jednotkou. Druhý zejména je kopie.

Pokud se neuvede jméno, předpoklada se ".\*" a kopíruje se bez změny jména. Filtr lze používat ve všech jménech i popisných částech jmen. Pokud se filtr použije v určení cesty, kopíruje se z prvního adresáře vybraným filtrem. Jedním povelom COPY nelze kopírovat více adresářů současně. Tuto konvenci lze samozřejmě použít jen u zařízení D:

Pod CP verze 2.x a vyšší lze používat přepínač "/A" kterým se umožní připojení prvního souboru na konec druhého. Při kopírování neplatí na rozdíl od spouštění programu a externích povelů automatické vyhledávání nenalezeného souboru v hlavním adresáři.

**Příklady:** COPY D:\*.PRN P:

Kopie všech souborů s popisnou částí PRN na tiskárnu.

COPY E: D:INPUT.BAT

Vytvoření povelového souboru kopií z editoru obrazovky. Ukončení vstupu se provede kombinací CTRL-3.

COPY E: P:

Přímá kopie z obrazovky na tiskárnu.

COPY GAME2 GAME1 /A

Připojení GAME2 za GAME1.

#### XCOPY

je povel pro kopírování pomocí jedné nebo dvou jednotek mezi formáty SpartaDOSu a/nebo formáty kompatibilními s Atari DOS 2 (s určitým omezením ohledně hustoty a počtu stop).

**Syntaxe:** XCOPY

**Typ a omezení:** Externí ve všech verzích CP.

#### Poznámky:

Tento program umožňuje přenos souborů z a do Sparta DOSu pomocí jedné i dvou jednotek. Kromě toho umožňuje kopírovat soubory z jedné diskety na jinou pomocí jedné jednotky a to bez ohledu na formát a hustotu kterékoliv z nich, protože se SpartaDOS při čtení i zápisu přepíná podle formátu a hustoty momentálně vložené diskety. Povel je řízen pomocí menu, přičemž je obrazovka rozdělena do čtyř základních okének:

horní ukazuje cestu a specifikaci souborů (ev. filtr) zdrojových,  
pod ní je cesta a filtr pro cílové soubory. Protože XCOPY ne-  
přejmenovává kopírované soubory, stačí u výstupní specifika-  
ce uvést jen cestu.

pravé horní ukazuje zvolená čísla diskových jednotek pro zdrojovou a cílovou disketu

pravé dolní zobrazuje momentálně platné povelové klávesy a jejich význam spoře se zprávami a výzvami programu.

levé obsahuje zvolený adresář právě vložené diskety. Soubory pro kopírování se volí pomocí označení v tomto okénku.

Zpracovávat lze maximálně 100 souborů.

#### Příklad: XCOPY

Na obrazovce se objeví menu v podobě popsaných okének. Standardně je zvoleno kopírování s jednou jednotkou z a do hlavního adresáře. Seznam souborů získáme klávesou START se zdrojovou disketou vloženou v jednotce. Seznam se zobrazí v levém okénku a šipka ukazuje momentálně zvolený soubor. Soubory pro kopírování označíme mezerníkem, klávesa SELECT posune šipku na další soubor. Po označení souborů se spustí kopie klávesou START, přičemž je uživatel podle potřeby vyzýván k výměně zdrojové a cílové diskety. Při přenosu ze SpartaDOSu do DOS 2.5 lze u DOS 2.5 zapisovat jen v oblasti jednoduché hustoty. V opačném směru není přenos omezen. Sparta DOS 3.2 čte v plném rozsahu diskety DOSu 2.5.

#### DUPDSK

Povel slouží k duplikaci celé diskety ve formátu SpartaDOS pomocí jedné nebo dvou jednotek. Jmenovka diskety se nekopíruje. Hustota a počet stop zdrojové i cílové diskety musí být stejná, jinak dojde k chybě.

#### Syntaxe: DUPDSK

Typ a omezení: externí ve všech verzích CP

#### Poznámky:

DUPDSK je kopírovací program, který kopíruje celou disketu ve formátu SpartaDOSu pomocí jedné nebo dvou jednotek. Tento povel neformátuje cílovou disketu a nekopíruje jmenovku z originálu. Z toho vyplývá, že cílová disketa musí být nejprve připravena povelom XINIT. Všechny soubory na cílové disketě se přepíšou. Cílová disketa je tedy až na jmenovku přesnou kopí originálu. Proto musí mít stejnou hustotu záznamu a stejný počet stop.

#### Příklad: DUPDSK

Povel se dotazuje na číslo zdrojové a cílové jednotky. Platná čísla jsou 1 až 8. Potom následují výzvy ke vkládání zdrojové či cílové diskety, nebo obou najednou pokud se jedná o kopii pomocí dvou jednotek. Po každé výzvě lze povel zrušit klávesou ESC. Jakákoliv jiná klávesa včetně BREAK znamená pokračování v činnosti.

**SCOPY**

je povel pro kopírování disket libovolného formátu po sektorech a pro uložení celé diskety ve formě souboru, nebo pro expanzi takto uloženého souboru na celou disketu.

**Syntaxe:**

**SCOPY Dn: [ [cesta>]jméno[.ext]] [/UR] Dn: [ [cesta>]jméno[.ext]] [/UR]**

kde:

- "U" aktivuje formát pro US doubler
- "R" znamená RAM-disk.

**Typ a omezení:** ve všech verzích CP externí.

**Poznámky:**

SCOPY je sektorový kopírák, který má tři operační mody. Může kopírovat disk na disk (jako DUPDSK, pouze s tím rozdílem že formátuje cílovou disketu přesně podle originálu a kopíruje vše včetně jmenovky. Lze jím kopírovat cokoliv.) Další dva operační mody umožňují buď komprese celé diskety do souboru na cílové disketě, nebo expanzi dříve komprimovaného souboru na cílovou disketu. Volba operačního modu je určena syntaxí povelu. První parametr vždy udává originál, druhý kopii. Tam, kde je uvedena jen specifikace jednotky jde o celý disk. Je-li navíc specifikován soubor, jde o komprezi, nebo expanzi. Soubor nelze specifikovat v obou parametrech najednou. Syntaxi lze pro jednotlivé případy rozložit takto:

Kopie disk -> disk      SCOPY Dn: [/UR] Dn: [/UR]

při kopii disk -> disk se zkopiují všechny sektory z jedné diskety na druhou. Před spuštěním kopie se program zastaví s výzvou ke vložení zdrojové (a cílové diskety je-li více jednotek). Příkopií na jedné jednotce vydává výzvy kvýměně disket. Pokud se v některém případě jedná o disketu s organizací sektorů pravý modus US doubleru (viz 2.3), musí se uvést přepínač U. Je-li jednouz jednotek RAM-disk, musí se uvést přepínač R. Použití RAM-disku přichází v úvahu jen u přístrojů s rozšířenou pamětí na 256 kB a více, protože by se jinak disketa do RAM-disku nevešla. Zde je třeba poznamenat, že jediný RAM-disk se kterým SCOPY funguje je instalován povelem RD. RD vytvoří skutečný model reálné disketové jednotky včetně možnosti formátování, konfigurace a podobně. Použití RAM-disku však nemá praktický význam, pro vícenásobné kopírování je vhodnější použít kombinaci dalších modů a příkopírování jednorázovém je mezi stupeň s RAM-diskem zbytečný. Přepínač U je u RAM-disku neúčinný.

Kopie disk -> soubor      SCOPY Dn: [cesta>]jméno[.ext] [/U ]

při kopii disk -> soubor se celý disk uloží na cílové disketě jako soubor určeného jména. V souboru jsou uloženy všechny potřebné údaje o formátu diskety. Operace se spustí bez přerušení, takže ji lze použít jen je-li více jednotek (včetně RAM-disku). Při komprezi disku do souboru se neprovádí

zhuštování dat, takže pro archivní účely se tento modus příliš nehodí. Snad jen proto, aby se dalo pracovat s disketou jako s jedním celkem, nebo ji pomocí dodatečně definovaného handlu archivovat na kazetu v systému Turbo. Hodí se pro vytváření vícenásobných kopií disket například pro distribuci.

**Kopie soubor -> disk** SCOPY Dn: [cesta>]jméno[ext] [/U ] Dn: [/UR]  
expanze dříve komprimované diskety ze souboru na celý disk.

SCOPY optimalizuje kopírovací proces tak, aby byl co nejrychlejší. Pracuje s disketami v jednoduché, rozšířené i dvojitě hustotě. Není určen pro speciality jako jsou osmipalcové, nebo oboustranné diskety.

#### Příklady:

Jako příklad uvedeme dvě spolupracující sekvence pro vícenásobné kopírování disket. První sekvence má název KOPIE .BAT a obsahuje povely:

```
RD D2:  
COPY SCOPY.COM D2:  
COPY ROZMNOZ.BAT D2:  
D2:  
, Vložte original. (RETURN)  
PAUSE  
SCOPY D: D2:DISK  
-ROZMNOZ
```

Druhá sekvence ROZMNOZ.BAT obsahuje tyto povely:

```
, Vložte kopii. (RETURN)  
PAUSE  
SCOPY D2:DISK D:  
-ROZMNOZ
```

Celý koloběh se spustí povellem "-KOPIE". Povel PAUSE umožňuje výměnu disket, protože SCOPY v tomto režimu běží bez zastávky.

#### 11.3.5. Údržba souborů a disket

Pod pojmem údržba souborů se skrývá přejmenovávání a mazání souborů a kontrola a změny jmenovek disket.

##### **ERASE**

slouží k rušení souborů. Není-li udána cesta, pracuje se s pracovním adresářem. Je možno rušit jeden soubor, nebo celou skupinu vybranou filtrem.

**Syntaxe:** ERASE [Dn:] [cesta>] jméno[ext]

**Typ a omezení:** Interní ve všech verzích CP.

##### **Poznámky:**

Místo jména lze použít filtr, ale při rušení skupin souborů je třeba opatrnos-

sti, protože rušení se již dále nepotvrzuje. Při filtrovi ".\*" se bez dalšího varování zruší celý obsah příslušného adresáře. Povelom ERASE lze rušit jen soubory, pod adresáře zůstávají bez změny. Obnovení zrušených souborů je možné povelom UNERASE. Není závadou, jestliže se po zkopírování a následném vymazání souboru změní počet volných sektorů o 1. Adresář a sektorové mapy nejsou vázány na konkrétní sektory a mohou se zvětšovat a zmenšovat podle potřeby. Nemusí být vždy ve velikosti a na místě předchozího umístění.

#### **UNERASE**

slouží k obnovení souborů bezprostředně po zrušení. Není-li uvedena cesta, obnovují se soubory v pracovním adresáři. Místo jména lze použít filtr. Pokud nelze soubor obnovit, přešle UNERASE zprávu o příčině.

**Syntaxe:** UNERASE [Dn:][cesta>] jméno [ext]

**Typ a omezení:** externí ve všech verzích CP.

**Pozor:** Používejte jen UNERASE.COM z roku 1985 nebo novější!

#### **Poznámky:**

Povel obnoví náhodně zrušené soubory ale jen v případě, jsou-li neporušeny. Je-li na disketu po zrušení souboru zapsán nějaký jiný soubor, může přepsat část nebo celý zrušený soubor. Ten je pak navždy ztracen.

Nepoužívejte UNERASE.COM z roku 1984! Tato verze zcela zničí obsah diskety formátované pod CP verze 2.x a 3.x!

**Příklad:** UNERASE \*\*

UNERASE prohlédne adresář a vypisuje jména obnovovaných souborů. Počet volných sektorů se po zrušení a obnovení souboru může lišit. Viz poznámku u ERASE.

#### **RENAME**

umožňuje měnit jména souborů

**Syntaxe:** RENAME [Dn:][cesta>]jméno [ext] jméno [ext]

**Typ a omezení:** Interní ve všech verzích CP.

#### **Poznámky:**

V obou specifikacích souboru lze použít filtr. Pravidla jsou stejná jako u DOS 2.5.

**Příklad:** RENAME FILEZ FILES

**CHVOL**

slouží ke změně jmenovky diskety.

**Syntaxe:** CHVOL [Dn:]název

**Typ a omezení:** externí ve všech verzích CP.

**Poznámky:**

Pomocí CHVOL lze měnit jmenovku diskety po formátování. Jméno může mít až 8 znaků a nesmí obsahovat mezery.

**11.3.6. Ochrana**

V této kapitole jsou povely pro ochranu jak jednotlivých souborů, tak celých disket proti přepsání.

**PROTECT**

slouží k zablokování souboru proti zápisu.

**Syntaxe:** PROTECT [Dn:][cesta>]jméno [.ext]

**Typ a omezení:** Interní od CP verze 2.x

CP verze 1.x nerozpoznává blokovaný stav.

**Poznámky:**

Protect zabrání náhodnému vymazání nebo přepsání souboru. Pokus o takovou akci končí hlášením chyby číslo 164. Na rozdíl od DOS 2.5 je povolen blokované soubory přejmenovávat. Blokovaný stav se indikuje při krátkém formátu výpisu adresáře povellem DIRS. Povel DIR blokovaný stav neukáže.

**UNPROTECT**

opačný pochod k PROTECT. Uvolňuje blokované soubory.

**Syntaxe:** UNPROTECT [Dn:][cesta>]jméno [.ext]

**Typ a omezení:** Interní v CP 2.x a novější.

CP 1.x nerozpoznává blokovaný status.

**Poznámky:**

Reverzní povel k PROTECT. Soubor musí být pro modifikaci nebo vymazání uvolněn.

**Ochrana disket**

SpartaDOS verze 2.x a vyšší má povely k zablokování nebo uvolnění celé diskety. Je to pochod podobný jako nalepení ochranné nálepky, ale jde o ryzí softwarovou ochranu, která nemůže zabránit smazání formátováním a funguje jen při používání SpartaDOSu. Ochrana nefunguje u povelu INIT, XINIT, AINIT a FORMAT.

## **LOCK**

blokuje proti zápisu celou disketu. Ochrana funguje jen u SpartaDOSu 2.x a vyššího.

**Syntaxe:**      **LOCK [Dn:]**

**Typ a omezení:** Interní pod CP verze 2.x a vyšší.  
CP 1.x nerozpoznává chráněný status.

### **Poznámky:**

Povel LOCK zapisuje na disketu ochranný byte, který může být změněn jen povelom UNLOCK. Status ochrany celé diskety lze zjistit povelom CHDKSK. Pokus o zápis na blokovanou disketu způsobí při práci v CP výpis "Disk write locked", v ostatních programech vede k chybě číslo 169.

## **UNLOCK**

uvolňuje softwarovou ochranu diskety verze 2.x (viz LOCK).

**Syntaxe:**      **UNLOCK [Dn:]**

**Typ a omezení:** Interní pod CP verze 2.x.  
CP verze 1.x nerozpoznává chráněný status

### **Poznámky:**

Povel přepisuje byte ochrany diskety a povouje na ni zápis. Povel CHDKSK ukazuje po provedení UNLOCK status "Write lock: OFF".

## **11.3.7. Operace pomocí menu**

Pro uživatele, kteří mají ráději volbu pomocí menu, než řízení povelovou řádkou má SpartaDOS povel MENU. Menu SpartaDOSu se liší od menu DOS 2.5 a pro vysvětlování funkce je nejlepší si jej pustit na obrazovku a vidět jak vypadá. Usnadní to pochopení jeho funkce podle popisu v následujícím textu.

## **MENU**

Zprostředkovává většinu povelů CP ve formě menu. Je přizpůsobeno pro operace s jedním i více soubory.

**Syntaxe:**      **MENU [R][n]**

**Typ a omezení:** Externí pod CP verze 2.x

### **Poznámky:**

Přepínač "R" způsobí, že menu se stane rezidentní, to jest zůstává trvale v paměti a o jeho délku se zvýší hodnota MEMLO. Pokud přejdeme z menu do modulu a vracíme se zpět povelom DOS, objeví se menu jen bylo-li vyvoláno s přepínačem R. Jinak sa dostaneme do CP a musíme si menu spustit znova.

Pro menu je k dispozici pomocný soubor MENU.HLP, který obsahuje vysvětlivky k jednotlivým funkcím. Parametr "n" určuje číslo jednotky, na které se tento soubor nachází. Máme tak možnost jej například umístit v RAM-disku a zrychlit tak práci s ním. Pokud se "n" nezadá, předpokládá se 1. Mezi parametry "R" a "N" se nedává mezera, to jest rezidentní menu s pomocným souborem na disku 2 zvolíme jako "MENU R2".

#### Činnost menu:

Mnoho funkcí menu může pracovat s větším počtem souborů (copy, erase, protect atd.). Soubory pro práci se musí nejprve vybrat a označit, protože funkce se provedejen pro označené soubory. Přiznačování souborů se používají následující klávesy:

- šipka nahoru - posunuje výběrový kurzor o jeden soubor vzhůru.  
Je-li kurzor na prvním souboru, přesune se na poslední.
- šipka dolů - Přesune kurzor na následující soubor.  
Je-li kurzor na posledním souboru, přesune se na první.
- mezera - Přepíná status souboru.  
Soubor vybraný pro operaci je zobrazen inverzně.

Dalším krokem je výběr povelu nebo funkce, která se má provést. Provede se to přesunutím povelového kurzu na příslušné okénko. Povel jsou uspořádány v pěti blocích po pěti povelech. Výběr se provádí těmito klávesami:

- OPTION - Přepnutí na následující blok povelů.  
Kurzor zůstává ve stejné pozici. Celkem je 5 bloků.
- SELECT - Posunuje kurzor na následující povel v bloku.
- šipka doprava - totéž co SELECT
- šipka doleva - Posunuje kurzor na povel vlevo.
- 1 .. 5 - Volí blok povelů.  
Jde o alternativu ke klávesě OPTION, která umožňuje rychlejší dosažení požadovaného bloku.
- A .. Z - Písmeno přesunuje kurzor na požadovaný povel.  
Tato možnost umožňuje po zapamatování si spojení písmen a povelu podstatně zrychlit ovládání menu. Pokud to bylo možné odpovídá písmenu začátečnímu písmenu povelu.
- HELP - Vypisuje v angličtině krátký popis povelu, na kterém stojí kurzor. Po přečtení popisu se vrátíme k práci klávesou (RETURN).

Jakmile máme kurzor nastaven na zvolenou funkci, spustíme ji klávesou (RETURN) nebo START. Obě jsou funkčně identické. Následuje popis jednotlivých funkcí. V závorce je uvedeno písmeno, kterým se dotyčná funkce volá.

?Files (F)	- povel načte adresář zadané jednotky, který se tak stane zdrojem souborů pro operace menu. Na dotaz "Which Drive?" odpovězte číslem jednotky (1 až 4) nebo (RETURN), volici 1.
Copy (C)	- povel kopíruje označené soubory, nebo soubor pod kurzorem, pokud nebyly žádné soubory označeny. Na dotaz "Dest Drive?" odpovíme číslem jednotky nebo (RETURN) pro jednotku 1. Dále zadáme cestu pro určení cílového adresáře, nebo (RETURN) pokud se kopíruje do hlavního. Při výzvách na výměnu disket se po provedení požadované akce stiskne (RETURN).
Erase (E)	- zruší všechny označené soubory. Další potvrzování se již nedělá, proto je třeba opatrnosti.
Rename (R)	- přejmenuje soubor pod kurzorem na zadané jméno. Při dotažu "Rename to?" napište nové jméno a (RETURN).
Exit (Q)	- opustí menu a aktivuje CP. Je-li aktivní zásuvný modul je třeba opatrnosti. Čtěte odstavec "Další poznámky".
RunCar (B)	- je-li aktivní zásuvný modul, přejde do něj. Čtěte odstavec "Další poznámky".
Load (L)	- načítá binární segmentovaný soubor, na kterém je právě kurzor. Před zavedením se vymaže obrazovka. Při zavádění se používají standardní vektory RUN a INIT (viz 3.5, 4.1.11). Po skončení programu se do menu dostaneme klávesou (RETURN), pokud ovšem zavedený program menu neporušil.
Save (S)	- ukládá binární soubor (viz 4.1.11). Je třeba zadat jméno a cestu.
Run (J)	- skočí na zadanou adresu. Pokud se žádná nezadá, skočí se na adresu definovanou posledním LOAD.
Exec/P (G)	- zavádí soubor pod kurzorem, ale před spuštěním umožňuje předat mu parametry, jako kdyby se spouštěl z povelové řádky. Je to způsob jak spouštět externí povely, které nejsou zahrnuty do menu. Před spuštěním se obrazovka vymaže a po vykonání programu se můžeme klávesou (RETURN) vrátit do menu.
Xinit (I)	- načte a spustí externí povel XINIT (viz 11. 3.2.). Po inicializaci diskety se do menu vrátíme klávesou ESC.
Ainit (A)	- formátuje disketu ve stylu Atari DOS 2. Zadává se číslo jednotky ((RETURN)-1) a po vložení se pomocí (RETURN) spustí formátování.
?Mem (M)	- vypíše obsah MEMLO a MEMHI. (RETURN) obnoví menu.
ChkDsk(Z)	- provádí povel CHKDSK. (RETURN) obnovuje menu
Help (H)	- vypisuje informaci o klávesách pro pohyb kurzoru a způsobu výběru. (RETURN) obnoví menu.
Prot (P)	- blokuje proti zápisu všechny označené soubory, nebo soubor pod kurzorem pokud nebyl žádný soubor označen.
UnProt (U)	- uvolňuje všechny označené soubory, nebo soubor pod kurzorem pokud nebyl žádný soubor označen.

- Lock (K) - blokuje proti zápisu celou disketu v standardní jednotce.
- UnLock (L) - uvolňuje disketu v standardní jednotce.
- Xfer (X) - velmi podobný povelu COPY, ale nepracuje s více soubory najednou. Dotazuje se na zdrojový a cílový soubor. Je třeba zadávat i jména zařízení. Před kopirováním se vymaže obrazovka a po něm se klávesou (RETURN) obnoví menu. Povel lze použít například pro kopii na obrazovku apod.
- ?Dir (V) - ukazuje momentální nastavení pracovního adresáře.
- >Dir (T) - zobrazí obsah adresáře pod kurzorem a nastaví jej jako pracovní adresář.
- <Dir (Y) - přejde do nadřízeného adresáře a zobrazí jeho obsah. Zobrazený adresář se stává pracovním.
- CreDir (N) - vytvoří nový podadresář, požaduje zadání jména.
- DelDir (D) - zruší adresář pod kurzorem, pokud neobsahuje žádné soubory.

#### Další poznámky k programu MENU

Některé povely zneplatňují uživatelskou paměť. Jsou to Copy, Xinit, Load a Exec/P.

Pokud je aktivní zásuvný modul je třeba zajistit, aby CP nekolidoval s menu. Přejdeme-li do Basicu, informuje nás WARMFLG, zda je uživatelská paměť platná. Jak CP, tak i MENU má vlastní kopii tohoto semaforu. Obě kopie se mohou lišit. Pokud se používá významně jen CP nebo jen MENU, problém nenastává, ale některým věcem je třeba zamezit pokud se používají oba. Jde o tyto případy, které budeme demonstrovat na příkladech:

1. Spustíme menu ("MENU R"), přejdeme do Basicu funkcí RunCar. Napišeme Basicovský program nebo nějaký zavedeme. Poté povelom DOS přejdeme do menu a provedeme povel ničící uživatelskou paměť (Copy). Menu opustíme ("Exit") a z CP přejdeme povelom CAR zpět do Basicu. V tomto případě CP neví, že byla uživatelská paměť poškozena.
2. Spustíme menu ("MENU R"), přejdeme do Basicu funkcí RunCar. Napišeme Basicovský program nebo nějaký zavedeme. Poté povelom DOS přejdeme do menu a opustíme jej ("Exit") a v CP provedeme povel poškozující uživatelskou paměť. Pak stiskneme RESET. V tomto případě MENU neví, že byla uživatelská paměť poškozena, a RESET použije jeho kopii WARMFLG.

**Důležité:** Při přechodu do modulu pomocí povelu se do WARMFLG dosadí hodnota z kopie naležející programu, z něhož jsme do Basicu přešli. Pokud se do modulu přechází klávesou RESET, používá se kopie patřící MENU.

### 11.3.8. Datum a čas

Jak je již vidět v výpisu adresáře povelenem DIR, ukládá SpartaDOS ke všem souborům datum a čas jejich vzniku. Není třeba zdůrazňovat, že to není samoučelná paráda, ale dobrý pomocník při rozmořitavání gordického uzlu různých verzí téhož programu vzniklých při zbesilém ladění a zkoušení. Pro práci s datem a časem je k dispozici několik povelů, které umožňují tyto údaje vypisovat a měnit. SpartaDOS má pro momentální datum a čas vyhrazeno v paměti místo, které po zavedení SpartaDOSu obsahuje standardní hodnotu. Pro SpartaDOS verze 2.x je to čas 3:59:00pm (odpoledne) a datum 1-1-84 (americká forma měsíc-den-rok). SpartaDOS verze 3.2 má datum 10-07-85 a čas 3:45:00pm. Pokud toto standardní datum a čas nezměníme, počítá se vše od něj. Zdrojem času a data mohou být buď hardwarové baterii napájené hodiny v zásuvném modulu R-time 8, které dodává též firma ICD, nebo softwarové hodiny odvozené od čítače vertikálních přerušení. Tyto softwarové hodiny jsou však vytvořeny pro americké systémy s frekvencí 60Hz, takže při používání na evropských verzích počítačů se dosti podstatně zpoždjují. Pro informaci o přesném času proto nemají velký význam, ale orientace o posloupnosti vzniku souborů na platnosti neztrácí. Nevhodou softwarových hodin je nutnost nastavení po každém studeném startu. Podpora času a data se u verzí 2.x a 3.x liší. Protože není praktický důvod používat starší verzi 2.x, budeme popisovat povely se zřetelem na poslední verzi 3.2.

#### **TIME**

zobrazí současný čas a umožňuje jeho změnu

**Syntaxe:**      **TIME**

**Typ a omezení:** Interní pod CP verze 3.2.

(Zcela odlišný od staršího externího povelu TIME)

#### **Poznámky:**

Povel vypisuje:      Current time is 12:58:18pm  
                          Enter new time: \_\_\_\_\_

Nový čas zadáváme ve formě hh:mm:ss [{a|p}], kde a znamená am čili dopoledne, p je pm čili odpoledne. Určení dopoledne či odpoledne lze vyněchat, pokud zadáme hodiny ve čtyřadvacetihodinovém cyklu, nebo pokud se jedná o dopoledne. SpartaDOS uchovává čas jedině ve dvanáctihodinovém rozsahu s určením dopoledne/odpoledne. Čas zadaný jako např. 14:05:20 převede na 2:05:20pm. Pokud jsme s vyspanou honotou spokojeni, stiskneme jen (RETURN).

**DATE**

zobrazí momentální datum a umožnuje jeho změnu.

**Syntaxe:**      **DATE**

**Typ a omezení:** Interní pod CP verze 3.2.

**Poznámky:**

Povel vytváří následující výpis:

Current date is 10-23-88

Enter new date: \_\_\_\_\_

Nyní lze zadat nové datum, nebo jen (RETURN) pokud se měnit nemá. Formát data je mm-dd-rr, kde mm je měsíc, dd den a rr rok. Jde o americkou formu data.

**TD**

povel umožňuje zapnout a vypnout řádku s informací o datu a času.

**Syntaxe:**      **TD {ON|OFF}**

**Typ a omezení:** Interní pod CP verze 3.2.

(Zcela odlišný od staršího externího povedu TD).

**Poznámky:**

Řádka s datem a časem není SpartaDOSem interně podporována. Pokud ji chceme na obrazovce mít, musíme ji nejprve aktivovat povelom TDLINE, který instaluje příslušný handler. Podrobnosti jsou vysvětleny dále.

**TDLINE**

instaluje handler pro řádku s datem a časem. Tento handler je fízen interním povelom TD.

**Syntaxe:**      **TDLINE**

**Typ a omezení:** Externí pod CP verze 3.2.

S dřívějšími verzemi nelze použít.

**Poznámky:**

Instaluje do systému handler, který je zpřístupňován pomocí vektorů data a času SpartaDOSu. Tento handler souvisí s funkcemi TDON a FMTTD. TDON způsobuje, že se handler zařadí do VBI rutin. Během přerušení zpětným během paprsku obrazovky volá FMTTD přes SpartaDOS (vraci formátovaný čas a datum), upraví je do kódu obrazovky a zobrazí do přidané horní řádky. Některé programy používající VBI k vlastním účelům kolidují s časovou řádkou a je třeba ji zakázat povelom TD OFF.

**RTIME8**

instaluje rutiny pro používání hardwarových hodin R-TIME8 v zásuvném modulu. Obsahuje handler zařízení "Z:" a funkci TDLINE.

**ZHAND**

instaluje handler zařízení "Z:" pro zpřístupnění data a času z uživatelských programů nebo Basicu. Pracuje jak s R-TIME8, tak se softwarovými hodinami.

**Syntaxe:** ZHAND

**Typ a omezení:** Externí pod CP verze 3.2.  
pod staršími verzemi nelze použít.

**Poznámky:**

Instaluje interfejs pro zpřístupnění funkcí data a času z vyšších programovacích jazyků. Konvertuje požadavky pomocí CIO do volání vektorů času a data SpartaDOSu a naopak. Popis handlu Z je uveden dále.

**CHTD**

mění datum a čas vytvoření souboru v adresáři.

**Syntaxe:** CHTD [Dn:][cesta>]jméno [.ext]

**Typ a omezení:** Externí pod CP verze 1.x, 2.x i 3.x.

**Poznámky:**

CHTD se používá ke změně nebo opravě data a času v hlavičce souboru. Může to být užitečné u souborů zkopiovaných z jiných DOSů, nebo když zapomeneme nastavit datum a čas po restartování SpartaDOSu. Povel zapíše do hlavičky momentální datum a čas v okamžiku vyvolání. Filtr lze použít pro změnu u skupiny souborů.

**Příklad:** CHTD D:/\*.\*

zapíše současný čas a datum do hlaviček všech souborů v pracovním adresáři na jednotce 1.

**11.3.9. Podpora komunikace**

SpartaDOS má podporu pro dva typy interfejsů RS232, a sice pro Atari 850 a pro ATR8000. Protože tyto interfejsy jsou u nás nedostupné a komunikace po telefonních linkách je zatím v plenkách, pojednáme o této problematice jen krátce.

**RS232**

instaluje handler pro sériovou komunikaci.

**Syntaxe:** ! RS232 nebo AT\_RS232

**Typ a omezení:** Externí pod všechny verzemi CP.

**Poznámky:**

Oba povely způsobí načtení handleru zařízení R: z interfejsu a jeho instalaci do systému. ATR8000 je inteligentní zařízení, které umí automaticky zapínat a vypínat konkurenční modus práce a proto je možno pomocí něj ovládat počítač dálkově z jiného zařízení.

Konkurenční modus je nutný při současném příjmu a vysílání, ale znemožňuje provádění jiných I/O operací přes rutiny SIO. Proto je nutno jej vypínat například kvůli přístupu na disk.

**PORt**

Konfiguruje sériový port.

**Syntaxe:** PORT [Dn:][cesta>]jméno[.ext]

**Typ a omezení:** Externí pod všechny verzemi.

**Poznámky:**

PORt posílá dvoubytový konfigurační soubor do zařízení R; a určuje tak přenosovou rychlosť a další parametry.

Byte 1:

110 baud	5	8 bitové slovo	0
300 baud	0	7 ..	16
1200 baud	10	6 ..	32
1800 baud	11	5 ..	48
2400 baud	12		
4800 baud	13	1 stop bit	0
9600 baud	14	2 stop bity	128

Byte 2:

překlad: na konci řádky:

std. ASCII	0	nepřidávat LF	0
ext. ASCII	16	přidávat LF	64
žádný (ATASCII)	32		

parita vstupní: parita výstupní:

žádná	0	žádná	0
lichá	4	lichá	1
sudá	8	sudá	2
ignoruj	12	mark	3

Hodnota bytů se vypočte součtem hodnot odpovídajících zvolené konfiguraci. Na systémové disketě SpartaDOSu verze 3.2 je jako příklad konfigurační soubor P\_4800.RC, který nastavuje rychlosť 4800 baudů, žádnou paritu, 8 datových a 1 stop bit, po CR přidává LF.

### **11.3.10. Přesměrování vstupu a výstupu (dávkový režim)**

#### **Přesměrování vstupu**

je schopnost systému řídit počítač povely z nějakého zařízení tak, jakoby je uživatel psal na klávesnici. To znamená, že určitý soubor může "psát" povely tak, jak je normálně psáme na klávesnici. Na rozdíl od ostatních DOSů, které umožňují dávkové zpracování přesměrovává SpartaDOS všechn vstup, ne jen povely pro CP. Tím je možno vkládat do dávkových souborů i povely pro zásuvný modul a podobně. SpartaDOS má možnost dávkové práce a přesměrování vstupu propracovanou ze všech DOSů pro počítače Atari nejlépe.

#### **Dávkové soubory**

slouží k provedení určité dávky povelů, která je v nich obsažena. Standardní jméno má koncovku ".BAT". Zvláštním případem je STARTUP.BAT, který se automaticky spustí po nastartování SpartaDOSu.

**Syntaxe:** -jméno[.ext]

**Typ a omezení:** Interní pod CP verze 1.x až 3.x

#### **Poznámky:**

Dávkové soubory obsahují proveditelné povely CP eventuelně modulu. Jde o soubory textové, které je možno vytvořit buď kopíí z obrazovky (viz 11.3.4. příklady povelu COPY), nebo libovolným textovým procesorem. Obsah dávkového souboru lze vypsat povelem TYPE. Standardní koncovka pro dávkové soubory je ".BAT". Pokud má dávkový soubor jinou koncovku, je třeba ji uvést. V souboru mohou být i komentáře na samostatné řádky začínající středníkem. Maximální délka řádky je 64 znaků. Zpracování dávky lze přerušit klávesou RESET. Výhodné je použít sekvence STARTUP.BAT pomocí níž můžeme vždy při startu provést základní inicializaci systému podle potřeby. Například můžeme chtít vždy po startu zadat datum a čas a vymazat obrazovku a inicializovat informační řádku s datem a časem:

```
TIME
DATE
,(ESC) (CTRL+CLEAR)
TDLINE
```

V dávkových souborech se mohou vyskytovat speciální povely, určené jen pro ně. Pro čtení dávkových souborů se používá kanál IOCB 5, který je však "znásilněn" tak, aby se pro ostatní programy jevil jako zavřený. Je to z toho důvodu, že například Basic po svém spuštění zavírá všechny kanály. Proto

se kanál 5 nesmí otevírat během vykonávání dávkového souboru, jinak mohou nastat nepředvídatelné následky. Některé povely jako SPCOPY, FORMAT, INIT a další reinitializují DOS, což může způsobit ukončení dávkové práce. Spojování dávkových souborů je možné od verze 2.x vždy na konci sekvence.

### **PAUSE**

Dočasné zastaví provádění dávkového souboru. Umožňuje například výměnu disket a podobně. Po stisknutí klávesy dávka pokračuje.

**Syntaxe:** PAUSE

**Typ a omezení:** Interní pod CP verze 2.x a 3.x.  
Externí pod CP verze 1.x

### **Poznámky:**

Během povelu PAUSE se nesmí měnit disk, ze kterého jede dávkový soubor, jinak by mohlo nastat přepsání informací na vložené disketě. PAUSE lze zrušit klávesou RESET.

**Příklady:** Předpokládejme následující sekvenci spouštěnou z disku 1. je určena pro model s rozšířenou pamětí:

```
RD D2:  
COPY COP.BAT D2:  
D2:  
-COP
```

Sekvence COP.BAT obsahuje toto:

```
, Vložte disketu s programy Basicu  
PAUSE  
COPY D:*.BAS  
BASIC ON  
CAR  
RUN "D2:MENU.BAS"
```

První sekvence nainstaluje RAM-disk a kopíruje do něj druhou sekvenci. Poté přepne standardní jednotku na D2 a spustí z ní druhou sekvenci. Změna jednotky umožňuje vyměnit disketu v jednotce 1 během povelu PAUSE. Druhá sekvence nakopíruje do RAM-disku programy v Basicu, přejde do modulu a v něm načte a rozbehne program MENU.

**TYPE**

vypisuje obsah ASCII souboru. Obvykle se užívá pro prohlídku dávkového souboru.

**Syntaxe:**     **TYPE [Dn:] [cesta>] jméno [.ext]**

**Typ a omezení:** Interní pod CP verze 1.x a dále.

**Poznámky:**

Soubor se čte řádek po řádku a vypisuje na obrazovku. Je-li délka řádky větší, než 64 znaků, dojde k chybě (zkrácená věta). Povel vypisuje jen jeden soubor. Téhož efektu lze dosáhnout povelom COPY, který však poškozuje uživatelskou paměť. Pomocí TYPE ve spojení s PRINT lze otisknout soubor na tiskárnu.

**Příklad:**     **TYPE STARTUP.BAT**

vypíše obsah inicializační sekvence.

**Přesměrování výstupu**

je schopnost opakovat všechno, co bylo napsáno na obrazovku na jiné výstupní zařízení, například do diskového souboru. Tím je možno mít zaprotokolován veškerý výstup na obrazovku. Jedinou výjimkou jsou programy, které piší přímo do obrazové paměti jako MENU.COM, nebo většina her.

**PRINT**

Specifikuje zařízení, na které se bude od této chvíle kopírovat veškerý výstup na obrazovku (na E: přes kanál 0)

**Syntaxe:**     **PRINT [Dn:] [cesta>] jméno [.ext] nebo**  
                 **PRINT zař.[n]:**    **nebo**  
                 **PRINT**

**Typ a omezení:** Interní pod CP verze 1.x a novější.

Volba "/A" je povolena od verze 2.x.

**Poznámky:**

Normálně se tento povel používá ke kopii obrazovkového výstupu na tiskárnu, třeba při tisku adresáře, ale stejně dobře lze archivovat výpisy do diskového souboru nebo jinam. PRINT s parametrem aktivuje výstup na zadané zařízení, bez parametrů výstup vypne.

Pokud je povel v činnosti a kopie výstupu se ukládá, je použit kanál 4. Proto jej v této době nesmí žádný program použít ani otevřít. Ze stejného důvodu jako u přesměrování vstupu je tento kanál upraven tak, aby se jevil jako zavřený, i když je v činnosti.

Příklady: PRINT P:

opis výstupu na tiskárnu

PRINT D:PROTOKOL

uložení v diskovém souboru.

PRINT

uzavře výstupní soubor a ukončí kopírování obrazovky.

#### Princip práce přesměrovaných I/O operací.

Přesměrování je umožněno existencí tabulky handlerů HATABS, ve které jsou písmenové názvy zařízení a ukazatele na tabulku adres rutin příslušného handleru. SpartaDOS si tuto tabulkou uschová ve své paměťové oblasti a do HATABS vnutí svou vlastní adresu. Tak může kontrolovat veškeré vstupní a výstupní operace, zda se jedná o kanál 0. Potom může při požadavku na čtení kanálu 0 "přistrčit" data z jiného kanálu, nebo je naopak poslat do jiného kanálu navíc.

#### Zákaz přesměrování

V některých případech, zejména u her spouštěných ze SpartaDOSu může technika přefložení tabulky zařízení způsobit zhroucení systému. V tomto případě je před spuštěním dotyčného programu nutno přesměrování zakázat. V CP verze 2.x a 3.x je k tomuto účelu povel XDIV.

#### XDIV

slouží k zákazu dávkové práce, povelu PRINT a přesměrování vstupu a výstupu. To je u některých programů nutné ke správné funkci.

Syntaxe: XDIV

Typ a omezení: Interní pod CP verze 2.x a 3.x

#### Poznámky:

XDIV vrátí tabulku zařízení do původní podoby a zakáže přesměrování. Tím se umožní funkce programům, které nesnáší ředirekci I/O operací. Zákaz je trvalý a platí až do nového restartování SpartaDOSu. U CP verze 1.x se k témuž účelu používal povel DIS\_BAT.

#### 11.3.11. Klávesový buffer

Klávesový buffer je prostředek ke zpříjemnění práce prorychlopísáče. Normálně si Atari pamatuje jen jeden znak z klávesnice a pokud je počítač zrovna zaměstnán třeba čtením diskety, všechno co jsme napsali v této době je až na 1 znak ztraceno a musíme psát znova. SpartaDOS instaluje automaticky klávesový buffer o velikosti 32 znaků. Kromě toho zrychluje autorepeat klávesnice. Díky bufferu lze napsat "do zásoby" až 32 znaků. Buffer funguje

pro celý systém i v Basicu a podobně. Některé programy však tento buffer nesnáší a před jejich spuštěním je nutno buffer zakázat. Typickým příkladem je ladící program DDT, který je součástí modulu MAC/65.

#### **KEY**

povoluje nebo zakazuje klávesový buffer.

**Syntaxe:** KEY {ON|OFF}

**Typ a omezení:** Interní pod CP verze 3.x

#### **11.3.12. Informační povely**

Zprostředkovávají různé informace o systému a diskových jednotkách.

##### **MEMLO a MEM**

vypisují hodnoty hranic volné paměti pro uživatele.

**Syntaxe:** MEM nebo  
MEMLO

**Typ a omezení:** MEM: Interní pod CP verze 2.x a vyšší.

MEMLO: Externí. Vypisuje jen dolní hranici.

##### **Poznámky:**

Povel MEM vypisuje v hexadecimální formě dolní a horní hranici uživatelské paměti. To je důležité, protože mnoho povelů a handlerů SpartaDOSu jsou relokativní soubory, které se umisťují na hranici MEMLO. Výpisem před a po zavedení je možno zjistit délku těchto souborů. Podle hodnoty MEMHI lze pohledem zjistit přítomnost modulu, nebo vestavěného BASICu.

##### **CHKDSK**

vypisuje jmenovku a sekvenci náhodných čísel identifikující disketu, velikost sektoru, celkový počet bytů a volných bytů na disketě a status ochrany diskety proti zápisu (jen diskety verze 2.x).

**Syntaxe:** CHKDSK [Dn:]

**Typ a omezení:** Interní pod CP verze 2.xc a 3.x

##### **Poznámky:**

Povel je určen k výpisu informací o disketě. Pokud je formátu 2.x, tedy inicializovaná povelom XINIT, zobrazí se kromě jména i náhodné a sekvenční číslo, která spolu s jmenovkou identifikují disketu a zajišťují, že SpartaDOS zjistí výměnu diskety i když má stejnou jmenovku. Sekvenční číslo se zvyšuje vždy při otevření souboru pro zápis.. Další údaj je délka sektoru v bytech (128 nebo 256), celkový počet bytů uskladnitelných na disketě a počet vol-

ných bytů. Poslední údaj je status softwarové ochrany proti přepsání disky. Diskety formátu 1.x nemají náhodné a sekvenční číslo, ani status ochrany. Diskety formátu Atari DOS 2 navíc postrádají jmenovku.

**Příklad:**                   CHKDSK

Výpis může vypadat třeba takto:

```
Volume: Games1 0A 25
Bytes/sector: 256
Total bytes: 184320
Bytes free: 123390
Write lock: ON
```

**RPM**

ukazuje rychlosť otáčení diskety.

**Syntaxe:**       RPM [Dn:]

**Typ a omezení:** Externí.

**Poznámky:**

Správný počet otáček u 1050 je 288, u XF551 je o něco vyšší.

### 11.3.12. Práce se strojovým kódem

Pro práci s strojovým kódem je u SpartaDOSu mnoho povelů, které umožňují nejen programy nahrávat, ukládat a spouštět, ale i analyzovat, vypisovat atd. Strojové programy můžeme z hlediska SpartaDOSu rozdělit na běžné binární segmentované soubory s eventuálními rozdíly v místech inicializačních adresami a na soubory externích povelů, které tyto vektory zpravidla nepoužívají a mohou jim být v povelové řádce předávány parametry.

**Externí povely**

jsou binární soubory složeného segmentovaného typu (viz 3.5 a 4.1.11), které mohou, ale nemusí mít definovaný rozdílový a inicializační adresy. Konvence umožňuje jim předávat parametry.

**Syntaxe:**       [Dn:][cesta>]jméno[.ext] parametry

**Typ a omezení:** Podporováno všemi verzemi CP.

**Poznámky:**

Externí povely jsou vykonatelné binární soubory se standardní koncovkou ".COM". Tato koncovka se standardně předpokládá a nepíše se. Pokud má externí povel koncovku jinou, nebo žádnou, musí se uvést buď koncovka nebo samotná tečka. Externí povely se zásadně spouštějí od začátku prvního segmentu, ale pokud jsou definovány standardní startovací vektory na adresách \$2E0 a \$2E2, mají přednost. Samozřejmě je možné psát podle konvencí pro externí povely i vlastní programy.

**Příklad:** TYPING

Spuštění externího povelu "TYPING.COM":

**LOAD**

zavede do paměti jakýkoliv binární soubor, ale nerozběhne jej a ignoruje jakékoliv startovací adresy.

**Syntaxe:** LOAD [Dn:][cesta>]jméno[.ext]

**Typ a omezení:** Interní.

**Poznámky:**

Povel se hodí k načítání znakových sad, binárních a jiných souborů, které se nemají rozběhnout. Poznamenejme, že povel LOAD lze spustit jen z disku, protože LOAD je součástí funkce XIO diskového handlezu.

**RUN**

znovuspuštění posledně spuštěného externího souboru (\*.COM), nebo rozbeh programu od dané adresy.

**Syntaxe:** RUN [adresa]

**Typ a omezení:** Interní

**Poznámky:**

Není-li adresa udána, je spuštěn poslední externí povel. Na adrese RUNLOC je uložena adresa posledního povelu (viz struktura SpartaDOSu). Pokud je adresa udána, začne běh programu zde a do RUNLOC se tato adresa také uloží. Adresa je zadána hexadecimálně. Uživatel plně odpovídá za smysluplnost zadáné adresy.

**Příklady:** RUN 4000

Spuští program začínající na adrese \$4000.

**RUN**

Spuští znovu naposledy prováděný externí povel. Spustíme například diskovou verzi MAC/65 a povelom DOS nebo CP přejdeme do SpartaDOSu. Pomocí RUN se vrátíme zpět do MACu, aniž by se musel znova načítat. Je samozřejmé, že v CP nesmíme mezitím použít povel ničící obsah paměti jako třeba COPY.

**SAVE**

Ukládá na disk obsah paměti.

**Syntaxe:**      **SAVE [Dn:]**[cesta>]jméno[.ext] [/A] adresa adresa

**Typ a omezení:** Interní.

Parametr /A je povolen jen u CP verze 2.x a vyšší.

**Poznámky:**

Ukládá určený úsek paměti na disk a vytváří binární segmentovaný soubor. Eventuální rozdílovou adresu lze definovat povelom PUTRUN.

**APPEND**

připojuje k binárnímu souboru další segment. Je ekvivalentní povelu SAVE s parametrem /A.

**Syntaxe:**      **APPEND [Dn:]**[cesta>]jméno[.ext] adresa adresa

**Typ a omezení:** Interní.

**Informační povely**

Následující povely jsou určeny pro zkušené programátory. DUMP a MDUMP vypisují hexadecimálně binární soubory z paměti a ze souboru, OFF\_LOAD umožňuje relokování programů.

**DUMP**

vypisuje soubor nebo jeho část ve formátech hexadecimálním, ASCII nebo ATASCII.

**Syntaxe:**      **DUMP [Dn:]**[cesta>]jméno[.ext][start[poč.byté]] [/P]

**Typ a omezení:** Externí.

Volitelné parametry "start" a "poč.bytů" nejsou povoleny u disket formátu Atari DOS 2.

**Poznámky:**

Povel DUMP vypisuje požadovanou informaci ze souboru. Parametr "start" udává číslo bytu, od kterého se začne výpis (standard 0). Pokus o zadání vyššího čísla, než je délka souboru vyúsťuje v chybu Address range error. "poč.byté" udává kolik bytů se vypíše. Výpis má nalevo pozici v souboru, potom osm hexadecimálních hodnot a nakonec tytéž hodnoty v reprezentaci kódu ATASCII. Parametr "/P" způsobí nahrazení CONTROL-znaků tečkami a tím výpis ve standardním ASCII. To je užitečné při přesměrování výstupu na tiskárnu povelom PRINT.

**Příklady:**      **DUMP TEST.OBJ 1000 5 /P**

Vypíše ze souboru TEST.OBJ hodnoty bytů \$1000 až \$1004 (relativně k začátku souboru) a zobrazí jejich ASCII ekvivalenty.

**MDUMP**

jako DUMP, ale vypisuje obsah paměti.

**Syntaxe:** MDUMP [adresa [poč.bytů]] [/P]

**Typ a omezení:** Externí.

**Poznámky:**

Povel MDUMP vypisuje požadovanou informaci z paměti. Výpis začíná od ustanovené adresy, obsahuje požadovaný počet bytů. Výpis má nalevo adresu, potom osm hexadecimálních hodnot a nakonec tytéž hodnoty v reprezentaci kódu ATASCII. Parametr "/P" způsobí nahrazení CONTROL-znaků tečkami a tím výpis ve standardním ASCII. To je užitečné při přesměrování výstupu na tiskárnu povelom PRINT.

**Příklady:** MDUMP 2E0 2

vypíše obsah adres \$2E0 a \$2E1 s jejich ekvivalenty v kódu ATASCII.

**OFF\_LOAD**

je služební povel který umožňuje zavádění jednotlivých segmentů binárního segmentovaného souboru posunutě na jiné adresy, než na které svým určením patří. Dále lze vypsat počáteční a koncové adresy jednotlivých segmentů, jejich polohu v souboru a zavedení nebo přeskročení segmentu lze řídit odpověďí na eventuelní dotaz. Povel může také relokovať sám sebe a založit nerelokovatelnou verzi OFF\_LOAD.

**Syntaxe:** OFF\_LOAD [Dn:]cesta>]jméno[.ext] posun [/SNPQ] nebo  
OFF\_LOAD -R adresa [Dn:]cesta>]jméno[.ext]

**Typ a omezení:** Externí pod CP verze 1.x až 3.x.

Parametry 'N' a 'Q' nelze použít u diskety ve formátu Atari DOS 2.

**Poznámky:**

První tvář OFF\_LOAD je povel sloužící k zavádění segmentů složeného binárního souboru (viz 3.5. a 4.1.11.) od určené adresy, která se získá součtem počáteční adresy segmentu a hodnoty "posun", která může nabývat hodnotu 0 až \$FFFF. Činnost se přitom řídí čtyřmi libovolně kombinovatelnými parametry S, N, P a Q. Parametr 'S' (Show) způsobí vypsání počáteční a koncové adresy segmentu spolu s adresou nového počátku po přičtení hodnoty posunu. Parametr 'N' udává, že se segmenty nebudou zavádět. To lze využít v případě, že chceme znát jen umístění segmentů a nechceme je zavést. Parametrem 'P' se zobrazí pozice segmentu v souboru (vzdálenost od začátku). Poslední parametr 'Q' má za následek že se zavádění na začátku každého segmentu zastaví a uživatel je sledován "Load this segment?" dotázán, zda se má právě násazený segment zavést. Odpověď je Y (ano) nebo N (ne).

Standardní OFF\_LOAD je relokativní. Zavádí se na adresu \$B400 a poté se překopíruje nad hranici MEMLO a rozběhne se. Nefunguje je-li zasunut modul nebo aktivován vnitřní BASIC.

Ve druhé formě se přemístí soubor OFF\_LOAD na udanou adresu a zapíše se do souboru udaného jména. Tak je možné přesunout OFF\_LOAD z adresové oblasti zásuvného modulu.

Příklady: OFF\_LOAD TEST.COM 0 /NS

zobrazí adresy segmentů, ale soubor nenačte.

#### **PUTRUN**

přidává k binárnímu segmentovanému souboru vektor obsahující startovní adresu externího povelového souboru. Tím se umožní souštět povel jako AUTORUN.SYS, u kterého se spouští pouze podle adres INIT/RUN.

**Syntaxe:** PUTRUN [Dn:][cesta>]jméno[.ext]

**Typ a omezení:** Externí.

#### **Poznámky:**

Povel je prakticky použitelný jen ke konverzi souborů externích povelů pro spouštění pod Atari DOS 2.5, nebo pro doplňování RUN adresy programů uložených z paměti povelom SAVE.

#### **11.3.13. Práce s diskem u SpariaDOSu**

Tato kapitola popisuje práci diskových handlerů a princip interfejsu na diskové jednotky a funkci povelu VERIFY. Něco obecně již bylo o práci disků řešeno, podrobnosti jsou v literatuře (Operating System User's Manual) proto zde jen krátce. Počítač musí být schopen předávat a přebírat informace z diskety podle pravidel, které určuje intelligentní řadič umístěný v každé disketové jednotce. Přístup na vlastní povrch diskety je řízen třemi úrovněmi interfejsu:

1) Interfejs mezi počítačem a diskovou jednotkou. Většinou se nazývá SIO a je používán ke komunikaci se všemi zařízeními na sériové sběrnici. Dialog probíhá v těchto krocích:

- Počítač nastaví vedení COMMAND do stavu LOW.
- Počítač vyšle povelový vzorec, skládající se ze čtyř bytů. Jsou to ID zařízení, povel (read, write, status, format), dva byty doplňujících informací AUX1 a AUX2. U disků to bývá číslo sektoru. Povelový vzorec je následován kontrolním součtem.
- Počítač nastaví vedení COMMAND do stavu HIGH.
- Zařízení (disk) určené pomocí ID odpovídá posláním ACK (povel platný) nebo NAK (povel neplatný).
- Pokud povel vyžaduje data (třeba při zápisu sektoru), pošle počítač datový balík následovaný kontrolním součtem.

- Pokud nastal předchozí případ, pošle jednotka signál ACK pokud byla data platná, nebo NAK pokud byla vadná.
  - Jednotka provede požadovanou operaci. Po dokončení pošle buď kód COMPLETE, nebo kód ERROR.
  - Při čtení posílá datový balík následovaný kontrolním součtem jednotka.
- 2) Interfejs mezi mikroprocesorem diskové jednotky a řadičem. V disketové jednotce je specializovaný čip nazývaný řadič nebo kontrolér, který má za úkol řídit fyzickou práci s disketou. Určuje její formát, vyhledává sektory a stopy a přijímá nebo předává data sektoru. Mikroprocesor diskové jednotky přebírá z řadiče povel, číslo stopy a sektoru. Potom posílá do řadiče ze svého bufferu data (při čtení), nebo je naopak do něj přijímá.
- 3) Interfejs mezi mikroprocesorem diskové jednotky a hardware.
- Poslední interfejs umožňuje mikroprocesoru diskové jednotky řídit motor jednotky, krokový motorek přesunující hlavu ze stopy na stopu, testovat senzory pro povolení zápisu a podobně.

#### **Práce Sparta DOSu s vyrovnavací pamětí (bufferem).**

SpartaDOS používá zcela odlišný způsob práce od DOS 2.5. Na rozdíl od všech ostatních DOSů přiděluje SpartaDOS sektorové vyrovnavací paměti dynamicky. To znamená, že nepotřebuje pro každou ohlášenou diskovou jednotku jeden buffer a stejně tak nepotřebuje buffer pro každý otevřený soubor. Teoreticky je možné mít na sedmi různých jednotkách otevřeno 7 souborů s použitím jediného bufferu pro jednoduchou a dvou pro dvojitou hustotu (i když to je na úkor rychlosti).

#### **Vektor přístupu na disk**

Použití tohoto vektoru umožňuje začleňovat do systému RAM-disky, nebo u ATR8000 přepínat konkurenční modus. Jde o to, že ve SpartaDOSu jdou veškeré operace s diskem přes jeden vektor. RAM-disk a ostatní handlery se do tohoto vektoru ohláší a při požadavku na akci, která se jich týká převezmou aktivitu. Tudy jdou také všechny povely SpartaDOSu a touto technikou je možno využívat přednosti rychlého I/O režimu s použitím US doubleru.

#### **US Doubler**

je doplňující zařízení pro diskovou jednotku 1050. Má dva soubory rutin pro sériovou komunikaci s počítačem, jeden pro standardní rychlosť, druhý pro zvýšenou. Rutina, která hlídá vedení COMMAND čte povelový vzorec v jedné rychlosti a pokud nastane chyba, zkouší číst okamžitě v další možné rychlosti. Jakmile se rychlosť najde a čtení proběhne bez chyby, stává se standardem. SpartaDOS posílá při startu povel '?' (kterému neupravená jednotka nerozumí) aby zjistil jak rychlý je I/O režim upravené jednotky. Pokud je dotaz jednotkou zodpovězen, pracuje se nadále v rychlém režimu, který funguje nejen s US Doublerem, ale i např. se Speedy 1050. Programy, které používají standardní vektor SIO \$E459 nemohou rychlý modus použít.

### **Verifikace**

SpartaDOS pracuje standardně bez verifikace zápisu ze dvou důvodů. Jedenak jsou diskové jednotky velmi spolehlivé a chyby při zápisu se téměř nevyskytují, jednak je zápis s verifikací třikrát pomalejší. Sektoru jsou totiž na standardní disketě uspořádány tak, že neupravená jednotka může přečíst nebo zapsat bez verifikace při jednoduché nebo rozšířené hustotě dva za sebou číslované sektory během jedné otáčky. 10 sektorů se tak zapíše za 5 otáček, což je asi 1 vteřina. Při zapnuté verifikaci však zápis sektoru trvá 1,5 otáčky. Pro bezpečnostní fanatiky kterým zpomalení nevadí, má SpartaDOS povel VERIFY.

### **VERIFY**

zapne nebo vypne zápis s verifikací.

**Syntaxe:**      VERIFY {ON|OFF}

**Typ a omezení:** Interní pod CP verze 2.x a 3.x.

### **Poznámky:**

Verifikace znamená zapsání sektoru, jeho opětné přečtení a porovnání se zapisovaným sektorem v paměti. Je to pomalé, ale lze tak zachytit eventualní chybu při zápisu. Snad stojí za to verifikaci zapnout jsou-li problémy s disketu.

## 11.4. SpartaDOS pro programátory

SpartaDOS poskytuje programátorovi nepřeberné množství různých služeb ať již pracuje v Basicu nebo v jiném jazyce. Část prostředků se dá používat pomocí rozšířených funkcí XIO, další větší část pomocí vektorů, rezidujících na určených místech paměti. Pro začátečníky bude tato kapitola patrně těžko srozumitelná, ale mohou se k ní vrátit po nabytí potřebných znalostí.

### 11.4.1. Funkce SpartaDOSu volané z Basicu

Následující seznam funkcí je přístupný buď standardně, nebo funkcí XIO. Pokud funkci odpovídá povel CP, je uveden v závorce. V popisu jsou uvedeny jen funkce, které pod DOS 2.5 nejsou, nebo jsou modifikovány či rozšířeny.

#### Otevření souboru

**Syntaxe:**      **OPEN #IOCB,AUX1,AUX2,"Dn:jméno [.ext]"**

kde: - znak # je třeba uvést

- IOCB je aritmetický výraz, nebo číslo určující číslo kanálu

- AUX1 je aritmetický výraz nebo číslo, určující druh prováděné operace:

4 - Pouze čtení.

6 - Formátované čtení adresáře diskety. Obsah je ve tvaru který produkuji povely DIR a DIRS. Volba se provádí hodnotou AUX2. Pro AUX2=128 je formát dlouhý, pro AUX2=0 krátký. Dlouhý formát je možný jen na disketách ve formátu SpartaDOS.

7 - SpartaDOS nemá !!

8 - Pouze zápis. Povel pracující se souborem v modu zápisu může pomocí parametru '/A' přepnout do otevření do modu append.

9 - append, připojování dat na konec souboru.

12- UPDATE. Soubor lze číst i psát.

20- Otevření pracovního adresáře pro čtení. Adresář je čten jako soubor. Tento modus je možno použít pouze na disketách formátu SpartaDOS.

24- Otevření pracovního adresáře v režimu UPDATE. Data adresáře lze číst a zapisovat jako obyčejný soubor. Tento modus je možno použít pouze na disketách formátu SpartaDOS.

36- Otevření podadresáře ve čtecím modu s tím, že se má podadresář číst jako normální soubor. Tento modus je možno použít pouze na disketách formátu SpartaDOS.

#### Poznámky:

Při neformátovaném čtení a aktualizaci adresáře je třeba maximální opatrnost, stejně jako při použití některých modů. Použije-li se například modus

40 (zápis do podadresáře), lze jej celý zničit. Vypsané mody jsou jediné, které mohou být uživateli užitečné. Jedinými legálními uživateli modů jako je 40 jsou funkce CREDIR a DELEDIR. Diskety formátu Atari DOS 2 lze číst se Spar-  
taDOSem verze 2.x, DOS 2.5 od verze 3.2.

### Přejmenování souboru (RENAME)

**Syntaxe:** XIO 33,\*IOCB,0,0,"[Dn:][cesta>]jméno [.ext]"

kde: - znak \* je třeba uvést

- IOCB je aritmetický výraz, nebo číslo určující číslo kanálu

### Poznámky:

Aby bylo možno tuto operaci provést, musí se dotyčný kanál nejprve uzavřít. Lze používat filtr a funkci lze aplikovat na disketu jakéhokoliv přípustného formátu.

### Blokování diskety (LOCK)

**Syntaxe:** XIO 34,\*IOCB,0,0,"Dn:"

kde: - znak \* je třeba uvést

- IOCB je aritmetický výraz, nebo číslo určující číslo kanálu

### Poznámky:

Lze použít jen přes uzavřený kanál. Funkce platí jen pro diskety formátu 2.x pod CP verze 2.x a novější.

### Blokování souboru

**Syntaxe:** XIO 35,\*IOCB,0,0,"[Dn:][cesta>]jméno [.ext]"

kde: - znak \* je třeba uvést

- IOCB je aritmetický výraz, nebo číslo určující číslo kanálu

### Poznámky:

Před použitím operace musí být kanál uzavřen. Ve jméně souboru lze použít filtr. Platí pro každý přípustný formát diskety a CP verze 2.x a novější.

### Uvoľnení souboru

**Syntaxe:** XIO 36,\*IOCB,0,0,"[Dn:][cesta>]jméno [.ext]"

kde: - znak \* je třeba uvést

- IOCB je aritmetický výraz, nebo číslo určující číslo kanálu

### Poznámky:

Před použitím operace musí být kanál uzavřen. Ve jméně souboru lze použít filtr. Platí pro každý přípustný formát diskety a CP verze 2.x a novější.

### **Nastavení pozice v souboru**

<b>Syntaxe:</b>	X-POS Y-0 <b>POINT =IOCB ,X,Y</b>	pro formát SpartaDOS
<b>nebo:</b>	Y-INT(POS/65536) POKE 846+IOCB*16,Y POS-POS-Y*65536 Y-INT(POS/256) POKE 845+IOCB*16,Y POKE 844+IOCB*16,POS-Y*256 XIO 37,=IOCB,0,0,"Dn:"	
<b>nebo:</b>	<b>POINT =IOCB,sektor,byte</b>	pro formát Atari

#### **Poznámky:**

Pro diskety formátu SpartaDOS: V první metodě musí být POS od 0 do 32767. Druhá metoda umožňuje zpřístupnit až 8 388 607 (\$FFFFFF) byte. Pozice udává vzdálenost bytu od začátku souboru. Je-li soubor otevřen v režimu UPDATE, lze nastavit pozici i mimo soubor. Prostor mezi koncem souboru a nastaveným místem se vyplní nulami, ale žádné sektory se nepoužijí k jejich fyzickému uložení. Tak lze dosáhnout soubor o délce 32K, zabírající na disketu 5 sektorů. Pokud se májí z této díry čist data, založí se okolo požadovaného místa vynulovaný sektor o délce 128 nebo 256 byte.

POINT u SpartaDOSu používá pozici vzhledem k začátku souboru. Je to způsob odlišný od DOS 2.5, kde se používá číslo sektoru a číslo byte v něm.

Pro diskety formátu Atari DOS: Používá se třetí metoda a význam parametrů je stejný jako u DOS 2.5.

### **Zjištění pozice v souboru**

<b>Syntaxe:</b>	<b>POINT =IOCB ,X,Y</b> pro formát SpartaDOS
	POS-X
<b>nebo:</b>	XIO 38,=IOCB,0,0,"Dn:" POS-PEEK(846+IOCB*16)*65536 POS-PEEK(845+IOCB*16)*256 POS-PEEK(844+IOCB*16)
<b>nebo:</b>	<b>NOTE =IOCB,sektor,byte</b> pro formát Atari

#### **Poznámky:**

Pro diskety formátu SpartaDOSu: V první metodě musí být POS od 0 do 32767. Druhá metoda umožňuje zpřístupnit až 8 388 607 (\$FFFFFF) byte. Pozice udává vzdálenost bytu od začátku souboru. NOTE u SpartaDOSu používá absolutní pozici vzhledem k začátku souboru. Je to způsob odlišný od DOS 2.5, kde se používá číslo sektoru a číslo byte v něm.

Pro diskety formátu Atari DOS: Používá se třetí metoda a význam parametrů je stejný jako u DOS 2.5.

## Zjištění délky souboru

**Syntaxe:** XIO 39,\*IOCB,0,0,"Dn;"  
 POS-PEEK(846+IOCB\*16)\*65536  
 POS-POS+PEEK(845+IOCB\*16)\*256  
 POS-POS+PEEK(844+IOCB\*16)

### Poznámky:

Takto zjistíme délku právě otevřeného souboru v bytech. Funguje to jen na disketách ve formátu SpartaDOSu. Pro formát Atari nemá ekvivalent.

## Načtení binárního souboru (LOAD)

**Syntaxe:** XIO 40,\*IOCB,4,X,"Dn:[cesta]>]jméno[.ext]"

### Poznámky:

Zavedení binárního souboru do paměti. Je-li X<128, použijí se inicializační a rozběhové vektory INIT/RUN. Kanál musí být otevřen.

## Uložení úseku paměti (SAVE, APPEND)

**Syntaxe:** XIO 41,\*IOCB,R,X,"Dn:[cesta]>]jméno[.ext] adr1 adr2"

- kde: - R je modus zápisu. R=8 znamená přepsání souboru, R=9 znamená připojení k existujícímu souboru.
- X určuje, zda bude na začátek segmentu zapsána hlavička \$FF FF. X<128 znamená zápis hlavičky.
- adr1, adr2 jsou začátek a konec ukládané oblasti.

### Poznámky:

Ukládá paměť mezi adresami adr1 a adr2 do udaného souboru. Adresy se píší hexadecimálně. R>128 je vhodné zadávat při připojování segmentů k existujícímu souboru. Kanál nesmí být otevřen.

## Založení podadresáře (CREDIR)

**Syntaxe:** XIO 42,\*IOCB,0,0,"Dn:cesta"

### Poznámky:

Založí nový podadresář, jehož jméno je určeno posledním jménem v cestě. Podrobnosti viz povel CREDIR. Kanál musí být uzavřen. Funkce nefunguje na disketách formátu Atari.

## Zrušení adresáře (DELDIR)

**Syntaxe:** XIO 43,\*IOCB,0,0,"Dn:cesta"

**Poznámky:**

Zruší podadresář, jehož jméno je určeno posledním jménem v cestě. Podrobnosti viz povel DELDIR. Kanál musí být uzavřen. Funkce nefunguje na disketách formátu Atari.

**Změna pracovního adresáře (CWD)**

**Syntaxe:** XIO 44,\*IOCB,0,0,"Dn:cesta"

**Poznámky:**

Zruší nový podadresář, jehož jméno je určeno posledním jménem v cestě. Podrobnosti viz povel DELDIR. Kanál musí být uzavřen. Funkce nefunguje na disketách formátu Atari.

**Funkce BOOT**

**Syntaxe:** XIO 45,\*IOCB,0,0,"Dn:[cesta>]jméno [.ext]"

**Poznámky:**

Kanál musí být uzavřen. Ve jméně lze použít filtr. Použitelné jen pro diskety formátu 2.x a pod CP verze 2.x a 3.x.

**Uvolnění diskety**

**Syntaxe:** XIO 46,\*IOCB,0,0,"Dn:"

**Poznámky:**

Kanál musí být uzavřen. Použitelné jen pro diskety formátu SpartaDOS.

**Formátování disket Atari DOS 2 (AINIT)**

**Syntaxe:** XIO 254,\*IOCB,0,0,"Dn:"

**Poznámky:**

Kanál musí být uzavřen. Pouze u CP verze 2.x a 3.x.

**Výpis adresáře (DIR, DIRS)**

**Syntaxe:**

```

10 DIM A$(40):TRAP 40
20 OPEN #IOCB,6,X,"Dn:[cesta>]jméno [.ext]"
30 INPUT #IOCB,A$?:A$":GOTO 30
40 CLOSE #IOCB

```

kde:

- X určuje druh výpisu (viz OPEN)
- IOCB kanál

#### **11.4.2. Vektorové tabulky SpartaDOSu**

SpartaDOS je systém řízený z volné části vykonavačem povelů a velká část jeho proměnných je uživateli zpřístupněnatak, aby mohl pod SpartaDOSem psát vlastní povely a vyvijet různé aplikace. Datová základna je zpřístupněna pod adresou COMTAB, její hodnota je uložena v DOSVEC (\$0A). U SpartaDOSu verze 3.2 je tabulka rozšířena a kromě toho je přidána další tabulka vektorů v oblasti RAM pod operačním systémem, zpřístupňující služby okolo data, času a podobně. Vektory z oblasti COMTAB jsou popsány v následujících řádcích se zřetelem na verzi 3.2, která bude jistě nejpoužívanější. Jednotlivé vektory jsou vždy uvedeny názvem a relativním umístěním vůči referenční adrese COMTAB, jejíž adresa je k nalezení na adrese 10 (\$0A) DOSVEC.

##### **DWARM**

[COMTAB-21]

Obsahuje kopii WARMFLG (8) a používá se při přechodu do modulu. Ne-nulová hodnota znamená, že uživatelská paměť je platná, 0 že byl pro-veden destruktivní povel, nebo zaveden binární soubor.

##### **DDENT**

[COMTAB-19]

Tabulka velikosti sektorů u každé disketové jednotky (1 až 8). 0 zna-me-na 256 byte/sektor, 128 128byte/sektor.

##### **LSIO**

[COMTAB-10]

Vektor ukazující na vlastní rutiny SIO SpartaDOSu. Tuto adresu můžeme nahradit i adresou vlastní implementace SIO. Mění ji například RAM-disk, aby zachytíl přístup na jednotku, kterou emuluje. Tento vektor je používán mnoha rutinami, které využívají rychlou rutinu SIO.

##### **ECHOFLG**

[COMTAB-8]

Tato adresa obsahuje index souboru, na který kopíruje výstup na obra-zovku, do tabulky HATABS. Hodnota \$FF znamená, že přesměrování výstupu není aktivní. Adresa platí od verze 2.x.

##### **BATFLG**

[COMTAB-6]

Tato adresa obsahuje index souboru, ze kterého se nahrazuje vstup z klávesnice, do tabulky HATABS. Hodnota \$FF znamená, že žádný dávkový soubor není aktivní. Adresa platí od verze 2.x.

##### **WRTCMD**

[COMTAB-2]

Tato adresa obsahuje zápisový povel SIO. Je-li zde 'W' zapisuje se s verifikací, je-li zde 'P' bez verifikace. Adresa platí od verze 2.x.

##### **WARMST**

[COMTAB-1]

U verze 3.2 se nepoužívá, u 2.x indikuje, že CP dělá studený start. Při každém vstupu do CP se nuluje.

**COMTAB**

[COMTAB]

Referenční bod tabulky. Obsahuje skok na vykonavač povelů CP. Tudy se zásuvné moduly a uživatelské programy dostávají "do DOSu".

**ZCRNAME**

[COMTAB+3]

Skok na rutinu CRNAME, používanou externími povely na získání následujícího jména souboru z povelové řádky, která je umístěna v 'LBUF'. Výsledné jméno je v 'COMFNAM'.

**ZDIVIO**

[COMTAB+6]

Adresa rutiny pro přesměrování vstupu a výstupu. Lze ji volat z assembleru nepřímo přes ZDIVIO. Jméno souboru musí být v COMFNAM a registr Y-0 při výstupu (PRINT), Y-1 při vstupu (-jméno).

**ZXDIVIO**

[COMTAB+8]

Adresa rutiny ukončující přesměrování vstupu a výstupu. Z assembleru ji lze volat nepřímo přes ZXDIVIO s volbou funkce podle registru Y. Y-0 znamená ukončení výstupu (PRINT), Y-1 vnutí signál EOF a ukončí dátovkový soubor.

**BUFOFF**

[COMTAB+10]

Aktuální index do povelové řádky. Tento ukazatel používá CRNAME při vyhledávání dalšího parametru v povelové řádce a při jeho přenosu do COMFNAM.

**ZORIG**

[COMTAB+11]

Tato adresa obsahuje startovací adresu SpartaDOSu. SPEED.DOS a STANDARD.DOS mají adresu \$600, ostatní \$700.

**DATER**

[COMTAB+13]

Datum ve formě dd/mm/yy (3 byty). Toto datum se ukládá v hlavičce nově zakládaného souboru nebo adresáře. Potlačení tohoto data viz TDOVER.

**TIMER**

[COMTAB+16]

Čas ve formátu hh/mm/ss (3 byty). Není ve formátu BCD, takže jej lze bez konverze číst z Basicu. Tento čas se umisťuje do adresáře při zakládání souboru nebo podadresáře. Potlačení tohoto času viz TDOVER.

**ODATER**

[COMTAB+19]

Alternativní datum ve stejném formátu jako DATER. Použije se místo DATER pokud je nastaven semafor TDOVER.

**OTIMER**

[COMTAB+22]

Alternativní čas ve stejném formátu jako TIMER. Použije se místo TIMER pokud je nastaven semafor TDOVER.

**TDOVER**

[COMTAB+25]

Semafor pro použití alternativního data a času. Pokud je nulový, použije se při zakládání nových souborů DATER/TIMER. Je-li \$FF, použije se alternativní sada ODATE/TIMER. Alternativní sada se používá při kopírování souborů pomocí kopirovacích programů jako jsou MENU nebo XCOPY pro zachování data a času stejného u kopie i originálu.

**TRUN**

[COMTAB+26]

Obsahuje RUN adresu zavedeného binárního souboru. TRUN se aktualizuje interní zaváděcí rutinou. Proto je možné z Basicu i jakéhokoliv jiného programu zjistit, jaká adresa to byla. Adresa RUNLOC je aktualizována z této adresy výhradně interpretrem povelů CP.

**SBUFF**

[COMTAB+28]

U CP verze 3.2 adresa začátku sektorových bufferů, u CP verze 2.x hodnota 128.

**DDENT**

[COMTAB+29]

Tabulka hustot disket v jednotkách 1 až 4 u CP verze 1.x. U ostatních verzí je tabulka nepřístupná.

**SMEMLO**

[COMTAB+30]

Zde je nejvyšší adresa obsazená SpartaDOSem v dolní části paměti. Handlery instalované během startu počítače leží v oblasti mezi SMEMLO a MEMLO.

**INCOMND**

[COMTAB+32]

Hodnota -1 v tomto semaforu ukazuje, že jsme v CP. Ø indikuje, že jsme v Basicu nebo jiném modulu. Semafor slouží k rozlišení, zda má inicializační rutina skočit do CP nebo modulu.

**COMFNAM**

[COMTAB+33]

Buffer pro výstup rutiny CRNAME, na kterou ukazuje ZCRNAME. Je dlouhý 28 byte (implicitní hranice pro délku jména a cesty) a vždy začíná 'Dn'. Pokud hledáme jen parametry, můžeme začít od COMFNAM + 3.

**RUNLOC**

[COMTAB+61]

Obsahuje startovací adresu naposledy spouštěného externího povelu, pokud byl startován přes CP. Pokud se při povelu RUN nezadá adresa, použije se adresa z RUNLOC.

**LBUF**

[COMTAB+63]

Na této adrese začíná vstupní buffer, ve kterém se ukládá povelová řádka. LBUF je dlouhý 64 znaků.

### Vektory služeb SpartaDOSu 3.2

Tyto vektory jsou ukryty v adresovém prostoru ROM operačního systému. Slouží k nastavování a čtení data a času, vykonávání povelových řádek, inicializaci systému a mnoha dalším činnostem. Jejich pole začíná na adrese \$FFC0 a zpřístupňují se následujícím mechanismem:

LDA	\$D301	,PIA
PHA		,uchovalení původní hodnoty portu
AND	#\$FE	,bit 0 do OFF
STA	\$D301	,zpřístup do RAM pod OS
JSR	VGETTD	,vyvolání požadované procedury
PLA		
STA	\$D301	,obnovení portu - zapnutí OS

Tyto vektory obsahují skokové instrukce na příslušné rutiny. Mechanismus je stejný, jako při volání standardních služeb OS přes jeho vektorovou tabulku. Pokud není funkce zpočátku implementována, obsahuje vektor místo skoku instrukce SEC a RTS. Podporovány jsou tyto vektory:

#### VGETTD \$FFC0

Funkce vraci čas a datum do TIMER a DATER. Pokud funkce selže, signalizuje to nastavený CARRY. Tento vektor se využívá k aktualizaci TIMER a DATER při otvírání souboru pro zápis. Dále jej používají povelky TIME, DATE, TDLINE a ZHAND.

#### VSETTD \$FFC3

Funkce nastavující datum a čas. Při vstupu je nové datum a čas v TIMER a DATER. Pokud nastane chyba, vraci funkce nastavený bit Carry. Ten-to vektor používají TIME, DATE a nastavovací funkce ZHAND.

#### VTDON \$FFC6

Funkce zapíná a vypíná řádku se zobrazením data a času. Na vstupu znamená 0 v registru Y požadavek na vypnutí, 1 na zapnutí řádky. Pokud funkce selže, vraci se s nastaveným bitem CARRY. Funkce není SpartaDOSem podporována interně. Vektor se instaluje spuštěním funkcí 38 a 39 handleru "Z:", nebo externím povelem TDLINE.

#### VFMTTD \$FFC9

Funkce vraci formátovanou řádku s datem a časem do uživatelem určeného bufferu. Na vstupu musí registry X a Y obsahovat horní a dolní byte adresy bufferu. Na výstupu signalizuje bit Carry úspěšnost provedení funkce. Tuto funkci SpartaDOS nepodporuje interně, ale je instalována s handlerem TDLINE povelem TDLINE nebo funkcemi ZHAND.

**VINITZ****\$FFCC**

Je volán poté, co SpartaDOS dokončil svou inicializaci po RESETu. Povel AUTOBAT.COM instaluje pomocí tohoto vektoru automatické spuštění zadáné sekvence po RESETu. Inicializaci se rozumí výsledek volání přes vektor DOSINI. OS volá tento vektor před předáním řízení modulu nebo DOSu.

**VINITZZ****\$FFCF**

Tento vektor používá SpartaDOS po dokončení neresetové inicializace. Některé povely SpartaDOSu (SCOPY, UNERASE apod.) inicializují DOS po provedení své funkce. Provádějí to skokem přes vektor DOSINI stejně, jako OS po RESETu.

**VXCOMLI****\$FFD2**

Tento vektor volá povelový procesor CP aby vykonal povelovou řádku v LBUF. Při volání musí být BUFOFF nulový. Eventuelní chyby v povelové řádce jsou obvyklým způsobem oznámeny. Před a po povelu nejsou žádné výzvy. Timto způsobem lze například psát menu.

**VCOMND****\$FFDS**

Pomocí tohoto vektoru je volán hlavní vstupní bod do vykonavače povelů CP. Uživatel může změnou tohoto vektoru aktivovat vlastní CP. CP SpartaDOSu vlastně pouze vypíše značku výzvy, přečte řádku, zavolá VXCOMLI a skáče na začátek. Touto metodou se vyvolávají povely CP z Basicu.

**VPRINT****\$FFD8**

Vektor ukazující na hlavní výpisovou rutinu SpartaDOSu. Způsob volání je tento:

```
JSR    VPRINT
.BYTE "Libovolny text",$9B,-1
```

**VKEYON****\$FFDB**

Funkce vypíná a zapíná buffer klávesnice podle obsahu registru Y. Nula vypíná, 1 zapíná buffer. Tuto funkci podporuje SpartaDOS interně.

**11.4.3. Formát disket SpartaDOSu**

SpartaDOS rozeznává čtyři typy sektorů: boot sektory, bitové mapy, sektorové mapy a datové sektory. Popis všech těchto typů následuje.

**Sektorové mapy**

Jsou seznamy sektorů, tvořících soubor. První dva údaje jsou odkaz na následující a předchozí sekutorovou mapu. Zbytek sektoru je seznam 62 (SD) nebo 126 (DD) čísel datových sektorů.

- následující: Číslo sektoru následující sektorové mapy. Jde-li o poslední mapu, je zde nula.
- předchozí: Číslo sektoru předchozí sektorové mapy. U první mapy je zde nula.
- data: Čísla datových sektorů souboru. Je-li číslo sektoru 0, není tato část souboru alokována. To může nastat při zápisu na začátek a potom blízko konce souboru, aniž by se zapisovalo doprostřed. Viz též POINT.

### **Bitové mapy**

Bitová mapa je posloupnost bitů, určujících zda je sektor použit nebo ne. Bit  $T$  reprezentuje první, bit 0 poslední ze skupiny osmi sektorů. První byte mapy odpovídá sektorům 0 až 7 (sektor 0 ovšem neexistuje), druhý 8 až 15 atd. Pokud je potřeba více bitových map než 1, jsou na disketě sekvenčně za sebou. Sektor je volný, je-li odpovídající bit 1.

### **Boot sektory**

Jsou první tři sektory na každé disketě formátu SpartaDOS. Obsahují program zavádějící DOS, připojující ho do systému a inicializující CP. První sektor obsahuje velkou datovou tabulku, která ukazuje na první bitovou mapu, sektorovou mapu hlavního adresáře, údaj o hustotě záznamu a volných sektorech. V seznamu znamená číslo vzdálenost od začátku sektoru:

- 9        adresa první sektorové mapy hlavního adresáře.
- 11      celkový počet sektorů na disketě
- 13      počet volných sektorů
- 15      počet bitových map
- 16      číslo sektoru první bitové mapy
- 18      číslo sektoru od kterého začíná hledání místa pro alokaci datových sektorů. Je to první sektor, od kterého se při zápisu souboru testuje, zda je volný.
- 20      číslo sektoru od kterého začíná alokace sektorů pro adresáře. Od tohoto sektoru se začíná hledat volné místo při rozširování adresáře, nebo zakládání nového. Sektory se přidělují pokud možno souvisle pro dosažení rychlejšího přístupu k informacím v adresářích.
- 22      jmenovka diskety. U diskety verze 1.x, nebo pro práci v CP verze 1.x musí být jedinečná.
- 30      počet stop. Jde-li o oboustrannou jednotku, je nejvyšší bit 1.
- 31      velikost sektorů. 0 indikuje 256, 128 znamená 128.
- 32      hlavní číslo formátu diskety. Možné hodnoty jsou \$11 pro verzi 1.x a \$20 pro 2.x.
- 33      počet sektorových bufferů u verze 1.x.
- 34      číslo standardní jednotky pro CP
- 35      VYHRAZENO
- 36      VYHRAZENO
- 37      pouze u verze 1.x

- 38 sekvenční číslo diskety. Zvyšuje se při každém otevření diskety (resp. souboru) pro zápis. Spolu s jmenovkou a náhodným číslem určuje, zda byla disketa vyměněna. Pouze u disket verze 2.x.
- 39 náhodné číslo diskety. Přiděluje se při formátování a jeho funkce je stejná jako u sekvenčního čísla. Pouze u disket verze 2.x.
- 40 adresa první sektorové mapy souboru určeného povelom BOOT. Pouze u disket verze 2.x.
- 42 Semafor blokování diskety proti zápisu. \$FF znamená, že disketa je chráněna, 0 že není. Pouze u disket verze 2.x.

#### Struktura adresářů

Adresář je u Sparta DOS užití soubor obsahující informace o každém souboru a podadresáři, který je v něm obsažen. Jedna věta adresáře obsahuje 23 bytů a obsahuje jméno souboru, datum a čas vzniku, délku, místo první sektorové mapy a status věty. První věta je zvláštní případ, popisuje svůj adresář jako soubor. Věta popisující v nadřízeném adresáři podadresář obsahuje totéž co věta popisující soubory s výjimkou délky. První věta obsahuje tuto informaci (čísla jsou relativní adresy vzhledem k začátku adresáře):

- 1 Adresa první sektorové mapy nadřízeného adresáře. Nula indikuje hlavní adresář.
- 3 Délka adresáře (3 byty)
- 6 Jméno adresáře (8 bytů)

Je-li adresář otevřen v neformátovaném modu (viz OPEN), pozice v souboru se automaticky nastaví na druhou větu. Pokud chceme číst informaci z první věty, musíme nastavit pozici na začátek povelom POINT. Druhá a další věty obsahují tuto informaci (čísla jsou pozice ve větě):

- 0 Status souboru. Nula indikuje konec adresáře.  
význam jednotlivých bitů:  
0 - soubor je blokován  
3 - pozice v adresáři je použita  
4 - soubor je zrušen  
5 - soubor je podadresář
- 1 adresa první sektorové mapy souboru
- 3 délka souboru (3 byty)
- 6 jméno souboru (8 byte)
- 14 koncovka jména - popisná část
- 17 datum založení (dd/mm/yy) 3 byty
- 20 čas založení souboru (hh/mm/ss) 3 byty

#### 11.4.4. Různé poznámky a informace

##### Další funkce Sparta DOSu volané přes CIO

Tyto speciální funkce nejsou začleněny do funkci XIO v Basicu, protože jejich použití z Basicu by bylo obtížné.

##### Status diskety (CHKDSK)

Vypisuje informace o disketě. Vstupní nastavení je toto:

iccom	- 47
icbal	- dolní byte adresy řetězce "Dn:"
icbah	- horní        -..-
icbll	- dolní byte adresy výstupního bufferu
icblh	- horní        -..-

Výstupní informace v bufferu:

buffer	- výsledek operace CHKDSK (17 byte)
+0	- číslo verze diskety (0-Atari DOS 2 nebo DOS 2.5)
+1	- počet byte v sektoru, 0->256
+2	- celkový počet sektorů (low,high)
+4	- počet volných sektorů (low,high)
+6	- jmenovka 8 byte (jen SpartaDOS)
+14	- sekvenční číslo 1 byte (SpartaDOS 2.x)
+15	- náhodné číslo 1 byte (SpartaDOS 2.x)
+16	- semafor blokování diskety, 0=false, neblokováno SpartaDOS 2.x

##### Zjištění cestý k pracovnímu adresáři (?DIR)

Vstupní parametry:

iccom	- 48
icbal	- dolní byte adresy řetězce "Dn:[cesta]"
icbah	- horní        -..-
icbll	- dolní byte adresy výstupního bufferu
icblh	- horní        -..-

Výstupní informace:

buffer	- výsledek operace ?DIR. Je to platná cesta pro přechod do zadaného adresáře. Pokud se na vstupu nezadá cesta, uloží se zde cesta do pracovního adresáře (nejčastější použití). Cesta je zakončena EOL.
--------	---

##### Identifikace verze SpartaDOSu

Od verze 2.5 je na adresách \$700 a \$701 uloženo ve dvou identických bytech číslo verze. \$700 obsahuje vždy hodnotu \$53, \$701 číslo verze (od \$25 do \$32).

## Poznámka k verzi 3.2

Protože DOS předává fízení po inicializaci do CP pokud je INCOMND = -1, přechází se do CP po stisku klávesy RESET nebo skoku na \$E474 i z uživatelského programu pokud není v modulu. Pokud je třeba tomu zabránit, musí program vložit na adresu \$702 hodnotu -1.

## Poznámky k BASIC XE

Z důvodu kompatibility s BASIC XE nepoužívá SpartaDOS oblasti \$D800 až \$FFFF nebo \$C000 až \$CBFF. Tyto oblasti jsou normálně použity pro povel AINIT a pro slovní hlášení chyb. Proto při používání BASIC XE nefunguje AINIT a chyby jsou hlášeny jen čísly. Počet bufferů se snižuje z 16-ti na 8.

### 11.4.5. Funkce handlu "Z:"

Instalace handlu se provádí povelením CHAIN (SpartaDOS 3.2). Handler se instaluje jak pro hardwarové hodiny, tak pro softwarové, pokud není modul R-TIME 8 instalován. Funkce handlu jsou po jeho instalaci přístupné z Basicu pomocí funkcí XIO. Jde o tyto funkce:

Zapnutí řádky s datem a časem (TD ON)

XIO 38,\*IOCB,0,0,"Z:"

#### Poznámky:

Objeví se informační řádka (Viz povel TDLINE). Nebyl-li handler TDLINE dosud instalován, dostaneme chybu 139 - Device NAK.

Vypnutí řádky s datem a časem (TD ON)

XIO 39,\*IOCB,0,0,"Z:"

#### Poznámky:

Informační řádka zmizí. Interní čas a datum SpartaDOSu tím není ovlivněn.

### Čtení formátovaného data

XIO 34,\*IOCB,0,0,"Z:"

INPUT \*IOCB,DATE\$

#### Poznámky:

Tato sekvence načte formátované datum do proměnné DATE\$. String musí být dlouhý nejméně 13 znaků. Datum je ve formě "29-Oct-88". Další čtení způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Pokud není v okamžiku volání instalován handler, hlásí se chyba 139.

### Čtení formátovaného času

XIO 32,\*IOCB,0,0,"Z:"

INPUT \*IOCB,TIME\$

**Poznámky:**

Tato sekvence načte formátovaný čas do proměnné TIME\$. String musí být dlouhý nejméně 10 znaků. Datum je ve formě "11:59:00am". Další čtení způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Pokud není v okamžiku volání instalován handler, hlásí se chyba 139.

**Čtení neformátovaného data**

(DATE)

```
XIO 35,#IOCB,0,0,"Z:"
GET #IOCB,DAY: GET #IOCB,MONTH: GET #IOCB,YEAR
```

**Poznámky:**

Tato sekvence načte datum do proměnných DAY, MONTH, YEAR. Rok je vrácen jako poslední dvojcísel. Další čtení způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Pokud není v okamžiku volání instalován handler, hlásí se chyba 139.

**Čtení neformátovaného času**

(TIME)

```
XIO 33,#IOCB,0,0,"Z:"
GET #IOCB,HOUR: GET #IOCB,MIN: GET #IOCB,SEC
```

**Poznámky:**

Tato sekvence načte čas do proměnných HOUR, MIN, SEC. Hodina je udávána ve 24 hodinovém cyklu. Další čtení způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Pokud není v okamžiku volání instalován handler, hlásí se chyba 139.

**Nastavení data**

(DATE)

```
XIO 37,#IOCB,0,0,"Z:"
PUT #IOCB,DAY: PUT #IOCB,MONTH: PUT #IOCB,YEAR
```

**Poznámky:**

Datum z proměnných DAY, MONTH, YEAR se použije pro nastavení data v počítači. Rok je udán jako poslední dvojcísel. Další zápis způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Tato funkce je interní a nezávisí na handleru "Z:". Pokud není funkce schopna datum nastavit, hlásí se chyba 139.

**Nastavení času**

(TIME)

```
XIO 36,#IOCB,0,0,"Z:"
PUT #IOCB,HOUR: PUT #IOCB,MIN: PUT #IOCB,SEC
```

**Poznámky:**

Datum z proměnných HOUR, MIN, SEC se použije pro nastavení času v počítači. Hodina je udávána ve 24 hodinovém cyklu. Další zápis způsobuje chybu 136. Před provedením operace se musí příslušný kanál otevřít. Tato funkce je interní a nezávisí na handleru "Z:". Pokud není funkce schopna datum nastavit, hlásí se chyba 139.

## 12. Popis chybových hlášení

V této kapitole jsou popsána chybová hlášení DOS 2.5 a hlášení ostatních systémů, pokud se od DOS 2.5 liší. Chybové kódy jsou uvedeny v tabulce, seříděny podle čísla chyby. Kódy specifické pro ostatní systémy jsou uvedeny jen pokud se liší od DOS 2.5. Tabulka uvádí přehled chybových hlášení. Jejich bližší popis je v dalším odstavci.

Tabulka chybových kódů podle čísel:

číslo chyby	název	systém	
dek	hex		
3	03	Last Byte of File Read. Next read EOF	MYDOS
128	80	BREAK abort	
129	81	IOCB already open	
130	82	Nonexistent Device	
131	83	IOCB Write only	
132	84	Invalid IOCB command	
133	85	Device/File IOCB not open	
134	86	Bad IOCB number	
135	87	IOCB Read Only	
136	88	End Of File	
		Attempt to read past EOF	MYDOS
137	89	Truncated Record	
138	8A	Device Timeout	
139	8B	Device NAK	
140	8C	Serial Frame Error	
141	8D	Kurzor mimo rozsah	
142	8E	Serial Bus Overrun	
143	8F	Checksum Error	
144	90	Device Done Error	
145	91	Illegal Screen Mode	
146	92	Function Not Implemented	
147	93	Insufficient RAM	
148	94	Not SpartaDOS diskette	SpartaDOS
149	95	Diskette no SpartaDOS version 2.x	SpartaDOS
150	96	Directory Not Found	SpartaDOS
151	97	File Exists. May not replace or delete file	SpartaDOS
160	A0	Drive Number Error	
161	A1	Too Many Open Files	
162	A2	Disk Full	
163	A3	Unrecoverable System I/O Error	
		Write Protected	Happy DOS
		Illegal Wild Card in Filename	SpartaDOS
164	A4	File Number Mismatch	
		File Erase Protected	SpartaDOS
165	A5	File Name Error	

166	A6	POINT Data Length Error Invalid POINT Byte Not Within File. Invalid POINT. Position Range Error		Happy DOS MYDOS SpartaDOS
167	A7	File Locked Cannot Delete Directory		SpartaDOS
168	A8	Device Command Invalid Illegal DOS Command/Not Implemented		SpartaDOS
169	A9	Directory Full Diskette Write Locked		SpartaDOS
170	AA	File Not Found		SpartaDOS
171	AB	POINT Invalid Bad Load File IOCB Not Open		Happy DOS MYDOS
172	AC	Illegal Append Kein Happy DOS II+ /D format Cannot Create.F.or D. of same name exists in parent Directory		Happy DOS MYDOS
173	AD	Bad Sectors at Format Time		
174	AE	Datei unmöglich zurueckholen Directory Not In Parent Directory		BIBODOS MYDOS
175	AF	Directory Not Empty. Cannot Delete		MYDOS
180	B4	Invalid File Structure for Loading Memory		MYDOS
181	B5	Invalid Address Range for Loading Memory .End<Begin		MYDOS

Pokud není v kolonce systém nic uvedeno, jde o hlášení podle standardu odpovídajícího DOS 2.5.

### 12.1. Popis chybových hlášení

3	\$03	Last Byte of File Read. Next read EOF	MYDOS
		Hlášení specifické pro MYDOS. Signalizuje přečtení posledního byte před koncem souboru. Další čtení již způsobí chybu 136.	
128	\$80	BREAK abort	standard
		Přerušení operace klávesou BREAK. Jde o informativní zprávu.	
129	\$81	IOCB already open	standard
		Pokus otevření kanálu, který je již otevřen. Povel se ignoruje, to znamená že parametry otevřeného kanálu zůstanou bez změny.	
130	\$82	Nonexistent Device	standard
		Pokus o otevření kanálu nebo jiné oslovení zařízení, jehož kód není v tabulce a není nainstalován handler. Zkontrolujte kód	

zařízení v povelu, nebo nainstalujte odpovídající handler.

- |     |      |   |          |
|-----|------|---|----------|
| 131 | \$83 | IOCB Write only   | standard |
|     |      | Pokus o čtení z kanálu otevřeného pro výstup. Pokud je třeba z kanálu čist, musí být otevřen pro čtení nebo aktualizaci (read/write, UPDATE).   |          |
| 132 | \$84 | Invalid IOCB command  | standard |
|     |      | Chybný kód povelu. Číslo kódu není podporováno handlerem osloveného zařízení.   |          |
| 133 | \$85 | Device/File IOCB not open   | standard |
|     |      | Pokus o operaci s kanálem, který není otevřen. Zkontrolujte správnost otevření kanálu.  |          |
| 134 | \$86 | Bad IOCB number   | standard |
|     |      | Chybné číslo kanálu (IOCB) v povelu I/O. V Basicu smí být číslo kanálu 1 až 7, v assembleru musí být násobek 16-ti a <128.  |          |
| 135 | \$87 | IOCB Read Only  | standard |
|     |      | Pokus o zápis do kanálu, který byl otevřen jen pro čtení. Je třeba zkontrolovat příkaz OPEN. Kanál musí být otevřen pro zápis nebo aktualizaci.   |          |
| 136 | \$88 | End Of File   | standard |
|     |      | Soubor již neobsahuje žádná data, která by bylo možno číst. Je třeba obsloužit konec vstupního souboru. SpartaDOS hlásí tuto chybu i při zápisu do zařízení "Z:" při nastavování času.        |          |
| --  |      | Attempt to read past EOF  | MYDOS    |
|     |      | MYDOS jako jediný systém varuje kódem 3 při přečtení posledních dat, že bude následovat konec. Význam chyby 136 je však standardní.   |          |
| 137 | \$89 | Truncated Record  | standard |
|     |      | Typická chyba při čtení delší věty, než je maximální délka věty pro CIO. Pokus použít čtení orientované na INPUT pro čtení dat zapsaných pomocí PUT.  |          |
| 138 | \$8A | Device Timeout  | standard |
|     |      | Zařízení, které bylo osloveno po sériové sběrnici neodpovědělo do doby, určené parametrem CIO. Může být způsobeno buď chybným kódem zařízení, nebo tím, že je zařízení vypnuto nebo odpojeno. |          |

139	\$8B	Device NAK	standard
		Zařízení nemohlo odpovědět díky chybným parametrům, nebo přijetí nesprávných dat z počítače.	
140	\$8C	Serial Frame Error	standard
		Bit 7 registru SKSTAT v obvodu POKEY je nastaven. Znamená to selhání sériové komunikace. Tato chyba se vyskytuje velmi zřídka a pravděpodobně znamená poruchu počítače. U kazetového magnetofonu zkuste zkontrolovat kably.	
141	\$8D	Kurzor mimo rozsah	standard
		Souřadnice pro kurzor vybočují z hranic stanovených pro daný grafický režim.	
142	\$8E	Serial Bus Overrun	standard
		Je nastaven bit 5 SKSTAT v obvodu POKEY. Znamená to, že počítač nebyl schopen dostatečně rychle odpovědět zařízení na sériové sběrnici. Pokud se chyba vyskytne více, než jednou, jde pravděpodobně o poruchu počítače.	
143	\$8F	Checksum Error	standard
		Nesouhlasí vypočtený kontrolní součet s přijatým. Buď chybě nastavená periferie, nebo vadné médium.	
144	\$90	Device Done Error	standard
		Zařízení není schopno vykonat platný povel. Buď jde o zápis na disketu chráněnou nálepkou, nebo o čtení vadného sektoru, nebo formátu, který není jednotka schopna zpracovat (ED na 810, DD na 1050 apod.). Také se vyskytuje při pokusech kopírovat firemní diskety s ochranou proti zápisu.	
145	\$91	Illegal Screen Mode	standard
		Chyběné číslo grafického modu při otevírání obrazovky. Zkontrolujte AUX2.	
146	\$92	Function Not Implemented	standard
		Handler zařízení nevykonává požadovaný povel. Může jít třeba o pokus zapisovat na klávesnici, nebo číst data z tiskárny.	
147	\$93	Insufficient RAM	standard
		Příliš malá paměť pro zvolený grafický modus. Je třeba zvolit jiný.	
148	\$94	Not SpartaDOS diskette	SpartaDOS
		Pokus provést operaci platnou jen na disketách formátu SpartaDOSu na disketě jiného formátu.	

149	\$95	Diskette no SpartaDOS version 2.x Pro operaci je požadována disketa formátu 2.x	SpartaDOS
150	\$96	Directory Not Found Adresář nenalezen	SpartaDOS
151	\$97	File Exists. May not replace or delete file Stává se při obnovování zrušených souborů (UNERASE) nebo při zápisu souboru stejného jména, jako má některý podadresář v příslušném adresáři.	SpartaDOS
160	\$A0	Drive Number Error Chybné číslo diskové jednotky	standard
161	\$A1	Too Many Open Files Příliš mnoho souborů otevřených současně. Nedostatečný počet sektorových bufferů	standard
162	\$A2	Disk Full Na disketu již není žádný volný sektor pro zapsání dat. Je třeba použít jinou disketu, na které ještě místo je.	standard
163	\$A3	Unrecoverable System I/O Error Chyba File Manageru. DOS nebo disketa jsou vadné.	DOS 2.5
--		Write Protected Pokus o zápis na disketu blokován proti zápisu.	Happy DOS
--		Illegal Wild Card in Filename Chybné použití filtru.	SpartaDOS
164	\$A4	File Number Mismatch Číslo souboru v sektoru nesouhlasí s pořadím v adresáři.	standard
--		File Erase Protected Soubor blokován proti zápisu	SpartaDOS
165	\$A5	File Name Error Chybné jméno (obsahuje neplatné znaky).	standard
166	\$A6	POINT Data Length Error Počet byte u příkazu POINT je větší, než 125.	DOS 2.5
--		Invalid POINT Počet byte u příkazu POINT příliš velký, nebo pokus o POINT u souboru otevřeného pro výstup.	Happy DOS

--	Byte Not Within File. Invalid POINT.	MYDOS
	Byte není v souboru. Chybný POINT.	
--	Position Range Error	SpartaDOS
	Chybná pozice v souboru (POINT).	
167 \$A7	File Locked	standard
	Pokus o jiný přístup než čtení u blokovaného souboru.	
--	Cannot Delete Directory	SpartaDOS
	Pokus o zrušení adresáře, který není prázdný, nebo o zrušení povelom ERASE.	
168 \$A8	Device Command Invalid	standard
	Pokus vykonat neplatný povel. Hlášení pochází od handluera.	
--	Illegal DOS Command/Not Implemented	SpartaDOS
	Neplatný nebo neimplementovaný povel DOSu. Může být způsobeno nenačtením externího handluera.	
169 \$A9	Directory Full	standard
	V adresáři již není místo na informace o dalším souboru. Tato chyba může nastat pokud zapisujeme velké množství krátkých souborů, zejména u BiboDOSu pokud pracujeme v oboustranné dvojitě hustotě. U MYDOSu lze tuto chybu fejít založením podadresáře, ale je to nutno udělat dříve než k této chybě dojde. SpartaDOS díky pružné struktuře tuto chybu nezná.	
--	Diskette Write Locked	SpartaDOS
	Pokus o zápis na softwarově blokovanou disketu.	
170 \$AA	File Not Found	standard
	Soubor otevřívaný pro čtení nebo aktualizaci nebyl nalezen. Většinou jde o chybně napsané jméno, nebo požadovaný soubor není na disketě.	
171 \$AB	POINT Invalid	standard
	Pokus o POINT na souboru, který není otevřen pro aktualizaci (AUX1-12)	
--	Bad Load File	Happy DOS
	Povelom XIO 39 jsou čtena data, která nemají formát binárního segmentovaného souboru	
--	IOCB Not Open	MYDOS
	Kanál není otevřen.	

172	\$AC	Illegal Append	standard
Pokus o připojování dat k souboru ve formátu DOS 1.			
--		Kein Happy DOS II+/D format	Happy DOS
Povel JOB uplatněný na diskuetu DOS 2.5, nebo v jednotce 1 ne- ní disketa.			
--		Cannot Create.F.or D. of same name exists in parent Directory	MYDOS
Nelze založit soubor nebo adresář, protože v nadřízeném adre- sáři existuje soubor nebo podadresář stejného jména.			
173	\$AD	Bad Sectors at Format Time	standard
Vadná disketa nebo zaledený otvor pro povolení zápisu.			
174	\$AE	Datei unmöglich zurückholen	BIBODOS
Nelze obnovit zrušený soubor.			
--		Directory Not In Parent Directory	MYDOS
Adresář není v nadřazeném adresáři.			
175	\$AF	Directory Not Empty, Cannot Delete	MYDOS
Nelze zrušit podadresář, protože není prázdný.			
180	\$B4	Invalid File Structure for Loading Memory	MYDOS
Neplatná struktura souboru pro zavádění paměti (viz L a N)			
181	\$B5	Invalid Address Range for Loading Memory .End<Begin	MYDOS
Soubor pro zavádění do paměti má počáteční adresu větší, než koncovou.			

### **13. Slovník základních pojmu**

Nejprve je uvedeno vysvětlení některých pojmu vyskytujících se v příručce. Potom následuje krátký slovník anglicko a německo český.

#### **13.1. Významový slovník**

##### **ADRESA**

Číslo označující byte v paměti. Při programování se obvykle používají hexadecimální čísla označená znakem dolar. Mohou mít rozsah \$0 až \$FFFF, dekadicky 0 až 65535.

##### **ADRESÁŘ**

Seznam souborů na disketě a informace o jejich délce a umístění. Hlavní adresář je umístěn ve zvláštní oblasti diskety.

##### **APPEND**

Režim, při kterém jsou nová data připojována na konec souboru.

##### **ASCII**

Zkratka pro American Standard Code for Information Interchange, česky americký standardní kód pro výměnu informací. Kod zahrnuje řídicí znaky, číslice, zvláštní znaky a písmena. Hodnoty 0 až 127 se označují jako standardní ASCII, 0 až 255 obsahuje rozšířený kód ASCII.

##### **ATASCII**

Varianta kódu ASCII používaná u osmibitových Atari. Místo řídicích kódů má grafické znaky a pro ukončení řádky používá znak EOL (\$9B, 155).

##### **AUTORUN.SYS**

Soubor s tímto vyhrazeným jménem je DOSem automaticky spuštěn.

##### **BANK**

Blok paměti předem určený velikostí a umístěním. Frepinání znamená výměnu různých banků paměti ve stejném adresovém prostoru. Tuto technikou lze u osmibitového počítače používat paměť větší, než 64 KB.

##### **BINÁRNÍ**

Číslo v dvojkové soustavě. Ve spojitosti souborem znamená obecný soubor, ale většinou se tak označují programy ve strojovém kódu.

##### **BIT**

Dvojková číslice, může nabývat hodnoty 0 nebo 1. Název je zkratka anglického Binary digit.

##### **BOOT**

Inicializace počítače po jeho zapnutí. Její součástí je zavedení primárního programového vybavení z diskety nebo kazety. Bootem je zaváděn i DOS.

##### **BREAK**

Klávesa, kterou je v mnoha případech možno zrušit probíhající operace, nebo volby z menu. Podobnou funkci mívá také klávesa ESC.

##### **BUFFER**

Vyrovnávací paměť pro dočasné uložení dat, obvykle při V/V operacích.

##### **BYTE (BAJT)**

Osmibitové číslo v rozsahu hodnot 0 až 255 (0 až \$FF). Může reprezentovat jeden znak.

**CESTA**

Popis způsobu, jak se dostaneme od hlavního adresáře k označenemu souboru.

**CIO**

Zkratka Central Input Output system - centrální vstupní a výstupní systém, který je nezávislý na zařízení. Jde o práci na úrovni souboru.

**ČÍSLO JEDNOTKY**

Číslo disketové jednotky.

**CLOSE**

Uzavření souboru, resp. V/V kanálu. Je nutné pro zaregistrování výstupního souboru.

**CP**

Příkazový nebo povelový procesor.

**CPU**

Jádro počítače vykonávající vejceré instrukce strojového kód u. U Atari je typu 6502C.

**CRC**

Cyclic Redundancy Check - kontrola správnosti uložení nebo přečtení dat metodou kontrolního součtu.

**DATA**

Informace, uložené na disketách.

**DÁVKA**

Způsob zpracování, kdy jsou povely pro počítač uloženy v souboru a poslupně za sebou plněny. Uživatel nemůže do zpracování zasahovat dokud dávka neskončí.

**DD**

Dvojitá hustota (Double Density). Takto formátovaná disketa má kapacitu 180 KB, 40 stop a 18 sektorů na stopu, délka sektoru 256 byte.

**DISKETA**

Pružný disk, floppy disk.

**DMA**

Přímý přístup do paměti bez použití CPU.

**DOS**

Diskový operační systém.

**DOS.SYS**

Soubor ve kterém je uložena rezidentní část diskového operačního systému.

**DRIVER**

Obslužný program periferního zařízení. Též handler.

**DS/DD**

Obostranná dvojitá hustota (Double Sided/Double Density). Takto formátovaná disketa má kapacitu 360 KB, 2 x 40 stop a 18 sektorů na stopu, délka sektoru 256 byte.

**ED**

Rozšířená hustota (Enhanced Density). Takto formátovaná disketa má kapacitu 127 KB, 40 stop a 26 sektorů na stopu, délka sektoru 128 byte.

**EOF**

Znak konce souboru (\$9B, 155).

**ESC**

Klávesa, kterou se u některých systémů ruší probíhající operace.

**FILTR**

Způsob označení skupiny souborů, při kterém je libovolný počet libovolných znaků označen hvězdičkou, jeden libovolný znak otazníkem.

**HANDLER**

Obslužný program periferního zařízení. Též driver.

**HEXADECIMÁLNÍ**

Číslo v šestnáctkové číselné soustavě, označuje se znakem \$.

**INDEXOVÁNÍ**

Metoda při které se čísla vět a hodnoty klíče ukládají do indexových souboru. Podle pořadových čísel vět se potom v souboru vyhledává.

**I/O**

Zkratka pro operace vstupu a výstupu.

**HUSTOTA**

Hustota záznamu diskety může být jednoduchá - SD, rozšířená - ED, dvojitá DD a oboustranná dvojitá DS/DD nebo Quad.

**INPUT**

Povel Basicu, kterým se čtou ze souboru jednotlivé věty.

**niocb**

Číslo vstupního/výstupního kanálu.

**IOCB**

Řídící blok vstupu a výstupu. Obsahuje informace potřebné pro otevření, čtení a zápis souboru. Těchto blokuje 8 a každý řídí jeden vstupní/výstupní kanál.

**JMÉNO SOUBORU**

Řetězec osmi ASCII znaků začínajících zpravidla písmenem. Často se doplňuje koncovkou a označuje soubor dat.

**KILOBYTE**

1024 bytu

**KONCOVKA**

Tříznaková popisná část jména, označující typ souboru.

**KURZOR**

Pozice na obrazovce, na kterou se umístí příští znak. Je označen čtverečkem.

**LADĚNÍ**

Odstrojení chyb v programech.

**NULOVÝ ŘETĚZEC**

Prázdný string neobsahující žádné znaky. V Basicu se vytvoří výrazem: A\$-".

**OCHRANA PROTI ZÁPISU**

Metoda znemožnění zápisu na disketu, dělá se zlepšením výřezu po straně diskety.

**OKTALOVÝ**

Číslo v osmičkové soustavě.

**OMEZOVAČ**

Znak, který odděluje části výrazu.

**OPEN**

Operace, která připravuje zpracování souboru dat a spojuje jej s určitým kanálem vstupu a výstupu.

**OS-ROM**

ROM ve které je uložen operační systém počítače Atari. Leží v rozsahu \$C000 až \$FFFF.

**OUTPUT**

Povel pro zápis dat.

**PODADRESÁŘ**

Adresář podřízený hlavnímu adresáři. V něm je uložen jako soubor a obsahuje informace o jemu podřízených souborech.

**PARAMETR**

Hodnota, která se dosazuje do hotových programů a řídí jejich činnost.

**PERIFERIE**

Vnější vstupní nebo výstupní zařízení počítače.

**PORTB**

Hardwareový registr který řídí zapínání vestavěných ROM OS, Basicu a Self-testu. U 130XE řídí ještě přepínání banků přídavné paměti.

**POVEL**

Požadavek uživatele na počítač, aby vykonal nějakou akci.

**PŘENOSOVÁ RYCHLOST**

Rychlosť přenosu po sériové sběrnici v bitech za sekundu.

**PROMĚNNÁ**

Místo v paměti, do kterého se ukládá nějaká hodnota. Je označováno jménem proměnné.

**QUAD**

Oboustranná dvojitá hustota DS/DD.

**RAM**

Vnitřní paměť počítače, do které lze zapisovat i z ní číst.

**RAM-disk**

Úsek RAM obsluhovaný zvláštním handlerem tak, aby se vzhledem k okolí jevil jako disketová jednotka. Je velmi rychlý.

**RELOKATIBILNÍ**

Tvar strojového programu, při němž nezáleží na konkrétním umístění v paměti.

**ŘETĚZEC**

Řetězec znaku kódu ATASCII, obvykle uzavřený do úvozovek.

**ROM**

Read-Only-Memory. Paměť kterou lze pouze číst.

**RS-232**

Celosvětový standard pro sériovou komunikaci. Oznacuje se též V24.

**SD**

Jednoduchá hustota (Enhanced Density). Takto formátovaná disketa má kapacitu 127 KB, 40 stop a 18 sektorů na stopu, délku sektoru 88 byte.

**SEKTOR**

Nejmenší jednotka dat, která může být přečtena z diskety, nebo na ni zapsána. Sektor má délku 128 (SD, ED) nebo 256 (DD, DS/DD) byte.

**SEKVENČNÍ PŘÍSTUP**

Postupný přístup k datum, kdy se čte jedna věta za druhou.

**SIO**

Rutina pro sériovou komunikaci s periferiem na nejnižší úrovni. Je součástí OS-ROM.

**SOUBOR**

Organizovaná ucelená jednotka dat, označená jménem. Dále se dělí na věty.

**STANDARD**

Standardní hodnoty parametrů, které se použijí pokud uživatel žádné neurčil.

**STOPA**

Záznamová kružnice na disketu, do které se ukládají data. Dělí se dále na sektory. U SD a DD je sektoru 18, u ED 26.

**STROJOVÝ KÓD**

Soustava dat představující instrukce procesoru počítače a k nim náležející parametry. Obvykle se tak označují programová celky které mohou obsahovat i jiná data, jako display-list apod.

**STUDENÝ START**

Základní inicializace při zapnutí počítače.

**SYNTAXE**

Pravidla pro tvorbu příkazu a povelu.

**SYSTÉMOVÁ DISKETA**

Disketa, na které je instalován DOS

**TEPLÝ START**

Malá inicializace probíhající po stisku klávesy RESET. Za určitých podmínek může přejít ve studený start.

**TOKENIZACE**

Proces přeměny textu na tokeny. Název každého povelu a proměnné se nahradí číslem.

**UKAZATEL**

Proměnná obsahující adresu objektu, na který ukazuje.

**ULOŽENÍ STROJOVÉHO PROGRAMU**

Uložení obsahu části paměti do souboru. Obvykle se jedná o program ve strojovém kódu, ale mohou to být i jiná data.

**VĚTA**

Blok dat ukončený znakem EOL (\$9B, 155).

**V/V**

Zkratka pro operace vstupu a výstupu.

**XIO**

Povel Basicu umožňující provádět rozšířené povely vstupu a výstupu.

U disketových jednotek umožnuje provádět některé operace se soubory.

**ZAŘÍZENÍ**

Logický nebo fyzický přístroj na vstup nebo výstup dat. U Atari se označuje písmenem, za kterým může být číslo zařízení a musí následovat dvojtečka.

**ZAVEDENÍ STROJOVÉHO PROGRAMU**

Zavedení obsahu binárního souboru do paměti počítače. Obvykle se jedná

o program ve strojovém kódu, ale mohou to být i jiná data. Program je většinou po načtení automaticky spuštěn.

#### ZDROJ

Jednotka nebo adresa obsahující data, která mají být poslána na cílovou jednotku nebo adresu.

#### ZDROJOVÝ KÓD

Text programu, ze kterého vznikne překladem strojový kód.

### 13. 2. Slovník anglicko - český

ADDRESS	adresa
APPEND	připojení dat na konec souboru
BATCH (processing)	dávka povělů, dávkové zpracování
BAUD RATE	přenosová rychlosť
BINARY	binární
BINARY LOAD	zavedení strojového programu
BINARY SAVE	uložení strojového programu
COLD START	studený start
COMMAND	povел, příkaz
COPY	kopirovat
CREATE	založit (vytvorit)
CURSOR	kurzor
DEBUG	ladění, odstraňování chyb v programech
DECIMAL	dekadický
DEFAULT	standardní hodnota
DELETE	zrušit
DELIMITER	omezovač
DENSITY	husťota, denzita
DESTINATION	cílový
DEVICE	zařízení
DIRECT ACCESS	přímý přístup
DIRECTORY	adresář
DISKETTE, DISK	disketa
DRIVE NUMBER	číslo (disketové) jednotky
DRIVE SPECIFICATION	specifikace (disketové) jednotky
DUPLICATE	duplikovat
ERASE	zrušit, vymazat
EXTENDER	koncovka
FILE	soubor
FILENAME	jméno souboru
FILESPEC	specifikace souboru
FORMAT	formátování
HEXADECIMAL	hexadecimální
INDEXING	indexování
LOCK	zamknout - blokovat proti zápisu.
MACHINE LANGUAGE	strojový kód
NULL STRING	nulový (prázdný) řetězec

OBJECT CODE	strojový kód
OCTAL	oktalový
PARAMETER	parametr
PATH	cesta
PERIPHERAL	periferie
POINTER	ukazatel, směrnik
PROTECT	chránit proti zápisu (zamknout, blokovat)
RECORD	věta
RELOCATIBLE	relokatibilní
RENAME	přejmenovat
SECTOR	sektor
SEQUENTIAL	sekvenční přístup
SOURCE	zdroj
SOURCE CODE	zdrojový kód (text)
STRING	řetězec
SUBDIRECTORY	podadresář
SYNTAX	syntaxe
TOKENIZING	tokenizace
TRACK	stopa
UNDELETE	obnovit, zrestaurovat soubor
UNLOCK	odemknout - uvolnit
UNPROTECT	odemknout - uvolnit
VARIABLE	proměnná
WARM START	teplový start
WILDCARD	filtr
WRITE	zapisovat
WRITE-PROTECT	ochrana proti zápisu

### 13. 3. Slovník německo český

ADRESSE	adresa
BENENNEN	přejmenovat
BINAER	binární
BINAER LOAD	zavedení strojového programu
BINAER SAVE	uložení strojového programu
CURSOR	kurzor
DATEI	soubor
DATEINAME	jméno souboru
DATEI SPEZIFIKATION	specifikace souboru
DATEN	data
DEZIMAL	dekadický
DICHTE	hustota (záznamu)
DIREKTZUGRIFF	přímý přístup
DISKETTE	disketa
DISKNUMMER	číslo jednotky
DRIVE SPEZIFIKATION	specifikace jednotky
DUPLIZIEREN	duplicovat

FORMATIEREN	formátování
FREIGEBEN	odemknout, uvolnit (soubor)
FREIGEBEN	odemknout, uvolnit
GERÄT	zafizení
HEXADEZIMAL	hexadecimální
INDEXIERUNG	indexování
KALTSTART	studený start
KOMMANDO	povel, příkaz
KOPIEREN	kopirovat
LÖSCHEN	zrušit
MASCHINENSPRACHE	strojový kód
NULLSTRING	nulový (prázdný) řetězec
OKTAL	oktalový
ORIGINAL (SOURCE)	zdroj
PARAMETER	parametr
PERIPHERIE	periferie
PUFFER	buffer
RELOZIERBAR	relokatibilní
SATZ (DETANSATZ)	věta
SCHREIBEN	zapisovat
SCHREIBSHUTZ	ochrana proti zápisu
SICHERN	zamknout, zablokovat
SICHERN	blokovat
SPUR	stopa
SYSTEMDISK	systémová disketa
TOKENISATION	tokenizace
TREIBER	obslužný program, driver
ÜBERTRAGUNGSGESCHWINDIGKEIT	přenosová rychlosť
UNTERVERZEICHNIS	podadresář
VERZEICHNIS	adresář
WARMSTART	tepílý start
ZIEL	cílový
ZURÜCKHOLEN	obnovit (soubor)

## **Seznam literatury**

### **Použitá literatura:**

- [1] Atari Tutorial, BYTE Magazin, 1985
- [2] Bee, P.. Reuß, E.: Inside Bibo-DOS, část 1 až 5, Compy-Shop, Compy-shop magazin na disketách, 1988
- [3] Dorndorf,S.: Happy DOS II - /D - Listing des Monats, Markt & Technik, časopis Happy Computer 3/86, str. 91-98
- [4] DOS 2.5: 1050 DISK DRIVE, Atari Corp, 1985
- [5] DOS 2.5: XF-551 DISK DRIVE, Atari Corp, 1988
- [6] Eichler, L., Grohmann, B.: Atari Intern, DATA BECKER GmbH, 1984
- [7] Kratochvílová,J., Minihofe, O.: Anglicko-český slovník výpočetní techniky, SNTL, 1986
- [8] Lawrow, S. D.: MAC/65, OSS, Inc., 1984
- [9] Marslett,CH., W.: MYDOS Version 4 User Guide, Revision 4.1, WORDMARK Systems, 1985
- [10] Morrison,G.: Advanced Atari Protection Techniques, Alpha Systems, 1986
- [11] Morrison,G.: Atari Software Protection Techniques, Alpha Systems, 1983
- [12] Operating System User's Manual, Atari, inc., 1982
- [13] RAMBO XL. Installation & Operations Manual, ICD, Inc., 1986
- [14] Reschke,J., Wiethoff,A.: Das ATARI Profibuch, SYBEX-Verlag GmbH, 1985
- [15] Reuß,E.: Die Formate der XF-551, Compy-Shop, Compy-Shop Magazin na disketách, 1988
- [16] Šatánek, A.: Mikropočítačový lexikon, SPN Praha, 1987
- [17] The Sparta DOS Construction Set, ICD, Inc., 1985
- [18] The Sparta DOS Construction Set, Supplement to SDCS Owners manual, ICD, Inc., 1985
- [19] Wilkinson, B.: Atari's Newest Drive, COMPUTE!, 12/87, str. 71

### **Doporučená literatura domácí:**

- [20] Bayer, Miloš a kol.: Důležité adresy systému a jejich použití při programování v Basicu, Atari klub 487, ZO Svazarmu, 1987
- [21] Chasin, M: Programování v assembleru pro počítače Atari, Atari klub 487, ZO Svazarmu Praha, 1988, překlad K. Šmuk
- [22] Fajta, V.: Základy programování mikroprocesoru 6502, Atari klub 487, ZO Svazarmu, 1988
- [23] Kodera,J.: Komentovaný výpis ROM operačního systému 800 XL, Atari klub 487, ZO Svazarmu, 1988
- [24] Lawrow, S. D.: MAC/65, překlad P. Jandík, Atari klub 487, ZO Svazarmu, 1988

### **Doporučená literatura zahraniční:**

- [25] De Re Atari, Atari, Inc.
- [26] Chadwick, Ian: Mapping the Atari, Compute Books
- [27] Peeks & Pokes, Data Becker 1985

## Rejstřík

Adresa	5, 14, 28, 32-35, 39, 49-52, 54-58, 67, 70, 73, 75-77, 85, 90-94, 96-100, 102, 104, 106, 109-111, 121, 123-127, 139-141, 145-148, 164, 182, 185-189, 195, 197-205
Adresát	15, 23, 27, 28, 36-38, 40-42, 46, 47, 64, 72, 78-81, 85-90, 94, 97-99, 107, 109, 115, 116, 119, 120, 122, 129-136, 142-144, 146-148, 153-158, 162-167, 170, 171, 174-178, 182, 192, 196, 198, 202-204, 211-213
Append	43, 44, 64, 97, 145, 148, 187, 192, 195
ASCII	182, 187, 188
Assembler	25, 32, 34, 45, 50, 52, 54, 57, 73, 74, 78, 85, 92, 98, 101, 102, 113, 127, 148, 189, 209
Atari - 1050	5, 7, 9-12, 17, 23-25, 40, 49, 98, 114, 115, 119, 120, 122, 125, 130, 141, 142, 149, 154, 159, 160, 185, 190, 210
- 130XE	14, 16, 17, 24, 40, 43, 55, 56, 76, 77, 82, 85, 105, 118, 119, 124, 140, 159, 162
- 850	8, 113, 178,
- XC 12	8
- XP 551	5-7, 9-11, 17, 24, 25, 42, 49, 105, 114, 115, 119, 120, 122, 125, 127, 130, 135, 141, 142, 148, 159, 160, 185
ATA SCSI	38, 187, 188
AUTORUN.SYS	29, 45, 46, 73, 74, 83, 84, 96, 105, 106, 148, 160, 189
Bank	77, 119, 124, 140, 162
Basic	2, 11-13, 20, 26-28, 40, 43, 45, 55, 56, 57, 60-63, 65, 69, 71, 73-75, 78, 83-85, 87, 88, 90, 91, 93, 95, 96, 98, 99, 102, 107, 109, 115, 116, 122, 126, 134, 140, 141, 144, 145, 159, 184, 189, 205
Binární - čísla	137
-data	57, 73
-formát	49
-segmentovaný soubor	29, 38, 39, 44, 50-52, 54, 70, 73, 91, 94, 97, 100, 106, 111, 113, 125, 139, 146, 174, 186-189, 197, 212
-soubor	57, 91, 100, 110, 120, 121, 185, 195, 199
Bit	32, 35, 37, 38, 96, 120, 123, 124, 144-147, 180, 200-203, 210
Boot	12, 28, 32-34, 83, 161, 201-203
-BOOT?	73, 96
-ERROR	12, 29
-formát	29, 31, 35, 42, 49
Buffer	32, 76, 93, 114, 119, 123, 125, 129, 145, 147, 156, 183, 184, 190, 199, 200-202, 204, 205, 211
Busy	7, 11, 18
CIO	27, 28, 122, 125-127, 134, 144, 178, 204, 209
CLOSE	63, 65, 73, 90, 95, 102, 145
COPY	13, 22, 42-44, 52, 53, 59, 78, 79, 86, 88, 101, 107, 111, 112, 156, 158, 165, 166, 175, 180, 182, 186
Create	14
Databanks	15, 26
DELETE	13, 23, 122, 146

Delete Back-Space	15
DOS SYS	15, 21, 27, 29, 35, 45, 48, 56, 57, 83, 84, 90, 101, 105, 106, 112, 114-116, 147
DUP SYS	14, 15, 21, 27, 35, 43, 45, 48, 56-58, 76, 77
Erase	102
ERROR	12, 28, 29, 42, 46, 71, 97, 120, 190
Filter	20, 22, 23, 40, 41, 43, 45-48, 50, 61, 64, 85-88, 99, 108, 109, 110, 115-117, 135, 136, 146, 155, 165, 166, 169, 170, 178, 193, 196, 211
Format	28, 29, 31, 34-37, 42, 49, 52, 60, 65, 78, 79, 86, 94, 97, 100, 111, 114, 120, 121, 125, 127-129, 130, 134, 136, 137, 145, 147, 148, 151, 152, 158-161, 166-168, 184, 185, 193-196, 202, 210-213
Formatování	10, 13-17, 21, 24, 41, 42, 47, 49, 59, 85, 90, 94, 97, 101, 112, 113, 119, 122, 124, 126, 129, 135, 137, 138, 139, 142, 147-150, 152, 154, 159-162, 168, 170, 171, 174, 203
GET	65, 70, 73, 102, 145
Hustota	12, 14, 17, 18, 23-25, 35-38, 41, 42, 49, 59, 64, 68, 72, 85-87, 90, 98, 101, 105, 108, 109, 112-116, 119-123, 127-130, 134, 135, 137-139, 142, 145, 147, 149, 152, 160, 161, 166, 167, 169, 190, 191, 199, 202, 212
INIT	49-55, 74, 101, 108, 112, 152, 159, 171, 174, 181, 189, 195
INPUT	65, 66, 145, 209
Interpret	27, 63, 65, 153, 154, 157, 199
Interrupt	125
Jmeno souboru	14, 19, 20-23, 38, 40, 44, 47, 50, 54, 58, 60, 63, 73, 78, 80, 81, 83, 88, 90, 91, 99, 110, 116, 121, 132, 135, 145, 147, 154, 180, 203
Kazeta	5, 8, 28, 29, 32, 60, 61, 65, 88, 96, 108, 111, 114, 116, 117, 169, 210
Koncovka	15, 19, 38, 40, 42, 44, 45, 59, 78, 83, 84, 100, 126, 136, 154, 155, 157, 159, 180, 185, 203, 213
Kontrolka	7, 11, 18
Lock	14, 72, 89, 90, 146, 172, 193
MEM SAV	43, 44, 52, 55-59, 77
MEM LO	32, 57, 75, 76, 105-107, 114, 115, 134, 172, 174, 184, 189, 199
MEM TOP	75
Menu	12, 13, 15, 16, 18-22, 27, 29, 40, 43, 52, 55-57, 62, 71, 73, 76-79, 81-83, 85, 87, 101, 105-109, 112, 114-116, 118, 120-122, 134, 135, 137, 146-149, 172-175, 201
OPEN	63, 73, 90, 95, 102, 127, 144, 145, 192, 209
Periferie	8, 26, 28, 62, 72, 73, 96, 210
Podadresář	24, 130-133, 125-137, 142-144, 145-148, 151-154, 162-165, 167, 192, 193, 195, 196, 203, 211, 213
PORTB	119, 124
Procesor	27, 32, 87, 92, 100, 105, 106, 112, 154, 156, 157, 190, 201
PUT	70, 73, 102, 145, 148, 209
RAM-disk	14, 15, 40, 42, 43, 56, 64, 71, 76-78, 80, 85, 96, 113, 114, 116, 118, 119, 122-125, 129-131, 134, 138, 140, 141, 144, 148, 149, 151, 161, 162, 165, 168, 173, 181, 190
READY	11, 12, 20, 21, 28, 43, 56
Rename	14, 72, 97, 102, 109, 145, 170, 193

RUN	13, 14, 20, 49, 50-55, 57, 61, 62, 74, 83, 88, 91, 96, 97, 100, 102, 109, 111, 121, 174, 184, 189, 195, 199
Sběrnice	7, 8, 23, 149, 189, 209, 210
Sektor	15, 17, 24, 27-29, 31, 32, 34-38, 40, 41, 44, 49, 55, 56, 65, 67-69, 72, 73, 75, 77, 78, 80-82, 85-87, 90, 93, 94, 112, 114, 120, 123-127, 130, 136, 138, 142, 145, 147, 149, 153, 154, 160, 162, 168, 170, 184, 189, 191, 194, 197, 198, 201-204, 210, 211
Seftest	12
Status	37, 38, 65, 71, 80, 93, 94, 130, 189, 203, 204
Stopa	17, 24, 29, 120, 127, 139, 142, 151, 159, 166, 167, 190, 202
VTOC	36, 58, 80, 81, 86, 119, 120, 137
Výpis	12, 13, 15, 19, 21, 23, 29, 37, 38, 40-42, 45, 48, 49, 62, 79, 80, 81, 87, 88, 92, 99, 103, 107, 109, 115, 116, 132, 134, 136-138, 153-158, 164, 171, 172, 176, 184, 187, 196, 201
Tiskárna	8, 9, 13, 29, 41, 42-45, 85, 98, 109, 115, 116, 134, 136, 158, 165, 182, 183, 187, 188, 210
Undelete	122
Unlock	14, 72, 89, 90, 146, 172

# Obsah

<b>1.</b>	<b>Úvod</b>	<b>5</b>
<b>2.</b>	<b>Disketová jednotka a DOS pro začátečníky</b>	<b>7</b>
2.1.	Zapojení a obsluha disketové jednotky	7
2.1.1.	Připojení disketové jednotky	7
2.1.2.	Připojení více periferních zařízení	8
2.1.3.	Nastavení čísla jednotky	9
2.1.4.	Péče o diskety	9
2.2.	Práce s DOS 2.5	11
2.2.1.	Zavedení DOSu do paměti.	11
2.2.2.	DOS s Basicem a bez něj	12
2.2.3.	Chyby při zavádění DOSu (BOOT ERROR)	12
2.2.4.	DOS menu	13
2.2.5.	Komunikace s DOSEM	14
2.2.6.	A - prohlídka obsahu diskety	15
2.2.7.	J - kopírování (duplikování) diskety	16
2.2.8.	Používání ochranných nálepek	17
2.2.9.	I - Formátování diskety	17
2.2.10.	Pojmenovávání souborů	19
2.2.11.	Práce s Basicem	20
2.2.12.	Kopírování souborů	21
2.2.13.	D - rušení souborů	23
2.3.	Typy disketových jednotek	23
<b>3.</b>	<b>Základní principy DOSu</b>	<b>26</b>
3.1.	Práce DOSu a jeho umístění v paměti.	26
3.2.	Studený start počítače	28
3.3.	Druhy a organizace diskových souborů	29
3.3.1.	Formát sekvenčního souboru (BOOT formát).	31
3.3.2.	Formát sektoru DOS (linkovaný formát)	35
3.4.	Struktura adresáře diskety	36
3.4.1.	Struktura VTOC	36
3.4.2.	Struktura věty adresáře souborů	37
3.5.	Stavba segmentovaného binárního souboru	38
<b>4.</b>	<b>DOS 2.5</b>	<b>40</b>
4.1.	Funkce menu DOS 2.5	40
4.1.1.	Výpis adresáře - DISK DIRECTORY	40
4.1.2.	Přechod do modulu - RUN CARTRIDGE	43
4.1.3.	Kopírování souborů - COPY FILE	43
4.1.4.	Rušení souborů - DELETE FILES	46
4.1.5.	Přejmenování souborů - RENAME FILE	47
4.1.6.	Blokování (zamknutí) souboru - LOCK FILE	47
4.1.7.	Uvolnění (odemknutí) souboru - UNLOCK FILE	48
4.1.8.	Instalace DOSu - WRITE DOS FILES	48

4.1.9.	Formátování - FORMAT DISK	49
4.1.10.	Duplikování disket - DUPLICATE DISK	49
4.1.11.	Ukládání úseků paměti - BINARY SAVE	49
4.1.12.	Načítání binárních souborů - BINARY LOAD	54
4.1.13.	Skok na adresu - RUN AT ADDRESS	55
4.1.14.	Založení MEM.SAV - CREATE MEM.SAV	55
4.1.15.	Duplikování souborů - DUPLICATE FILE	58
4.1.16.	Formátování SD - FORMAT SINGLE	59
4.2.	Basic a DOS 2.5	60
4.2.1.	Čtení a ukládání programů	60
4.2.2.	Řídící bloky vstupu a výstupu - IOCB	62
4.2.3.	Povely pro otvírání a zavírání souborů OPEN/CLOSE	63
4.2.4.	Čtení a zápis vět	INPUT/PRINT
4.2.5.	Přímý přístup do souboru	NOTE/POINT
4.2.6.	Vstup a výstup po jednotlivých bytech PUT/GET	70
4.2.7.	Status kanálu STATUS	71
4.2.8.	Náhrada povelů DOS menu funkcí XIO	71
4.3.	Různé informace	72
4.3.1.	Opravování poškozených souborů	72
4.3.2.	Soubor AUTORUN.SYS	73
4.3.3.	Obsazení paměti DOsem 2.5	74
4.3.4.	Zrychlení zápisu dat a volba počtu jednotek	75
4.3.5.	Funkce RAM-disku u 130 XE	77
4.4.	Služební programy DOS 2.5 (Utility)	78
4.4.1.	Program COPY32.COM	78
4.4.2.	Program DISKFIX.COM	79
4.4.3.	Program SETUP.COM	81
<b>5.</b>	<b>Happy DOS II+ /D</b>	<b>85</b>
5.1.	Zvláštní rysy Happy DOSu	86
5.1.1.	Konvence jmen a konstrukce filtru	86
5.1.2.	Řízení zápisu	86
5.1.3.	Odlišnosti VTOC a adresáře	86
5.1.4.	Zvláštní funkce kláves	87
5.2.	Příkazy pro řízení Happy DOSu	87
5.2.1.	Výpis adresáře	87
5.2.2.	Přechod do modulu	88
5.2.3.	Kopirování souborů	88
5.2.4.	Rušení souborů	89
5.2.5.	Přejmenování souboru	89
5.2.6.	Blokování a uvolňování souborů	89
5.2.7.	Instalace Happy DOSu	90
5.2.8.	Formátování diskety	90
5.2.9.	Uložení úseku paměti	90
5.2.10.	Načtení a spuštění binárního souboru	91
5.2.11.	Skok na adresu	91
5.2.12.	Změna aktuální disketové jednotky	91
5.2.13.	Monitor	92
5.2.14.	Uložení příkazového souboru	93

5.3.	Nové povely Basicu	93
5.3.1.	Nové povely XIO	93
5.3.2.	Čtení a zápis úseků paměti, status disku	94
5.4.	Různé informace	96
6.	<b>OS/A.</b>	<b>98</b>
6.1.	Interní povely OS/A.	98
6.2.	Externí povely	100
6.3.	Tvorba povelového souboru pro dávkové zpracování	102
6.4.	Vazba programu na OS/A.	104
7.	<b>DOS XL</b>	<b>105</b>
7.1.	Start DOS XL a volba varianty uložení v paměti.	105
7.2.	Menu	107
7.3.	Povelový procesor (command processor) DOS XL.	109
7.3.1.	Interní povely	109
7.3.2.	Externí povely	111
7.4.	Tvorba povelového souboru pro dávkové zpracování	113
7.5.	Vazba programu na DOS XL	113
7.6.	Možnost RAM-disku	113
8.	<b>Bibo DOS</b>	<b>114</b>
8.1.	Manu Bibo DOS	115
8.2.	Bibo DOS pro programátory	122
8.2.1.	Doplňky funkcí CIO volaných z Basicu	122
8.2.2.	Tabulka parametrů	122
8.2.3.	Tabulka vektorů	125
8.2.3.	Programování XF551	127
8.3.	Služební programy pro Bibo DOS	129
8.3.1.	Program CONV32D.COM	129
8.3.2.	Program FILECOPY.COM	129
8.4.	Kompatibilita s DOS 2.5 a přechod na dvojitou hustotu.	130
9.	<b>Stromová struktura - podadresáře</b>	<b>131</b>
9.1.	Podadresáře a jejich postavení v systému souborů	131
9.2.	Pracovní adresář	132
10.	<b>MYDOS 4.3B</b>	<b>134</b>
10.1.	Funkce menu MYDOS 4.3B	134
10.2.	Nové a změněné funkce CIO	144
10.3.	Struktura a organizace disku ve formátu MYDOS.	147
10.4.	Obsazení paměti	147
10.5.	Změny parametrů a instalace RAM-disku.	148
10.6.	Komunikace s diskem přes SIO	149

<b>11.</b>	<b>Sparta DOS verze 3.2</b>	<b>151</b>
11.1.	Všeobecný přehled SpartaDOSu	152
11.2.	Syntaxe SpartaDOSu	154
11.3.	Popis povelů SpartaDOS	158
11.3.1.	Základní povely	158
11.3.2.	Inicializace diskety	159
11.3.3.	Podadresáře	162
11.3.4.	Kopírování	165
11.3.5.	Údržba souborů a disket	169
11.3.6.	Ochrana	171
11.3.7.	Operace pomocí menu	172
11.3.8.	Datum a čas	176
11.3.9.	Podpora komunikace	178
11.3.10.	Přesměrování vstupu a výstupu (dávkový režim)	180
11.3.11.	Klávesový buffer	183
11.3.12.	Informační povely	184
11.3.12.	Práce se strojovým kódem	185
11.3.13.	Práce s diskem u SpartaDOSu	189
11.4.	SpartaDOS pro programátory	192
11.4.1.	Funkce SpartaDOSu volané z Basicu	192
11.4.2.	Vektorové tabulky SpartaDOSu	197
11.4.3.	Formát disket SpartaDOSu	201
11.4.4.	Různé poznámky a informace	204
11.4.5.	Funkce handleru "Z:"	205
<b>12.</b>	<b>Chybová hlášení</b>	<b>207</b>
12.1.	Popis chybových hlášení	208
<b>13.</b>	<b>Slovnik základních pojmu</b>	<b>214</b>
13.1.	Významový slovník	214
13.2.	Slovnik anglicko - český	219
13.3.	Slovnik německo český	220
<b>Seznam literatury</b>		<b>222</b>
<b>Rejstřík</b>		<b>223</b>



## Poznámky:

**Publikované zo súhlasom - vid' Prohlášení představitelů AK Praha.**