K A P I T O L A

3

P O U Ž I T Í M O D U C 64

ČÁST 9

Používání BASICu 2.0

Celý jazyk Commodoru 64 BASIC 2.0 je zahrnut do jazyky BASIC 7.0 Commodoru 128. Příkazy BASICu 2.0 lze používat jak v modu C 128, tak v modu C 64. Další podrobnosti o těchto příkazech naleznete v částech 3. a 4. Kapitoly 2.

Používání znaků z klávesnice

Na zobrazení klávesnice, které je uvedeno v části 3. vystínovaná tlačítka jsou ta, která se používají v modu C 64. V tomto modu má klávesnice tytéž dva soubory znaků, jako Commodore 128:

- velká písmena/grafika
- velká/malá písmena,

Po aktivování modu C 64 je klávesnice v modu s velkými písmeny/grafikou, takže všechna tištěná písmena budou velká. V modu C 64 je možné používat současně pouze jednu sadu znaků. Aby se přešlo od jedné sady k druhé, stiskněte tlačítko SHIFT současně s tlačítkem Cx (tlačítko COMMODORE).

Používání tlačítek jako psacího stroje

Stejně jako v modu C 128, i v modu C 64 lze používat tlačítka jako psací stroj s velkými a malými písmeny. Mimoto je možné

tisknout i číselná tlačítka v řadě tlačítek nahoře na základní klávesnici a tisknout grafické znaky uvedené na přední straně tlačítek.

Používání příkazových tlačítek

Většina příkazových tlačítek (t.j. tlačítek, která posílají počítači celá hesla, například RETURN, SHIFT, CTRL atd), pracují v modu C 64 stejně jako v modu C 128. Jediný rozdíl je, že v modu C 64 kursor může být přemístěn pouze pomocí dvou tlačítek CRSR v pravém dolním rohu základní klávesnice (v modu C 128 je kromě toho možné použít tlačítka se šipkami umístěná vpravo nahoře na hlavní klávesnici).

Posouvání kursoru v modu C 64

V modu C 64 se pro posouvání kursoru používají tlačítka CRSR na hlavní klávesnici a tlačítko SHIFT, jak to bylo popsáno v Části 3.

Programování funkčních tlačítek v modu C 64

Čtyři tlačítka umístěná na pravé straně klávesnice, nad číselnou klávesničkou, se nazývají funkční tlačítka a jsou označena F1, F3, F6, F7 na horní straně a F2, F4, F6 a F8 na přední straně. Tato tlačítka

mohou být naprogramována - t.j. jsou určena k provádění specifikovaných úloh nebo funkcí. Z tohoto důvodu tato tlačítka často nazýváme programovatelná funkční tlačítka.

Abychom obdrželi funkci spojenou s tím, co je uvedeno na přední straně tlačítka - tj. F2, F4, F6 a F8 - přidržte tlačítko SHIFT. Z tohoto důvodu se tato tlačítka někdy nazývají programovatelná funkční tlačítka s SHIFT.

Funkčním tlačítkům v modu C 64 nejsou přiřazeny tištěné znaky. Jsou jim však přiřazeny kódy CHR$. Skutečně, každé s těchto tlačítek má dva kódy CHR$ - jeden odpovídající normálnímu stisknutí tlačítka, druhý odpovídá současnému přidržení tlačítka SHIFT. Abychom obdrželi tlačítkové funkce odpovídající sudým číslům, přidržte tlačítko SHIFT současně s požadovaným tlačítkem. Například, abyste aktivovali F2, stiskněte SHIFT s F1.

Kódy CHR$ pro tlačítka F1 - F8 jsou 133 - 140. Kódy však nejsou přiřazeny tlačítkům v číselném pořadí, ale následovně:

F1 CHR$(133) F2 CHR$(137)
F3 CHR$(134) F4 CHR$(138)

F5 CHR$(135) F6 CHR$(139)
F7 CHR$(136) F8 CHR$(140)

Funkční tlačítka lze použít různým způsobem v programu pomocí příkazu GET (viz část 4. s podrobnostmi o příkazu GET), například, následující program přiřadí tlačítku F1 funkci tisknout hlášení na obrazovce:

```
10 ?"STISKNI F1 ABYS POKRACOVAL"
20 GET A$: IF A$="" THEN 20
30 IF A$ CHR$(133) THEN 20
40 ?"JE STLACENO F1"
```

Řádky 20 a 30 provádějí v tomto programu většinu práce. Řádek 20 přikáže počítači čekat, dokud nebude stisknuto nějaké tlačítko, a teprve poté pokračovat dále. Všimněte si, že jestliže příkaz bezprostředně následující za THEN je GOTO, stačí pouze uvést číslo řádku. Všimněte si mimoto, že příkaz GOTO lze použít rovněž pro skok na řádek, v němž byl použit. Řádek 30 přikazuje počítači vracet se zpět a čekat na stisknutí nějakého tlačítka, dokud nebude stisknuto právě tlačítko F1.

ČÁST 10

Vytvořený program můžeme definitivně uložit pro budoucí použití na diskdrajvu Commodoru 128 nebo na přehrávači Dattassett Commodore.

Formátování disku v modu C 64

Pro uložení programu na prázdnou nebo netknutou disketu musíme především připravit disketu tak, aby mohla přijímat data. Tato operace se nazývá formátování disku. Dříve než disketu vložíte, přesvědčte se, že diskdrajv je zapnut. Nový disk se v modu C 64 formátuje následujícím příkazem:

OPEN 15,8,15: PRINT#15,"NO:JMÉNO,ID" RETURN

Tam kde je v příkazu uvedeno jméno uvěďte požadované jméno diskety, jehož délka nepřekročí 16 znaků. Na místě ID uveďte libovolný kód ze dvou znaků (např. W2 nebo 10). V průběhu formátování se kursor ztratí z obrazovky. Jakmile se znovu objeví, přikažte:

CLOSE 15 RETURN

Poznámka: Poté co jste disketu naformátovali v modu C 64 nebo C 128, můžete ji použít v obou modech.

Příkaz SAVE

Příkaz SAVE se používá k ukládání programů na disku nebo na kazetu.

Ukládání na disk

Jestliže vlastníte jednoduchý diskdrajv Commodore, program uložíte příkazem

SAVE "JMENO PROGRAMU",8 RETURN

Číslo 8 oznamuje počítači, že na uložení programu má použít diskdrajv.
Pro JMENO PROGRAMU platí táž pravodla při použití disku i kazety. Jménem programu může být jakékoliv jméno sestávající z písmen, číslic a/nebo symbolů, o délce maximálně 16 znaků. Zapamatujte si, že JMENO PROGRAMU musí být uvedeno v uvozovkách. Během ukládání programu kursor zmizí z obrazovky a objeví se znovu po dokončení operace.

Ukládání na kazetu

Jestliže k uložení programu použijete Datassett, vložte do přehrávače netknutou kazetu, převiňte ji na začátek (pokud je to nutné) a napište:

SAVE "JMENO PROGRAMU" RETURN

Příkazy LOAD a RUN

Po uložení programu je možné znovu jej přenést do paměti počítače a provést, pokaždé, kdy je to zapotřebí.

Přepsaní a provedení programu z disku

Pro přepsání programu z disku napište:

LOAD "JMENO PROGRAMU",8 RETURN

I v tomto případě číslo 8 ukazuje, že počítač má pracovat s diskdrajvem.
Aby se program provedl, napište RUN a stiskněte RETURN.

Přepsání a provedení programu z kazety

Pro přepsání programu z kazety napište

LOAD "JMENO PROGRAMU" RETURN

Jestliže neznáte jméno programu napište

LOAD RETURN

V tomto případě se přepíše nejbližší program na kazetě.
Pro určení bodu, kde začíná záznam programu můžete použít počítadlo DAtassettu. Jestliže pak chcete program vyvolat, převiňte kazetu z polohy 000 na počátek daného programu a napište

LOAD RETURN

V tomto případě nemusíte uvést JMENO PROGRAMU; program se nahraje automaticky, protože se vezme první následující program na kazetě.

Poznamka: Během procesu přehrávání se nahrávaný program na kazetě nevymaže, ale pouze se zkopíruje do počítače. V procesu nahrávaní programu se však automaticky zruší jakýkoliv jiný program BASIC, který se nacházel v paměti počítače.

Jiné příkazy pro disk

Kontrola programu

Pro kontrolu, zda byl program bez chyb uložen nebo přehrán zpět, použijte příkaz:

VERIFY "JMENO PROGRAMU",8 RETURN

Jestliže je program v paměti počítače totožný s programem na disku, na obrazovce se objeví "OK".

Příkaz VERIFY lze provést rovněž s programem na kazetě. Příkažte:

VERIFY "JMENO PROGRAMU" RETURN

V tomto případě nepište čárku a číslo 8, protože 8 znamená opět, že se pracuje s diskem.

Zobrazení obsahu disku

Abyste zobrazili seznam programů na disku nejprve napište:

LOAD "$",8 RETURN

Během této procedury kursor zmizí. Jakmile se znovu objeví napište

LIST RETURN

V tomto okamžiku se objeví seznam programů na disku. Pozor, při přepisu seznamu program, který se eventuálně nalézal v paměti počítače bude vymazán.

Inicializace diskdrajvu

Jestliže okénko na diskdrajvu bliká, znamená to, že se narazilo na chybu disku. Aby se disk vrátil do stavu, v němž se nacházel před objevením chyby, používa se procedura zvaná INICIALIZACE. Abyste inicializovali diskdrajv, napište:

OPEN 1,8,15"I": CLOSE 1 RETURN

Jestliže okénko bliká dále, vyjměte disketu, vypněte a poté zapněte diskdrajv.

Další informace o ukládaní a nahrávání programů naleznete v manualu diskdrajvu nebo Datassettu a v popisu příkazů SAVE a LOAD v Encyklopedii BASICu 7.0 v Kapitole 5.

K A P I T O L A

4

P O U Ž I T Í M O D U CP/M

ČÁST 11

Co to je CP/M 3.0

CP/M je výrobkem Digital Research Inc. Verse CP/M používaná Commodorem 128 je CP/M Plus verse 3.0. V této kapitole, kde pojednáváme v hlavních rysech o použití CP/M na Commodoru 128, budeme CP/M nazývat CP/M 3.0 nebo prostě CP/M.

CP/M 3.0 je operační systém, používaný v mikropočítačích nejčastěji. Jako každý operační systém, CP/M 3.0 řídí a ovládá kapacity počítače, včetně paměti a ukládání na disk, konsoly (obrazovka a klávesnice), tiskáren a komunikačních zařízení. CP/M 3.0 mimoto ovládá informace uložené v diskových fajlech. CP/M 3.0 může kopírovat fajly z disku do paměti počítače nabo na periferijní zařízení, například na tiskárnu. Pro provádění těchto operací má CP/M 3.0 uložené v paměti různé programy a provádí je na příkaz z konsoly. Po přepsání do paměti počítače program provede řadu operací, které přikáží počítači splnit určité úkoly. CP/M může být použit při tvorbě vlastních programů, nebo můžete použít řadu aplikačních programů, které jsou v CP/M k dispozici.

Prostředky nutné pro využití CP/M 3.0

CP/M 3.0 vyžaduje následující prostředky: počítač se zabudovaným procesorem Z 80, konsolu sestávající z klávesnice a obrazovky a alespoň jednoduchý diskdrajv. V osobním počítači Commodore 128 je mikroprocesor Z 80 zabudován; konsolu tvoří kompletní klávesnice Commodoru 128 a monitor s 80 sloupci; diskdrajvem je nový rychlý diskdrajv 1570 Commodore. Mimoto, k počítači jsou přibaleny jedna nebo dvě diskety s CP/M. Jestliže jsou dodány dvě diskety, jedna obsahuje systém CP/M 3.0 a řídící program HELP a druhá obsahuje jiné uživatelské programy. Jestliže je dodávána jedna disketa, pak na jedné straně jsou zaznamenány uživatelské systémové programy a program HELP a na druhé - další uživatelské programy.

Poznámka: Jestliže se CP/M používá s monitorem o 40 sloupcích, zobrazení bude sice mít 80 sloupců, ale zobrazí se vždy pouze 40 sloupců. Abyste zviditelnili 80 sloupců musíte nechat zobrazení proběhnout ve vodorovném směru pomocí tlačítka CONTROL a vhodného tlačítka kursoru (vlevo nebo vpravo).

Obsah disku

CP/M+.SYS

Jedná se o základní systémové fajly CP/M Plus a patří sem všechny části operačního systému, které jsou neustále přítomné v paměti: INPUT/OUTPUT systém BASICu (BIOS), který je zaznamenán v horní části paměti, diskový operační systém BASICu (BDOS), který je zaznamenán v paměti bezprostředně za BIOS a systémové parametry, které jsou zaznamenány na poslední stránce paměti.

CCP.COM

Při přepisování CP/M se příkazový procesor konsole (CCP) zaznamená do paměti bezprostředně za BDOS. Zbylá paměť pod CCP a nad stránkou 0 se nazývá přechodná paměťová oblast (TPA) a je to místo, do něhož se zapisují aplikační programy. CCP je program, který zpracovává všechny vstupy (obecně zaváděné z klávesnice) jako odpověď na požadavek systému A . CCP obsahuje 6 zabudovaných příkazů (viz Tabulku 14.1) a mimoto 14 příkazů na modifikování konsole (Tabulka 13.1).
CCP zpracuje jakékoliv slovo (které není jedním ze zabudovaných příkazů) zavedené jako odpověď na žádost systému, jako by to byl dočasný příkaz a poté se pokouší nalézt a provést fajl s tímto jménem a s rozšířením .COM. Jestliže CCP nenalezne fajl tohoto jména na disku, který je použit, zobrazí toto slovo následované otazníkem a dále systémovou odezvou.
Jestliže jako odpověď na systémovou odezvu zavedete další slovo, budou všechna slova následující za prvním chápána jako parametry přechdu k dočasnému příkazu. Jazyk nebo aplikační program bude zapsán a proveden jako kdyby se prováděl příkaz. Všechny programy CP/M zahrnují fajl .COM.

Fajly .COM

Všechny fajly .COM obsahují přechodné příkazy (viz seznam v tab. 14.2).
HELP.COM zobrazí hlášení obsažená v HELP.HLP (jehož rozšíření .HLP ukazuje, že se jedná o datový fajl, ne programový), který je prohlížen systémem CP/M C 128 a jeho příkazy. Jestliže nejste dobře obeznámeni s CP/M a nevlastníte jiné publikace nebo manuály o něm, doporučujeme použít příkaz HELP. Jestliže přitom chcete nasměrovat výstup na tiskárnu, stiskněte CONTROL a P. Dalším stisknutím CONTROL a P se tato funkce zruší.

OBSAH DISKU CP/M 3.0

Směrníky pro drajv A : Uživatel 0

Název		Bytů	Resc	Atributy
CCP	COM	4 K	25	Dir RW
DIR	COM	15 K	114	Dir RW
HELP	COM	7 K	56	Dir RW
KEYFIG	COM	10 K	75	Dir RW
PIP	COM	9 K	68	Dir RW
CPM+	SYS	23 K	182	Dir RW
FORMAT	COM	5 K	35	Dir RW
HELP	HLP	83 K	664	Dir RW
KEYFIG	HLP	9 K	72	Dir RW

Celkem bytů = 165 K
Celkem záznamů (recs)= 1291
Fajlů = 9
Celkem bloků 1 K = 165
Použito/max Dir pro drajv A : 15/64

Směrníky pro drajv B: Uživatel 0

Název		Bytů	Záznamů	Atributy
DATE	COM	8 K	25	Dir RW
DATEC	RSX	2 K	3	Dir RW
DIR	COM	30 K	114	Dir RW
DUMP	COM	2 K	8	Dir RW
ERASE	COM	8 K	29	Dir RW
GET	COM	14 K	51	Dir RW
PATCH	COM	6 K	19	Dir RW
PUT	COM	14 K	55	Dir RW
SAVE	COM	4 K	14	Dir RW
SETDEF	COM	8 K	32	Dir RW
SUBMIT	COM	12 K	42	Dir RW
DATEC	ASM	2 K	5	Dir RW
DEVICE	COM	16 K	58	Dir RW
DIRLBL	RSX	4 K	12	Dir RW
ED	COM	20 K	73	Dir RW
GENCOM	COM	30 K	116	Dir RW
INITDIR	COM	64 K	250	Dir RW
PIP	COM	18 K	68	Dir RW
RENAME	COM	6 K	23	Dir RW
SET	COM	22 K	81	Dir RW
SHOW	COM	18 K	66	Dir RW
TYPE	COM	6 K	24	Dir RW

Celkem bytů = 314 K Celkem 1 K bloků= 314
Celkem záznamů = 1168 Fajlů = 22
Použito/max Dir pro drajv A: 23/64

Napište HELP a obdržíte seznam různých argumentů, nebo HELP C 128 CP/M pro podrobnější informace (znak uprostřed C 128 CP/M se dostane stisknutím leve šipky v horní části klavesnice). V průběhu tisku, aby nevznikaly přerušení po zaplnění každé obrazovky, zavedte HELP C 128 CP/M |NO PAGE.
Další podrobnosti viz Část 14.

Jiné fajly

- ASM označuje zdrojový fajl v jazyce assembler
- RSX označuje rozšíření residenčního systému, t.j. přepisovaný automaticky z příkazového fajlu, jakmile je požadován.

Uvedení CP/M 3.0 do provozu

Následující stati ukazují, jak zapnout a přepsat CP/M 3.0, jak zavádět a ménit příkazový řádek a jak vyrobit rezervní kopii disku CP/M.

Nahrání CP/M 3.0

Přepsat CP/M 3.0 znamená přečíst kopii operačního systému ze systémového disku CM/M 3.0 a zavést ji do paměti počítače.
Přepsání může být provedeno různými způsoby. Jestliže je počítač vypnut, CP/M se nahraje tak, že nejprve zapneme diskdrajv a vložíme do něj systémový disk CP/M 3.0,a poté zapneme počítač. CP/M 3.0 se nahraje automaticky.

Jestliže se již nalézáme v modu C 128 BASIC, CP/M 3.0 se nahraje vložením systémového disku CP/M do drajvu, napsáním příkazu BASICu - BOOT nasledovaného RETURN. V tomto okamžiku se CP/M nahraje. V modu C 128 CP/M může být rovněž nahrán vložením systémového disku a stisknutím tlačítka RESET.
Jestliže se nacházíme v modu C 64 a chceme aktivovat mod CP/M, nejprve vypneme počítač, poté vložíme systémový disk CP/M do drajvu a zapneme počítač.
Pozor: Přesvědčte se, zda je disk zcela zasunut do drajvu dříve než zavřete okénko drajvu.

V CP/M 3.0 Commodoru 128 má uživatel k dispozici 59 K TPA (oblast dočasného programu), což je prakticky RAM uživatele.

Výchozí obrazovka CP/M

Po nahrání CP/M 3.0 do paměti počítače se na obrazovce objeví:

```
CP/M 3.0 On the Commodore 128  3 JUNE 85
         80 column display
A>
```

Důležitou součástí hlášení o uvedení systému do provozu je:

```
A>
```

Je to systémová "ozvěna" CP/M 3.0. Toto hlášení informuje, že CP/M je připraven číst příkazy zaváděné z klávesnice. Hlášení rovněž informuje, že implicitním drajvem je drajv A. Znamená to, že CP/M, pokud nedostane jiné informace, zapisuje programy a datové fajly na drajv A. Mimoto hlášení říká, že k systému byl jako uživatel připojen uživatel č. 0, protože žádné jiné číslo není před A uvedeno.
Poznámka: V CP/M je jednoduchý diskdrajv identifikován s drajvem A. A je ekvivalentní zařízení č. 8, drajv 0 v modu C 128 a C 64. Normálně maximální počet drajvů v CP/M je čtyři. Další drajvy jsou označeny B, C atd.

Příkazový řádek

CP/M 3.0 provádí operace podle určitých příkazů, zaváděných z klávewnice. Tyto příkazy se zobrazují na obrazovce a nazývají se příkazový řádek. Příkazový řádek v CP/M sestává z kl klíčového slova příkazu a ze závěru příkazu, který není povinný. Klíčové slovo pžíkazu určuje příkaz (program), který má být proveden. Závěr pčíkazu může obsahovat dodatečné informace o příkazu, například jméno fajlu nebo parametry. Například v

```
A > DIR MUJFAJL
```

je DIR klíčovým slovem příkazu a MUJFAJL je

závěr příkazu. Aby se příkazový řádek CP/M 3.0 provedl, stiskněte tlačítko RETURN. Zatímco tisknete znaky na klávesnici, objevují se na obrazovce. Kursor se přitom posunuje doprava. Jestliže se při psaní dopustíte chyby, stiskněte tlačítko INST/DEL nebo kombinace CTRL-H, čímž posunete kursor doleva a vymažete chybu. CTRL označujeme tlačítko CONTROL. Při specifikování řídícího znaku podržte tlačítko CTRL a stiskněte požadované písmeno (v Části 13. jsou uvedeny řídící znaky a jejich použití). Klíčové slovo a závěr příkazu mohou být vytištěny libovolnou kombinací velkých i malých písmen. CP/M 3.0 interpretuje všechna písmena příkazového řádku, jako by to byla písmena velká. Obvykle musí být příkazový řádek vypsán za systémovou ozvěnou, CP/M však dovoluje vložit mezery mezi ozvěnu a klíčové slovo příkazu.

Typy příkazů

CP/M 3.0 uznává dva různé typy příkazů: příkazy vestavěné a příkazy pro občasné použití. Příkazy vestavěné provádějí programy přítomné v paměti počítače jako součást operačního systému. Tyto příkazy se provedou okamžitě. Příkazy pro občasné použití jsou uloženy na disku jako programové fajly. Aby je bylo možné

provést, musí být nejprve přepsány z disku. Programové fajly pro občasné použití jsou odlišeny na zobrazení obsahu disku, protože za jejich jménem následuje tečka a COM (.COM). V Části 14. naleznete seznam příkazů CP/M a to jak zabudovaných, tak pro občasné použití. U programů s občasným použitím CP/M 3.0 kontroluje pouze klíčové slovo příkazu. Většina uživatelských programů vyžaduje určitý závěr příkazu. Závěr příkazu může obsahovat maximálně 128 znaků.

Jak CP/M čte příkazový řádek

Nyní použijeme příkaz DIR, abychom předvedli, jak CP/M čte příkazový řádek. DIR je zkratkou directory (adresář) a přikazuje CP/M zobrazit na obrazovce seznam fajlů na disku. Napište slovo DIR za systémovou "ozvenou" a stiskněte RETURN:

A DIR RETURN

CP/M odpoví na tento příkaz zobrazením jmen všech fajlů, které jsou uloženy na disku, vloženém do drajvu A. Například, jestliže v drajvu A je založena systémová disketa CP/M, na obrazovce se objeví seznam jmen fajlů následovně:

A: PIP COM: ED COM: CCP COM: HELP COM HELP HLP
A: DIR COM: CPM SYS

CP/M 3.0 uznává pouze správně napsaná příkazová slova. Jestliže se dopustíte při psaní chyby a stisknete RETURN, aniž byste chybu opravili, CP/M 3.0 zopakuje příkazový řádek následovaný otazníkem. Předpokládejme například, že jsme chybně napsali příkaz DIR:

A>DJR RETURN

CP/M odpoví:

DJR?

Znamená to, že CP/M nemůže nalézt klíčové slovo příkazu nazvaného DJR. Pro opravení chyby tohoto typu použijte tlačítko INST/DEL k vymazání chybného písmene. Jiný způsob vymazání znaku je pomocí tlačítek CTRL přidrženého současně se stisknutím H. Přitom se kursor posune dolema. CP/M poskytuje mnoho jiných řídících znaků, umožňujících modifikovat příkazové řádk V Části 13. ukážeme jak používat řídící znaky při provádění změn v příkazovém řádku a v informacích zadávaných z konsoly.
DIR připouští jako závěr příkazu jméno fajlu. DIR může být použit se jménem fajlu aby se prověřilo, zda se daný fajl nachází na disku. Například kontrola, zda se programový fajl MUJFAJL skutečně naléza na disku se provede příkazem:

A>DIR MUJFAJL RETURN

CP/M provede tuto operaci a buď zobrazí na obrazovce jméno určeného fajlu nebo hlášení:

No File (fajl není)

Vynechte alespoň jednu mezeru za DIR, bay se klíčové slovo oddělilo od závěru příkazu. Jestliže to neuděláte, CP/M 3.0 hlásí následující chybu:

```
A> DIRMUJFAJL   RETURN
DIRMUJFAJL?
```

Kopie disku CP/M 3.0

Dříve než započnete s jakýmikoliv operacemi, je nutné vyrobit kopii systémového disku CP/M pomocí jednoho nebo dvou diskdrajvů. Jestliže používáte dva drajvy, mohou to být dva 1541, dva 1570 nebo jeden 1541 a jeden 1570. Kopírovací disketa může být nová i použitá. Pro pořízení kopie použijte uživatelské programy FORMAT a PIP ze systémového disku CP/M.

Formátování disku

Formátujte disk pomocí programu FORMAT na 1541 nebo 1570.
Zaveďte příkaz FORMAT, zvolte typ požadovaného disku pomocí tlačítka pro pohyb kursoru směrem dolů, stiskněte RETURN a řiďte se instrukcemi na obrazovce. Stiskněte Y (ano) nebo

N (ne) v odpověď na otázku "Do you want to format another disk" (chcete formátovat jiný disk).

Kopie fajlu

Pro zkopírování fajlu použijte program PIP na druhé straně originální diskety. Zaveďte PIP a obvyklé hlášení (A>) bude nahrazeno hlášením PIP(*).
Jestliže používáte jediný diskdrajv, použijte A jako zdrojový drajv a E jako adresovaný. Drajv E se nazývá virtuální drajv, protože neexistuje jako součást počítačové konfigurace. Vložte disketu pro kopírování do drajvu a napište E:=A:*.* (bude zobrazeno hlášení pokaždé, když zdrojový disk a adresovaný disk vyměníte).
Jestliže používáte dva diskdrajvy, vložte zdrojovou disketu do drajvu A (zařízení č. 8 a disketu teprve formatovanou do drajvu B (zařízení č. 9 - začněte sklopením přepínače DIP umístěného zleva vzadu na 1570, když je ještě zdroj vypnut) a napište B:=A:*.*, aby se zkopírovaly všechny fajly.
Originální disketa CP/M je "flippy", t.j. je nahrána po obou stranách. Z tohoto důvodu, aby se přečetla druhá strana, musíme disketu vyjmout a otočit ji. Tato operace je nutná,

jestliže se používá 1570, který je jednostranným drajvem. S tímto drajvem musíte tudíž kopírovat dvě strany na dvě zvláštní diskety. Po zkopírování první strany otočte původní disketu a zkopírujte druhou stranu na tentýž cílový disk tak, že znovu provedete kopírovací příkazy jako odpověď na hlášení PIP.
Abyste vyrobili disketu pouze se systémovými fajly, použijte PIP pro zkopírování fajlů CPM+.SYS a CCP.COM na již formátovanou disketu.
Poznámka: Tyto dva fajly jsou nutné pouze na disketách používaných pro zavedení CP/M do počítače.
Jestliže používáte jediný drajv, napište E:=A:CPM+.SYS aby se zkopíroval první fajl a E:+A:CCP.COM aby se zkopíroval druhý. Po skončení použití PIP stiskněte RETURN a místo slova PIP se vám ohlásí operační systém.
Jestliže chcete dostat úplný popis PIPu napište
HELP PIP, HELP PIP OPTIONS a HELP PIP EXAMPLES.

Aplikační jazyky a softvér

CP/M je pouze operační systém, t.j. prostředek pro dosažení určitého cíle, ne cíl sám. Sám o sobě OS nemůže udělat nic. Abyste mohli psát vlastní programy, musíte mít jazyk - assembler nebo nějaký vyšší jazyk. Abyste se

pobavili videohrami nebo využili pracovní aplikace počítače, musíte mít aplikační progra

Jaké programy jsou k mání

CP/M je implementován prakticky na všech počítačích používajících CPU Intel 8080 nebo Zilog Z 80. Existuje proto spousta programů prováděných systémem CP/M. Nejlepší bude, kd se obrátíte za informacemi na obchodníky s p gramy. Obecně CP/M používá pro záznam informací na diskety modifikovanou frekvenční modulaci (MFM). DOS Commodoru normalně používá grupový kódový záznam (GCR). Diskdrajv Commodoru 1570 může číst pouze GCR. Prodávané softvérové pakety CP/M jsou diskety MFM. Rovněž mezi MFM existují různé formáty: 1570 může číst pouze diskety formátované pro:

Epson QX10 (sektory po 512 bytech, dvoustranné, 10 sektorů na stopu)

IBM 8 SS (CP/M-86) (sektory po 512 bytech, jednostranná, 8 sektorů na stopu

IBM-8DS(CP/M-86) (sektory po 512 bytech, dvstranná, 8 sektorů na stopu)

IBM-9DS(CP/M-86) (sektory po 512 bytech, dvostranná, 9 sektorů na stopu)

KayPro II (sektory po 512 bytech, jednostranná, 10 sektorů na stopu)

KayPro IV (sektory po 512 bytech, dvoustranná, 10 sektorů na stopu)

Osborne DD SS (sektory po 1024 bytech, jednostranná, 5 sektorů na stopu)

Osborne DD SS (sektory po 1024 bytech, dvoustranná, 5 sektorů na stopu)

Jestliže získáte program v CP/M musí být na disku v jednom z formátů uvedených výše. Mimoto, pamatujte, že C 128 provádí programy psané pro CP/M 2.2 a CP/M Plus (což je nové jméno systému, nazývaného původně verse 3). V každém případě, CP/M-86 je verse CP/M navržená pro použití s 16-bitovými procesory; programy CP/M-86 nefungují s 8-bitovým procesorem Z 80 C 128, ačkoliv 1570 dovoluje číst datové fajly CP/M-86.
Jestliže máte diskdrajv 1541 musíte převést programy z formátu MFM do formátu GCR. Pro tuto operaci se obraťte na obchodníka s programy s tím, že eventuálně uhradíte náklady na zkopírování.

Instalování na C 128

Existuje množství počítačů používajících operační systém CP/M. Z tohoto důvodu většina programů CP/M musí být představena v závislosti na aparatuře, na níž má pracovat. Proces implantování programu na C 128 znamená stanovit některé parametry programu. Manuál programu popisuje, jak instalovat program, jestliže je nutné tuto operaci provést. Většina programů je vybavena seznamem terminálů. Jestliže tento seznam obsahuje rovněž ADM31, berte jej.

Jestliže naopak není zahrnut, musíte instalov program osobně.
Níže uvádíme seznam toho, co musí být zahrnuto během provádění WINSTAL.COM (instalační program, který je součástí bloku Wordstar), spolu s informacemi nezbytnými pro instalová jiných programů, ačkoliv ne všechny programy je vyžadují.

Jméno terminálu	Commodore 1
Rozměry obrazovky	
výška	24
šířka	80
Umístění kursoru	
sekvenční funkční kód	1Bh 3Dh
Znaky, které je třeba umístit mezi číslo řádku a sloupce	žádné
Znaky, které je třeba umístit za číslem řádku a číslem sloupce	žádné
Umísťuje se číslo sloupce před číslo řádku?	ne
Jaký znak je poslán terminálu, aby označil řádek 1?	20h
Jaký znak je poslán terminálu, aby označil sloupec 1?	20h
Kódy jakého typu se zasílají, aby označili čísla řádků a sloupců?	Jed bajtová bin. h
Terminálový začátek sekvence funkčních kódů	1Bh 59h 1Bh 1 1Bh 60h
Terminálový výstup sekvence funkčních kódů	žádný

Přehled zapnut sekvence funkčních kódů	1Bh 1Bh 1Bh 52h
Přehled vypnut sekvence funkčních kódů	1Bh 1Bh 1Bh 51h
Vymaž do konce řádku	1Bh 54h
Zruš řádek	1Bh 52h
Dosaď řádek	1Bh 45h
Používá váš terminál poslední znak na obrazovce jako příkaz scroll?	ano

Většina tiskáren Commodoru vyžaduje stejnou instalaci jako standartní tiskárna, bez komunikačního protokolu a prvního seznamového zařízení jako drajvu tiskárny.

Poznámka: Písmeno h u čísel v předchozím seznamu indikuje, že se jedná o hexadecimální čísla (která používají jako základ 16 místo základu desítkového).

ČÁST 12

Co je to fajl

Jedním z nejdůležitějších úkolů CP/M je umožnit přístup do fajlů na disku a řídit je. Fajly jsou v CP/M stejně jako v modech C 128 a C 64 základními soubory informací. Přesto však CP/M ovládá fajly jinak než v modech C 128 a C 64. V tomto paragrafu popisujeme dva druhy fajlů, používané v CP/M; ukazujeme jak je vytvářet, definovat a vstupovat do nich a popisujeme, jak lze fajly ukládat na disky CP/M.

Jak bylo řečeno, fajl v CP/M 3.0 je souborem informací. Každému fajlu musí být přiřazeno jméno, podle něhož jej CP/M může identifikovat. Na každém disku je uložen seznam. Tento seznam obsahuje jména všech fajlů uložených na disku a umístění těchto fajlů na daném disku. Existují dva typy fajlů CP/M: programové fajly (příkazy) a datové fajly. Programové fajly obsahují řadu příkazů, které počítač provede, aby dosáhl požadovaného výsledku. Datový fajl je obvykle souborem souvztažných informací (například seznam jmen a adres, inventář, účetní záznamy podniku, text dokumentace).

Vytváření fajlů

Existují různé způsoby jak vytvořit fajl v CP/M. Jedním z nich je textovy editor. Tex-

tový editor ED CP/M se používá pro vytvoření a přiřazení jména fajlu. Fajl může být rovněž vytvořen jako kopie (na novém místě) fajlu již existujícího; v tomto procesu se rovněž změnilo jméno fajlu. V CP/M lze použít příkaz PIP pro kopírování a přejmenování fajlu. Nakonec jiné programy (jako MAC - program ve strojovém jazyku CP/M) vytvářejí výstupní fajly v průběhu zpracování vstupního fajlu. O příkazech ED a PIP pojednáme stručně v Části 14. spolu s jinými příkazy často používanými v CP/M. Další podrobnosti o těchto a všech dalších příkazech CP/M 3.0 naleznete v Uživatelském manuálu CP/M Plus.

Přiřazení jména fajlu

Specifikování fajlu

CP/M identifikuje každý fajl pomocí jednoznačné specifikace fajlu. Specifikace fajlu může sestávat ze čtyř částí: identifikátor drajvu, jméno fajlu, typ fajlu a klíčové slovo (pass word). Jedinou povinnou částí specifikace je jméno fajlu.

Identifikátor drajvu

Identifikátorem drajvu je jedno písmeno (od A do P) následované dvojtečkou. Každému systémovému diskdrajvu je přiřazeno písmeno. Identifikátor drajvu ve specifikaci fajlu

přikazuje CP/M hledat daný fajl na disketě, která se momentálně nachází ve specifikovaném drajvu. Například, při:

B:MUJFAJL RETURN

CP/M hledá fajl MUJFAJL na drajvu B. Jestliže identifikátor drajvu není uveden, CP/M hledá fajl na "implicitním" drajvu (obvykle jím je drajv A).

Jméno fajlu

Jméno fajlu může sestávat maximálně z 8 znaků, například:

MUJFAJL

Specifikace fajlu může obsahovat pouze jméno fajlu. Používejte jako jméno fajlu, které má nějaký vztah k jeho obsahu. Například pro fajl obsahující seznam klientů můžete zvolit jméno: KLIENTI., takže ukazuje na obsah fajlu.

Typ fajlu

Abychom ukázali, že fajl patří do určité kategorie, CP/M umožňuje připojit ke jménu fajlu rozšíření, tvořené jedním až třemi znaky, nazývané typ fajlu. Jestliže chceme uvést typ fajlu, musíme ho napsat oddělený od jména fajlu tečkou. Rovněž v tomto případě je radno používat rozšíření, které pomáhá rozpoznat kategorii fajlu. Například k fajlu, obsahujícímu seznam klientů můžeme přidat:

KLIENTI.JMN

Jestliže CP/M zobrazuje specifikaci fajlu, přidá mezery za jménem fajlu, aby bylo možné snadno porovnat typ fajlu. Programové fajly, CP/M uložené na disku mají typ fajlu .COM. Nepoužívejte tento typ fajlu ve vašich vlastních specifikacích.

Klíčové slovo

V CP/M Commodoru 128 je možné zahrnout do specifikace fajlu rovněž klíčové slovo. Klíčové slovo může sestávat maximálně z 8 znaků a musí být oddělené od typu fajlu (nebo od jména fajlu, jestliže typ fajlu nebyl specifikován) středníkem například:

KLIENTI.JMN;UCTY

Klíčové slovo není povinné, ale jestliže je fajl chráněn tímto klíčovým slovem, musíte klíčové slovo uvést ve specifikaci fajlu, abyste k němu získali přístup.

Příklad specifikace fajlu

Specifikace fajlu obsahující všechny čtyři možné prvky sestává z identifikátoru drajvu, primárního jména fajlu, typu fajlu a klíčového slova; vzájemně jsou oddělené příslušnými znaky, jako v následujícím příkladu:

A:DOKUMENT.LEG;MARIO RETURN

Číslo uživatele

CP/M 3.0 identifikuje všechny fajly rovněž tím, že jim přiřadí číslo uživatele od 0 do 15. CP/M 3.0 přiřadí číslo uživatele fajlu během vytváření tohoto fajlu. Čísla uživatelů umožňují rozdělit fajly do 16 skupin. Číslo uživatele vždy předchází identifikátor drajvu, s výjimkou uživatele č. 0, což je implicitní číslo uživatele a neuvádí se v hlášení systému. Uvedeme některé příklady čísel uživatelů a jejich význam:

```
4A>   uživatel č. 4, drajv A
 A>   uživatel č. 0, drajv A
2B>   uživatel č. 2, drajv B
```

Číslo uživatele zavádíme a měníme pomocí příkazu USER:

```
A>USER 3    RETURN
3A>
```

Číslo uživatel i drajv můžeme měnit současně pomocí příkazu:

```
A>3B:  RETURN
3B>
```

Většina příkazů umožňuje přístup pouze do fajlu s daným číslem uživatele. Jestliže však fajl má číslo uživatele 0 a jsou mu přiřazeny ostatní znaky fajlu, je tento fajl přístupný pro uživatele s libovolným číslem.

Používání proměnných znaků pro přístup do několika fajlů zároveň

Některé příkazy CP/M, zabudované i pro občasné použití, mohou vybírat a zpracovávat různé fajly, jestliže jméno nebo typ fajlu obsahují speciální proměnné znaky. Proměnný znak to je znak, který se může použít na místě jiného znaku. CP/M 3.0 používá jeko proměnné znaky hvězdičku (*) a otazník (?). Například použití ? jako třetího znaku ve jménu fajlu sděluje CP/M, že ? zde nahrazuje libovolný znak použitelný v tomto místě. Podobně * přikazuje CP/M nahradit jméno fajlu postačujícím počtem otazníků, aby vyplnily celé jméno fajlu. Specifikace fajlu, obsahující proměnné znaky se nazývá neurčená specifikace fajlu a může se vztahovat na několik fajlů současně, protože poskytuje CP/M pouze model pro nalezení jména fajlu. CP/M hledá v seznamu disku a vybere všechny fajly, jejichž jméno nebo typ odpovídá danému "modelu". Například při

```
?????TAS.LIB  RETURN
```

CP/M vybere z disku všechny fajly, jejichž jméno má 8 znaků, končí TAS a jejichž typ je LIB.

Rezervované znaky

Znaky uvedené v tabulce 12-1 mají v CP/M 3.0 zvláštní význam a z tohoto důvodu je nelze

použít ve specifikacích fajlů jinde, než stanoví pravidla.

Tabulka 12-1. Rezervované znaky CP/M 3.0

Znak	Význam
<S,!(>[]	terminátory specifikace fajlů tabulátor space,return
:	terminátor drajvu ve specifikaci fajl
.	terminátor typu fajlu ve specifikaci fajlu
;	terminátor klíčového slova ve specifikaci fajlu
;	terminátor komentáře na začátku příkazového řádku
*?	proměnné znaky v neurčené specifikaci fajlu
< > &[] +-	terminátory specifikací opcí
[]	terminátory seznamu opcí pro opce globální a specifické
()	terminátory pro pozměňující násobky v hranatých závorkách pro opce s modifikátory
/$	terminátory opcí v příkazovém řádku

Typy fajlů rezervované

CP/M 3.0 má již ustálené různé skupiny fajlů. V tabulce 12-2 jsou uvedené různé typy fajlů s krátkým popisem každého z nich.

Tabulka 12-2. Rezervované typy fajlů v CP/M

Typ fajlu	Význam
ASM	Zdrojový fajl assembleru
BAS	Zdrojový program BASIC
COM	Program procesoru Z 80 nebo ekvivalentního strojového jazyka
HEX	Výstupní fajl z MAC (používaný HEXCOMem)
HLP	Fajl hlášení HELP
$$$	Dočasný fajl
PRN	Fajl tisku MAC nebo RMAC
REL	Výstupní fajl RMAL (používaný LINKem)
SUB	Seznam příkazů, které musí být provedeny SUBMITem
SYM	Fajl symbolů MAC, RMAC nebo LINK
SYS	Systémový fajl

ČÁST 13

Tato část manuálu popisuje, jak CP/M 3.0 komunikuje s konsolou a tiskárnou, jak vyvádí a ukončuje výstupy na konsolu a tiskárnu a jak modifikuje příkazy zaváděné z klávesnice.

Ovládání výstupů na konsolu

Informace zobrazované CP/M 3.0 se ukazují na obrazovce příliš rychle a tudíž by nebylo možné číst je. Abychom systému přikázali počkat, dříve než dokončí zobrazování, je třeba stisknout současně tlačítka CTRL a S. Posloupnost CTRL a S zastaví zobrazení. Jestliže chcete pokračovat, stiskněte CTRL Q. Stisknutí tlačítka NO SCROLL vyvolá svstémovou pauzu a zobrazí okno "pauza" na spodku obrazovky (řádek 25). Zobrazení obnovíte novým stisknutím NO SCROLL. Jestliže stisknete jakékoliv jiné tlačítko než CTRL Q nebo NO SCROLL během tohoto přerušení, CP/M 3.0 generuje akustický signál.

Jiné uživatelské programy CP/M 3.0 (například DIR a TYPE) obsahují automatické stránkování konsole. Znamená to, že když je výstup programu příliš dlouhý na to, aby se vešel na obrazovku, zobrazování výstupu se automaticky zastaví jakmile se obrazovka zcela zaplní.

V tomto případě, aby zobrazování pokračovalo, musíte stisknout RETURN. Tato volba může být aktivována nebo deaktivována příkazem SETDEF.

Ovládání výstupu na tiskárnu

Abychom převedli výstupy na tiskárnu je rovněž možné použít řídící příkazy. Pro vyvedení výstupů na tiskárnu stiskněte CTRL-P. Systém informuje uživatele akustickým signálem, že tato funkce byla aktivována. Ukončíte tuto funkci opštovným stlačením CTRL-P (v tomto okamžiku se žádný akustický signál negeneruje). Dokud je výstup na tiskárnu aktivován všechny znaky zobrazované na obrazovce se tisknou na tiskárně.

Výstup na tiskárnu se může použít s příkazem DIR pro obdržení výpisu fajlu uloženého na disketě. Abyste obdrželi kopii určité části fajlu použijte rovněž CTRL-P spolu s CTRL-S a CTRL-Q. Pro vyvedení fajlu na obrazovku používejte příkaz TYPE. Jakmile se dostanete k místu, které chcete vytisknout, stiskněte CTRL-S čímž zastavíte zobrazení, CTRL-P čímž vyvedete výstupy na tiskárnu, a potom CTRL-Q, aby se obnovilo zobrazování a začal tisk. Pro deaktivování výstupu na tiskárnu použijte potom posloupnost CTRL-S, CTRL-P a CTRL-Q.

Změna řádku na konsole

Jak jsme viděli, je možné opravovat chyby vzniklé při tištění příkazů pomocí tlačítek

INST/DEL nebo CTRL-H. CP/M 3.0 obsahuje i další funkce pro opravování řádků, které se realizují pomocí řídících znaků. Řídící znaky lze používat při opravování řádků nebo vstupních dat ve většině programů.

Použití řídících znaků pro změnu řádku

Pomocí řídících znaků, uvedených v tabulce 13-1 je možné pohybovat kursorem vlevo nebo vpravo, abychom doplnili nebo zrušili znaky v příkazovém řádku. Tímto způsobem se vyhneme nutnosti znovu tisknout vše, co se nalézá napravo od opravovaného znaku.

V následujícím příkladu uživatel napsal chybně PIP a CP/M 3.0 reagoval chybovým hlášením. Uživatel znovu vyvolá příkazový řádek, obsahující chybu pomocí CTRL-W a chybu opraví (podtržení reprezentuje kursor):

A>POP A:=B:*.*_	chyba tisku (POP)
POP?	
A>POP A:=B:*.*_	CTRL-W znovu vyvolalo řádek
A>POP A:=B:*.*	CTRL-B přemístilo kursor na začátek řádku
A POP A:=B:*.*	CTRL-F posunulo kursor o jedno místo doprava
A>PP A:=B:*.*	CTRL-G zrušilo chybný znak
A>PIP A:=B:*.*	stisknutím I jsme opravili chybu

Po opravení příkazového řádku může uživatel stisknout RETURN i když se kursor nachází uprostřed řádku. Po stisknutí RETURN (nebo jednoho z ekvivalentních řídících znaků) se nejen provede příkaz, ale příkaz se rovněž uloží do bufferu, takže je možné pomocí CTRL-W znovu jej vyvolat a opravit nebo opakovaně provédt.

Jestliže vložíme znak dovnitř řádku, znaky nacházející se vpravo od kursoru se posunou doprava. Přitom se může stát, že znaky zcela vpravo v příliš dlouhém řádku se ztratí z obrazovky. Tyto znaky však nejsou ztraceny a znovu se objeví, jestliže například zrušíme nějaké znaky v řádku. Objeví se rovněž jestliže stiskneme CTRL-E v případě, že kursor se právě nalézá někde uprostřed řádku. CTRL-E vyvolá přenesení znaků napravo od kursoru na následující řádek.

Tabulka 13-1 obsahuje úplný seznam řídících znaků CP/M 3.0 pro modifikovaní řádků na Commodoru 128.

Tabulka 13-1. Řídící znaky pro modifikování řádků v CP/M 3.0

Znak	Význam
CTRL-A	Posune kursor o jeden znak doleva
CTRL-B	Přemístí kursor na začátek příkazového řádku, aniž by se změnil obsah řádku. Jestliže se kursor nachází na začátku řádku, CTRL-B jej přemístí na konec řádku.
CTRL-E	Vyvolá nucený "návrat vozíku", neodešle však příkazová řádek do CP/M 3.0. Přemístí kursor na začátek následujícího řádku, aniž by zrušil předchozí vstupy.
CTRL-F	Posune kursor o jeden znak doprava.
CTRL-G	Zruší znak v poloze kursoru. Kursor se přitom neposune, ale znaky vpravo od kursoru se posunou o jedno místo doleva.
CTRL-H	Zruší znak vlevo od kursoru a posune kursor o jedno místo doleva. Znaky napravo od kursoru se posunou o jedno místo doleva.
CTRL-I	Přemístí kursor do následující polohy tabulátoru. Tabulátorové body se nalézají v každém osmém sloupci. Tato kombinace má stejný výsledek jako tlačítko TAB.
CTRL-J	Odešle příkazový řádek do CP/M 3.0 a umístí kursor na začátek následujícího řádku. Tato kombinace tlačítek má stejný výsledek jako RETURN nebo CTRL-M.

CTRL-K Zruší celý řádek počínajíc polohou kursoru.

CTRL-M Odešle příkazový řádek do CP/M 3.0 a vrátí kursor na začátek následujícího řádku. Tato kombinace tlačítek má stejný efekt jako RETURN nebo CTRL-J.

CTRL-R Znovu vytiskne příkazový řádek. Vloží znak # v místě, kde se nacházel kursor, přemístí kursor na následující řádek a znovu vytiskne všechny dílčí příkazy, vytištěné až do tohoto okamžiku.

CTRL-U Zruší všechny znaky v příkazovém řádku, vloží znak # v běžné poloze kursoru a přemístí kursor na následující řádek. Pomocí CTRL-W je možné vyvolat všechny znaky, které se nalézaly vlevo od kursoru než jsme stiskli CTRL-U.

CTRL-W Vyvolá a zobrazí příkazový řádek zavedený jako poslední jak na úrovni operačního systému, tak během provádění programu, jestliže CTRL-W je první operace provedená jako odpověď na systémové hlášení. CTRL-J, CTRL-M, CTRL-U a RETURN jsou příkazy, po nichž může být příkazový řádek znovu vyvolán. Jestliže příkazový řádek obsahuje nějaké znaky, CTRL-W přemístí kursor na konec příkazového řádku a po stisknutí RETURN CP/M 3.0 provede vyvolaný příkaz.

CTRL-X Zruší všechny znaky vlevo od kursoru a přemístí kursor na začátek běžného řádku. CTRL-X zachovává znaky vpravo od kursoru.

ČÁST 14

Jak jsme viděli v části 11., příkazový řádek CP/M 3.0 sestává z klíčového slova příkazu, volitelného příkazového kódu a stisknutí tlačítka RETURN. V dané části manuálu popíšeme dva typy příkazů, které mohou být určeny klíčovým slovem příkazu a uvedeme další obecné údaje o jednotlivých příkazech a jejich funkci. Poskytneme rovněž příklady jiných často používaných příkazů a vysvětlíme pojem logického a fyzického zařízení v CP/M 3.0. Ukážeme rovněž, jak CP/M 3.0 vyvolává programové fajly z disku, jak provádí vícenásobné příkazy a jak znovu vyvolat operační systém. Nkonec, vysvětlíme jak používat příkaz HELP přímo z klávesnice, abychom získali informace o různých argumentech CP/M 3.0 včetně tvaru příkazů a jejich použití.

Dva typy příkazů v CP/M 3.0

CP/M 3.0 zahrnuje dva typy příkazů:

- vestavěné příkazy, které určují programy v paměti počítače
- příkazy pro občasné použití, které určují programové fajly na disku.

CP/M 3.0 ovládá 6 vestavěných příkazů a více než 20 příkazů pro občasné použití. K systému lze přidat uživatelské programy, včetně prodávaných aplikačních programů kompatibilních s CP/M 3.0. Programátor může psát i

vlastní programy pro použití v CP/M 3.0.

Vestavěné příkazy

Vestavěné příkazy jsou součástí CP/M 3.0, které lze použít vždy, nezávisle na tom, jak disketa je momentálně vložena do diskdrajvu. Vestavěné příkazy se zavedou do paměti počítače při nahrání CP/M 3.0 a provádějí se rychleji, než programy pro občasné použití. V tabulce 14-1 jsou uvedeny vestavěné příkazy CP/M 3.0 Commodoru 128.

Některé vestavěné příkazy obsahují volby, které vyžadují pomoc ze strany programů pro občasné použití. Příslušný příkaz pro občasné použití má stejné jméno, jako vestavěný příkaz, a typ fajlu COM.

Tabulka 14-1. Vestavěné příkazy

Příkaz	Funkce
DIR	Zobrazí jména všech fajlů ze seznamu, kromě fajlů označených SYS.
DIRSYS	Zobrazí jména fajlů označených (systémovým) přívlastkem SYS ze seznamu.
ERASE	Zruší jméno fajlu v seznamu diskety a uvolní pamětový prostor obsazený tímto fajlem.
RENAME	Přiřadí nové jméno fajlu na disku.
TYPE	Zobrazí na obrazovce obsah fajlu ASCI (TEXT)
USER	Přejde na uživatele s jiným číslem.

Příkazy pro občasné použití

Příkazy CP/M 3.0 pro občasné použití jsou uvedeny v tabulce 14-2. Jestliže zavedeme klíčové slovo příkazu pro občasné použití, CP/M 3.0 přepíše programový fajl z disku a přenese na tento fajl data nebo parametry, které se zadávají v příkazovém kódu. DIR, RENAME a TYPE jsou vestavěné příkazy s rozšířením o přechodné příkazy.

Tabulka 14-2. Příkazy pro občasné použití

Příkaz	Funkce
DATE	Určí a zobrazí datum a hodinu
DEVICE	Přiřadí logické zařízení CP/M jednomu nebo několika fyzickým zařízením, mění protokol drajvu a rychlost přenosu nebo určí šířku obrazovky.
DIR	Zobrazí obsah fajlu a jeho charakteristiky
DUMP	Zobrazí fajl ve formátu ASCII a hexadecimálním.
ED	Vytváří a modifikuje fajly ASCII.
ERASE	Používá se pro zrušení proměnných znaků.
FORMAT	Formátuje disk CP/M. Zruší data na již použitém disku.
GENCOM	Vytváří speciální fajl COM s fajlem RSX v řadě.
GET	Dočasně přijímá input z fajlu na disku místo z klávesnice.

HELP	Zobrazí informace jak používat příkaz CP/M 3.0.
INITDIR	Inicializuje adresář diskety, aby bylo možné tisknout hodinu a datum.
KEYFIG	Umožní modifikovat definici tlačítek.
PATCH	Zobrazí nebo instaluje korekční rutinu CP/M.
PIP	Kopíruje a kombinuje fajly.
PUT	Převede dočasně výstup z tiskárny nebo z konsole na fajl na disku.
RENAME	Změní jméno fajlu nebo skupiny fajlů proměnnými znaky.
SAVE	Zkopíruje obsah paměti počítače do fajlu.
SET	Stanoví volitelné charakteristiky fajlů, včetně značení disket, přívlastků fajlů, tvar, v němž se zobrazí datum a hodina a ochranu klíčovým slovem.
SETDEF	Stanoví systémové volby, včetně pořadí hledání na drajvu.
SHOW	Zobrazí charakteristiky disku a drajvu
SUBMIT	Automaticky provede násobné příkazy
TYPE	Zobrazí na obrazovce (nebo na tiskárně, pokud je to požadováno) obsah textového fajlu (nebo skupiny fajlů, jestliže se použijí proměnné znaky).

Přepínání VSTUP/VÝSTUP

Příkaz PUT CP/M 3.0 umožňuje dirigovat výstupy na konsolu nebo na tiskárnu či na diskový fajl. Je možné použít příkaz GET pro přenos

vstupů z diskového fajlu do CP/M 3.0 nebo do uživatelova programu.
Příklad, který následuje, umožňuje ilustrovat další možnosti GET a PUT.
Příkaz PUT lze použít k převedení výstupu z konsole na fajl na disku nebo na konsole.
Příkazem PUT lze vytvářet diskové fajly obsahující adresář všech fajlů na této disketě, tak jak je to ukázáno v tab. 14-1.

```
A>PUT CONSOLE OUTPUT TO FILE DIR.PRN
  (přenes výstupy z konsole na fajl DIR.PRN)
A>DIR
A:FILENAME   TEX:FRONT  BAK:ONE  BAK:THREE  TEX
A:FOUR       TEX:ONE    TEX:LINEDIT
```

Tab. 14-1. Příklad příkazu PUT

Příkaz GET může přikázat CP/M 3.0 nebo nějakému programu číst vstupy z diskového fajlu místo z klávesnice. Jestliže fajl má být přečten systémem CP/M 3.0, musí obsahovat řádek standartních příkazů CP/M 3.0. Jestliže má být fajl přečten uživatelovým programem, musí obsahovat vstupy vhodné pro tento program. Fajl může obsahovat jak příkazové řádky CP/M 3.0, tak vstupy pro program, pokud ovšem obsahuje rovněž příkaz vyvolat tento program.

Přiřazování logických zařízení

Minimální hardvérová konfigurace pro CP/M 3.0 na Commodoru 128 zahrnuje konsolu, sestávající z klávesnice a obrazovky, a diskdrajv 1570 K systému lze přidat i jiná zařízení, například tiskárnu nebo modem (měnič měnící digitální signály v akustické, které lze poslat po telefonní lince, a naopak).V tabulce 14-3 jsou uvedena jména logických zařízení CP/M 3.0 s různými fyzickými zařízeními pro vstup a výstup. Mimoto tabulka ukazuje fyzická zařízení, přiřazovaná logickým zařízenám systému CP/M 3.0 Commodoru 128.

Tabulka 14-3. Logická zařízení CP/M 3.0

Jméno logic. zařízení	Typ zařízení	Přiřazení fyzického zařízení
CONIN	Vstup z konsole	klávesnice
CONOUT	Výstup na konsolu	obrazovka 80 sloupců
AUXIN	Pomocný vstup	není
AUXOUT	Pomocný výstup	není
LST	Výstup výpisů	PRT1 nebo PRT2

Toto přiřazení lze změnit pomocí příkazu DEVICE. Například, lze přiřadit AUXIN a AUXOU modemu, takže počítač může prostřednictvím te lefonní linky komunikovat s jinými uživateli

počítače nebo s informační službou.

Hledání programového fajlu

Jestliže klíčové slovo příkazu poukazuje na uživatelův program, CP/M 3.0 vyhledá fajl na "implicitním" drajvu nebo na drajvu daného uživatele a pod uživatelem 0 tento fajl opatří přívlastkem SYS. Vlibovolném okamžiku během hledání CP/M 3.0 přeruší hledání při nalezení programového fajlu. Poté CP/M 3.0 přepíše program do paměti počítače a provede jej. Po skončení programu CP/M 3.0 zobrazí systémové hlášení a čeká na nový příkaz. Jestliže CP/M 3.0 nenalezl příkazový řádek, zobrazí tento příkaz následovaný otazníkem a čeká na nový příkaz.

Provádění složených příkazů

V ukázkách uváděných až dosud CP/M 3.0 prováděl pokaždé pouze jediný příkaz. CP/M 3.0 však může provádět rovněž posloupnosti příkazů. Posloupnost příkazů zavedete jako odpověď na systémové hlášení nebo může být uložena ve fajlu na disku pomocí často používaného "SUB" jako typ fajlu. Po uložení této posloupnosti na disku ji lze v libovolném okamžiku provést pomocí příkazu SUBMIT.

Uzavření programu

Abychom ukončili provádění programu nebo znovu vyvolali operační systém, použijeme kombinaci tlačítek CTRL-C tak, že současně stlačíme tlačítka CTRL a C.

Většina aplikačních programů, které fungují v CP/M a mnoho programů pro občasné použití lze pomocí kombinace CTRL-C přerušit. Jestli však zkoušíme ukončit program v okamžiku výstupu na obrazovku, musíme stisknout CTRL-S, abychoz zastavili zobrazování, a teprve poté CTRL-C.

Jak přivolat pomoc

CP/M 3.0 ovládá příkaz nazývaný HELP, který zobrazí přehled tvarů a způsoby použití nej používanějších příkazů CP/M. Napište následující příkaz:

A>HELP RETURN

Místo, abyste tiskli slovo HELP po znacích sledované RETURN stačí stisknout tlačítko HELP.

Poté se zobrazí seznam programů následovně:

Argumenty, které jsou k dispozici

COMMANDS CNTRLCHARS DATE DEVICE DIR
DUMP ED ERASE FILESPEC GENCOM GET
HELP HEXCOM INITDIR LIB LINK MAC

PATCH PIP(COPY) PUT RENAME RMAC SAVE
SET SETDEF SHOW SID SUBMIT TYPE
USER XREF

Například, napište:
HELP>PIP RETURN

a CP/M zobrazí následující informace:

PIP (COPY)
Syntax:
DESTINATION SOURCE
PIP d: Gn filespec[Gn] =filespec[o],...d:[o]
Vysvětlení:
Program pro kopírování fajlů kopíruje fajly, spojuje a přenáší fajly na disk, tiskárnu, konsolu nebo jiné zařízení, připojené k počítači. První specifikovaný fajl (filespec) je cílový (adresovaný), Druhý je zdrojový. Uveďte dva nebo více zdrojových fajlů oddělených čárkami, jestliže je chcete sloučit do jednoho fajlu. [o] je libovolná kombinace dostupných volitelných parametrů. Volba [Gn] ve specifikaci cílového fajlu přikáže PIP zkopírovat fajl s tímto číslem uživatele.
PIP bez příkazového kódu zobrazí * a čeká na řadu příkazů, zaváděné a prováděné jeden po druhém. Zdrojem nebo cílem mohou být libovolná logická zařízení CP/M 3.0. Funkce

HELP dodává informace, jak jsme ukázali na příkladu, o všech vestavěných i občasně používaných příkazech CP/M 3.0. Jestliže požadujete specifické informace, napište HELP argument za systémovým hlášením, kde "argumentem" je příkazový kód, o němž vyžadujeme informace, například:
A>HELP PIP
A>HELP DIRSYS
HELP lze použít pokaždé, když potřebujeme informace o určitém příkazu nebo může být použit, když chceme rozšířit svoje znalosti o CP/M 3.0.

ČÁST 15

Klávesnice

Commodore je v CP/M vybaveno různými možnostmi rozšíření. Tato zdokonalení spojují výhody Commodoru 128 a CP/M 3.0. V dané části manuálu popíšeme tato rozšíření.

Všechna tlačítka klávesnice mohou být redefinována tak, aby mohla produkovat daný kód nebo funkci, kromě následujících tlačítek:

SHIFT vlevo
SHIFT vpravo
Commodore Cx
CTRL
RESTORE
40/80
CAPS LOCK

Při redefinování tlačítek klávesnice reaguje na následující speciální operace. Abychom specifikovali specifikovali nějakou funkci, přidržte stlačenými tlačítko CTRL a tlačítko SHIFT vpravo současně s tlačítkem požadované funkce.

Tlačítko	Funkce
KURSOR vlevo	definuje tlačítko
KURSOR vpravo	definuje řetězec
ALT	mění funkce tlačítek

Definování tlačítka

Uživatel může stanovit kód generovaný daným tlačítkem. Každé tlačítko má čtyři možné definice: normální, velká písmena se SHIFT a s CONTROL. Velká abeceda s SHIFT se aktivuje/deaktivuje stisknutím tlačítka Commodore.
Po aktivování tohoto modu se ve vnitřní části obrazovky ukáže obdélník. První stisknuté tlačítko bude definované tlačítko. Zobrazí se běžná hexadecimální hodnota (HEX) tlačítka; v tomto okamžiku uživatel může natisknout nový HEX kód tohoto tlačítka, nebo celou operaci anulovat tím, že stiskne nehexadecimální tlačítko. Níže uvedeme definice kódů, které mohou být tlačítku přiřazeny (v modu ALT budou kódy vracet aplikace - viz mod ALT).

Kód	Funkce
00h	nic (stisknutí tlačítka nevyvolá žádnou reakci)
od 01h do 7Fh	normální kódy ASCII
od 80h do 9Fh	přiřazení stringu
od A0h do AFh	barva znaků (80 sloupců)
od B0h do BFh	barva pozadí (80 sloupců)
od C0h do CFh	barva znaků (40 sloupců)
od D0h do DFh	barva pozadí (40 sloupců)
od E0h do EFh	barva rámečku (40 sloupců)
F0h	aktivuje/deaktivuje stav disku

F1h	systém a pauza
F2h	nedefinováno
F3h	okno pravé obrazovky (40 sloupců)
F4h	okno levé obrazovky (40 sloupců)
od F5h do FFh	nedefinováno

Definování řetězce

Tato funkce umožňuje přiřadit více tlačítkových kódu jedinému tlačítku. Všechna tlačítka stisknutá v tomto modu budou zavedena do řetězce. Natištěné znaky se zobrazí ve čtverci v dolní části obrazovky.
Poznámka: Některá tlačítka by mohla nezobrazit symbol na nich uvedený. Aby uživatel mohl kontrolovat proces zavádění dat, je k dispozici pět uvedených speciálních funkčních tlačítek. Abyste zpřístupnili tyto funkce, stiskněte tlačítko CTRL a tlačítko SHIFT vpravo spolu s požadovaným funkčním tlačítkem.

Tlačítko	Funkce
RETURN	zakončí definici řetězce
+(hlavní kláv.)	vloží mezeru do řetězce
-(hlavní kláv.)	zruší znak na kursoru
šipka doleva	kursor doleva
šipka doprava	kursor doprava

Využití modu ALT

Mod ALT je komunikační funkcí. Jeho implicitní hodnotou je OFF. Tato funkce umožnuje zasílat kódy o 8 bitech.

Obrazovka

Obrazovka CP/M 3.0 imituje terminál ADM31. Následující funkce obrazovky imitují funkce ADM31, které jsou podsouborem operací na terminálu ADM31.

CTRL G	akustický signál
CTRL H	kursor doleva
CTRL J	kursor dolů
CTRL K	kursor nahoru
CTRL L	kursor vpravo
CTRL M	kursor na začátek běžného řádku
CTRL Z	kursor do polohy HOME a vymazání obrazovky
CSC=RC	Poloha kursoru, kde R je umístění řádku (s hodnotou mezery po 8) a C je umístění sloupce (mezery 10) vzhledem k stávajícímu řádku

Jiné funkce v modu ADM31 jsou

ESC T ESC t	zrušení řádku až do konce
ESC Y ESC y	zrušení až do konce obrazovky
ESC:	kursor do polohy HOME a vymazání

ESC*	obrazovky (včetně stávajícího řádku)
ESC Q	vkládání znaků
ESC W	zrušení znaku
ESC E	vložení řádku
ESC R	zrušení řádku

ESC ESC ESC barva stanoví jednu z 16 barev obrazovky, viz Kapitola 2., Část 6., tabulka 6-2. Číslo barvy se určí následovně:

od 20h do 2Fh	barva znaků
od 30h do 3Fh	barva pozadí
od 40h do 4Fh	barva rámečku (pouze v modu 40 sloupců)

Viditelný efekt následujících funkcí může být obdržen pouze ve formátu obrazovky s 80 sloupci:

ESC	poloviční intenzita
ESC	plná intenzita
ESC G4	aktivováno inverzní video
ESC G3	aktivováno podtržení*
ESC G2	aktivováno blikání
ESC G1	volba alternativní řady znaků*
ESC G0	deaktivuje všechny funkce ESC G

*Poznámka: toto nejsou normální sekvence ADM31.

x x x x x x x x x x

Jednotlivé části této kapitoly poskytly obecné informace o struktuře a mnohostranných možnostech CP/M 3.0.

K A P I T O L A 5

E N C Y K L O P E D I E

B A S I C u 7.0

Č Á S T 16

Struktura encyklopedie

V této kapitole ilustrujeme prvky jazyka BASIC 7.0 spolu se syntaktickými pravidly BASICu 7.0 pro Commodore 128 a povšechným popisem všech techti prvků.
BASIC 7.0 zahrnuje všechny prvky BASICu 2.0. Základní prvky BASICu uvedené v následujících sekcích se dělí na:

1. Příkazy a instrukce: instrukce používané pří tvorbě, ukládání v paměti a rušení programů a příkazy jazyka BASIC používané v šislovaných řádcích programu
2. Funkce: funkce stringů, tisku a numerické funkce.
3. Proměnné a operátory: všechny možné typy proměnných, jména přisuzovaná proměnným, aritmetické a logické operátory.
4. Rezervovaná slova a symboly: slova a symboly, rezervované pro jazyk BASIC 7.0, která nelze použít pro jiné cíle.

Příkazy a instrukce

Příkaz ——> AUTO

Popis ——> Zapíná/vypíná automatické číslování řádků

Tvar ——> AUTO [řádek#]

Vysvětlení-> Tento příkaz uvádí do chodu automatické číslování řádků. Usnadňuje operace při zavádění programu tím, že se automaticky píší čísla řádků (namísto, aby to dělal uživatel) pokaždé, když se stiskne tlačítko RETURN na konci každého řádku programu. Kursor se posune o dvě mezery vpravo od čísla řádku. Argument "řádek#" značí přírustek požadovaný mezi čísly řádků. AUTO bez uvedení argumentu "řádek#" vypíná automatické číslování, které rovněž může být deaktivováno příkazem RUN. Tento příkaz se používá pouze v přímém modu (vně programu).

Příklady:

Příklady-> AUTO 10 : čísluje automaticky řádky programu s přírustkem 10.

AUTO 50 : automaticky čísluje řádky programu s přírustkem 50.
AUTO : vypíná automatické číslování řádků.

Řádek, který ukazuje tvar příkazu, sestává z několika prvků:

DLOAD "jméno programu"[D0,U8]

klíčové slovo — argument — pomocné argumenty (nepovinné)

Části příkazu nebo instrukce, které musí být napsány přesně tak, jak jsou uvedeny, budou psány velkými písmeny. Slova dodávaná uživatelem, jako například jméno programu, budeme psát malými písmeny.
Jestliže cokoliv bude označeno uvozovkami ("") (obvykle jméno programu nebo fajlu), je nutné tyto uvozovky uvést rovněž tak, jak jsou ukázány v příkladech na tvar příkazu.
Klíčová slova, nazývaná rovněž rezervovaná slova, obsahují velká písmena. Klíčová slova je možné psát jak s použitím celého slova, tak pomocí zkratek (úplný seznam zkratek naleznete v Doplňku K). Klíčové slovo nebo zkratka musí být zavedeny do počítače přesně, jinak se objeví chybové hlášení. Chybová hlášení BASICu a DOSu jsou uvedena příslušně v Dplňcích A a B.

Klíčová slova jsou podstatnou součástí jazyka BASIC. Jsou ústřední částí příkazů a instrukcí, informujících počítač, jakou akci má provést. Tato slova nesmí být použita jako jména proměnných. Úplný seznam rezervovaných slov a symbolů naleznete v části 20.
Argumenty, nazývané rovněž parametry, budou uváděny velkými písmeny. Argumenty doplňují klíčová slova a poskytují specifické informace o příkazu nebo instrukci. Například klíčové slovo "load" nařizuje počítači přečíst program a argument obsahuje specifikované jméno programu. Druhý argument říká, z kterého drajvu se program čte. Argumenty zahrnují jména fajlů, proměnných, čísla řadků atd.
Hranaté závorky [] označují opční (nepovinné) argumenty. Uživatel buď argument zvolí, nabo ne, podle potřeby.
Ostré závorky < > znamenají, že uživatel MUSÍ zvolit jeden z vyjmenovaných argumentů.
Svislá čárka | odděluje prvky v seznamu argumentů, jestliže volba je omezena vyjmenovanými argumenty. Jestliže se svisla čárka objeví v seznamu uzavřeném do hranatých zavorek, volba je omezena na prvky seznamu, avšak uživatel může nepoužít žádný argument.

Pomlkové tečky ..... ukazují, že opce nebo argument se může opakovat vícekrát.
Uvozovky "" uzavírají stringy (řetězce) znaků, jména fajlů a jiné výrazy. Jestliže jsou argumenty v uvozovkách, musí být uvozovky uvedeny i v příkazu nebo instrukci. Uvozovky nejsou konvencí používanou pro popis formátů, nicméně jsou nedílnou součástí příkazu nebo instrukce.
Závorky (), pokud obsahují argumenty, musí následovat za příkazem nebo instrukcí. Závorky se nepoužívají obvykle pro popis tvarů, jsou však nedílnou součástí příkazu.
Proměnné jsou jakákoliv platná jména proměnných BASICu, například X, A$, T% atd.
Výrazy jsou jakékoliv platné výrazy BASICu, například A+B+2, .5*(X+3) atd.
Čárka (,), dvojtečka (:) a středník (;) musí být zahrnuty do příkazů a instrukcí jako jejich podstatná část.

Grafické a zvukové příkazy

Nepovinné (opční) parametry grafických a zvukových příkazů vypadají následovně:

[,parametr]

Jestliže parametry vynecháme, musíme napsat čárky, protože na parametry se odvoláváme v určité poloze. Nesmíme však psát čárku za

posledním ještě specifikovaným parametrem.
Příklad:

ENVELOPE n[,att][,dec][,sos][,ril][,fo][,li]

Jestliže chceme určit pouze hodnotu parametru li, napíšeme:

ENVELOPE n,,,,ril

Prvé tři čárky ukazují, že bylo vynecháno attacco, decadimento a sostegno, zatímco čtvrtá ukazuje polohu rilascio, Pro vynechaný tvar vlny (fo) a délku impulsu (li) čárky uvádět nemusíme.
V grafických příkazech, pokaždé když specifikujeme souřadnici (pomocí X a Y) můžeme nahradit tuto souřadnici vektorem (X;Y); v tomto případě X je délka vektoru a Y úhel ve stupních (0= směrem nahoru, 90= doprava atd.).
Příklad:

```
LOCATE 160,100
DRAW TO 40;45
```

nakreslí úsečku pod 45^{0} o délce 40.

Příkazy pro disk

Opční parametry v příkazech pro disk vypadají následovně:

[,parametr]

Čárka není povinná, jestliže by byla první za příkazem. Jestliže vynecháváme některé parametry vyžadující čárky, vynecháme rovněž tyto čárky.

Příklad: _

DIRECTORY [Ddraj][<ON|,> Unčíslo prostředku] [,proměnný znak]

zplodí:

DIRECTORY DO ON U8,"AB*"

Jestliže specifikujeme pouze proměnný znak, čárku vynecháme:

Příklad:

DIRECTORY "AB*"

Jestliže se v příkazu pro disk používají proměnné, musí být uzavřeny do závorek ().

Příklad:

DIRECTORY D(DV),(A$)

ČÁST 17

APPEND

APPEND #číslo logického fajlu,"jméno fajlu" [,Dčíslo drajvu][<ON|,> Uzařízení]

Tento příkaz otevírá fajl definovaný jménem fajlu a umísťuje pointer na konec tohoto fajlu. Následující příkaz záznamu PRINT# vyvolá postavení na konec uvedeného fajlu. Neuvedené hodnoty čísla drajvu a zařízení se předpokládají 0 a 8 respektive. Proměnné nebo výrazy použité jako jméno fajlu musí být v uvozovkách.

Příklady:

APPEND #8,"MUJ FAJL"	Otevírá logický fajl 8 definovaný jako "MUJ FAJL" pro nastavení následujícího příkazu PRINT#
APPEND #7,(A$),D0,U9	Otevře logický fajl definovaný proměnnou A$ na drajvu 0, číslo zařízení 9 a připraví nastavení.

AUTO

- Aktivuje/deaktivuje automatické číslování řádků

AUTO [řádek#]

Tento příkaz zapíná automatické číslování řádků. Usnadňuje to činnost při zavádění programu do paměti, protože se automaticky píší čísla řádků na potřebném místě pokaždé, když uživatel stiskne tlačítko RETURN na konci předchozího řádku programu. Kursor bude umístěn o dvě mezery vpravo od čísla. Argument "řádek#" označuje požadovaný přírůstek čísel řádků. AUTO bez argumentu "řádek#" vypíná automatické číslování, které rovněž může být zrušeno příkazem RUN. Tento příkaz se může používat pouze v přímém modu (vně programu).

Příklady:

AUTO 10	Automaticky čísluje řádky programu s přírustkem 10.
AUTO 50	Automaticky čísluje řádky programu s přírůstkem 50.
AUTO	Ruší automatické číslování řádků.

BACKUP

- Kopíruje obsah disku na jiný na dvojitém diskdrajvu

BACKUP zdrojový Dčíslo drajvu TO cílový Dčíslo drajvu [<ON|,> Uzařízení]

Tento příkaz kopíruje všechny údaje ze zdrojové diskety na cílovou disketu za použití dvojitého diskdrajvu. S příkazem BACKUP je možné použít novou disketu, aniž bychom ji museli předem formátovat, protože příkaz BACKUP kopíruje veškeré informace obsažené na disketě, včetně formátů. Z tohoto důvodu příkaz BACKUP ničí veškeré informace přítomné případně na cílové disketě. Jestliže použijeme tento příkaz s použitou disketou, přesvědčete se, že neobsahuje programy, které je třeba zachovat. Z obezřetelnosti počítač žádá ujištění (ARE YOU SURE? - jste si jistý?), dříve než operaci provede. Stiskněte tlačítko "Y" (yes - ano), aby se příkaz provedl, nebo jakékoliv jiné tlačítko, aby se zrušil. Radíme vždy pořídit kopii vašich vlastních disket na případ, že originální disketu poškodíte nebo ztratíte. Podívejte se rovněž na příkaz COPY. Číslo zařízení, pokud není uvedeno, se předpokládá 8.

Poznámka: Tento příkaz můžete používat pouze s dvojitým diskdrajvem. Tento příkaz se nemůže používat pro pořízení kopií chráněných disket (jako je většina komerčních programů).

Příklady:

BACKUP D0 TO D1 Zkopíruje veškerá data z diskety v drajvu D0 na disketu

v drajvu 1 diskdrajvového zařízení č. 8.

BACKUP D0 TO D1 ON U9 Zkopíruje veškeré informace z drajvu 0 na drajv 1 diskdrajvu 9.

BANK

- Vybírá jednu z 16 paměťových bank (od 0 do 15).

BANK číslo banky

Tento příkaz specifikuje číslo banky a odpovídající konfiguraci paměti v paměti Commodore 128. Bez uvedení čísla se předpokládá banka č. 15. Uvedeme tabulku konfigurací BANK, které jsou k dispozici v paměti Commodoru 128:

BANKA	KONFIGURACE
0	Pouze RAM(0)
1	Pouze RAM(1)
2	Pouze RAM(2)*
3	Pouze RAM(3)*
4	ROM vnitřní, RAM(0), I/O
5	ROM vnitřní, RAM(1), I/O
6	ROM vnitřní, RAM(2), I/O*
7	ROM vnitřní, RAM(3), I/O*
8	ROM vnější, RAM(0), I/O

```
 9   ROM vnější, RAM(1), I/O
10   ROM vnější, RAM(2), I/O*
11   ROM vnější, RAM(3), I/O*
12   ROM vnitřní a Kernal (dolní),RAM(0),I/O
13   ROM vnější a Kernal (dolní),RAM(0),I/O
14   ROM BASIC a Kernal,RAM(0), ROM znaků
15   ROM BASIC a Kernal, RAM(0), I/O
```

*Pro použití verze s rozšířenou pamětí vnitřní např. 256 K. V nerozšířené verzi neexistuje RAM v těchto bankách a 2 se hlásí jako 0 a 3 se hlásí jako 1.

Abyste získali přístup do určité banky,napište BANK n (n=0-15) a poté můžete použít PEEK/POKE nebo SYS. Monitor předešle hexadecimální číslici (0 - F) čtyřmístnému hexadecimálnímu číslu adresy, která je zobrazena.

BEGIN/BEND

Struktura používaná s IF ... THEN ELSE, umožňující zahrnout více programových řádků mezi počátek (BEGIN) a konec (BEND) struktury.

```
IF podmínka THEN BEGIN: příkaz
příkaz
......
příkaz BEND: ELSE BEGIN
příkaz
.........
příkaz BEND
```


Příklad:

```
10 IF X=1 THEN BEGIN: PRINT "X=1 PLATI"
20 PRINT "PROVEDE SE TATO CAST PROGRAMU"
30 PRINT "JESTLIZE X JE ROVNE 1"
40 BEND: PRINT "KONEC BLOKU BEGIN/BEND":GOTO 60
50 PRINT "X NENI ROVNO 1": PRINT "PRIKAZY MEZI
    BEGIN A BEND SE IGNORUJI"
60 PRINT "ZBYLA CAST PROGRAMU"
```

Jestliže je podmínka na řádku 10 splněna provedou se příkazy mezi klíčovými slovy BEGIN a BEND, včetně všech příkazů na řádku s BEND. Jestliže podmínka (IF .... THEN) není splněna na řádku 10, ignorují se všechny příkazy mezi BEGIN a BEND včetně těch, které jsou na jednom řádku s BEND, a program pokračuje na prvém řádku programu bezprostředně následujícím za řádkem, obsahujícím BEND. BEGIN/BEND zachází s řádky od 10. do 40. jako s jediným řádkem.

Tato pravidla platí i v případě, že se specifikuje klauzule ELSE: BEGIN. Jestliže podmínka není splněna, provedou se všechny příkazy mezi ELSE: BEGIN a BEND, včetně příkazů na řádku s BEND. Jestliže podmínka je splněna, program přejde na řádek bezprostředně následující zá řádkem obsahujícím toto BEND.

BLOAD

- Přečte binární fajl a uloží jej počínajíc specifikovanou adresou v paměti počítače.

BLOAD "jméno fajlu" [,Dčíslo drajvu][,ON]
[Učíslo zařízení] [,B číslo banky]
[,P počáteční adresa]

kde:

- jméno fajlu je jméno fajlu
- číslo banky umožňuje zvolit jednu z 16 bank
- výchozí adresa je místo v paměti, od něhož se začne ukládat fajl

Binární fajl je fajl (program nebo data), který byl uchován pomocí monitoru strojového kódu nebo pomocí příkazu BSAVE. Příkaz BLOAD uloží binární fajl v místě určeném počáteční adresou.

Příklady:

BLOAD "SPRITE",B0,P3584 Uloží binární fajl "SPRITE" počínajíc adresou 3584 banky 0

BLOAD "DATA1",D0,U8,B1,P4096 Uloží binární fajl "DATA1" na místě 4096 (BANKA 1) z drajvu č. 1, jednotky 8

Pokud počáteční adresa není uvedena, fajl se přenese na tu adresu, z níž byl přečten a uložen.

BOOT

- Přečte, uloží a provede program uložený v binárním fajlu.

BOOT ["jméno fajlu"][,Dčíslo drajvu]
[,ON][,] Uzařízení]

Příkaz zapíše proveditelný binární fajl a začne provádět program od předem určené počáteční adresy. Číslo zařízení, pokud není uvedeno, se předpokládá 8, drajv č. 0.

Příklady:

BOOT Zapíše a provede proveditelný program (např. CP/M plus)
Je to speciální případ a vyžaduje stanovit specifický sektor disku.

BOOT "GRAP",D0,U9 Zapíše a provede program "GRAP" z jednotky 9, drajv 0

Provádění programu začíná počáteční adresou řádku programu, z něhož začalo zapisování.

BOX

- Nakreslí čtverec v určitém místě obrazovky.

BOX [zdrojová barva],X1,Y1[,X2,Y2][,úhel]
[,vybarvení]

kde:

zdrojová barva - 0=barva pozadí
1=barva popředí
2= multicolor 1 } pouze v grafic.
3=multicolor 2 } modu 3 a 4

X1,Y1 - souřadnice horního levého rohu

X2,Y2 - dolní pravý roh proti X1,Y1 ; pokud není uveden, je to místo [illegible]PK

úhel - pootočení ve stupních ve směru hodinových ručiček; pokud není uvedeno bude to 0 stupňů

vybarvení - vybarvení obrázku
0 = bez barvy
1= barevný
(není uvedeno = 0)

Tento příkaz dovoluje uživateli nakreslit na obrazovce čtverec libovolných rozměrů. Pootočení se provádí kolem středu čtverce. Pixelový kursor (PK) je umístěn na X2,Y2 po provedení příkazu BOX. Číslo zdroje barvy musí být nula (0) nabo jedna (1) v modu bodové matrice, nebo 2 či 3 v modu vícebarevné bodové matrice. Viz rovněž příkaz GRAPHIC pro volbu grafického modu, užívaný s číslem zdrojové barvy.BOXu. Viz rovněž příkaz LOCATE s dalšími informacemi o pixelovém kursoru.

Příklady:

BOX 1,10,10,60,60 Nakreslí obvod čtverce

BOX 1,10,10,60,60,45,1 Nakreslí čtverec vybarvený a pootočený (kosočtverec)

BOX ,30,90,,45,1 Nakreslí pootočený mnohoúhelník

BOX 1,20,20,,,1 Kreslí plný čtverec 20 pixelů vpravo a 20 pixelů směrem dolů vzhledem k okamžité poloze pixelového kursoru

Libovolný parametr lze vynechat s podmínkou, že na jeho místě napíšeme čárku, jako v posledních dvou příkladech.
NB: X2,Y2 jsou posuzovány jako jediný parametr, takže je nutná pouze jedna čárka. Ke změně řádku dojde, jestliže hodnota ve stupních je větší než 360, takže 360=0,(450=90).

BSAVE

- Ukládá binární fajl ze specifikované části paměti.

BSAVE "jméno fajlu"[,D číslo drajvu][,U číslo zařízení][,Bčíslo banky], Padresa počáteční TO P adresa konečná

Kde:
- jméno fajlu je jméno připsané fajlu

- číslo drajvu je 0 nebo 1 pro dvojitý diskdrajv (0 pokud není uvedeno pro jednoduchý diskdrajv)
- číslo zařízení je číslo diskdrajvu (8 pokud není explicite uvedeno)
- číslo banky je specifikované číslo banky (od 0 do 15)
- počáteční adresa je startovací adresa na níž byl uložen program
- konečná adresa je poslední adresa v paměti, totiž konečná adresa musí být nadřazena odpovídajícímu bytu posledního uloženého bytu.

To je vše, co se týká užití příkazu SAVE monitoru strojového kódu.

Příklady:

BSAVE "DATA SPRAJTU",B0,P3584 TO P 4096

Uloží binární fajl nazvaný "DATA SPRAJTU" počínajíc adresou 3584 a končíc 4095 (banka č. 0)

BSAVE "PROGRAM.SCR",D0,U9,B0,P3182 TO P8000

Uloží binární fajl nazvaný "PROGRAM.SCR" z adres od 3182 do 7999 (banka č. 0) na drajv 0, jednotky č.9)

CATALOG

- Zobrazí seznam disku.

CATALOG [Dčíslo drajvu][<ON|,>Učíslo zařízení][,proměnný string]

Příkaz CATALOG zobrazí seznam specifikovaného drajvu, přesně jako příkaz DIRECTORY. Viz příkaz DIRECTORY s dalšími příklady (DIRECTORY a CATALOG jsou zaměnitelné)

Příklad:

CATALOG zobrazí seznam disku v drajvu 0, zařízení č. 8.

CHAR

- Zobrazí na obrazovce znaky v určité poloze.

CHAR [zdrojová barva][,x,y[,řetězec][,RVS]

Tato funkce je určena v zásadě k zobrazování znaků na obrazovce s bodovou matricí, avšak lze ji rovněž použít na textové obrazovce. Vysvětlíme význam parametrů:

Zdrojová barva
- 0 = pozadí
- 1 = popředí
- 2 = multicolor 1
- 3 = multicolor 2

x — sloupec znaků (0-79)(v modu se 40 sloupci přeskočí na následující řádek)

y - řádek znaků (0-24)
řetězec - tištěný řetězec
RVS - flag pole inverze (0=vypnuto, 1=zapnuto)

Text (alfanumerický řetězec) může být zobrazen na obrazovce libovolného typu a v libovolném místě pomocí příkazu CHAR. Data znaků se čtou z oblasti znaků v ROM Commodore 128. Uživatel zadá souřadnice x a y počátečního bodu a uvede řetězec zobrazovaného textu. Zdrojová barva a inverzi zobrazení není povinné uvádět. Řetězec bude pokračovat na následujícím řádku, jestliže překročí pravý okraj obrazovky. Jestliže použijeme textový mod, řádek tištěný příkazem CHAR funguje přesně jako řetězec v PRINTu s týmiž operacemi kurzoru a ovládáním barvy. Tyto řídící funkce uvnitř řetězce nepůsobí, jestliže se příkaz CHAR používá k zobrazení textu v modu bodové matrice. Také ovládání velká/malá písmena (CHR$(14) nebo CHR$(142)) fungují v modu bodové matrice.
Vícebarevné znaky se ovládají jinak, než standartní znaky. Následující tabulka ukazuje, jak dostat všechny možné kombinace:

			Flag inverze 0 (vypnut)	Flag inverze 1 (zapnut)
Zdrojová barva	0	Text	1	1
		Pozadí	2	3
Zdrojová barva	1	Text	1	0
		Pozadí	0	1
Zdrojová barva	2	Text	2	0
		Pozadí	0	2
Zdrojová barva	3	Text	3	0
		Pozadí	0	3

Příklad:

```
10 COLOR 2,3 :REM multicolor 1=červená
20 COLOR 3,7 :REM multicolor 2=modrá
30 GRAPHIC 3,1
40 CHAR 0,10,10,"TEXT",0
50 CHAR 0,10,11,"TEXT",1
```

CIRCLE

- Kreslí na obrazovce kružnice, elipsy, oblouky atd. v zadané poloze.

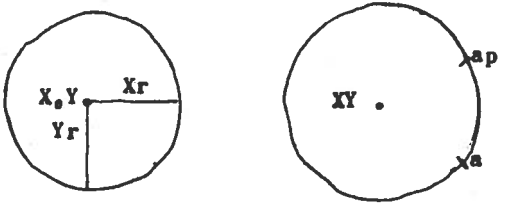
CIRCLE [zdrojová barva] ,X,Y,Xr [,Yr] [,ap] [,a] [,úhel] [,inc]

kde:

zdrojová barva - 0 = barva pozadí
1 = barva popředí

2 = multicolor 1 } grafický mod
3 = multicolor 2 } 3 a 4

X,Y - souřadnice středu kružnice
Xr - poloměr X
Yr - poloměr Y (= Xr pokud není uveden)
ap - úhel počátečního bodu oblouku (=0 stupňů pokud není uveden)
a - úhel koncového bodu oblouku (=360° pokud není uveden)
úhel - pootočení ve směru hodinových ručiček ve stupních (=0 stupňů pokud není uveden)
inc - stupně mezi segmenty (=2 stupně pokud není uvedeno)



Pomocá příkazu CIRCLE může uživatel kreslit kružnice, elipsy, oblouky, trojúhelníky, osmiúhelníky a další mnohoúhelníky. Pixelový kursor (PK) bude umístěn na obvodu kružnice na úhlu počátku oblouku. Všechny rotace

jsou vztaženy ke středu. Všechny oblouky se kreslí počínajíc výchozím úhlem do konečného úhlu ve směru hodinových ručiček. Přírůstek (inc) řídí regulárnost tvaru; při použití menších hodnot dostaneme přesnější obrazec. Při použití hodnoty dvojnásobně větší dostaneme přibližnější obrazec.
Viz rovněž příkaz LOCATE s dalšími podrobnostmi o pixelovém kursoru.

Příklady:

```
CIRCLE,160,100,65,10          nakreslí elipsu
CIRCLE,160,100,65             nakreslí kružnici
CIRCLE,60,40,20,18,,,,45      nakreslí osmiuhelník
CIRCLE,260,40,20,30,,,,90     nakreslí kosočtverec
CIRCLE,60,140,20,18,,,,120    nakreslí trojúhelník
```

Parametry lze vynechávat, avšak v takovém případě na jejich místě musíte uvést čárku. Vynechané parametry nabývají opčních hodnot.

CLOSE

- Uzavírá logický fajl.

CLOSE číslo fajlu

CLOSE
CLR
CMD

Tento příkaz uzavírá fajly používané příkazem DOPEN nebo OPEN. Číslo následující za slovem CLOSE je číslo uzavíraného fajlu.

Příklad:

CLOSE 2 Logický fajl 2 se uzavře.

CLR

- Vynuluje proměnné v programu

CLR

Tento příkaz vynuluje všechny proměnné nalézající se v paměti, ponechávajíc program beze změny. CLR se provede automaticky při provedení příkazu RUN nebo NEW.

CMD

- Převádí výstupy na obrazovku

CMD číslo logického fajlu ,seznam písma

Tento příkaz zasílá výstupy, které jsou obecně směřovány přímo na obrazovku (například příkaz PRINT produkuje vždy listink, avšak ne na obrazovce),na jiné zařízení, například dává fajl na disk, nebo na tiskárnu. Toto zařízení, případně fajl, musí proto být nejprve otevřeno. Příkaz CMD musí být následo-

ván číslem nebo číselnou proměnnou, která určuje fajl. Seznam písma může být proměnná nebo alfanumerický řetězec. Tento příkaz se používá při tisku stránek programu.

Příklad:

OPEN 1,4	Otevírá zařízení #4 (tiskárna)
CMD 1	Jakýkoliv výstup půjde nyní na tiskárnu
LIST	Výpis programu jde na tiskárnu, ne na obrazovku (rovněž slovo READY)
PRINT#1	Posílá výstup opět na obrazovku
CLOSE 1	Uzavírá fajl

COLLECT

- Uvolňuje nepřístupnou oblast na disku.

COLLECT [Dčíslo drajvu][< ON!,> Uzařízení]

Používejte tento příkaz k uvolnění na disku prostoru, který je vymezen uzavřeným fajlům a ke zrušení odkazů na tyto fajly v seznamu. Tento typ fajlů je v seznamu označen hvězdičkou za jménem. Jestliže zařízení není uvedeno, rozumí se 8.

Příklad:

COLLECT D0 Uvolní veškerý prostor, který je k dispozici, vymezený pro nepotřebné uzavřené fajly.

NB: Tento příkaz uvolní rovněž prostor vymezený pro přímý přístup. Další podrobnosti naleznete v manuálu diskdrajvu.

COLLISION

- Stanoví ovládání při přerušení při srážce sprajtů.

COLLISION typ [,příkaz]

typ - Typ přerušení:
1 = srážka sprajt/sprajt
2 = srážka sprajt/zobrazená data
3 = optické pero

příkaz - číslo řádku BASIC podprogramu

Jestliže nastane specifikovaná situace, BASIC přeruší provádění příkazů v průběhu provádění programu a provede GOSUB na řádek specifikovaný číslem. Po skončení podprogramu (který musí končit příkazem RETURN) BASIC pokračuje v provádění programu v místě, kde došlo k přerušení. Činnost přerušení trvá, dokud není specifikován příkaz COLLISION tohoto typu bez čísla řádku. Můžeme aktivovat více typů přerušení současně, avšak každý je ovládán individuálně (nejsou tedy možné žádné rekurse nebo sdružování přerušení). Příčina přerušení může trvat a vyvo-

lávat jiná přerušení během určitého časového období, ledaže se situace změní nebo jsou přerušení zrušena/deaktivována). Abyste určili, které sprajty jsou do srážky zapleteny (srážka která vyvolala poslední zásah), použijte funkci BUMP.

Příklad:

COLLISION 1,5000 Při srážce sprajt/sprajt se řízení předá podprogramu na řádku 5000.

COLLISION 1 Zruší se přerušení definované předchozím příkazem.

COLLISION 2,1000 Při srážce sprajt/data se řízení programu předá podprogramu na řádku 1000.

Poznámka: Sprajty mohou vstoupit do srážky i když se nacházejí mimo viditelné pole, ledaže by byl příkaz deaktivován.

COLOR

- Určuje barvy různých oblastí obrazovky.

COLOR zdrojové číslo, číslo barvy

Tento příkaz přiřadí barvu jedné z následujících oblastí:

Oblast	Zdroj
0	Pozadí - 40 sloupců (VIC)
1	Popředí - 40 sloupců (VIC)
2	Multicolor 1
3	Multicolor 2
4	Rámeček - 40 sloupců (VIC)
5	Barva znaků (obrazovka 40 nebo 80 sloupců)
6	Barva pozadí - 80 sloupců

Používané barvy mají čísla od 1 do 16

Kód barvy	Barva	Kód barvy	Barva
1	černá	9	oranžová
2	bílá	10	hnědá
3	červená	11	světločervená
4	azurová	12	tmavošedá
5	purpurová	13	šedá
6	zelená	14	světlezelená
7	modrá	15	světlemodrá
8	žlutá	16	světlešedá

Čísla barev ve formátu 40 sloupců

Kód barvy	Barva	Kód barvy	Barva
1	černá	9	tmavopurpurová
2	bílá	10	hnědá
3	tmavočervená	11	světločervená
4	světleazurová	12	tmavoazurová
5	světlepurpurová	13	šedá
6	světlezelená	14	světlezelená
7	tmavomodrá	15	svetlemodrá
8	světležlutá	16	světlešedá

Čísla barev ve formátu 80 sloupců.

Příklad:

COLOR 0,1 Změní barvu pozadí na černou (40 sloupců)

COLOR 5,8 Změní barvu znaků na žlutou

CONCAT

- Spojuje dva datové fajly.

CONCAT "fajl 2"[,Dčíslo drajvu] TO "fajl 1"
[,Dčíslo drajvu][<ON[,> Uzařízení]

Příkaz CONCAT připojí fajl 2 na konec fajlu 1 a zachová jméno fajlu 1. Pokud není uvedeno, číslo zařízení se předpokládá 8 a číslo drajvu - 0.

Příklad:

```
CONCAT "FILE B" TO "FILE A"   FILE B bude při-
                              pojen k FILE A a
                              výsledný fajl bude
                              nazván FILE A

CONCAT (A$) TO (B$),D1,U9     Fajl označený B$
                              se stává novým faj-
                              lem s týmž jménem
                              a s fajlem označe-
                              ným A$ připojeným
                              na konec fajlu B$
                              - to vše bude pro-
                              vedeno na jednotce
                              č. 9, drajv 1 (dvo-
                              jitý diskdrajv)
```

Jestliže se jako jméno fajlu použije proměnná, jako v posledním příkladu, proměnná obsahující jméno fajlu musí být uzavřena v závorkách.

Poznámka: Používejte krátká jména fajlů (maximálně 10 znaků), protože řídící buffer na disku je omezený.

CONT

- Pokračuje v provádění programu.

CONT

Tento příkaz se používá, abychom znovu uvedli do chodu program, který byl přerušen

tlačítkem STOP, příkazem STOP nebo příkazem END. Program pokračuje v místě, kde byl přerušen. CONT nevyvolá pokračování v provádění programu jestliže byl změnen řádek, jestliže byl nějaký řádek přidán k programu nebo jestliže na obrazovce se uskuteční nějaká operace, modifikující program. Jestliže byl program přerušen v důsledku chyby; nebo jestliže dojde k chybě dříve, než se pokusíme znovu obnovit provádění programu, CONT nemá žádný účinek. V tomto případu se objeví chybové hlášení: CAN'T CONTINUE ERROR (chyba - nemohu pokračovat).

COPY

- Kopíruje fajl z jednoho drajvu na jiný fajl na jiném drajvu dvojitém nebo jednoduchém.

COPY <["jméno zdrojového fajlu"][,Dčíslo drajvu]> TO <["jméno cílového fajlu"] [,Dčíslo drajvu][<ON|,> Uzařízená]

Tento příkaz zkopíruje fajl z disku (zdrojový fajl) na jiný (cílový fajl), za použití dvojitého diskdrajvu. Sjednoduchým drajvem

je možné vytvořit kopii fajlu na téže disketě, používajíc však jiné jméno fajlu. Jméno fajlu ovšem může být stejné, jestliže se kopírování provádí z jednoho drajvu na jiný. Příkazem COPY můžeme rovněž kopírovat všechny fajly z jednoho disku na jiný s dvojitým diskdrajvem. V tomto případě musí být specifikována čísla diskdrajvů a jména zdrojového a cílového fajlu se vynechají. Neuvedené parametry příkazu COPY se předpokládají: 8 pro číslo zařízení a 0 pro drajv.
Poznámka: Kopírování mezi dvěma jednotkami jednoduchých nebo dvojitých drajvů není možné. Viz BACKUP.

Příklady:

COPY "pokus",D0 TO "progpokus",D1 Zkopíruje "pokus" z drajvu 0 n na drajv 1 a přiřadí jméno "progpokus" fn fajlu na drajvu 1.

COPY "program",D0 TO "program", D1 zkopíruje "program" z drajvu 0 na drajv 1

COPY D0 TO D1 Zkopíruje všechny fajly z drajvu 0 na drajv 1.

COPY "prog.lov" TO "backup" Zkopíruje "prog.lov" jako fajl nazvaný "backup" na tentýž disk (drajv 0).

DATA

- Definuje data používaná programem.

DATA seznam konstant

Za tímto příkazem následuje seznam datových prvků pro zavedení do paměti počítače příkazem READ. Prvky mohou být čísla nebo řetězce a jsou odděleny čárkami. Stringová data nemusí být nutně v uvozovkách, ledaže obsahují jeden z následujících znaků: mezera, dvojtečka, čárka. Jestliže mezi dvěma čárkami není žádný údaj, bude přečten jako nula pokud je považován za číselnou hodnotu, nebo jako prázdný řetězec. Podívejte se rovněž na příkaz RESTORE, který umožňuje Commodoru 128 znovu číst táž data.

Příklad:

DATA 10,200,MARIO,"AHOJ HONZO",,š,lč,ABCL23

DCLEAR

- Vyprázdní všechny kanály otevřené pro diskdrajv.

DCLEAR [Dčíslo drajvu][<ON|,> Uzařízení]

D O D A T E K E

Kódy ASCII a CHR$

V tomto dodatku uvádíme znaky, které se zobrazí při použití příkazu PRINT CHR$(X) pro všechny dovolené hodnoty X. Mimoto jsou v dodatku uvedeny číselné hodnoty, které obdržíme příkazem PRINT ASC("X"), kde "X" je libovolný zobrazitelný znak. Používá se to při zobrazení znaku přijímaného v příkazu GET, pro konverzi malých a velkých písmen a pro tisk příkazů založených na těchto znacích (například aktivování modu velká/malá písmena), které nelze uzavřít do uvozovek.

PRINT	CHR$	PRINT	CHR$	PRINT	CHR$
	0		11		21
	1		12		22
	2	RETURN	13		23
	3	Dolní			24
	4	poloha	14		25
—	5		15		26
	6		16		27
	7	CRSR ↓	17	HELP	28
SHIFT Cx	8	—	18	CRSR →	29
SHIFT Cx	9	HOME	19	—	30
	10	INST	20	—	31

PRINT	CHR$	PRINT	CHR$	PRINT	CHR$
PACE	32	:	58	T	84
!	33	;	59	U	85
"	34	<	60	V	86
#	35	=	61	W	87
$	36	>	62	X	88
%	37	?	63	Y	89
&	38	@	64	Z	90
.	39	A	65	[	91
(	40	B	66	£	92
)	41	C	67	]	93
*	42	D	68	↑	94
+	43	E	69	←	95
,	44	F	70		96
-	45	G	71		97
.	46	H	72		98
/	47	I	73		99
0	48	J	74		100
1	49	K	75		101
2	50	L	76		102
3	51	M	77		103
4	52	N	78		104
5	53	O	79		105
6	54	P	80		106
7	55	Q	81		107
8	56	R	82		108
9	57	S	83		109

PRINT	CHR$
	110
	111
	112
	113
	114
	115
	116
	117
	118
	119
	120
	121
	122
	123
	124
	125
	126
	127
	128
oranž.	129
	130
	131
	132
F1	133
F3	134
F5	135
F7	136

PRINT	CHR$
F2	137
F4	138
F6	139
F8	140
SHIFT RETURN	141
Horní poloha	142
	143
	144
CRSR	145
	146
HOME	147
INST	148
hnědá	149
červ.	150
tm šedá	151
šedá	152
zelen	153
modrá	154
sv šedá	155
	156
CRSR	157
	158
	159
SPACE	160
	161
	162

PRINT	CHR$
	163
	164
	165
	166
	167
	168
	169
	170
	171
	172
	173
	174
	175
	176
	177
	178
	179
	180
	181
	182
	183
	184
	185
	186
	187
	188
	189

PRINT	CHR$
	190
	191

Kódy 192-233 jako 96-127
Kódy 224-254 jako 160-190
kód 255 jako 126

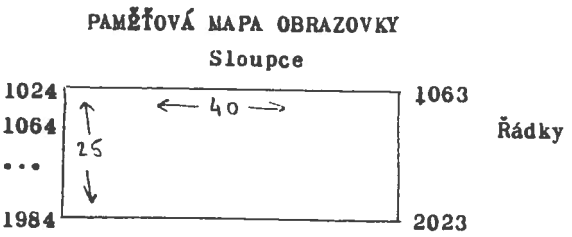
Poznámka: Uvedené kódy jsou pro mod C 64.
Speciální kódy pro mod C 128 viz Dodatek I.

DODATEK F

Mapa paměti obrazovky a barev - mod C 128, 40 sloupců a mod C 64.

Mapa ukazuje paměťové buňky používané v modu se 40 sloupci (C 128 a C 64) k uložení znaků pro obrazovku a jejich barvy. Každá mapa je řízena nezávisle a obsahuje 1000 pozicí. Znaky zobrazované na mapě lze přímo ovládat pomocí příkazu POKE.

PAMĚŤOVÁ MAPA OBRAZOVKY

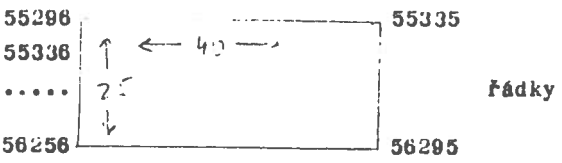


Paměťovou mapu obdržíme pomocí příkazu POKE následovaného číselnou hodnotou zobrazovaného znaku (viz Dodatek D), Například

POKE 1024,13

zobrazí písmeno M v levém horním rohu obrazovky.

PAMĚŤOVÁ MAPA BAREV



Jestliže použijeme pro paměťovou mapu barev příkaz POKE následovaný číslem barvy, způsobí to změnu barvy znaku. Například

POKE 55296,1

změní barvu písmene M na bílou.

Kódy barev - 40 sloupců

0	černá	8	oranžová
1	bílá	9	hnědá
2	červená	10	světločervena
3	azurova	11	tmavošeda
4	purpurova	12	šeda
5	zelená	13	světlezelená
6	modrá	14	světlemodrá
7	žlutá	15	světlešedá

Paměťová buňka určující barvu rámečku je 53280

Paměťová buňka určující barvu pozadí je 53281.

D O D A T E K G

Odvozené trigonometrické funkce

Funkce	Ekvivalent v BASICu
sekans	SEC(X)=1/COS(X)
kosekans	CSC(X)=1/SIN(X)
konatngens	COT(X)=1/TAN(X)
arkussinus	ARCSIN(X)=ATN(X/SQR(1-X*X))
arkuskosinus	ARSCOS(X)=-ATN(X/SQR(1-X*X))
arkussekans	ARSEC(X)=ATN(X/SQR(X*X-1))
arkuskosekans	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1)*π/2
arkuskotangens	ARCOT(X)=ATN(X)+π/2
sinus hyperbolický	SINH(X)=(EXP(X)-EXP(-X))/2
kosinus hyperbolický	COSH(X)=(EXP(X)+EXP(-X))/2
tangens hyperbolický	TANH(X)=2*EXP(-X)/(EXP(X)+EXP(-X))+1
sekans hyperbolický	SECH(X)=2/(EXP(X)+EXP(-X))
kosekans hyperbolický	CSCH(X)=2/(EXP(X)-EXP(-X))
kotangens hyperbolický	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2
arkussinus hyperbolický	ARCSINH(X)=LOG(X+SQR(X*X+1))
arkuskosinus hyperbolický	ARCCOSH(X)=LOG(X+SQR(X*X-1))

Arkustangens hyperbolický	ARCTANH(X)=LOG((1+X)/(1-X))/2
arkussekans hyperbolický	ARCSECH(X)=LOG(SQR(1-X*X) +1/X)
arkuskosekans hyperbolický	ARCCSCH(X)=LOG(SGN(X)* SQR(X*X+1)/X)
arkuskotangens hyperbolický	ARCCOTH(X)=LOG((X+1)/(X-1))/2

DODATEK H

Mapa systémové paměti

RAM C 128		ROM C 128	
FFFF	NMI RST IRQ kódy RAM CP/M kódy RAM Krnl	FFFF	Tabulka skoků Kernal a hardvérové vektory
FF05	Registry konf. MMU	FF4D	
FF00		FF05	Zasílací kódy přerušení Kernal
		FF00	Registry konf. MMU
	Oblast textu BASIC (text BASIC začíná na $1C00 jestliže není aktivována bodová matrice)	FC80	ROM rezervovaná pro verze cizích jazyků
		FA00	Tabulka editoru
		E000	Kódy ROM Kernal
		D000	Prostor I/O
		C000	Kódy ROM editor
		B000	ROM monitor
4000		4000	Kódy ROM BASIC

RAM C 128

4000	
	Obrazovka VIC a bodova matrice
2000	
	Obrazovka VIC barev (Vm#2)
1C00	
	Rezervovano pro softvér funkčních tlačátek
1800	
	Rezervovano pro systém s cizím jazykem
1400	
1300	
	Proměnné BASIC absolutní
1200	
	Proměnné BASIC DOS/VSP
1108	
	Kódy RESET CP/M
1100	
	Buffer funkčních tlačítek
1000	
	Oblast sprajtů
0E00	
	Buffer otputu RS-232
0D00	
	Buffer inputu RS-232
0C00	
	stran. zazn. na disk
0BC0	
	Buffer kazety
0B00	
	absolutní hodnoty monitoru a Kernalu

0A00	
	čas provádění stack BASIC
0800	
	Textova obrazovka VIC (Vm#1)
0400	
	kódy RAM BASIC
0380	
	Tabulka Kernal
033C	
	adresy
02FC	
	Kódy RAM Kernal
02A2	
	buffer inputu BASICu a monitoru
0200	
	systémový stack
0149	
	použití BASIC DOS
0110	
	BUFFER F
0100	
	Kernal Z.P.
0090	
	Basic Z.P.
0002	
0000	

D O D A T E K I

Řídící kódy ESC

Řídící kódy

CHR$	Sled tlačítek	Funkce	Mod C64	Mod C128
CHR$(2)	CTRL B	Podtržení (8)		x
CHR$(5)	CTRL 2 CTRL E	Určuje znaku bílou barvu	x	x
CHR$(7)	CTRL G	Generuje akustický signál		x
CHR$(8)	CTRL H	Deaktivuje změnu souboru znaků	x	
CHR$(9)	CTRL I	Aktivuje změnu souboru znaků Umístí kursor na následující tab.	x	x
CHR$(10)	CTRL J	Vrátí kursor a posune řádek Posune řádek	x	x
CHR$(11)	CTRL K	Deaktivuje změnu souboru znaků		x
CHR$(12)	CTRL L	Aktivuje změnu souboru znaků		x
CHR$(13)	CTRL M	Nařídí počítači návrat kursoru a posunutí řádku a uloží řádek BASIC	x	x
CHR$(14)	CTRL N	Aktivuje soubor znaků velká/malá písmena	x	x
CHR$(15)	CTRL O	Aktivuje blikání (80)		x

CHR$	Sled tlačítek	Funkce	Mod C64	Mod C128
CHR$(17)	CRSR dolů CTRL Q	Přemístí kursor o jeden řádek dolů	x	x
CHR$(18)	CTRL 9	Tiskne inverzní znaky	x	x
CHR$(19)	HOME CTRL S	Přemístí kursor "domů" (levý horní roh) na obrazovce (v běžném okně)	x	x
CHR$(20)	DEL CTRL T	Vymaže poslední natištěný znak a posune o jednu mezeru vlevo všechny znaky napravo od vymazaného znaku	x	x
CHR$(24)	CTRL X	Zavádí/anuluje tabulační body		x
CHR$(27)	ESC CTRL [	Zavede znak ESC		x
CHR$(28)	CTRL 3 CTRL/	Stanoví červenou barvu pro znaky (80) a (40)	x	x
CHR$(29)	CRSR CTRL]	Přemístí kursor o jeden sloupec doprava	x	x
CHR$(30)	CTRL 6 CTRL ↑	Stanoví zelenou barvu pro znaky (40) a (80)	x	x
CHR$(34)	"	Vytiskne uvozovky " na obrazovce a uvede editor do podřízeného modu	x	x

CHR$	Sled tlačítek	Funkce	Mód C64	Mód C128
CHR$(129)	Cx 1	Stanoví oranžovou (40) nebo tmavopurpurovou (80) barvu znaků	x	x
CHR$(130)		Deaktivuje podtržení (80)		x
CHR$(131)		Provede program. Tento kod CHR$ nefunguje s PRINT CHR$(131), ale z bufferu klávesnice	x	x
CHR$(133)	F1	Kód CHR$ rezervovaný pro tlačítko F1	x	x
CHR$(134)	F3	Kód CHR$ rezervovaný pro tlačítko F3	x	x
CHR$(135)	F5	Kód CHR$ rezervovaný pro tlačítko F5	x	x
CHR$(136)	F7	Kód CHR$ rezervovaný pro tlačítko F7	x	x
CHR$(137)	F2	Kód CHR$ rezervovaný pro tlačítko F2	x	x
CHR$(138)	F4	Kód CHR$ rezervovaný pro tlačítko F4	x	x
CHR$(139)	F6	Kód CHR$ rezervovaný pro tlačítko F6		
CHR$(140)	F8	Kód rezervoyaný pro tlačítko F8	x	x

CHR$	Sled tlačítek	Funkce	Mód C64	Mód C128
CHR$(141)	SHIFT RETURN	Vyvolá navrat vozíku a přechod na nový řádek, aniž by se uložil řádek BASICU	x	x
CHR$(142)		Zavede soubor znaků velká písmena/grafika	x	x
CHR$(143)		Deaktivuje blikání (80)		x
CHR$(144)	CTRL 1	Zavede černou barvu pro znaky (40) a (80)	x	x
CHR$(145)	CRSR nahoru	Přemístí kursor nebo umístí tisk nad řádek	x	x
CHR$(147)	HOME	Vymaže obrazovku a umístí kursor v levém horním rohu obrazovky	x	x
CHR$(146)	CTRL 0	Ukončí zobrazení inverzního pole	x	x
CHR$(148)	INST	Určí tmavošedou (40) nebo tmavomodrou barvu pro znaky (80)	x	x
CHR$(149)	Cx 2	Stanoví šedou barvu pro znaky (40) a (80)	x	x
CHR$(150)	Cx	Stanoví světlezelenou barvu znaků (40) a (80)	x	x
CHR$(151)	Cx 4	Stanoví světlemodrou barvu znaků (40) a (80)	x	x

CHR$	Sled tlačítek	Funkce	Mod C64	Mod C128
CHR$(152)	Cx 5	Stanoví světlešedou barvu znaků (40) a (80)	x	x
CHR$(153)	Cx 6	Stanoví purpurovou barvu znaků (40) a (80)	x	x
CHR$(154)	Cx 7	Přenese kursor vlevo o jeden sloupec	x	x
CHR$(155)	Cx 8	Stanoví azurovou (40) nebo světleazurovou (80) barvu znaků	x	x
CHR$(156)	CTRL 5	Posune znaky, počínajíc polohou kursoru, vpravo o jeden sloupec	x	x
CHR$(157)	CRSR vlevo	Stanoví hnědou (40) nebo tmavožlutou (80) barvu znaků	x	x
CHR$(158)	CTRL 4	Stanoví světločervenou barvu znaků (40) a (80)	x	x

Poznámka: (40) ... pouze pro zobrazení se (40) sloupci

(80) ... pouze pro zobrazení se (80) sloupci.

Kódy ESC

Dále uvedeme řadu funkcí ESC , které jsou k dispozici na Commodoru 128. Posloupnost ESC se zavádí stlačením a puštěním tlačítka "ESC" a stisknutím jednoho dalšího tlačítka.

Funkce ESC	Tlačítko ESC
Anuluje mod podílový a vkládání	ESC O
Vymaže až do konce běžný řádek	ESC Q
Vymaže od začátku běžný řádek	ESC P
Vymaže až do konce obrazovku	ESC @
Přenese na začátek běžného řádku	ESC J
Přenese na konec běžného řádku	ESC K
Aktivuje mod autovkládání	ESC A
Deaktivuje mod autovkládání	ESC C
Zruší běžný řádek	ESC D
Uloží řádek	ESC I
Stanoví tabulační body (8 mezer)	ESC Y
Zruší všechny tabulační body	ESC Z
Aktivuje "klouzání" obrazovka	ESC L
Deaktivuje klouzání obrazovky	ESC M
Klouzání nahoru	ESC V
Klouzání dolů	ESC W
Aktivuje akustický signál (s CTRL G)	ESC G
Deaktivuje akustický signál	ESC H

Funkce ESC	Tlačítko ESC
Uvede kursor do modu bez blikání	ESC E
Uvede kursor do modu s blikáním	ESC F
Umístí dolní část okna v poloze kursoru	ESC B
Umístí horní část okna v poloze kursoru	ESC T
Převádí do zobrazení se 40 sloupci, jestliže jsme byli v 80 sloupcích a do modu 80 sloupců, jestliže jsme byli v 40 sloupcích	ESC X

Posloupnost ESC zobrazená níže platí pouze pro zobrazení s 80 sloupci (viz Část 8 - podrobnosti o použití modu s 80 sloupci).

Zavede kursor ve tvaru pomlčky	ESC U
Zavede kursor ve tvaru čtverečku	ESC S
Zavede inverzní video	ESC R
Vrací obrazovku do normálního (neinverzního) stavu	ESC N

D O D A T E K J

Monitor strojového kódu

ÚVOD

Commodore 128 je vybaven programem pro zabudovaný monitor strojového jazyka, který umožňuje uživateli psát a účinně používat programy ve strojovém jazyku. MONITOR Commodoru 128 sestává z monitoru strojového jazyka, miniassembleru a disassembleru. Zabudovaný monitor funguje pouze v modu C 128, se 40 i 80 sloupci.

Programy ve strojovém jazyce, psané pomocí monitoru Commodoru 128 mohou být použity samostatně, nebo jako velmi rychlé subroutiny v normálním programu BASICu. Monitor Commodoru 128 tedy může pracovat spolu s BASICem. Programy v jazyce assembler je nutné umístit v paměti počítače tak, aby na ně program BASIC nemohl být "naložen".

Monitor BASICu zapnete příkazem MONITOR RETURN

Příkazy monitoru Commodoru 128

ASSEMBLE	Assembluje řádek kódu 8502.
COMPARE	Porovná dvě sekce paměti a indikuje rozdíl mezi nimi.
DISASSEMBLE	Disassembluje řádek kódu 8502.
FILL	Zaplní úsek paměti určeným bytem.
GO	Začne provádět program počínajíc určenou adresou.

HUNT Hledá v určené části paměti kolikrát se tam vyskytne řada bytů.

JUMP Přeskočí do podprogramu

LOAD Přečte fajl z kazety nebo disku

MEMORY Zobrazí hexadecimálně obsah paměťových buněk

REGISTERS Zobrazíregistry 8502

SAVE Uloží na kazetu nebo disk

TRANSFER Přenese kód z jedné části paměti do jiné

VERIFY Porovna obsah paměti s obsahem kazety nebo disku

EXIT Deaktivuje monitor Commodoru 128

(tečka) Assembluje řádek kódu 8502

> Mění paměť

; (středník) Změní zobrazení registru 8502

@ Zobrazí stav disku, pošle příkaz disku, zobrazí jeho obsah.

Commodore 128 zobrazuje hexadecimální adresy v monitoru strojového jazyka 5 číslicemi. Obecně, hexadecimální číslo representující povolenou hodnotu adresy sestává ze 4 číslic. Dodatečna pátá číslice vlevo specifikuje paměťovou konfiguraci (BANK) (v okamžiku v němž byl daný příkaz proveden)). Tabulka paměťových konfigurací:

0 = pouze RAM 0
1 = pouze RAM 1
2 = pouze RAM 2
3 = pouze RAM 3
4 = ROM INT, RAM 0, I/O
5 = ROM INT, RAM 1, I/O
6 = ROM INT, RAM 2, I/O
7 = ROM INT, RAM 3, I/O
8 = ROM EXT, RAM 0, I/O
9 = ROM EXT, RAM 1, I/O
A = ROM EXT, RAM 2, I/O
B = ROM EXT, RAM 3, I/O
C = KERNAL+INT(dolní), RAM 0, I/O
D = KERNAL+EXT(dolní), RAM 1, I/O
E = KERNAL+BASIC, RAM 0, ROM CAR
F = KERNAL+BASIC, RAM 0, I/O

Souhrn deskriptorů pole monitoru

Následující deskriptory předcházejí datová pole monitoru (popř. přepis paměti). Jestliže se tyto deskriptory použijí jako příkazy, dojde ke změně obsahu paměti nebo registru monitorem prostřednictvím určených dat.

. <tečka> předchází řádku disassemblovaného kodu

> <ostrá závorka> předchází řádek pro přepsání paměti

; <středník> předchází řádek pro přepsání registru

Následující deskriptory předcházejí číselným polím (například adresa) a uvádějí číselný základ těchto čísel. Jestliže se použijí jako příkazy, přikazují deskriptory monitoru zobrazit takovou hodnotu.

<nic> před hexadecimálním číslem

$ <dolar> před hexadecimálním číslem (základ 16)

+ <plus> před desítkovým číslem (zaklad 10)

& <komerční a> před oktálovým číslem (základ 8)

% <procenta> před binárním číslem (základ 2)

Následující znaky používá monitor jako oddělovače polí nebo terminátory řádků (s vyjímkou uvnitř řádku ASCII).

<mezera> , oddelovač - odděluje dvě pole

, <čárka> , oddělovač - odděluje dvě pole

: <dvojtečka> , terminátor - logický konec řádku

? <otazník> , terminátor - logický konec řádku

Poznamka: Symbol <> uzavírá požadované parametry, symbol [] uzavíra nepovinné parametry.

Popis příkazů monitoru Commodoru 128

Pamatujte, že jakékoliv číselné pole (např. adresa, číslo zařízení, datový byte) může být zadáno ve tvaru mocniny. Platí to rov-

něž pro pole operandu příkazu ASSEMBLE. Viz rovněž příručku syntaxe seznamu příkazů pro disk. Jestliže pracujete s monitorem, můžete používat automatickou funkci chybových hlášení Kernal. Znamená to, že Kernal zobrazí I/O ERROR a kód chyby pokaždé, když monitor nalezne I/O chybu. Tato funkce je deaktivována po vystoupení z monitoru.

Příkaz: A

Účel: Uvádí řádek v kódu assembleru

Syntax: A <adresa> <mnemotechnický kód operace> <operand>

adresa - číslo indikující paměťovou bunku do níž se zavádí operační kod

mnemot. operační kód - standartní MOS kód jazyku assembler, například LDA, STX, ROR.

operand - operandem, pokud je požadován, může být libovolný způsob dovoleného adresování.

Pro označení konce assemblovaného řádku se používá RETURN. Jestliže jsou v řádku chyby, objeví se otazník ukazující na chybu a kursor se přenese na následující řádek. Pro opravení chyby nebo chyb na tomto řádku můžete použít editorovou obrazovku.

Příklad:

```
.A01200LDX#$00
.A01202
```

Poznámka: tečka (.) je ekvivalentní příkazu ASSEMBLE.

Příklad:

.02000LDA#$23

Příkaz: C

Účel: porovná dvě oblasti paměti

Syntax: C<adresa 1> <adresa 2><adresa 3>

<adresa1> číslo udávající počáteční adresu porovnávané paměti

<adresa 2> hexadecimální číslo udávající konečnou adresu porovnávané části paměti

<adresa 3> číslo udávající počáteční adresu druhé porovnávané části paměti. Na obrazovce budou zobrazeny všechny adresy, které nejsou shodné.

Příkaz: D

Účel: Disassembluje strojový kód v mnemotechnický a operandy jazyka assembleru

Syntax: D[<adresa 1>][<adresa 2>]

adresa 1 číslo určující adresu, od níž má začít disassemblování

adresa 2 koncová adresa (nepovinná) kodu, který má být disassemblován

Tvar výsledku disassemblování obdrženého tímto způsobem se poněkud liší od běžného assembleru. Prvním znakem je tečka, kdežto

A

Výpis disassemblovaných příkazů může být změněn pomocí obrazkového editoru. Proveďte změny mnemotechnických kódů nebo operandů na obrazovce a poté stiskněte tlačítko RETURN. Tímto způsobem uložíte znovu řádek a znovu se vyvolá assembler, aby bylo možné provést další změny. Disassemblování se může provádět rovněž po stránkách. Jestliže natiskneto D RETURN zobrazí se vždy následující stránka disassemblovaného programu.

Příklad:

```
D 3000 3003
 .03000 A900        LDA#$00
 .03002 FF          ???
 .03003 D02B        BNE $3030
```

Příkaz: F

Účel: Uloží určený byte do řady paměťových buněk

Syntax: F<adresa 1> <adresa 2> <byte>

<adresa 1> první buňka, do níž se má uložit <byte>

<adresa 2> poslední buňka, do niž se má uložit <byte>

<byte> číslo, které se má uložit

Tento příkaz se používá pro inicializování datových struktur nebo jiné oblasti RAM.

Příklad:

F 0400 0518 EA

Zaplní se paměťové buňky od $0400 do $0518 hodnotou $EA (příkaz NOP).

Příkaz: G

Účel: Začne provádět program od určené adresy.

Syntax: G [<adresa>]

<adresa> adresa od níž má začít provádění programu. Jestliže adresa není uvedena, provádění začne od běžného PC (běžný PC můžeme můžeme zobrazit pomocí příkazu R).

Příkaz GO obnoví všechny registry (které lze zobrazit pomocí příkazu R) a začne provádět program od určené počáteční adresy. Doporučujeme používat příkaz GO obezřetně. Abyste se po provedení programu ve strojovém kódu vrátili do modu monitor Commodoru 128, použijte na konci programu příkaz BRK.

Příklad:

G 140C

Začne se provadět program od adresy $140C.

Příkaz: H

Účel: Hledá v určené části paměti řadu bytů pokaždé, když je porovnává.

Syntax: H<adresa 1><adresa 2><data>

<adresa 1> počáteční adresa pro proces vyhledávání

<adresa 2> koncová adresa pro proces vyhledávání

<data> data, která mají být vyhledána mohou být čísla nebo řádek ASCII

Příklad:

H A000 A101 A9 FF 4C

Hledá data $A9, $FF, $4C od A000 do A 101

H 2000 9800 'CASSA'

Hledá řetězec CASSA.

Příkaz: L

Účel: Zapíše do paměti fajl z kazety nebo z disku.

Syntax: L<"jméno fajlu">[,<zařízení>[,adresa zastavení záznamu]]

<"jméno fajlu"> libovolné jméno fajlu dovolené pro Commodore 128

<zařízení> číslo zařízení, z něhož se zapisuje, 1=kazeta, 8=disk (nebo 9,A atd.)

<adresa zastavení záznamu> nepovinná, pro závznam fajlu až do určené adresy

Příkaz LOAD přepíše fajl do paměti. Jestliže není použitá jiná adresa záznamu, fajl se zaznamená na adrese v bance 0., která je uvedena v záhlaví na kazetě nebo v prvních dvou bytech fajlu na disku. Adresa záznamu se uvádí, jestliže chceme fajl zapsat počínajíc jinou adresou, jejíž hodnota však musí ležet mezi $0000 a $FFFF.

Příklad:

L "PROGRAM",$12000 Zaznamená fajl nazvaný PROGRAM z disku do banky č. 1 počínajíc adresou $2000.

Příkaz: M

Účel: Zobrazí paměť jako soubor hexadecimálních nebo ASCII znaků pro zadané adresy.

Syntax: M [<adresa 1>] [<adresa 2>]

<adresa 1> první adresa v paměti. Nepovinná. Jestliže tato adresa není uvedena, zobrazí se stránka. První byte je číslo banky, která se má zobrazit, další čtyři dávají adresu, která se má zobrazit.

<adresa 2> poslední adresa v paměti Nepovinná. Jestliže tato adresa není uvedena, zobrazí se stránka. První byte je číslo zobrazované banky, následu- poslední adresu, která ještě má být zobrazena.

Paměť se zobrazí v následujícím tvaru:

> 03000 45 58 2E 56 41 46 55 45: EX.VALUE

Obsah paměti lze měnit pomocí obrazovkového monitoru. Umístěte kursor na data, která chcete změnit, vytiskněte požadovanou opravu a stiskněte RETURN. V případě chybné buňky RAM nebo pokusu změnit ROM se objeví chybové návěští (?). Vpravo od hexadecimálních dat se objeví ASCII data zobrazená inverzně (aby se odlišila od dat zobrazených na obrazovce). Jestliže nějaký znak nemůže být zobrazen, objeví se na jeho místě inverzní zobrazení tečky. Jako v případě příkazu DISASSEMBLE, posun obrazovky směrem dolů se provede stisknutím M a RETURN.

Příklad:

M F41F1 F4201

> F41F1 20 43 4F 4D 4D 4F 44 4F: COMMODO
> F41F9 52 45 20 45 4C 45 43 54: RE ELECT
> F4201 52 4F 4E 49 43 53 2C 20: RONICS,

Uvedené zobrazení bude produkováno pro 40 sloupců.

Příkaz: R

Účel: Zobrazí důležité registry 8502. Zobrazí se: stavový registr programu, počítač programu, akumulátor, inde-

xové registry X a Y a stack pointer.

Syntax: R

Příklad:

R

```
   PC  SR AC XR YR SP
;01002 01 02 03 04 F6
```

Poznámka: může být použit symbol ; (středník) pro změnění zobrazení registrů, přesně stejně jako může být použito pro změnění paměťových registrů.

Příkaz: S

Účel: Uloží určenou oblast paměti na kazetu nebo disk.

Syntax: S <"jméno fajlu">, <zařízení>, <adresa 1>, <adresa 2>

<"jméno fajlu">	jakékoliv jméno fajlu přípustné pro Commodore 128. Pro uložení dat jméno fajlu musí být uzavřeno v uvozovkách
<zařízení>	číslo určující zařízení, na nějž má být přenesen fajl. Kazeta=1, disk=8,9
<adresa 1>	počáteční adresa v paměti, která se má uložit
<adresa 2>	podlední adresa v paměti, která se ještě uloží, + 1

Všechna data až po tuto adresu a ji vyjímaje budou uložena.

Fajl může být znovu vyvolán pomocí příkazu L. Jestliže se uložení provádí na kazetu, bude to možné pouze z banky 0.

Příklad:

S"HRA"8,0400,0C00

Uloží na disku obsah paměti od $0400 po $0C00.

Příkaz: T

Účel: Přenáší segmenty paměti z jedné části paměti do druhé.

Syntax: T <adresa 1> <adresa 2> <adresa 3>

<adresa 1>	počáteční adresa dat, která mají být přenesena
<adresa 2>	koncová adresa dat, která mají být přenesena
<adresa 3>	počáteční adresa oblasti do níž budou data přenesena

Data mohou být přenesena z horní paměti do dolní a naopak. Další segmenty paměti o libovolné délce mohou být přesunuty z jedné paměti do druhé. Provede se automaticky "porovnání" během přenosu z bytu a všechny rozdíly budou vyjmenovány za adresou.

Příklad:

T 1400 1600 1401	Počínajíc adresou $1400 do a včetně) adresy $1600 se data v pamětí posunou o jeden byte.

Příkaz: V

Účel: Porovná fajl na kazetě nebo na dis s obsahem paměti počítače.

Syntax: V<"jméno fajlu">[,<zařízení>[,<počáteční adresa>]]

<"jméno fajlu">	libovolné jméno fajlu dovolené p. Commodore 128
<zařízení>	číslo zařízení, n němž se nachází fajl: kazeta=1, disk=8,9 atd.
<počáteční adresa>	nepovinná; od této adresy se začne porovnávat

Příkaz VERIFY porovná fajl s obsahem paměti počítače. Commodore 128 napíše VERIFYING. Jestliže narazí na chybu, objeví se rovněž slovo ERROR, jestliže je výsledek porovnání kladný, objeví se znovu kursor.

Příklad:

V "ZAZNAM",8

Příkaz: X

Účel: Návrat k BASICu.

Syntax: X

Příkaz: > (větší než)

Účel: Může být použit pro označení konce osmi nebo šestnácti buněk.

Syntax: >[<adresa>[<datové byty..8/16>]]

<adresa>	prvá adresa paměti
<datové byty..8/16>	data k zavedení do paměťových buněk následujících za adresou s jednou mezerou před každým novým bytem.

Maximální počet bytů, které lze takto zavést je 8 (v modu se 40 sloupci) nebo 16 (80 sloupců). Po stisknutí tlačítka RETURN se zobrazí obsah 8/16 buněk za adresou.

Příklady:

>2000 Zobrazí řádek bytů za $2000

>2000 31 32 38 Zavede čísla do $2000 a zobrazí řádek bytů za $2000.

Příkaz: @

Účel: Používá se pro zobrazení stavu disku.

Syntax: @[<jednotka#>],<diskový příkaz>

<jednotka#>	číslo jednotky (nepovinné)
<diskový příkaz>	příkaz pro disk.

Poznámka: Při použití samotného symbolu @ dostaneme stav diskdrajvu.

Příklady:

@ 00,OK,00,00	prověří stav disku
@ ,I	inicializuje drajv č. 8
@ ,S	zobrazí obsah drajvu 8

D O D A T E K K

Zkratky BASICu 7.0

Poznámka: Zkratky uvedené níže lze použít v modu velká písmena/grafika. Stiskněte tlačítko nebo tlačítka uvedených písmen, poté stiskněte tlačítko SHIFT současně s písmenem uvedeným za SHIFT.

Klíčové slovo	Zkratka
ABS	A SHIFT B
APPEND	A SHIFT P
ASC	A SHIFT S
ATN	A SHIFT T
AUTO	A SHIFT U
BACKUP	BA SHIFT C
BANK	B SHIFT A
BEGIN	B SHIFT E
BEND	BE SHIFT N
BLOAD	B SHIFT L
BOOT	B SHIFT O
BOX	není
BSAVE	B SHIFT S
BUMP	B SHIFT U
CATALOG	C SHIFT A
CHAR	CH SHIFT A
CHR$	C SHIFT H
CIRCLE	C SHIFT I
CLOSE	CL SHIFT O
CLR	C SHIFT L
CMD	C SHIFT M
COLLECT	COLL SHIFT E
COLINT	není
COLLISION	COL SHIFT L
COLOR	COL SHIFT O

Klíčové slovo	Zkratka
CONCAT	C SHIFT O
CONT	není
COPY	CO SHIFT P
COS	není
DATA	D SHIFT A
DEC	není
DCLEAR	DCL SHIFT E
DCLOSE	D SHIFT C
DEF FN	není
DELETE	DE SHIFT L
DIM	D SHIFT I
DIRECTORY	DI SHIFT R
DLOAD	D SHIFT L
DO	není
DOPEN	D SHIFT O
DRAW	D SHIFT R
DSAVE	D SHIFT S
DVERIFY	D SHIFT V
EL	není
END	není
ENVELOPE	E SHIFT N
ER	není
ERR$	E SHIFT R
EXIT	EX SHIFT I
EXP	E SHIFT X
FAST	není
FETCH	F SHIFT E
FILTER	F SHIFT I
FOR	F SHIFT O
FRE	F SHIFT R
FNxx	není
GET	G SHIFT E
GETKEY	GETK SHIFT E
GET#	není
GOSUB	GO SHIFT S
GO64	není
GOTO	G SHIFT O
GRAPHIC	G SHIFT R
GSHAPE	G SHIFT S

Klíčové slovo	Zkratka
HEADER	HE SHIFT A
HELP	HE SHIFT L
HEX$	H SHIFT E
IF ... GOTO	není
IF ... THEN ... ELSE	není
INPUT	není
INPUT#	I SHIFT N
INSTR	IN SHIFT S
INT	není
JOY	J SHIFT O
KEY	K SHIFT E
LEFT$	LE SHIFT F
LEN	není
LET	L SHIFT E
LIST	L SHIFT I
LOAD	L SHIFT O
LOCATE	LO SHIFT C
LOG	není
LOOP	LO SHIFT O
MID$	M SHIFT I
MONITOR	MO SHIFT N
MOVESHAPE	není
MOVSPR	M SHIFT O
NEW	není
NEXT	N SHIFT E
ON ... GOSUB	ON ... GO SHIFT S
ON ... GOTO	ON ... G SHIFT O
OPEN	O SHIFT P
PAINT	P SHIFT A
PEEK	PE SHIFT E
PEN	P SHIFT E
PI	není
PLAY	P SHIFT L
POKE	PO SHIFT O

Klíčové slovo	Zkratka
POS	není
POT	P SHIFT O
PRINT	?
PRINT=	P SHIFT R
PRINT USING	?US SHIFT I
PUDEF	P SHIFT U
RBUMP	RB SHIFT U
RCLR	R SHIFT C
RDOT	R SHIFT D
READ	RE SHIFT A
RECORD	R SHIFT E
REM	není
RENAME	RE SHIFT N
RENUMBER	REN SHIFT U
RESTORE	RE SHIFT S
RESUME	RES SHIFT U
RETURN	RE SHIFT T
RGR	R SHIFT G
RIGHT$	R SHIFT I
RLUM	není
RND	R SHIFT N
RREG	R SHIFT R
RSPCOLOR	RSP SHIFT C
RSPPOS	R SHIFT S
RSPR	není
RSPRITE	RSP SHIFT R
RUN	R SHIFT U
RWINDOW	R SHIFT W
SAVE	S SHIFT A
SCALE	SC SHIFT A
SCNCLR	S SHIFT C
SCRATCH	SC SHIFT R
SGN	S SHIFT G
SIN	S SHIFT Y
SLEEP	S SHIFT L
SLOW	není
SOUND	S SHIFT O
SPC(	není

Klíčové slovo	Zkratka
SPRCOLOR	SPR SHIFT C
SPRDEF	SPR SHIFT D
SPRITE	S SHIFT P
SPRSAV	SPR SHIFT S
SQR	S SHIFT Q
SSHAPE	S SHIFT S
STASH	S SHIFT T
STATUS	není
STEP	ST SHIFT E
STOP	ST SHIFT O
STR$	ST SHIFT R
SWAP	S SHIFT W
SYS	není
TAB(	T SHIFT A
TAN	není
TEMPO	T SHIFT E
TI	není
TI$	není
TO	není
TRAP	T SHIFT R
TROFF	TRO SHIFT F
TRON	TR SHIFT O
UNTIL	U SHIFT N
USR	U SHIFT S
VAL	není
VERIFY	V SHIFT E
VOL	V SHIFT O
WAIT	W SHIFT A
WHILE	W SHIFT H
WIDTH	WI SHIFT D
WINDOW	W SHIFT I
XOR	X SHIFT O

D O D A T E K L

Souhrn příkazů pro disk

Tento dodatek uvádí souhrn příkazů používaných pro operace s diskem v modech C 128 a C 64. Commodoru 128. Podrobnější informace o příkazech viz Encyklopedie BASICu 7.0, Kapitola 5. Další informace jsou uvedeny v manuálu diskdrajvu.

Nové příkazy BASICu 7.0 se mohou používat pouze v modu C 128. Všechny příkazy BASICu 2.0 mohou být použity v obou modech.

Příkaz	Použití	BASIC 2.0	BASIC 7.0
APPEND	Seřadí data do fajlu*		x
BLOAD	Přepíše do paměti počítače binární fajl od určené pamětové bunky		x
BOOT	Přepíše a provede program		x
BSAVE	Uloží binární fajl z daného místa paměti		x
CALALOG	Zobrazí na obrazovce obsah seznamu disku*		x
CLOSE	Uzavře logický fajl na disku	x	x
CMD	Přesměruje výstupy z obrazovky na diskový fajl	x	x
COLLECT	Uvolní nepřístupnou oblast na disku*		x

Příkaz	Použití	BASIC 2.0	BASIC 7.0
CONCAT	Spojí dva datové fajly*		x
COPY	Zkopíruje fajl z jednoho disku na druhý*		x
DCLEAR	Formuje a inicializuje diskdrajv*		x
DCLOSE	Uzavře logický fajl na disku		x
DIRECTORY	Zobrazí na obrazovce seznam obsahu disku*		x
DLOAD	Přepíše program BASIC z disku*		x
DOPEN	Otevře diskový fajl pro operace čtení a/nebo záznamu*		x
DSAVE	Uloží program BASIC na disku*		x
DVERIFY	Porovná program v paměti s programem na disku*		x
GET#	Čte z otevřeného diskového fajlu	x	x
HEADER	Formátuje disk*		x
LOAD	Přepíše fajl z disku	x	x
OPEN	Otevře fajl pro vstup nebo výstup	x	x
PRINT#	Posílá data do fajlu	x	x
RECORD	Umísťuje pointer relativních fajlů*		x
RENAME	Změní jméno fajlu na disku*		

Příkaz	Použití	BASIC 2.0	BASIC 7.0
RUN jmé- no fajlu	Provede program BASIC z disku		x
SAVE	Uloží na disk program z paměti počítače	x	x
VERIFY	Porovná program v paměti počítače s programem na disku	x	x

*Ačkoliv BASIC 2.0 neobsahuje jednoduchý ekvivalent tohoto příkazu, existuje ekvivalentní několikapříkazová instrukce. Viz manuál diskdrajvu ohledně těchto příkazů v BASICU 2.0.

Tento příkaz uzavře a vyprázdní všechny kanály otevřené pro zařízení specifikované číslem. Pokud zařízení není uvedeno, je to U8. Tento příkaz je analogem OPEN 0,8,15,"I0" CLOSE 10

Příklad:

DCLEAR D0 Zruší všechny kanály otevřené pro drajv 0, zařízení č. 8.

DCLEAR D1,U9 Zruší všechny fajlové kanály otevřené pro drajv 1, zařízení 9

Poznámka: Všechny fajly budou zrušeny; data není možné opět získat z fajlů, v nichž byla původně zapsána. Viz CLOSE/DCLOSE.

DCLOSE

- Uzavírá diskový fajl.

DCLOSE [#číslo logického fajlu][[,ON],>Uzařízení]

Tento příkaz uzavírá jednotlivý fajl nebo všechny fajly běžně otevřené na diskové jednotce. Jestliže není uvedeno žádné číslo logického fajlu, uzavřou se všechny fajly běžně otevřené. Číslo zařízení, pokud není uvedeno, bude 8. Všimněte si následujících příkladů:

DCLOSE Uzavře všechny fajly běžně otevřené na jednotce č. 8.

DCLOSE #2 Uzavře fajl spojený s číslem logického fajlu 2.

DCLOSE ON U9 Uzavře všechny fajly běžně otevřené na jednotce č. 9.

DEF FN

- Vrací hodnotu fukkce definované uživatelem.

DEF FNjméno(proměnná) = výraz

Tento příkaz umožňuje definovat složitější výpočet jako funkci. V případě dlouhého vzorce používaného vícekrát v tomtéž programu toto klíčové slovo umožní ušetřit místo v programu. Jméno přiřazené funkci začíná písmeny FN, následovanými jakýmkoliv povoleným jménem číselné proměnné. Nejprve definujte funkci pomocí příkazu DEF následovaného jménem funkce. Za jménem dejte dvojici závorek (), v nichž je uvedeno jméno fiktivní číselné proměnné. Této proměnné bude přiřazena hodnota pouze jestliže se objeví napravo od symbolu rovnosti (=). Totéž jméno proměnné může být použito v jiné části programu, avšak bude zcela nezávislé na jeho použití ve funkci. Ve výrazu lze použít i jiné

proměnné a/nebo funkce, které budou vyhodnoceny při volání funkce. Poté musíme zavést symbol rovnítka, následovaný definičním výrazem. Funkce může být použita při záměně X (argumentu) libovolným číslem ve tvaru, zobrazeném na řádcích 20, 40 a 50 následujícího programu.

Příklad:

```
10 DEF FNEG(L0)=INT((V1*L0!2)*100)/100   L0 je lokální proměnná pro tento řádek
20 L0=15                                 Proměnná norm. progr.
30 V1=3.141593                           Přibližně
40 PRINT FNEG(5)        Přiřadí          ve funkci hodnotu 5
50 PRINT FNEG(1)        lokální proměnné L0
60 PRINT INT((V1*L0!2)*100)/100          Použije 1 jako L0 ve funkci
                                         Proměnná L0 použita
70 PRINT                                 Zůstává nezměněno
```

DELETE

- Ruší řádky programu BASIC v určeném rozmezí.

DELETE <[počáteční řádek][počáteční řádek -] [počáteční řádek - koncový řádek] [- koncový řádek]>

Tento příkaz může být proveden pouze v přímém modu.

Příklady:

DELETE 75	Zruší řádek 75.
DELETE 10-50	Zruší řádky od 10 do 50 včetně.
DELETE-50	Zruší všechny řádky od začátku programu do řádku 50 včetně.
DELETE 75-	Zruší všechny řádky od 75. až do posledního řádku včetně.

DIM

- Udává počet prvků matice.

DIM proměnná (indexy)[,proměnná(indexy)][...]

Dříve než použijeme proměnno matici program musí provést příkaz DIM, aby stanovil dimenze (rozměry) matice (ledaže je v každé dimenzi matice obsaženo 11 nebo méně než 11 prvků) Příkaz DIM je následován jménem matice, kterým může být libovolné povolené jméno proměnné. Za ním, uzavřený v závorkách, musí následovat počet (nebo číselná proměnná) prvků v každé dimenzi. Matice s více než jednou dimenzí se nazývá mnohoměrná matice. Můžete použít libovolný počet dimenzí, pamatujte však, že celý seznam vytvořených proměnných zabírá poměrně rozsáhlou část paměti, která je k dispozici, jestliže použijete příliš mnoho

dimenzí. Uvedeme počet bytů zaujímaných jednou maticí:

5 bytů pro jméno matice
2 byty na každou dimenzi
2 byty/prvek pro celočíselné proměnné
5 bytů/prvek pro normální číselné proměnné
3 byty/prvek pro stringové proměnné
1 byte pro každý znak v každém prvku řetězce

Celočíselné matice zabírají dvě pětiny místa v porovnání s maticemi s pohyblivou desetinnou čárkou (např. DIM A%(100) vyžaduje 209 bytů, DIM A(100) vyžaduje 512 bytů).

Poznámka: Prvky se číslují od 0. Například DIM A(100))dává 101 prvků.

Příkazem DIM může být určena dimenze více než jedné matice, jestliže jména proměnných oddělíme čárkami. Jestliže program obsahuje vícero příkazů DIM pro tutéž matici, objeví se chybové hlášení "RE DIM ARRAY ERROR" (chyba - opakovaná DIM matice). Doporučujeme umístit všechny příkazy DIM na začátek programu.

Příklad:

```
10 DIM A$(40), B7(15),CC%(4,4,4)
     41 prvků  16 prvků  125 prvků
```


DIRECTORY

- Zobrazí na obrazovce obsah seznamu disku.

DIRECTORY [Dčíslo drajvu][,<ON|,> Uzařízení] [,proměnné znaky]

Funkční tlačítko F3 v modu C128 zobrazí seznam pro zařízení č. 8, drajv č. 0. Použijte CONTROL S nebo NO SCROLL, abyste zastavili zobrazení; pro nové spuštění zobrazení stiskněte jakékoliv tlačítko. Tlačítko COMMODORE zpomalí "klouzání" zobrazení. Příkaz DIRECTORY se nemusí použít při tisku kopie na papír, protože některé tisky interferují s daty posílanými z diskdrajvu. Abyste pořídili kopii na papír, musíte přečíst seznam disku (LOAD"$,8), přičemž zrušíte běžný program v paměti. Pokud není uvedeno číslo zařízení, předpokládá se 8 a číslo drajvu 0.

Příklady:

DIRECTORY	Vypíše všechny fajly z disku na jednotce č. 8.
DIRECTORY D1,U9,"PRACE"	Vypíše fajl nazvaný "PRACE" z drajvu č. 1 na jednotce č. 9.

DIREC TORY
DLOAD

DIRECTORY "CO^{X}" Vypíše všechny fajly, jejichž jméno začíná písmeny CO, například COMPUTER, COPIA atd., všechny z drajvu jednotky 8. Hvězdička označuje proměnné znaky, takže všechny fajly začínající CO budou zobrazeny.

DIRECTORY DO,"FILE"?.BAK" Symbol ? je proměnný znak, odpovídající libovolnému znaku v této poloze, například FILE 1.BAK, FILE 2.BAK, FILE 3.BAK atd.

DIRECTORY D1,U9,(A$) Vypíše fajl se jménem uloženým v proměnné A$ ze zařízení č.9, drajv 1. Odvolává se k obsahu proměnné v závorkách, jenž bude použit jako jméno fajlu.

Poznámka: Abychom vytiskli seznam disku z drajvu 0, zařízení č.8, použijte následující příkazy:

```
LOAD "$0", 8
OPEN 4,4: CMD 4: LIST
PRINT 4: CLOSE 4
```

DLOAD

- Přenese program BASIC z disku do paměti počítače.

DLOAD "jméno fajlu" [,Dčíslo drajvu] [<ON>,]
Učíslo zařízení]

DLOAD
DO/LOOP/WHILE/UNTIL/EXIT

Tento příkaz přeoíše program BASIC z disku do paměti počítače (pro přepsání programu z pásku použijte LOAD). Program musí být specifikován jménem fajlu, obsahujícím maximalně 16 znaků. DLOAD předpokládá číslo zařízení 8 a drajvu 0 pokud nejsou uvedena explicite.

Příklady:

DLOAD "REC" Vyhledá na disku program "REC" a přepíše jej do paměti.

DLOAD (A$) Vyhledá na disku program, jehož jméno je uloženo v proměnné A$. Jestliže je A$ prázdná proměnná, objeví se chybové hlášení. Připomínáme, že musíte uzavřít do závorek proměnnou, používanou jako jméno fajlu.

Příkaz DLOAD se může použít v programu BASIC, aby se přepsal z disku do paměti jiný program. Taková operace se nazývá sdružování.

DO/LOOP/WHILE/UNTIL/EXIT

- Definuje a ovládá smyčku v programu.

DO [UNTIL podmínka] [WHILE podmínka] příkaz
[EXIT] LOOP [UNTIL podmínka] WHILE podmínk

Smyčková struktura provádí příkazy mezi příkazem DO a příkazem LOOP. Jestliže příkazy DO nebo LOOP nejsou doprovozeny příkazem UNTIL

nebo WHILE, provádění těchto příkazů pokračuje nekonečně. Jestliže se v tělese smyčky DO narazí na příkaz EXIT, řízení se předá prvnímupříkazu za příkazem LOOP. Do smyčky DO lze zavést blok příkazů podle pravidel pro strukturu FOR - NEXT.
Jestliže je specifikován parametr UNTIL, program pokračuje v provádění smyčky dokud příslušná podmínka je splněna. Parametr WHILE je v podstatě opakem parametru UNTIL = program provádí smyčku dokud podmínka není splněna. Jestliže podmínka není více splněna, řízení programu se předá příkazu bezprostředně následujícímu za příkazem LOOP. Příkladem podmínky (booleovsky argument) může být A=1 nebo G 65.

Příklad:

```
10 X=25
20 DO UNTIL X=0
30 X=X-1
40 PRINT "X=";X
50 LOOP
60 PRINT "KONEC SMYCKY"
```

V této ukázce se provádějí příkazy X=X-1 a PRINT "X=";X až když X=0. Jestliže X=0, program předá řízení příkazu PRINT "KONEC SMYCKY", bezprostředně následujícímu za LOOP.

17 -38


```
10 DO WHILE A$=" ": GET A$: LOOP
20 PRINT "TLACITKO";A$;"JE STISKNUTO"
```

A$ zůstává prázdnou proměnnou, dokud nestiskneme nějaké tlačítko. Jakmile stiskněte nějaké tlačítko, řízení programu se předá příkazu bezprostředně následujícímu za LOOP: PRINT "TLACITKO"; A$;"JE STISKNUTO". V ukázce se provádí GET A$ pokud A$ zůstává prázdným řetězcem. Tato smyčka neustále kontroluje, zda je nějaké tlačítko klávesnice stisknuto. (Poznámka: GETKEY A$ má stejný účinek na řádku 10).

```
10 DOPEN#8,"FILESEQ"
20 DO
30 GET#8,A$
40 PRINT A$;
50 LOOP UNTIL ST
60 DCLOSE#8
```

Tento program otevře fajl "FILESEQ" a čte z něho data, dukud systémova proměnná ST nenabude hodnotu, odpovídající tomu, že všechna data byla do počítače zavedena.

DOPEN

- Otevírá fajl na disku pro operace čtení a/nebo záznamu.

DOPEN#číslo logického fajlu,"jméno fajlu" [,<S|P>] [,L délka záznamu] [,Dčíslo drajvu] [<ON|,> Učíslo zařízení] [,W]

kde:

S - sekvenční fajl

P - prgramový fajl

L - délka záznamu = pouze relativní délka záznamu

w - záznam (jestliže tento parametr není uveden, předpokládá se čtení).

Tento příkaz otevírá sekvenční fajl, relativní nebo náhodného přístupu, pro záznam nebo čtení. Délka záznamu (L) se vztahuje na relativní fajl, který může mít délku maximálně 254 znaků. Parametr w se specifikuje pouze v případě operace záznamu (PRINT#) v sekvenčním fajlu. Jestliže není uvedeno jinak, diskdrajv předpokládá, že prováděnou operací je čtení. Relativní fajly jsou otevřeny současně pro čtení i záznam Číslo logického fajlu spojuje číslo z fajlu jako zprávu pro následující diskové operace, například čtení (INPUT#) nebo záznam (PRINT#). Číslo logického fajlu může být v rozmezí od 1 do 255. Číslo logického fajlu větší než 128 automaticky veací válec a posunuje řádek při každém příkazu tisku (PRINT#). Číslo logického fajlu menší než 128 vrací pouze válec, což však lze vyloučit zavedením středníku na konec příkazu PRINT#. Jestliže zařízení uvedeno,

není, předpokládá se 8, v případě čísla drajvu se předpokládá 0.

Příklady:

```
DOPEN#1,"ADR",W     Otevře sekvenční fajl 1 (ADR)
                    pro záznam.

DOPEN#2,"RIC",D1,U9 Otevře sekvenční fajl 2
                    (RIC) pro čtení ze zařízení 9, drajv č. 1.

DOPEN#3,"KNIHY",L128 Otevře relativní fajl 3
                    (KNIHY) pro čtení i záznam. Délka záznamu je
                    128 znaků.
```

DRAW

- Kreslí na obrazovce body, úsečky a obrazce na určeném místě.

DRAW [zdrojová barva],X1,Y1 [TO X2,Y2].....

Tento příkaz kreslí body, úsečky a jednotlivé obrazce. Hodnoty parametrů:

Zdrojová barva 0=pozadí bodové matrice
1=popředí bodové matrice
2=multicolor 1 grafický
3=multicolor 2 mod 3 a 4

X1,Y1 výchozí souřadnice

X2,Y2 koncové souřadnice

Viz rovněž příkaz LOCATE s údaji o pixelovém

kursoru.

Příklady:

DRAW 1,100,50 Nakreslí bod

DRAW ,10,10 TO 100,60 Nakreslí úsečku

DRAW ,10,10 TO 10,60 TO 100,60 TO 10,10 Nakreslí trojúhelník.

Je možné vynechat parametry, je však nutné zavést čárku, která následuje za neuvedeným parametrem.

DSAVE

- Uloží programový fajl v BASICU na disku.

DSAVE "jméno fajlu"[,Dčíslo drajvu][<ON|,>
Učíslo zařízení]

Tento příkaz uloží program BASIC na disku (viz SAVE pro uložení programu na pásek). Za příkazem musí být uvedeno jméno fajlu, sestávající maximálně z 16 znaků. Neuvedené explicite číslo zařízení bude 8, zatímco neuvedené číslo drajvu je 0.

Příklady:

DSAVE "REC" Uloží program "REC" na disk.

DSAVE (A$) Uloží na disk program pojmenovaný v proměnné A$.

DSAVE "PROG3",D1,U9 Uloží program "PROG3" na disketu, zařízení č. 9, drajv 1 (dvojitý diskdrajv

DVERIFY

- Porovná program v paměti s programem na disk

DVERIFY "jméno fajlu"[,Dčíslo drajvu][<ON|,>
Učíslo zařízení]

Tento příkaz porovná pomocí Commodoru 128 určený program na disku s programem v paměti počítače. Neuvedené číslo drajvu se předpokládá 0 a neuvedené číslo zařízení bude 8.

Poznámka: Jestliže nějaká grafická oblast byla přidělena po SAVE, vyvolá to chybové hlášení. Technicky je tato operace korektní. Text BASIC bude umístěn do původního místa, kde byla přidělena grafická oblast matrici bodů. Z tohoto důvodu původní místo, kde Commodore 128 provádí porovnání uloženého programu, bude jiné. V tomto případě VERIFY, který porovnává byte po bytu, nebude fungovat i když program bude v pořádku.

Pro porovnávání binárních dat viz příkaz VERIFY "jméno fajlu",8,1 v popisu příkazu VERIFY.

DVERIFY

END

ENVELOPE

Příklady:

DVERIFY"C128" Porovná program "C128" z drajvu 0, jednotka 8.

DVERIFY "SPRITE",D0,U9 Porovná program "SPRITE" na drajvu 0, zařízení č. 9.

END

- Ukončí provádění programu.

END

Jestliže program narazí na příkaz END, okamžitě se zastaví jeho provádění. Aby program pokračoval příkazem následujícím (pokud takový existuje) za příkazem END, použijte příkaz CONTINUE.

ENVELOPE

- Určuje "zabarvení" hudebního nástroje.

ENVELOPE n[,att][,dec][,sos][,ril][,fo][,li]

kde:

n - číslo zabarvení (0-9)

att - attacco (0-15)

dec - decadimento (0-15)

sos - sostegno (0-15)

ril - rilascio (0-15)

fo - tvar vlny:
0 = trojúhelníkový
1 = pilovitý
2 = variabilní impuls (hranatý)
3 = šum
4 = kruhová modulace

li - délka impulsu (0-4095)

Nespecifikovaný parametr nabývá hodnotu apriori určenou nebo nebo průběžně predefinovanou. Délka impulsu má smyslo pouze v případě vlny s variabilním impulsem (fo = 2) a je určena vztahem li/40,95. Například li = 2048 produkuje čtvercový impuls, kdežto 0 a 4095 produkují konstantní vstup. Commodore 128 je vybaven následujícími 10 "zabarveními" zvuků:

	n	A	D	S	R	fo	li	nastroj
ENVELOPE	0	0	9	0	0	2	1536	klavír
ENVELOPE	1	12	0	12	0	1		harmonika
ENVELOPE	2	0	0	15	0	0		kaliope
ENVELOPE	3	0	5	5	0	3		buben
ENVELOPE	4	9	4	4	0	0		flétna
ENVELOPE	5	0	9	2	1	1		kytara
ENVELOPE	6	0	9	0	0	2	512	klavičemb
ENVELOPE	7	0	9	9	0	2	2048	varhany
ENVELOPE	8	8	9	4	1	2	512	trouba
ENVELOPE	9	0	9	0	0	0		xylofon

Pro zahrání předem definovaných hudebních nástrojů stačí jednoduše specifikovat číslo "zabarvení" (n) v příkazu PLAY (viz PLAY). Není tedy nutné použít přitom příkaz ENVELOPE, protože tento příkaz se používá pro změnu zabarvení

FAST

- Aktivuje operační mod 2 MHz.

FAST

Tento příkaz uvádí do chodu mod o 2 MHz a vyvolává deaktivování obrazovky o 40 sloupcích.pro VIC. Všechny operace (s vyjímkou input/output) se viditelně zrychlí. Může být použit i v grafickém modu, který ovšem bude zobrazitelný pouze po příkazu SLOW.

FETCH

- Vybírá data z rozšířené paměti (modul RAM).

FETCH#byte,inic,inexp,nbexp

kde:

byte - počet bytů vybíraných z rozšířené paměti (0-65535)
inic - počáteční adresa v základní RAM (0-65535)
nbexp- číslo banky RAM rozšíření 64 K (0-15)
inexp- počáteční adresa rozšířené RAM (0-65535)

FILTER

- Určuje zvukové parametry filtru (čip SID).

FILTER [freq][,fb][,fp][,fa][,res]

kde:

freq - obřezávací kmitočet filtru (0-2047)
fb - zapnutí (1) nebo vypnutí (0) basového filtru
fp - zapnutí (1) nebo vypnutí (0) pásmovéh filtru
fa - zapnutí (1) nebo vypnutí (0) altového filtru
res - rezonance (0-15)

Jestliže parametry nejsou specifikovány nemění se běžné hodnoty.

Je možné používat více filtrů najednou. Například basový a altový filtr mohou být použity společně, aby vytvořily pásmový blok rychlosti čela vlny. Pro dosaření slyšitelného účinku filtru potřebujeme zvolit alespoň jeden typ filtru, skrz který musí projít alespoň jeden hlas.

Příklady:

FILTER 1024,0,1,0,2 Stanoví obřezavací kmitočet 1024, zapne pásmový filtr a úroven rezonance 2.

FILTER 2000,1,0,1,1,10 Stanoví obřezavací kmitočet 2000, volí basový a altový filtr (pásmový blok rychlosti čela) a zavádí úrovén rezonance 10.

FOR/TO/STEP/NEXT

- Určuje strukturu opakované smyčky v programu.

FOR proměnná=počáteční hodnota TO koncová hodnota [STEP přírůstek]

Tento příkaz spolu s příkazem NEXT umožňuje opakovat určitou část programu tolikrát, kolikrát chceme (příkl. smyčky). Tato funkce se používá, jestliže je nutné provést součet nebo jestliže jisté operace musí být provedeny opakovaně daný počet krát (například tisk). Tento příkaz opakovaně provádí všechny příkazy, které se nacházejí mezi příkazy FOR a NEXT v souladu se stanovenými počátečními a koncovými hodnotami. Počáteční a koncová hodnota reprezentují počátek a konec odečtu proměnné smyčky. Proměnná smyčky přibývá nebo ubývá během provádění FOR/NEXT. Logika příkazu FOR/NEXT je následující: Nejprve se proměnné smyčky přiřadí počáteční hodnota. Jestliže program dojde do řádku programu, obsahujícího příkaz NEXT přičte se přírustek STEP (pokud není uveden, pak = 1) k proměnné smyčky a zkontroluje se, zda tato proměnná je větší nebo rovna koncové hodnotě. Jestliže

tato je menší než koncová hodnota, smyčka se provede znovu počínaje příkazem bezprostředně následujícím za příkazem FOR. Jestliže proměnná smyčky je větší než koncová hodnota, smyčka se ukončí a program pokračuje příkazem, bezprostředně následujícím za NEXT. Jestliže je hodnota kroku (STEP) záporná, provedou se opačné akce.

Příklad (A):

```
10 FOR L=1 TO 10
20 PRINT L
30 NEXT L
40 PRINT "KONEC! L=";L
```

Příklad (B):

```
10 FOR L=10 TO 1 STEP-\
20 PRINT L
30 NEXT L
40 PRINT "KONEC! L=";L
```

Program (A) tiskne čísla od 1 do 10, následovaná hlášením KONEC! L=11. Program (B) vytiskne čísla od 10 do 1 a poté KONEC! L=0. Koncová hodnota smyčky může být následována slovem STEP a dalším číslem nebo proměnnou. V tomto případě hodnota následující za STEP se pokaždé přičte namísto jedničky. Umožňuje to počítat v opačném směru, přičítat zlomková čísla nebo přírustky větší než jedna. Uživatel může použít smyčku uvnitř jiné smyčky. Říkáme tomu sdružování smyček. Jestliže se smyčky sdružují, je poslední smyčka od počátku vždy první smyčkou od konce.

Příklad:

```
10 FOR L=1 TO 100
20 FOR A=5 TO 11 STEP 0.5
30 NEXT A
40 NEXT L
```

Smyčka FOR...NEXT na řádcích 20 a 30 je vložena do smyčky na řádcích 10 a 40. Přírustek STEP 0.5 se používá, aby ukázal, jak jsou možné indexy v pohyblivé desetinné čárce. Viz rovněž příkaz NEXT.

GET

- Odebírá vstupní data z klávesnice po znacích, aniž by čekal na stisknutí nějakého tlačítka.

GET seznam proměnných

Příkaz GET čte jakýkoliv znak, natištěný uživatelem. Jestliže uživatel vytiskne nějaké znaky, uloží se v paměti počítače (v části nazývané BUFFER klávesnice). V této části je uloženo vždy nanejvýš 10 znaků a veškeré další znaky budou ztraceny. Příkaz GET přečte první znak z bufferu, posune ostatní směrem nahoru a tím uvolní paměť. Slovo GET je následováno jménem proměnné, číselné nebo stringové. Jestliže v bufferu žádné znaky nejsou,

GET vrátí nulový (prázdný) znak.
GET nevyvolává přestávku v provádění programu jestliže v bufferu není žádný znak (viz GETKEY).
Jestliže C 128 čeká na stisknutí nějakého tlačítka a bude stisknuto číselné tlačítko, program se zastaví a objeví se chybové hlášení. Příkaz GET může být místo toho vložen do smyčky pro přezkoumání neplatného výsledku. V tomto případě může být použit rovněž příkaz GETKEY. Další podrobnosti viz GETKEY.
Příkazy GET a GETKEY mohou být použity pouze v programu.

Příklad:

```
10 DO: GET A$: LOOP UNTIL A$="A"   Tento řádek
                                   čeká, dokud nestisk-
                                   nete tlačítko A a po-
                                   tom pokračuje dále.
20 GET B,C,D   Přijme číselné proměnné B, C, D
               z klávesnice, aniž by čekal na
               stisknutí nějakého tlačítka.
```

GETKEY

- Odebírá vstupní data z klávesnice, znak po znaku, a čeká na stisknutí nějakého tlačítka.

GETKEY seznam proměnných

Příkaz GETKEY se velmi podobá příkazu GET.

GETKEY
GET=

Na rozdíl od příkazu GET však GETKEY čeňá na stisknutí nějakého tlačítka na klávesnici uživatelem, jestliže v bufferu klávesnice žádný znak není. Umožňuje to počítači vyčkat stisknutí jednoho znaku. Tento příkaz může být proveden pouze v programu.

Příklad:

10 GETKEY A$ Tento řádek očekává až bude stisknuto nějaké tlačítko, aby program pokračoval.

10 GETKEY A$,B$,C$ Tento příkaz čeká zadání alfanumerických znaků z klávesnice. GETKEY může být použit rovněž pro číselná tlačítka.

Poznámka: GETKEY nemůže vracet nulové (prázdné) znaky.

GET#

- Odebírá vstupní data z pásku, disku nebo RS232.

GET# číslo fajlu, seznam proměnných

Tento příkaz zavádí znak po znaku předem otevřený fajl. V každém jednotlivém případě funguje jako příkaz GET. Tento příkaz se může používat pouze v programu.

Příklad:

10 GET# 1,A$ V tomto případě počítač přijme

GET#
GO 64
GOSUB

jeden znak a uloží jej v proměnné A$, z fajlu č. 1. Předpokládá se, že fajl č. 1 byl předtím otevřen. Viz příkazy OPEN/DOPEN.

GO 64

- Aktivuje mod C 64.

GO 64

Tento příkaz provádí přechod z modu C 128 do modu C 64. Na obrazovce se objeví otázka "Are You Sure?" (Jsi si jistý?) jako odpověď na příkaz GO 64. Jestliže stisknete Y (yes - ano) program přítomný v paměti bude ztracen a řízení se předá modu C 64; v opačném případě, po stisknutí libovolného jiného tlačítka, počítač zůstává v modu C 128. Tento příkaz může být použit v přímém modu i v programu. V programovém modu se otázka neobjeví.

GOSUB

- Volá podprogram uvedeným číslem řádku.

GOSUB číslo řádku

Tento příkaz se podobá příkazu GOTO s tou výjimkou, že se Commodore vrátí po dokončení podprogramu (subrutiny) do výchozího místa. Jestliže počítač narazí na řádku obsahující

příkaz RETURN program se vrátí na příkaz bezprostředně následující za příkazem GOSUB. To, na co se odvolává příkaz GOSUB, se nazývá sabrutina (podprogram). Sabrutiny jsou užitečné, jestliže určité operace musíme opakovat v programu vícekrát. Místo abychom opakované psali určitou část programu zavedeme sabrutinu a použijeme příkaz GOSUB na potřebných místech programu. Viz rovněž příkaz RETURN.

Příklad:

```
20 GOSUB 800
.....
799 END
800 PRINT "NAZDAR"
801 RETURN
```

V této ukázce se volá sabrutina začínající na řádku 800. Každá sabrutina musí být zakončena příkazem RETURN. Řádek 799 ukončí program, který by jinak znovu provedl sabrutinu.

GOTO/GO TO

- Přenáší provádění programu na uvedený řádek.

GOTO číslo řádku

Jestliže počítač narazí na příkaz GOTO v programu, pak provede příkaz, jehož číslo řádku je uvedeno v příkazu GOTO (jdi na). Jestliže se použije v přímém modu, GOTO začne provádět program počínajíc řádkem, jehož číslo je uvedeno, aniž by přitom vynuloval proměnné.

Tato operace se podobá příkazu RUN až na to, že RUN vynuluje hodnoty proměnných.

Příklady:

```
10 PRINT "COMMODORE"
20 GOTO 10
```

GOTO na řádku 20 způsobí opakování řádku 10, dokud nestisknete RUN/STOP

```
GOTO 100
```

Uvede program do chodu p počínajíc řádkem 100, aniž by vynuloval paměťovou část pro proměnné.

GRAPHIC

- Aktivuje grafické mody.

GRAPHIC mod [,mazání][,s]

nebo

GRAPHIC CLR

Tento příkaz uvede Commodore 128 do jednoho z grafických modů:

mod	popis
0	text, 40 sloupců
1	standartní grafika s bodovou matricí
2	standartní grafika s bodovou matricí (dělená obrazovka)
3	multibarevná grafika s bodovou matricí
4	multibarevná grafika s bodovou matricí (dělená obrazovka)
5	text, 80 sloupců

Parametr "mazání" určuje, zda obrazovka s bodovou matricí bude vymazána (1) při provádění

programu, nebo zda bude ponechána beze změny (0). Parametr S ukazuje číslo prvního řádku na obrazovce v případě grafických modů č. 2 nebo 4 (mod dělené obrazovky se standartní nebo multibarevnou bodovou matricí). Pokud počáteční řádek není uveden, obrazovka je dělena 19. řádkem.

Při provedení příkazu GRAPHIC 1-4 se v paměti vyčlení oblast bodové matrice 9 K. Počátek textové oblasti BASICu se posuna nad oblast pro bodovou matrici a všechny programy BASIC se automaticky přemístí. Tato oblast se vyčlení rovněž v případě, že se uživatel vrátí do textového modu (GRAPHIC 0). Jestliže parametru mazání přiřadíme hodnotu 1, obrazovka se vymaže. Příkaz GRAPHIC CLR zruší oblast 9 K pro bodovou matrici, která opět bude k dispozici textu BASIC. Jakýkoliv program BASIC bude opět posunut.

Příklady:

GRAPHIC 1,1 Zvolí standartní mod bodové matrice a vymaže bodovou matrici.

GRAPHIC 4,0,10 Zvolí multibarevný mod bodové matrice s dělenou obrazovkou, nevymaže bodovou matrici a rozdělí obrazovku na řádku 10.

GRAPHIC 0 Ustanoví text s 40 sloupci.

GRAPHIC 5 Ustanoví text s 80 sloupci.

GRAPHIC CLR Zruší a vymaže obrazovku s bodovou matricí.

HEADER

- Formátuje disk.

HEADER "jméno disku"[,i.d.][,Dčíslo drajvu]
[<ON|,> Učíslo zařízení]

kde:

jméno disku jakékoliv jméno sestávající maximálně z 16 znaků.

i.d. dva alfanumerické znaky, libovolné s vyjímkou mezery

Dříve než poprvépoužijete nový disk musíte jej formátovat pomocí příkazu HEADER. Tento příkaz můžete rovněž použít pro vymazání dříve formátovaného disku, který lze tímto způsobem znovu použít.

Při zavedení příkazu HEADER v přímém modu se objeví otázka ARE YOU SURE? (jste si jistý?). V programovém modu se tato otázka neobjeví. Tento příkaz rozdělí disk na části, nazývané bloky a vytvoří ukazatel fajlů, nazývaný seznam. Každému disku musí být přiřazeno výlučné číslo i.d. Používejte HEADER pozorně, protože tento příkaz vymaže všechna uložená data.

Jestliže disketa již byla formátována, pří-

kaz HEADER se může použít rychlejším způsobem tak, že se vynechá i.d., které se zavádí pro nový disk; v tomto případu se použije staré i.d. Tato operace se může provést pouze jestliže byl disk dříve formátován, protože pak se pouze zruší seznam a disk se neformátuje. Jestliže není explicite specifikováno, pak číslo zařízení je 8 a číslo disku je 0.

Systém se ujišťuje otázkou ARE YOU SURE? (jste si jistý?) dříve než Commodore 128 operaci provede. Stiskněte tlačítko Y jestliže jste si jistý, nebo stiskněte libovolné jiné tlačítko, jestliže chcete operaci zrušit.

Příkaz HEADER přečte chybový kanál diskových příkazů a jestliže zjistí chybu, objeví se hlášení "?BAD DISC" (špatný disk).

Příkaz HEADER je analogem příkazů BASICu 2.0:
OPEN 1,8,15,"N0: jméno disku,i.d."

Příklady:

HEADER "MUJ DISK",I51,D0 Formátuje "MUJ DISK" s použitím i.d. 51 na drajvu 0, zařízení 8 (neuvedeno explicite)

HEADER "REC",I45,D1 ON U9 Formátuje "REC" za použití i.d. 45 na drajvu 1, zařízení č. 9.

HEADER "PROG C128",D0 Operace zrychleného formátování na drajvu 0, zařízení č. 8, přičemž disk již byl formátován. Použije se staré číslo i.d.

Poznámka: Jako i.d. nelze použít stringovou proměnnou.

HELP

- Upřesní řádek, v němž došlo k chybě.

HELP

Příkaz HELP se používá po identifikování chyby v programu. Jestliže HELP použijeme ve formátu 40 sloupců, vypíše se řádek, v němž byla zjištěna chyba s částí, obsahující chybu, zobrazenou inverzně. Ve formátu 80 sloupců část řádku, v níž byla identifikována chyba, bude podtržena. Stisknutí tlačítka HELP způsobí natisknutí HELP a vrátí automaticky hlavu.

IF/THEN/ELSE

- Vyhodnotí podmíněný výraz a provede tu část programu, která závisí na hodnotě výrazu.

IF výraz THEN příkazy BASIC 2.0

IF výraz THEN příkazy [:ELSE klauzule jinak] BASIC 7.0

Příkaz IF....THEN vyhodnotí výraz BASIC a provede jednu z dvou možných činností, v závislosti na výsledku vyhodnocení. Jestliže je výraz pravdivý, provede se příkaz (=y) následující za THEN (jakýkoliv příkaz BASICu). Jestliže výraz není pravdivý, program pokračuje řádkem programu bezprostředně následujícím za řádkem, obsahujícím příkaz IF, pokud se nepoužila klauzule ELSE. Vnitřní příkaz v IF ..... THEN může obsahovat maximálně 160 znaků (80 v modu C 64). Viz rovněž BEGIN/BEND. Klauzule ELSE, pokud se použije, se musí nacházet na následujícím řádku za příkazem IF..THEN a musí být oddělena od klauzule THEN dvojtečkou (:). Jestliže se kaluzule ELSE použije, provede se pouze když je výraz nepravdivý. Vyhodnocovaným výrazem může být proměnná nebo vzorec a jsou považované za pravdivé, jestliže nejsou nulové nebo za nepravdivé, když jsou nulové. Většinou však se jedná o výraz, zahrnující relační operátory (=, < , > , <=, >=, <>). Příkaz IF ... THEN může být vyjádřen ve dvou tvarech:

IF výraz THEN číslo řádku

nebo

IF výraz GOTO číslo řádku

Tyto dva tvary přenesou provádění programu na řádek, jehož číslo je uvedeno, pokud je výraz pravdivý. V opačném případě se program provádí dále, počínajíc řádkem bezprostředně následujícím za řádkem s příkazem IF.

Příklad:

```
50 IF X> 0 THEN PRINT "OK": ELSE END
```

V tomto řádku se prověřuje hodnota X. Jestliže je X větší než 0, provede se příkaz bezprostředně následující za klíčovým slovem THEN (PRINT "OK") a klauzule ELSE se ignoruje. Jestliže X je menší nebo rovno 0, provede se klauzule ELSE a příkaz bezprostředně následující za ELSE THEN se ignoruje.

```
10 IF X=0 THEN 100
20 PRINT "X se nerovna 0"
.....
99 STOP
100 PRINT "X se rovna 0"
```

V této ukázce se vyhodnocuje hodnota X. Jestliže se X rovná 0, řízení se předá řádku 100, a objeví se nápis "X se rovna 0". Jestliže X není rovno 0, program pokračuje na řádku 20, C128 vytiskne "X se nerovna 0" a program se zastaví.

Poznámka: Klauzule ELSE se nesmí použít v modu C 64.

INPUT

- Přijímá stringová data nebo čísla z klávesnice a čeká na stisknutí RETURN uživatelem.

INPUT ["libovolný řetězec";] seznam proměnných

Příkaz INPUT požaduje, aby byla zadána data během provádění programu a přiřadí tato data proměnné nebo proměnným. Program se zastaví, zobrazí otazník (?) na obrazovce a čeká na odpověď a na stisknutí tlačítka RETURN. Za slovem INPUT může následovat řetězec znaků a dále následuje jméno proměnné nebo seznam jmen proměnných, oddělených čárkami. Hlášení - řetězec znaků v uvozovkách - zobrazuje údaje zavedené uživatelem. Jestliže se toto hlášení do příkazu zavede, pak za závěrečnými uvozovkami musí následovat středník (;).
Jestliže je zahrnuto více proměnných, musí být odděleny čárkami. Počítač vyžaduje zavedení zbylých proměnných dvojitým otazníkem (??). Jestliže stisknete tlačítko RETURN, aniž byste zadali nějakou hodnotu proměnné, proměnná v INPUT si podrží předešlou hodnotu. Příkaz INPUT může být použit pouze v programu.

Příklad:

```
10 INPUT "ZADEJ CISLO";A
```



```
20 INPUT "A SVOJE JMENO";A$
30 PRINT A$;"ZADAL(A) CISLO";A
```

INPUT#

- Zavádí do paměti počítače data z fajlu.

INPUT# číslo fajlu, seznam proměnných

Tento příkaz funguje jako input, ale přenáší data z předem otevřeného fajlu, obecně z disku nebo z pásku, místo z klávesnice. Nepoužívá žádná pomocná hlášení. Tento příkaz lze použít pouze v programu.

Příklad:

```
10 OPEN 2,8,2,"DATA,S,R"
20 INPUT#2,A$, C, D$
```

Tento příkaz zavádí data z fajlu DATA a uloží je do proměnných A$, C a D$.

KEY

- Definuje nebo vypisuje význam funkčních tlačítek.

KEY [číslo tlačítka, řetězec]

Commodore C 128 má osm funkčních tlačítek (F1 - F8). Čtyři se tisknou na klávesnici samostatně, čtyři současně se SHIFT. Commodore 128 umožňuje provést takto určitou funkci

nebo operaci pokaždé, když stisknete určené funkční tlačítko. Definice přiřazená určitému tlačítku může obsahovat data, příkaz nebo řadu příkazů. KEY bez uvedení parametrů vrací výpis všech momentálních definicí tlačítka KEY. Jestliže jsou tlačítku přiřazena data, zobrazí se po stisknutí odpovídajícího tlačítka na obrazovce. Maximální délka celé definice činí 246 znaků.

Příklad:

```
KEY 7,"GRAPHIC 0" + CHR$(13) + LIST + CHR$(13)
```

Tento příkaz přikáže počítači přejít do modu textové obrazovky (VIC) se 40 sloupci a vypsat program pokaždé, když se stiskne tlačítko F7.(v přímém modu). CHR$(13) je ASCII kod pro RETURN a má stejný výsledek jako stisknutí RETURN. Pro ESC používejte CHR$(27).Pro zahrnutí uvozovek do řetězce KEY použijte CHR$(34). Tlačítka lze předefinovat rovněž v programu. Například:

```
10 KEY 2,"PRINT DS$" + CHR$(13)
```

Řádek přikazuje počítači prověřit a zobrazit proměnnou chybového kanálu diskdrajvu (PRINT DS$) pokaždé, když se stiskne funkční tlačítko F2. Abyste uvedli všechna funkční tlačítka do implicitního stavu BASICu, stiskněte tlačítko RESET a uveďte Commodore 128 do výchozího stavu.

LET

- Přiřadí proměnné hodnotu.

[LET] proměnná = výraz

Slovo LET se v programech používá pouze vzácně, protože je zbytečné. Jestliže je definována proměnná, nebo jestliže je proměnné přiřazována hodnota, LET se vždy rozumí samo sebou. Jméno proměnné, která přijímá výslednou hodnotu výrazu, je vžda nalevo od znaku rovnosti, zatímco čísla, řetězce nebo výrazy jsou napravo. S příkazem LET (rozumějícím se samo sebou) je možné přiřazovat pouze jedinou hodnotu. Například LET A=B=2 je nezákonné.

Příklad:

```
10 LET A=5    Přiřadí hodnotu 5 čísel. proměnné A
20 B=6        Přiřadí hodnotu 6 čísel. proměnné B
30 C=A^B+3    Přiřadí číselné proměnné C výsledek
              operace 5 krát 6 plus 3.
40 D$="AHOJ"  Přiřadí řetězec "AHOJ" stringové
              proměnné D$.
```

LIST

- Vypíše program BASIC nacházející se momentálně v paměti počítače.

LIST [řádek|první-|první-poslední|-poslední]

Příkaz LIST zobrazí výpis programu BASIC, který byl zaveden a uložen v paměti Commodoru 128 tak, že je možné program číst a měnit. Jestliže se LIST použije samotný (nenásledovaný čísly) Commodore zobrazí na obrazovce výpis celého programu. Tento proces může být zpomalen přidržením tlačítka Commodore současně s tlačítkem CONTROLS nebo NO SCROLL a může být znovu obnoven stisknutím libovolného tlačítka. Proces zastavíte stisknutím tlačítka RUN/STOP. Jestliže za slovem LIST následuje číslo žádku, Commodore 128 zobrazí pouze tento řádek. Jestliže natisknete LIST s dvěma čísly oddělenými pomlčkou, zobrazí se všechny řádky, počínajíc prvním uvedeným číslem a končíc druhým číslem včetně. Jestliže za LIST následuje číslo a pomlčka, Commodore 128 zobrazí všechny řádky počínaje tímto číslem až do konce programu. Nakonec, jestliže za LIST následuje pomlčka a potom číslo, zobrazí se všechny řádky od začátku programu do řádku s uvedeným číslem. Pomocí těchto variant můžeme vypsat

na obrazovce libovolnou část programu pro případnou kontrolu nebo provedení změn. V modu C 128 můžeme LIST použít v programu.

Příklady:

LIST	Vypíše celý program.
LIST 100-	Výpis od řádku 100 do konce programu
LIST 10	Vypíše pouze řádek 10.
LIST -100	Vypíše všechny řádky od začátku programu do řádku 100 včetně.
LIST 10-200	Vypíše řádky od 10. do 200. včetně.

LOAD

- Přenese do paměti počítače program z periferního zařízení, například z diskdrajvu nebo kazetového přehrávače.

LOAD ["jméno fajlu"][,číslo zařízení][,flag uložení]

Tento příkaz se používá pro vyvolání programu, uloženého na disku nabo na pásku. Jméno fajlu je jméno programu, obsahující maximálně 16 znaků uzavřených v uvozovkách. Jméno je následováno čárkou (za uvozovkami) a číslem, které reprezentuje číslo zařízení, na němž je program uložen (disk nebo přehrávač). Jestliže toto číslo není uvedeno, Commodore 128 dosadí číslo zařízení 1 (přehravač kazetový Dataset).

Flag uložení je číslo (0 nebo 1), které určuje, kde bude program uložen v paměti počítače. Flag uložení 0 přikazuje Commodoru 128 uložit program na začátku programové oblasti BASIC. Flag 1 přikazuje počítači uložit program na místo, kde byl původně dislokován. Implicitní (tedy neuvedená) hodnota flagu je 0. Hodnota parametru 1 se obecně používá při ukládání programů ve strojovém kódu.

Zařízení obvykle používané s příkazem LOAD je diskdrajv (zařízení číslo 8), ačkoliv v případě práce s diskdrajvem je pohodlnější používat příkaz DLOAD. Jestliže použijeme LOAD bez argumentů, následované RETURN, C 128 předpokládá, že se čte z pásku a zobrazí "PRESS PLAY ON TAPE" (stiskni PLAY na přehrávači). Po stisknutí PLAY Commodore 128 začne na pásku hledat program. Ponnalezení programu Commodore 128 napíše FOUND "jméno fajlu" (NALEZL "jméno fajlu"), kde jméno fajlu je jméno prvního fajlu, nalezeného na pásku v DATASETu. Stiskněte tlačítko Commodore jestliže byl nalezen požadovaný fajl nebo stiskněte klávesu mezery, jestliže chcete, aby hledání pokračovalo. Po přenesení programu do paměti počítače jej lze provést, vypsat případně upravovat.

Poznámka: Stisknutí klávesy mezery v modu C 64 způsobí hledání následujícího fajlu.

Příklady:

LOAD	Čte nejbližší program na pásku.
LOAD "AHOJ"	Hledá na pásku program nazvaný AHOJ a přenese jej do paměti počítače, pokud jej nalezne.
LOAD A$,8	Přenese z disku do paměti počítače program, jehož jméno je uloženo v proměnné A$. Tento příkaz je ekvivalentní DLOAD (A$)
LOAD "AHOJ",8	Hledá program pod jménem AHOJ na diskdrajvu 8, drajv 0 (je ekvivalentní DLOAD "AHOJ")
LOAD "LINGMAC",8,1	Uloží program (v strojovém kódu) nazvaný LINGMAC v místě, z něhož byl uložen na disk.

Příkaz LOAD můžeme použít v programu BASIC, aby byl nalezen a proveden následující program z pásku nebo z disku. Tato operace se nazývá sdružování.

LOCATE

- Umístí na obrazovce pixelový kursor bodové matrice.

LOCATE X,Y

Příkaz LOCATE umístí pixelový kursor (PK) v určených pixelových souřadnicích na obrazovce.

MONITOR

Pixelový kursor (PK) je souřadnice obrazovky s bodovou matricí, z něhož se začínají kreslit kružnice, čtverce, čáry a body a v níž začíná vybarvování. Souřadnice PK se mění od hodnoty 0,0 do hodnoty 319,199. Na rozdíl od textového kursoru PK není viditelný, ale je ovládán grafickými příkazy (BOX, CIRCLE, DRAW atd). Jestliže není uvedeno jinak, je pixelový kursor umístěn v bodě se souřadnicemi X,Y, které jsou uvedeny v každém grafickém příkazu. V těchto případech není nutné příkaz LOCATE používat.

Příklad:

LOCATE 160,100 Umístí PK ve středu obrazovky s bodovou matricí. Kursor není viditelný, dokud nezačneme něco kreslit.

PK lze likalizovat pomocí funkce RDOT(0) - souřadnice X a RDOT(1) - souřadnice Y. Barvu PK můžeme identifikovat pomocí příkazu RDOT(2).

MONITOR

- Aktivuje monitor strojového kódu Commodoru 128.

MONITOR

Viz Doplněk J s informacemi o monitoru strojového kódu Commodoru 128.

MOVSPR

- Umísťuje nebo přemísťuje sprajty po obrazovce.

MOVSPR číslo,xl,yl Umístí daný sprajt v bodě s absolutními souřadnicemi xl,yl.

MOVSPR číslo+|-x,+|-y Posune sprajt z běžné polohy.

MOVSPR číslo,x;y Přemístí sprajt na vzdálenost x a pod úhlem y z dané polohy.

MOVSPR číslo,úhel x#rychlost y Přemístí sprajt pod úhlem x (ve směru hodinových ručiček) z původní polohy danou rychlostí y.

kde:

číslo je číslo sprajtu (od 1 do 8)

<,xl,yl> je souřadnice polohy sprajtu

úhel je úhel (0-360) pohybu(odečítaný ve směru hodinových ručiček) z původní polohy sprajtu

rychlost je rychlost (0-15) pohybu sprajtu

Tento příkaz umístí sprajt v určeném místě obrazovky v souladu s konvencí pro souřadnice sprajtu (liší se od konvence pro bodovou matrici) nebo vyvolá pohyb sprajtu se zadanou rychlostí. Viz MOVSPR v části 6. manuálu s podrobným vysvětlením souřadnic sprajtů.

Příklady:

MOVSPR 1,150,160 Umístí sprajt 1 nedaleko

	středu obrazovky v souřadnici X,Y: 150,160
MOVSPR 1,+20,-30	Posune sprajt 1 doprava o 20 dílků a nahoru o 30 dílků.
MOVSPR 4,-50,+100	Posune sprajt 4 doleva o 50 dílků a dolů o 100 dílků.
MOVSPR 5,45#15	Přemísťuje sprajt 5 pod úhlem 45 stupňů (odečteno ve směru hodinových ručiček) z jeho původní polohy se souřadnicemi x a y. Sprajt se přemísťuje největší rychlostí (15).

Poznámka: Po stanovení úhlu a rychlosti ve třetím tvaru příkazu MOVSPR musíte poté předepsat rychlost 0, aby se pohyb sprajtu zastavil.

NEW

- Zruší program a proměnné v paměti počítače.

NEW

Tento příkaz zruší celý program nacházející se v paměti počítače a zruší všechny používané proměnné. Jestliže program nebyl včas uložen na disk nebo na pásku, bude po této operaci ztracen. Používejte tedy tento příkaz opatrně. Příkaz NEW můžete rovněž použít jako příkaz programu BASIC. Jakmile Commodore 128 narazí na řádek programu s NEW, program se zruší a počítač se zastaví.

ON

- Podmíněný skok na řádek programu určený v závislosti na výsledku zadaného výrazu.

ON výraz ⟨GOTO|GOSUB⟩ řádek#1 [,řádek#2,...]

Tento příkaz umožňuje, aby příkazy GOTO a GOSUB fungovali jako speciální varianta příkazu IF (podmínka). Za slovem ON následuje výraz, dále GOTO nebo GOSUB a seznam čísel řádků, oddělených čárkami. Jestliže hodnota výrazu je 1, provede se skok na první řádek seznamu. Jestliže výsledek je 2, provede se skok na druhý řádek atd. Jestliže je výsledek 0 nebo větší než počet položek v seznamu, program pokračuje příkazem bezprostředně následujícím za příkazem ON. Jestliže je toto číslo záporné, objeví se chybové hlášení ILLEGAL QUANTITY ERROR (nepřípustná hodnota).

Příklad:

```
10 INPUT X: IF X<0 THEN 10
20 ON X GOSUB 30,40,50,60
25 GOTO 10
30 PRINT "X=1":RETURN
40 PRINT "X=2":RETURN
50 PRINT "X=3":RETURN
60 PRINT "X=4":RETURN
```

Jestliže X=1 ON přeskočí na řádek s prvním číslem v seznamu (30). Jestliže X=2 ON přeskočí na druhý v pořadí řádek (40) atd.

OPEN

- Otevíra fajl pro vstup a výstup.

OPEN číslo logického fajlu, číslo zařízení
[,sekundární adresa][<,"jméno fajlu,
typ fajlu, mod"|,řetězec cmd>]

Příkaz OPEN umožňuje Commodoru 128 během provádění programu přístup k fajlům na takových zařízeních jako jsou diskdrajv, kazetový přehrávač, tiskarna nebo obrazovka.Commodoru 128.
Za slovem OPEN nasleduje číslo logického fajlu, na něž se poté odvolávají všechny další příkazy BASICu pro vstup/výstup, například příkazy PRINT# (zaznam), INPUT# (čtení) atd.
Toto číslo může být v rozmezí 1-255.
Druhé číslo, nazývané číslem zařízení, následuje za číslem logického fajlu.
Číslo zařízení:

0 klávesnice Commodoru 128
1 kazetovy přehravač
3 obrazovka Commodoru 128
4-7 tiskárny
8-11 diskdrajvy.

Doporučuje se přiřadit fajlu číslo rovné číslu příslušného zařízení, takže se snadno zapamatovává.

Bezprostředně za číslem zařízení následuje třetí parametr - sekundární adresa. Jestliže se používá kazetový přehrávač, vedlejší adresa musí být 0 pro čtení, 1 pro záznam a 2 pro záznam s ukazatelem KONEC PÁSKU na konci.
V případě, že používáte disk, sekundární adresa udává číslo kanálu. Viz manuál diskdrajvu s dalšími podrobnostmi o kanálech a jejich číslování. Pokud se týká tiskárny, sekundární adresy se používají pro volbu některých programovatelných funkcí.
Za sekundární adresou můžete specifikovat jméno fajlu pro použití na disku nebo na pásku, nebo řetězec (cmd), kterým může být příkaz pro diskdrajv nebo kazetový přehrávač, nebo jméno fajlu na pásku či disku. Jestliže se specifikuje jméno fajlu, typ a mód se vztahují pouze na diskové fajly. Typy fajlů jsou: programový, sekvenční, relativní a uživatelský, mód je čtení nebo záznam.

Příklady:

```
10 OPEN 3,3          Otevře obrazovku jako fajl
                     číslo 3.
20 OPEN 1,0          Otevře klávesnici jako fajl
                     číslo 1.
30 OPEN 1,1,0,"BOD"  Otevře kazetový nahrávač
                     pro čtení jako fajl 1 s
                     použitím BOD jako jméno
                     fajlu
```

OPEN 4,4 Otevře tiskárnu jako fajl č. 4.

OPEN 15,8,15 Otevře příkazový kanál disku jako fajl 15. Sekundární adresa č. 15 je rezervována pro chybový kanál diskdrajvu.

OPEN 8,8,12,"PROV,SEQ,SCRIT" Otevře sekvenční diskový fajl pro záznam, název fajlu PROV, číslo fajlu 8, sekundární adresa 12.

Viz rovněž příkazy CLOSE, CMD, GET#, INPUT# a PRINT# a sytémové proměnné ST, DS a DS$.

PAINT

- Vyplňuje plochu barvou.

PAINT [zdrojová barva],x,y[,mod]

kde:

zdrojová barva - 0 popředí bodové matrice
1 pozadí bodové matrice
2 multicolor 1
3 multicolor 2

x,y - výchozí souřadnice (pokud není uvedena je to poloha pixelového kursoru (PK)

mód - 0=barví určenou oblast zvolenou zdrojovou barvou
1= barví určenou oblast zdrojovou barvou jinou než pozadí

Příkaz PAINT vyplní plochu barvou: oblast vybarvení je dána obvodem, s vyloučením určitých souřadnic X a Y. Není možné vybarvit ty body, jejichž zdrojovou barvou je barva pixelového kursoru.

Jestliže mod=0 vyplňovaná oblast musí být omezena zdrojovou barvou, jakákoliv jiná zdrojová barva uvnitř tohoto obvodu bude "přebarvena" (příklad č. 1).

Jestliže mod=1 za obvod obrazce se považuje jakákoliv zdrojová barva (kromě 0). Žádná zdrojová barva nemůže být "přebarvena" (znamená to, že pouze nevybarvené oblasti mohou být vyplněny jestliže mod=1)(příklad 2).

Příklad 1:

```
10 COLOR 0,1: COLOR 1,2: COLOR 2,5: COLOR 3,7
20 GRAPHIC 3,1          Multibarevná grafika
30 CIRCLE 1,80,100,30   Kreslí kružnici zdrojovou barvou
40 CIRCLE 3,80,100,35   Kreslí kružnici zdrojovou barvou 3
50 BOX 2,80,100,90,110,45,1  Nakreslí plný kosočtverec zdrojovou barvou 2.
60 PAINT 3,70,100,0     Vybarví vnitřekkružnic zdrojovou barvou 3 a zarazí se pouze na této zdrojové barvě (přebarví tedy kosočtverec i menší kružnici).
```

Příklad 2:

Jako příklad 1, změnte však řádek 60, následovně:

```
60 PAINT 3,70,100,1  Vybarví vnitřek kružnice a zarazí se na jakékoliv zdrojové barvě jiné než pozadí.
```

Příklad 3:

Jako příklad 2, přidejte následující řádky 70 a 80:

```
70 COLOR 2,8  Změní zdrojovou barvu 2 na žlutou
80 PAINT 2,90,110,1 Zkouší přebarvit čtverec (pokus se nepodaří, protože zdrojová barva v PAINT a v bodu (90,110) je táž (2)).
```

PLAY

- Definuje a hraje hudební noty a prvky.

PLAY "[Vn][On][Tn][Un][Xn][prvky][,....]"

kde:

Vn = hlas (n=1-3)

On = oktáva (n=0-6)

Tn = zabarvení motivu (n=0-9)

0 = klavír	5 = kytara
1 = harmonika	6 = klavičembalo
2 = kaliope	7 = varhany
3 = buben	8 = trubka
4 = flétna	9 = xylofon

Un = hlasitost (n=0-9) 0-vypnuto,9-plná síla (VOL 15)

Xn = filtr zapnut (n=1), vypnut (n=0)

Prvky: Noty: A,B,C,D,E,F,G

- # křížek, zvyšovací znaménko*
- $ snižovací znaménko*
- W celá nota
- H poloviční
- Q čtvrtinová
- I osminová
- S šestnáctinová
- . prodloužená nota*
- R pauza
- M čeká až všechny hlasy, které právě zní, ukončí právě hraný takt.

Příkaz PLAY umožňuje zvolit hlas, oktávu a zabarvení motivu (zahrnující deset predefinovaných hudebních nástrojů), hlasitost a požadované noty. Všechny tyto řídící znaky musí být uzavřeny v uvozovkách.

Všechny prvky, kromě R a M, předcházejí hudební noty v řetězci PLAY.

* Tyto symboly musí předcházet každou notu, pro kterou platí.

Příklady:

```
PLAY "V1O4T0U5X0CDEFGAB"  Zahraje noty C,D,E, F,G,A,B prvním hlasem ve 4. oktávě zabarvením 0 (klavír), hlasitostí 5 s vypnutým filtrem.
```

PLAY "V305T6U7X1↳B$AW.CHDQEIF" Zahraje noty zvýšené B, snížené A, prodloužené celé C, poloviční D, čtvrtinové E a osminové F.

Poznámka: abyste vyslechli co nejkvalitnější tóny musíte zapnout filtr (zkuste FILTER 1024,1)

POKE

-- Mění obsah paměťových buněk RAM.

POKE adresa, hodnota

Příkaz POKE umožňuje měnit kteroukoliv hodnotu v RAM Commodoru 128 a umožňuje měnit obsah většiny registrů I/O Commodoru 128. Za klíčovým slovem POKE vždy následují dva parametry. Prvé z nich určuje místo v paměti Commodoru 128 (s hodnotou od 0 do 65535). Druhý parametr s hodnotou od 0 do 255 nahrazuje v uvedeném místě jakoukoliv předešlou hodnotu. Hodnota paměťové adresy určuje seskupení adresových bitů paměti. V modu C 128 POKE působí v průběžně určené bance RAM. Adresa POKE závisí na čísle banky. Viz příkaz BANK v této Encyklopedii, kde naleznete příslušné vhodné konfigurace BANK.

Příklad:

10 POKE 53280,1 Změní barvu rámečku VICu (banka č. 15 v modu C 128).

Poznámka: PEEK, funkce spřažená s POKE, která vrací obsah zadané paměťové buňky, je popsána v části FUNKCE.

PRINT

- Vyvolá výstup na textovou obrazovku.

PRINT [seznam tisků]

Příkaz PRINT je nejdůležitějším příkazem výstupu v jazyce BASIC. Je to první příkaz BASICu, který jsme se naučili a může být různě modifikován. Za slovem PRINT musí následovat některá z následujících dat:

Znaky v uvozovkách	("text")
Jména proměnných	(A, B, A$, XA)
Funkce	(SIN(23),ABS(33))
Rozdělovací znaménka	(;,)

Znaky v uvozovkách budou přesně v stejném tvaru zobrazeny na obrazovce. Mimoto budou vytištěny hodnoty proměnných (číselných nebo stringových) a numerické hodnoty funkcí.
Rozdělovací znaménka se používají pro uspořádání dat na obrazovce. Čárkou oddělená data budou vytištěna s 10 mezerami, zatímco středník způsobí tisk těsně vedle sebe (viz část 3., tisk čísel). Obě oddělovací znaménka

můžete použít jako poslední znak příkazu. Následující příkaz tisku se pak provádí, jako by byl pokračováním tohoto příkazu.

Příklady:	Tisk:
10 PRINT "AHOJ"	AHOJ
20 A$="PRITELI":PRINT "AHOJ";A$	AHOJ PRITELI
30 A=4:B=2:PRINT A+B	6
40 J=41: PRINT J;:PRINT J-1	41 40
50 PRINT A;B;: D=A+B	
PRINT D;A-B	4 2 6 2

Viz rovněž funkce POS, SPS a TAB.

PRINT#

- Odesílá data do fajlu.

PRINT# číslo fajlu, seznam tisků

Existují jisté rozdíly mezi tímto příkazem a příkazem PRINT. Nejdůležitější z nich je, že slovo PRINT# musí být následováno číslem, reprezentujícím datový fajl, otevřený již dříve. Číslo následující za čárkou je seznam prvků, odesílaných do fajlu. Středník funguje stejným způsobem pro mezery na tiskárnách jako v příkazu PRINT, čárka zavádí 10 mezer. Žádné zařízení nefunguje s TAB a SPC.

Příklad:

10 OPEN 4,4	Odešle AHOJ a proměnné A$,B$ na tiskárnu.
20 PRINT#4,"AHOJ",A$,B$	
10 OPEN 2,8,2,"FILE,S,W"	Odešle proměnné A, B$, C a D do diskového fajlu číslo 2.
20 PRINT#2,A,B$,C,D	

Poznámka: Příkaz PRINT# se používá také samotný pro obnovení kanálu k zařízení po vyslání prostřednictvím CMD a dříve než uzavřeme fajl, například:

```
OPEN 4,4
CMD 4
PRINT#4
CLOSE 4
```

Viz rovněž příkaz CMD.

PRINT USING

- Určuje formát výstupu.

PRINT[#číslo fajlu,] USING "formát";seznam tisků

Tento příkaz určuje tvar stringových nebo číselných prvků při výstupu dat na obrazovku, na tiskárnu nebo jiné zařízení. Tvar je uzavřen v uvozovkách - je to t zv. formát. Přidejte za něj středník a seznam veličin, které mají být vytištěny tímto formátem - seznam tisků.

Seznam tisků musí obsahovat proměnné nebo jiné hodnoty určené k tisku, oddělené řárkami.

Formátový řetězec používaný pro

Znaky	čísla	stringy
Symbol čísla (#)	X	X
Znak plus (+)	X	
Znak minus (-)	X	
Desetinná tečka (.)	X	
Čárka (,)	X	
Znak dolaru ($)	X	
4 aksanty (^^^^)	X	
Symbol rovnosti (=)		X
Znak větší než (>)		X

Znak čísla (#) rezervuje místo pro jeden znak ve výstupním poli. Jestliže prvek numerických dat obsahuje více znaků, než kolik bylo symbolů # ve formátovém poli, pole bude vyplněno hvězdičkami (*) a hvězdičky budou vytištěny.

Příklad:

```
10 PRINT USING "####";X
```

Pro různé hodnoty X formát zobrazí:

```
X=12.34          12
X=567.89         568 (zaokrouhlí)
X=123456         ****
```


Pro stringové prvky budou stringová data uříznuta na délku formátového pole. Vytiskne se tolik znaků, kolik bylo symbolů čísla (#) ve formátovém poli. Uřezává se zprava.

Znaky plus(+) a minus (-) se mohou použít na prvním místě formátového pole, ne však v obou polohách. Pak se vytiskne znak plus jestliže číslo je kladné, nebo znak minus jestliže číslo je záporné.

Jestliže použijete znak minus a číslo je kladné, na místě znaku indikujícího znak minus bude ponechána prázdná mezera.

Jestliže ve formátovém poli pro prvky číselných dat nepoužijete ani znak plus, ani znak minus a číslo je záporné, vytiskne se před první číslicí znak minus nebo znak dolaru. Jestliže je číslo kladné, netiskne se žádný symbol. Znamená to, že se tiskne dodatečný znak - znak minus - jestliže je číslo záporné. Jestliže je znaků více, než se může umístit do pole určeného počtem symbolů čísla a znaků plus/minus, dojde k overflow (přetečení) a pole bude vyplněno hvězdičkami (*).

Znak desetinné tečky (.) označuje polohu desetinné tečky v čísle. V každém formátovém poli může být pouze jedna desetinná tečka

Jestliže formátové pole neobsahuje desetinnou tečku, číslo se zaokrouhlí na nejbližší celé číslo a bude vytištěno bez desetinné tečky.

Jestliže je poloha desetinné tečky určena, počet číslic před desetinnou tečkou (včetně znaku minus, jestliže je číslo záporné) nesmí překročit počet symbolů čísla předdesetinnou tečkou. Jestliže nějaké číslice přebývají, dojde k overflow a pole bude vyplněno hvězdičkami (*).

Čárka (,) umožňuje zavést čárky do číselného pole. Poloha čárky ve formátu ukazuje místo, kde se čárka objeví ve vytištěném čísle. Tiskne se čárka pouze uvnitř čísla. Nevyužitá čárka vlevo od první číslice se objeví jako vyplňující znak (např. mezera). Před první čárkou v poli musí být alespoň jeden symbol čísla (#).

Jestliže je v číselném poli uvedena čárka a číslo je záporné, vytiskne se znak minus jako první znak rovněž v případě, že poloha tohoto znaku bude specifikována jako čárka.

Symbol dolaru ($) ukazuje, že v čísle bude vytištěn symbol dolaru. Jestliže tento symbol je pohyblivý (má být vytištěn vždy bezprostředně před číslem), musí se před ním nacházet symbol symbol čísla (#). Jestliže je symbol dolaru uveden bez předcházejícího mu symbolu čísla, vytiskne se symbol dolaru vždy v té pozici, v které je uveden ve formátu. Jestliže je ve formátovém poli specifikován symbol plus nebo minus spolu se symbolem dolaru, program vytiskne znak plus nebo minus před symbolem dolaru.

Příklady:

Pole	Výraz	Výsledek	Komentář
###.#	-.1	-0.1	Přidá se nula jako první číslice
##.#	1	1.0	Přidá se poslední nula
####	-100.5	-101	Zaokrouhlí se
####	-1000	****	Dojde k overflow, protože čtyři číslice a znak minus se do pole neumístí.
##.	10	10.	Přidá se desetinná tečka.
#$##	1	$1	Přidá se symbol dolaru na začátek.

Šipky nahoru, nebo akcenty (^^^^) určují, že číslo se má tisknout ve formátu E (vědecké značení). Ke specifikování šířky pole musí být použit symbol čísla za čtyřmi akcenty.

Aksanty musí být umístěny za symbolem čísla ve formátovém poli. Abychom vytiskli číslo ve formátu E musíme tedy uvést čtyři aksanty. Jestliže uvedeme méně než čtyři, objeví se syntaktická chyba. Jestliže naopak specifikujeme více než čtyři aksanty, vezmou se do úvahy pouze první čtyři a zbylé budou interpretovány jako textové symboly. Pro umístění řetězce v poli se používá symbol rovnosti (=). Šířka pole se určuje počtem znaků (symbolů čísel a rovnítek) ve formátovém poli. Jestliže řetězec obsahuje méně znaků než může být umístěno v poli, řetězec bude vystředován v poli. Jestliže řetězec obsahuje více znaků než kolik může pole obsáhnout, znaky zcela napravo budou odříznuty a řetězec zcela zaplní pole. Pro upravení zprava řetězce v poli se používá symbol větší než (>). Do formátového řetězce lze zahrnout další znaky, s kterými bude zacházeno jako s alfanumerickými znaky. Umožňuje to vytvářet tabulky a grafy. Viz řádek 30 následujícího programu.

Příklad:

```
5 X=32: Y=100.23: A$="KOCKA": B$="POCITAC"
6 F$="##=######.##=####.##"+CHR$(13)
```



```
10 PRINT USING "$###.##";13.25,X,Y
20 PRINT USING "###>#";"CBM",A$
30 PRINT USING F$;A$,X,B$,Y
```

CHR$(13) je RETURN

Po provedení řádek 10 vytiskne:

```
$13.25$32.00$*****
```

Pět hvězdiček (*****) se vytiskne místo hodnoty Y, protože Y má 5 cifer, což neodpovídá formátu (jak bylo vysvětleno výše).

Řádek 20. vytiskne:

```
CBM KOCKA
```

s dvěma mezerami před prvním řetězcem, jak to určuje formát.

Řádek 30. vytiskne:

```
  KOCKA                $23.00
  POCITAC              $100.23
```

PUDEF

- Předefinuje symboly v příkazu PRINT USING.

PUDEF "nnnn"

nnnn je kombinace znaků, maximálně čtyř, a PUDEF umožňuje redefinovat kterýkoliv z následujících čtyř symbolů příkazu PRINT USING: mezera, čárka, desetinná tečka a symbol dolaru. Tyto čtyři symboly mohou být převedeny na jiné znaky uvedením těchto nových znaků na správném místě v ovládacím řetězci PUDEF.

Pozice 1 je vyplňujícím znakem. Jestliže není uveden, bude jím prázdná mezera. Uveďte na tomto místě nový znak, který má zaměnit mezeru. Pozice 2 je čárka. Pokud zde není nic uvedeno, zůstane tímto znakem čárka. Pozice 3 odpovídá desetinné tečce. Pokud není znak uveden, zůstane jím desetinná tečka. Pozice 3 je znak dolaru. Pokud není jiný znak uveden, zůstane jím dolar.

Příklady:

10 PUDEF "m" Tiskne se m místo prázdných mezer.

20 PUDEF "<" Tiskne se < na místě čárky.

Poznámka: Musí být specifikovány všechny pozice až do té, kterou je třeba zaměnit. Například PUDEF " $" tiskne dolar na místě symbolu dolaru, zatímco desetinná tečka, čárka a výplnkový znak budou přeměněny v mezery.
PUDEF se používá pouze v numerických formátech, t.j. PUDEF "0" přemění výplnkové mezery v číslech na 0, ale nezmění výplnkové mezery v řetězcích.
Znak zaměnující symbol $ nemá vliv, ledaže mu předchází symbol # (pohyblivý).

READ

- Čte data z příkazu DATA a zavádí je do paměti počítače (během prováděcí fáze programu).

READ seznam proměnných

Tento příkaz vybírá informace z příkazu DATA a uloží je do proměnných, takže data mohou být použita programem během jeho provádění. Seznam proměnných příkazu READ může obsahovat jak stringové, tak číselné proměnné. Je nutné se vyvarovat čtení řetězce jestliže příkaz READ čeká číslo a naopak, jinak se objeví hlášení TYPE MISMATCH ERROR (chyba - míchání typů).
Data v příkazu READ se čtou po řadě. Každý příkaz READ přečte jeden nebo více datových prvků. Každá proměnná v příkazu READ vyžaduje jeden datový prvek. Jestliže nebude tento datový prvek dodán, objeví se hlášení OUT OF DATA ERROR (chyba - nejsou data).
V programu je možné přečíst všechna data a potom je číst znovu pomocí příkazu RESTORE. RESTORE umístí datový pointer na počátek, takže data lze začít číst znovu. Viz příkaz RESTORE.

Příklady:

```
10 READ A,B,C
20 DATA 3,4,5
```

Čte 3 datové prvky (musí jimi být čísla, aby se neobjevilo chybové hlášení) do proměnných A, B, C.

```
10 READ A$,B$,C$
20 DATA JITKA,HONZA,
   FRANTA
```

Přečte tři řetězce z příkazu DATA.

```
10 READ A,B$,C
20 DATA 1200,MARIE,345
```

Přečte číselnou proměnnou, řetězcovou proměnnou a další číselnou proměnnou.

RECORD

- Umísťuje pointery relativního fajlu.

RECORD #číslo logického fajlu, číslo záznamu [,byte]

Tento příkaz umístí pointer relativního fajlu, aby se vybraly všechny byty (znaky) z celého záznamu v relativním fajlu. Číslo logického fajlu je z rozmezí od 1 do 255. Číslo záznamu může mít hodnotu od 1 do 65535. Číslo bytu může být od 1 do 254. Podrobnosti o relativních fajlech viz v manuálu diskdrajvu.
Jestliže číslo záznamu bude mít hodnotu větší než číslo posledního záznamu ve fajlu, stane se následující:

V případě záznamu (PRINT#) se vytvoří dodatečný záznam, aby se rozšířil fajl, a přidá se potřebný počet záznamů.
V případě čtení (INPUT#) bude vrácen nulový záznam a objeví se hlášení RECORD NOT PRESENT ERROR (chyba - záznam není přítomen).

Příklady:

```
10DOPEN #2"ZAKAZNICI"
20 RECORD #2,10,1
30 PRINT#2,A$
40 DCLOSE#2
```

V této ukázce se otevírá na řádku 10. relativní fajl nazvaný ZAKAZNICI jako fajl číslo 2. Řádek 20. umístí pointer relativního fajlu na první bajt záznamu č. 10. Řádek 30. zapíše data A$ do fajlu.

Příkaz RECORD chápe proměnné jako parametry. Doporučujeme používat příkaz RECORD ve smyčkách FOR....NEXT a DO. Viz rovněž DOPEN.

REM

- Zavádí komentáře nebo poznámky, pomocí nichž se sleduje vývoj programu.

REM [text]

Příkaz REM představuje uživatelovy poznámky, které vysvětlují výpis programu. REM může vysvětlovat část programu, obsahovat

jméno autora tohoto programu atd. Příkaz REM neovlivňuje průběh programu, ovlivňuje však jeho délku (zabírá prostor v paměti). Počítač neinterpretuje jako příkaz to, co následuje vpravo za klíčovým slovem REM. Tudíž v řádku, kde se nalézá REM nesmí být prováděné příkazy.

Příklad:

```
1010 NEXT X: REM Konec smycky hlavniho progra-
     mu
```

RENAME

- Mění jméno fajlu na disku.

RENAME [Dčíslo drajvu,] "staré jméno fajlu" TO "nové jméno fajlu" [<ON|,>Učíslo zařízení]

Tento příkaz se používá při přiřazení nového jména fajlu na disku. Diskdrajv nepřejmenuje fajl jestliže je fajl otevřen.

Příklady:

```
RENAME D0,"ZKOUSKA" TO "ZKDEF"   Změní jméno fajlu
                                 ZKOUSKA na ZKDEF

RENAME D0,(A$) TO (B$),U9        Změní jméno fajlu uvedené v A$
                                 na jméno specifikované v B$ na drajvu 0, zařízení 9. Jestliže se jako
```


jméno fajlu použije stringová proměnná, uzavřete ji do závorek

RENUMBER

- Přečísluje řádky programu BASIC.

RENUMBER [nové číslo prvního řádku][,přírustek][,staré číslo prvního řádku]

Nový počáteční řádek znamená číslo prvního řádku programu po přečíslování; jestliže není uveden bude to 10.
Přírůstek je interval mezi čísly řádků (tj. 10, 20, 30 atd.); pokud není uveden, bude 10.
Staré číslo počátečního řádku je číslo prvního řádku před přečíslováním programu. Umožňuje to přečíslovat pouze určitou část programu. V tomto případě se za neuvedenou hodnotu dosadí číslo prvního řádku. Tento příkaz se používá pouze v přímem modu.
Jestliže počítač narazí na odvolání na neexistující řádek, oznámí "UNRESOLVED REFERENCE ERROR" (chyba - nerozluštitelné odvolání se). Jestliže přečíslovaní rozšíří program za dovolené meze, oznámí počítač OUT OF MEMORY (nepostačující paměť). Zobrazí se LINE NUMBER TOO LARGE ERROR (chyba - příliš velké číslo řádku), pokud příkaz

RENUMBER vygeneruje řádek s číslem 64000 nebo větším. Obě tato chybová hlášení nemají vliv na program.

Příklady:

RENUMBER	Přečísluje řádky programu počínajíc 10. s přírustkem 10.
RENUMBER 20,20,15	Přečísluje program počínajíc řádkem 15. Rádek 15 se stane řádkem 20, a všechny další řádky budou mít přírustek 20.
RENUMBER ,,65	Přečísluje program s přírustkem 10, počínaje řádkem 65. Řádek 65 se stane řádkem 10. Jestliže chcete vynechat parametr, musíte na jeho místě uvést čárku. Neměly by být žádné řádky mezi 10 a 64 včetně.

Uložte vždy program dříve než budete měnit jeho číslování, protože příliš dlouhé programy mohou způsobit potíže s přečíslováním.

Vězte rovněž, že dlouhé programy mohou být přečíslovány rychleji - přečíslování může normálně trvat dost dlouho, až 30 minut pro program 55 K rychlým způsobem.

Jestliže používáte mod se 40 sloupci, napište FAST: RENUMBER..... RETURN, potom napište SLOW: RETURN. Během provádění přečíslování se na obrazovce nic neobjeví, až po skončení operace.

Jestliže používáte mod s 80 sloupci, zvolte obrazovku s 80 sloupci dříve než napíšete FAST.

RESTORE

Přemístí pointer READ tak, že data bude možné číst znovu.

RESTORE	mod C64
RESTORE [řádek#]	mod C128

Jestliže se tento příkaz provede v programu, pointer prvků příkazu DATA, která mají být čtena, se nastaví na první prvek příkazu DATA. Umožní to číst data znovu. Jestliže za příkazem RESTORE následuje číslo řádku, pointer READ se nastaví na první datový prvek následující za tímto číslem řádku programu. V opačném případě bude pointer nastaven na začátek programu BASIC.

V modu C 64 není možné specifikovat číslo řádku a RESTORE se může použít pouze počínajíc začátkem programu.

Příklady:

```
10 FOR I=1 TO 3
20 READ X
30 T=T+X
40 NEXT
50 RESTORE
60 GOTO 10
70 DATA 10,20,30
45 PRINT T
```

Tento program čte data na řádku 70 a ukládá je do číselné proměnné X. Přidá je k součtu všech číselných dat. Po vytištění součtu všech tří dat bude pointer READ přenesen na začátek programu (řádek 10) a program se provádí do nekonečna.

```
10 READ A,B,C
20 DATA 100,500,750
30 READ Y,X,Z
40 DATA 36,24,38
50 RESTORE 40
60 READ S,P,Q
70 PRINT A,B,C
80 PRINT X,Y,Z
90 PRINT S,P,Q
```

V této ukázce se pointer DATA vrací na začátek datových prvků na řádku 40. Při provedení řádku 60 čte pointer data 36,24,38 na řádku 40, protože již není potřebné číst znovu data na řádku 20.

Poznámka: Jestliže specifikujete číslo řádku přesvědčte se, že řádek v programu skutečně existuje. Není přitom možné použít proměnnou, například RESTORE LR.

RESUME

- Určuje bod, v němž se znovu začne provádět program poté, co došlo k chybě.

RESUME [řádek# | NEXT]

Tento příkaz se používá pro obnovení provádění programu poté, co byla zaregistrována chyba pomocí TRAP. Jestliže v RESUME není žádný parametr, počítač se pokusí provést znovu řádek, v němž objevil chybu. RESUME NEXT pokračuje příkazem bezprostředně následujícím za příkazem, v němž došlo k chybě.; RESUME následované číslem řádku způsobí přeskok na specifikovaný řádek a provádění programu pokračuje odtud. RESUME může být použito pouze v programovém modu.

Příklad:

```
10 TRAP
20 INPUT "ZADEJ CISLO",A
30 B=100/A
40 PRINT "VYSLEDEK=",B: PRINT "KONEC"
50 PRINT "CHCES TO PROVEST ZNOVU (A/N)"
   :GETKEY Z$: IF Z$="A" THEN 20
60 STOP
100 INPUT "ZADEJ JINE CISLO (NENULOVE)";A
110 RESUME
```

V této ukázce dojde k chybě "dělení nulou" v řádku 30. jestliže bude zadána 0 v řádku 20. Jestliže zadáte nulu program přeskočí na řádek 100, kde vyžaduje zadání nenulového

čísla. Řádek 110 vrací program na řádek 30., aby byl výpočet dokončen. Řádek 50. se ptá, zda požadujete zopakování programu. V případě souhlasu stiskněte tlačítko A.

RETURN

- Způsobí výstup z podprogramu.

RETURN

Tento příkaz se vždy používá s příkazem GOSUB. Jestliže program narazí na příkaz RETURN, přeskočí na příkaz bezprostředně následující za posledním provedeným příkazem GOSUB. V případě, že předtím nebyl žádný příkaz GOSUB proveden, objeví se hlášení RETURN WITHOUT GOSUB ERROR (chyba - RETURN bez GOSUB) a program se zastaví.
Všechny podprogramy musí být zakončeny příkazem RETURN.

Příklad:

```
10 PRINT "ZACATEK PODPROGRAMU"
20 GOSUB 100
30 PRINT "KONEC PODPROGRAMU"
.............
90 STOP
100 PRINT "PODPROGRAM 1"
110 RETURN
```


V této ukázce se volá podprogram na řádku 100, který vytiskne PODPROGRAM 1 a vrátí řízení na řádek 30, na zbývající část programu.

RUN

- Provede program BASIC.

1) RUN [řadek#]
2) RUN "jméno fajlu"[,Dčíslo drajvu][,Učíslo zařízení]
   Pouze v modu C 128.

Po zavedení nebo přepsání programu do paměti počítače jej příkaz RUN provede. Dříve než začne provádět program, RUN vynuluje všechny proměnné v tomto programu. Jestliže za příkazem RUN následuje číslo, program se začne provádět na řádku s tímto číslem. Jestliže je příkaz RUN následován jménem fajlu, bude tento fajl přepsán z diskdrajvu a proveden, aniž by uživatel podnikl cokoliv dalšího. RUN lze použít i v programu. Pokud není uvedeno, je číslo drajvu 0 a číslo zařízení 8.

Příklady:

RUN	Zahájí provádění (od začátku) programu, který je v daném okamžiku v paměti.
RUN 100	Začne provádět program počínajíc řádkem 100.
RUN "PRG1"	Přepíše (DLOAD) "PRG1" z diskdrajvu 8 a provede tento program od začátku.
RUN (A$)	Přepíše program pojmenovaný v proměnné A$ a provede jej od začátku.

SAVE

- Uloží program z paměti počítače na disk nebo pásek.

SAVE ["jméno fajlu"][,číslo zařízení][,flag EOT]

Tento příkaz uloží na pásek nebo na disk program, který se právě nachází v paměti. Jestliže SAVE uvedete samotné, fajl bude uložen na pásek beze jména. Pásek je sekvenční systém a uživatel se musí přesvědčit, že na pásku nejsou již uloženy informace, které ještě bude potřebovat (viz VERIFY). Jméno programu přiřadíte tak, že požadované jméno uzavřete v uvozovkách

(nebo použijete stringovou proměnnou) hned za SAVE. Jméno smí sestávat nanejvýš z 16 znaků.

Poznámka: Ve fázi nahrávaní na disk musíte specifikovat jméno fajlu, jinak se objeví hlášení MISSING FILE NAME ERROR (chyba - chybí jméno fajlu).

Pro specifikování jména zařízení (například 1 pro kazetový přehrávač) napište čárku následovanou číslem zařízení za uvozovkami uzavírajícími jméno fajlu.

Poslední parametr (EOT) následuje za číslem zařízení a rovněž je oddělen čárkou. Jestliže použijete diskdrajv, nemá tento parametr žádný vliv, jestliže však použijete kasetovou jednotku, musí mít jednu z následujících hodnot:

0	žádná činnost (rovněž když EOT není uveden)
1	uloží fajl tak, že parametry nového nahrání programu do paměti (příkazu LOAD) nemají žádný vliv, t.j. program bude vždy nahrán na tu adresu, z níž byl uložen
2	zavede značku EOT - zkrtk. end of tape = konec pásku - na konci fajlu. Jestliže se pokusíme přepsat nějaké informace z oblasti za koncem fajlu objeví se hlášení FILE NOT FOUND ERROR (chyba, fajl nebyl nalezen)

3 uloží fajl ve formátu znemožňujícím jeho nahrání jinam než odkud byl uložen (1) a zavede EOT (2).

Poznámka: Jestliže je spedifikováno číslo zařízení nebo parametr EOT, musíte zavést jméno fajlu (a číslo zařízení). Jestliže používáte kasetovou jednotku můžete použít prázdný znak "". Viz následující příklady.

Příklady:

SAVE "AHOJ"	Uloží na pásek program pod jménem AHOJ
SAVE A$,8	Uloží na disk program pod jménem, uloženým v proměnné A$
SAVE "AHOJ",8	Uloží na disk pod jménem AHOJ (operace je ekvivalentní DSAVE "AHOJ")
SAVE "AHOJ",1,2	Uloží na pásek pod jménem AHOJ a udělá značku konec pásku za programem.
SAVE "",1,3	Uloží na pásek bez jména, zapíše značku konce pásku (EOT) za koncem programu a znemožní přepsání programu jinam, než odkud byl uložen.

SCALE

- Změní měřítko v grafickém modu.

SCALE n[,xmax,ymax]

kde:

n = 1 (aktivuje) nebo 0 (deaktivuje)

xmax nabývá hodnot od 320 do 32767; pokud není uvedeno, bude 1023 (vysoké rozlišení) nebo 2047 (multicolor)

ymax nabývá hodnot od 200 do 32767, pokud není uvedeno bude 1023

Příkaz mění měřítko souřadnic zobrazení v bodové matrici v multibarevném modu a v modu s vysokým rozlišením. Změní se rovněž měřítko souřadné v příkazu MOVSPR. Vícero logických bodů se zmapuje na jediný fyzický bod.

Tato funkce se používá, jestliže potřebujeme načrtnout data s velkým rozsahem hodnot. Multibarevný mod používa 2 pixely (fyzické) na bod ve směru osy x, takže normální zobrazení bude

X = od 0 do 159, Y = od 0 do 199

proti

X = od 0 do 319, Y = od 0 do 199.

Jestliže je potřebí použít tytéž souřadnice pro multicolor i pro vysoké rozlišení, použijte příkaz SCALE 1,640, 200 pro zavedení multibarevné obrazovky a použijte "implicitní" hodnoty parametrů SCALE pro druhý typ obrazovky.

Poznámka: Příkaz GRAPHIC deaktivuje měřítko, je tedy ekvivalentní GRAPHIC .....: SCALE 0

Příklad:

```
10 GRAPHIC 1: GOSUB 100
20 SCALE 1: GOSUB 100
30 SCALE 1,5000,5000: GOSUB 100
40 END
100 CIRCLE 1,160,100,60: RETURN
```

SCNCLR

- Vymaže obrazovku.

SCNCLR [číslo modu]

Mody jsou následující:

Číslo modu	Mod
0	text 40 sloupců (VIC)
1	bodová matrice*
2	bodová matrice s dělenou obrazovkou*
3	multibarevná bodová matrice*
4	multibarevná bodová matrice s dělenou obrazovkou*
5	text 80 sloupců (8563)

Tento příkaz bez argumentu vymaže grafickou obrazovku, v opačném případě vymaže obrazovku běžného textu.**

Příklady:

SCNCLR 5 Vymaže textovou obrazovku 80 sloupců

SCNCLR 1 Vymaže obrazovku s bodovou maticí (VIC)

SCNCLR 4 Vymaže obrazovku s multibarevnou bodovou maticí a dělenou obrazovkou (VIC)

Poznámky: *Oblast bodové matrice je stejná pro vysoké rozlišení a multicolor a různá čísla modu určují další parametry vymazání, například RAM barev (3 a 4), textu se 40 sloupci (2 a 4).

**Jestliže byla vytvořena grafická obrazovka, ale nebyla zvoleny (GRAPHIC=0), bude vymazána. Jestliže se používají dvě obrazovky (80 sloupců pro text a 40 sloupců pro grafiku), SCNCLR vymaže jak grafickou, tak textovou obrazovku, jestliže jsme vyvolali obrazovku s 80 sloupci.

SCRATCH

- Zruší fajl v seznamu disku.

SCRATCH "jméno fajlu"[,Dčíslo drajvu][, ON][,
Učíslo zařízení]

Tento příkaz zruší fajl v seznamu disku. Z obežřetelnosti se počítač zeptá "ARE YOU

SURE" (pouze v přímém modu) dříve než dokončí operaci. Stiskněte Y aby se příkaz SCRATCH provedl nebo libovolné jiné tlačítko, aby se operace zrušila. Používejte tento příkaz ke zrušení nepoužívaných fajlů a k uvolnění prostoru na disku. Jméno fajlu může obsahovat proměnné znaky (?, * atd). Číslo drajvu se předpokládá 0 a číslo zařízení 8, pokud nejsou uvedeny explicite.

Příklad:

SCRATCH "MUJ FAJL",D0

Příkaz zruší fajl MUJ FAJL na disku v drajvu 0, zařízení 8.

SLEEP

- Pozdrží provádění programu na určenou dobu.

SLEEP n

kde n je číslo od $0 \leq n < 65535$.

Příliš dlouhé pozastavení programu se deaktivuje stisknutím tlačítka RUN.

SLOW

- Přikáže Commodoru 128 pracovat normálně na kmitočtu 1 MHz.

SLOW

V Commodoru 128 mikroprocesor 8502 může pracovat rychlostí 1 nebo 2 MHz.
Příkaz SLOW zpomalí mikroprocesor z 2 MHz na 1 MHz. Příkaz FAST uvede Commodore 128 do modu s 2 MHz. Commodore 128 může provádět příkazy mnohem větší rychlostí s 2 MHz, než je to možné s 1 MHz.
V každém případě, obrazovka se 40 sloupci nemůže být používána při 2 MHz.

SOUND

- Produkuje zvukové efekty a hudební tóny.

SOUND v,f,d [,dir][,m][,g][,o][,l]

kde:

v = hlas (1-3)

f = kmitočet (0-65535)

d = délka (0-32767)

dir= směr přírustku (0-vzhůru, 1-dolů nebo 2-kolísání), implicitní hodnota=0

m = minimální kmitočet (používa se s glisandem) (0-63535), implicite=0

g = hodnota přírustku glisanda (0-65535), implicitní hodnota = 0

o = tvar vlny (0=trojúhelníková, 1=pilovitá, 2=impulsní, 3=šum), implicitní hodnota=2

l = délka impulsu (0-4095), implicitní hodnota = 2048

Příkaz SOUND umožňuje generovat rychle a účinně zvukové efekty a hudební tóny. Tři povinné parametry v, f a d určují hlas, kmitočet a dobu trvání zvuku. Délka se uvádí v jednotkách zvaných "jiffy" (anglicky jiffy - džify = mžik). 60 džifů se rovná jedné vteřině.

Příkaz SOUND umožňuje kolísání kmitočtu, které produkuje zvukové efekty v širokém rozmezí tónů. Směr změny kmitočtu se určuje parametrem dir. Minimální kmitočet glisanda se zadává parametrem m, hodnota přírustku pomocí g. Vhodný tvar vlny se volí pomocí o, šířka vlny ve tvaru proměnného impulsu (jestliže byla zvolena pomocí o) se určuje parametrem l.

Příklady:

SOUND 1,40960,60 Generuje zvuk s kmitočtem 40960, hlas 1 trvající 1 vt vteřinu.

SOUND 2,2000,50,0,2000,100 Generuje měnící se zvuk s kmitočtem 2000 a s přírustkem směrem nahoru rovným 100 jednotek.

SOUND 3,5000,1,2,3000,500,1 Nyní se generuje stupnice zvuků počínajíc nejmenším kmitočtem 3000 až do 5000 s přírustkem 500. Směr glisanda je vpřed a zpět (osciluje). Tvar vlny byl zvolen pilovitý, hlas č. 3.

SPRCOLOR

- Stanoví multicolor 1 a/nebo multicolor 2 pro všechny sprajty (poze pro zobrazení se 40 sloupci).

SPRCOLOR [smcr-1][,smcr-2]

kde:

smcr-1 stanoví multicolor 1 pro všechny sprajty

smcr-2 stanoví multicolor 2 pro všechny sprajty

Každý z těchto parametrů může mít libovolnou barvu od 1 do 16.

Příklady:

SPRCOLOR 3,7 stanoví multicolor 1 sprajtů jako červený a multicolor 2 jako modrý

SPRCOLOR 1,2 stanová multicolor 1 sprajtů jako černý a multicolor 2 jako bílý.

SPRDEF

- Aktivuje mod definování sprajtů pro generování a obměňování obrázků sprajtů (pouze pro zobrazení 40 sloupců).

SPRDEF

Příkaz SPRDEF má za následek zobrazení pracovní oblasti sprajtu na obrazovce (24 znaků šířka krát 21 výška). Každa pozice znaková na mříži odpovídá jednomu pixelu sprajtu, přičemž reálný sprajt se zobrazuje vpravo od pracovní oblasti. Uvedeme nyní operace prováděné v modu SPRDEF s příslušnými tlačítky:

Operace uživatele	Popis
tlačítko RETURN	výstup z modu tvoření sprajtů po slovech SPRITE NUMBER?
1 - 8	určují čísla sprajtů
A	zapíná a vypíná automatický posun kursoru
tlačítka CRSR	přemísťují kursor
tlačítko RETURN	přenese kursor na začátek nového řádku

tlačítko HOME	přemístí kursor do levého horního rohu pracovní oblasti
tlačítko CLR	vymaže vnitřek mříže
1 - 4	volba zdrojové barvy: 1 - vymazání 2 - popředí 3 - multicolor 1 4 - multicolor 2
tlačí CTRL,1-8	volba barvy popředí sprajtu (1-8)
tlač. Cx, 1-8	volba barvy popředí sprajtu (9-16)
tlačítko STOP	zruší změny a vrátí nápis
SHIFT RETURN	uloží sprajt v paměti a vrátí hlášení SPRITE NUMBER?
X	roztáhne sprajt ve směru X (vodorovně)
Y	roztáhne sprajt ve směru Y (svisle)
M	multibarevný sprajt
C	kopíruje data jednoho sprajtu na jiný sprajt

Poznámka: Vyvolání SPRDEF způsobí vymazání obrazovky s bodovou matricí.

SPRITE

- Aktivuje a deaktivuje, barví, roztahuje a určuje prioritu sprajtu na obrazovce.

SPRITE <číslo> [,1\0][,popr][,priorita] [,x-rozt][,y-rozt][,mod]

Příkaz SPRITE ovládá většinu charakteristik sprajtu.

Parametr	Popis
číslo	číslo sprajtu (1-8)
1 0	aktivuje (1) nebo deaktivuje (0) sprajt
popr	určuje barvu popředí sprajtu (1-16)
priorita	priorita je 0 jestliže se sprajty mají ukazovat před předměty zobrazenými na obrazovce a 1, jestliže se sprajty mají objevovat za předměty na obrazovce.
x-rozt	vodorovné roztažení je aktivováno (1) nebo deaktivováno (0)
y-rozt	svislé roztažení je aktivováno (1) nebo deaktivováno (0)
mod	volí sprajt standartní (0) nebo multibarevný (1).

Parametry, které nejsou v příkazu SPRITE uvedeny, si podrží hodnoty z předchozího sprajtového příkazu. Charakteristiky sprajtu lze obdržet pomocí funkce RSPRITE.

Příklady:

SPRITE 1,1,3	Aktivuje sprajt č. 1 a vybarví jej červeně.
SPRITE 2,1,7,1,1,1	Aktivuje sprajt č. 2 a vybarví jej modře. Sprajt se bude přemístovat za předměty na obrazovce a bude roztažen jak vodorovně, tak svisle.
SPRITE 6,1,1,0,0,1,1	Aktivuje sprajt č. 6 a vybarví jej černě. První 0 oznamuje počítači, že sprajt bude zobrazen před předměty na obrazovce. Druhá 0 a následující 1 přikazují počítači roztáhnout sprajt pouze ve svislém směru. Poslední 1 specifikuje multibarevný mod. Použijte příkaz SPRCOLOR pro volbu multicoloru sprajtu.
SPRITE 7,,,,1	Roztáhne sprajt č. 7 ve vodorovném směru, všechny ostatní charakteristiky zůstávají nezměněny.

SPRSAV

- Uloží data sprajtu ze řetězcové textové proměnné do oblasti paměti, určené k ukládání sprajtů, a naopak.

SPRSAV zdroj, určení

SPRSAV

SSHAPE/GSHAPE

Tento příkaz přenáší obrázek sprajtu ze stringové proměnné do oblasti paměti, určené k ukládání sprajtů. Mimoto může přenést data z oblasti paměti pro sprajty do řetězcové proměnné. Jak zdroj, tak určení mohou být číslem sprajtu nebo stringovou proměnnou, nemohou však být oba stringovou proměnnou. Jestliže se nějaký řetězec přenáší do sprajtu, použije se pouze prvních 63 bytů. Zbylé byty se ignorují, poněvadž sprajt může obsahovat pouze 63 bytů dat.

Příklady:

SPRSAV 1,A$	Přenese zobrazení sprajtu 1 do stringu nazvaného A$
SPRSAV B$,2	Přenese data ze stringové proměnné B$ do sprajtu 2.
SPRSAV 2,3	Přenese data sprajtu 2 do sprajtu č. 3.

Poznámka: String SPRSAV sprajtu vytváří string téhož tvaru jako příkaz SSHAPE tak, že může být použit příkazem GSHAPE k vyvedení sprajtu na obrazovku s vysokým rozlišením. String má délku 67 znaků.

SSHAPE/GSHAPE

- Uloží/vyvolá obrázek do/z stringové proměnné.

SSHAPE a GSHAPE se používají pro uložení a vyvolání obdélníkové oblasti obrazovky v multicoloru nebo s bodovou matricí do/z stringové proměnné BASICu. Příkaz pro uložení oblasti obrazovky do stringové proměnné je:

SSHAPE stringová proměnná, X1,Y1[,X2,Y2]

kde:

stringová proměnná	- jméno stringu, v němž budou uložena data
X1,Y1	- souřadnice rohu (od 0,0 do 320,200)
X2,Y2	- souřadnice rohu opačného k X1, Y1 (jestliže není uvedeno, bude to polola PK)

Protože BASIC omezuje délku stringu na 255 znaků, rozměry oblasti, která může být uložena, jsou rovněž omezeny. Požadovaná délka stringu může být spočtena z jednoho z následujících vzorců:

L(v-r)=INT((ABS(X1-X2)+1)/8+.99)*(ABS(Y1-Y2)+1)+4

L(mclr)=INT((ABS(X1-X2)+1)/4+.99)*(ABS(Y1-Y2)+1)+4

Poznámka:Jestliže X1-X2=23 a grafickým modem je vysoké rozlišení,(mod 1 nebo 2), string produkovaný příkazem SSHAPE může být použit ke generování sprajtu (viz SPRSAV).

Příkaz pro vyvolání (přepsání) dat ze stringové proměnné a jejich zobrazení na určitém místě obrazovky má tvar:

GSHAPE stringová proměnná [,X,Y][,mod]

kde:

string - obsahuje tvar náčrtku

X,Y - horní levá souřadnice (od 0,0 do 319,199) ukazující, kde nakreslit obrázek (pokud není uvedena, je jí momentální poloha pixelového kursoru)

mod - mod zobrazení:
0_: umístí obrázek tak jak je (implicitní hodnota)
1 : invertuje obrázek
2 : provede s obrázkem logické OR s oblastí
3 : provede s obrázkem AND s oblastí
4 : provede s obrázkem XOR s oblastí

Tento mod dovoluje modifikovat data v stringové proměnné: inerze tvaru, provede logické OR, výlučné OR nebo AND s obrázkem. Viz rovněž příkaz LOCATE s dalšími údaji o pixelovém kursoru.

Příklady:

SSHAPE A$,10,10 Uloží do stringové proměnné A$ obdélníkovou oblast od souřadnic 10,10 do pixelového kursoru.

SSHAPE B$,20,30,47,51 Do stringové proměnné B$ uloží obdélníkovou oblast mezi levým horním bodem (20,30) a pravým dolním bodem (47,51)

GSHAPE A$,120,20 Vyvolá obrázek obsažený ve stringové proměnné A$ a zobrazí jej se souřadnicemi levého horního rohu 120,20.

GSHAPE B$,30,30,1 Vyvolá obrázek uložený ve stringové proměnné B$ a zobrazí jej s levými horními souřadnicemi (30,30). Obrázek bude inverzní, díky určenému modu zobrazení 1.

Poznámka: Věnujte pozornost použití modů 1-4 s multibarevnými obrázky. Můžete dostat překvapující výsledky.

STASH

- Přenese obsah základní paměti do RAM rozšíření.

STASH #byte, iniz, inesp, nbes

Viz příkaz FETCH pro popis parametrů.

STOP

- Přeruší provádění programu.

STOP

Tento příkaz přeruší program. Objeví se nápis BREAK IN LINE XXX (přerušení na řádku XXX, pou ze v programovém modu), kde XXX je číslo řádku, obsahujícího příkaz STOP. Program může být znovu spuštěn od příkazu následujícího za STOP pomocí příkazu CONT, jestliže nebyly provedeny nějaké změny. Příkaz STOP se často používá v řadě bodů programu.

SWAP

- Vzájemně vymění obsahy základní RAM a RAM rozšíření.

SWAP #byte,iniz,inesp,nbes

Popis parametrů viz příkaz FETCH.

SYS

- Vyvolá a provede podprogram ve strojovém kódu na určené adrese.

SYS adresa mod C 64
SYS adresa [,a] [,x] [,y] [,s] mod C 128

Tento příkaz vyvolá podprogram ve strojovém kódu na dané adrese, z paměťové konfigurace stanovené příkazem BANK. Nepovinné argumenty a, x, y, s budou zapsány do registrů akumulátorů X a Y a dojde k tomu dříve, než se zavolá podprogram. Stupnice adres je od 0 do 65535. Příkaz vyvolá provedení programu ve strojovém kódu počínajíc tímto místem paměti. Viz rovněž příkaz BANK.

Příklady:

SYS 40960 Vyvolá a provede program ve strojovém kodu z místa 40960.

SYS 8192,0 Vyvolá a provede program ve strojovém kodu z bunky 8192 a zapíše 0 do akumulátoru.

TEMPO

- Určuje rychlost hudební produkce.

TEMPO n

kde n je relativní délka (od 1 do 255). Účinná délka celé noty se zrčuje z následujícího vzorce:

délka celé noty = 23,06/n vteřin

Implicitní hodnotou n je 8, délka noty se zmenšuje s růstem n.

Příklady:

TEMPO 16 Stanoví tempo 16.
TEMPO 1 Stanoví minimální tempo.
TEMPO 250 Stanoví tempo 250.

TRAP

- Identifikuje a řídí programové chyby během provádění programu BASIC.

TRAP [řádek#]

Když je aktivován, příkaz TRAP identifikuje většinu chybových podmínek (vyjma chybová hlášení DOS, ale včetně tlačítka STOP). V případě jakékoliv chyby při provádění programu je zřízen chybový flag a provádění programu se přenese na řádeğ, jehož číslo je uvedeno v příkazu TRAP. Číslo řádku, v němž došlo k chybě, může být nalezeno pomocí systémové proměnné EL. Podmínka specifikované chyby je obsažena v systémové proměnné ER. Stringová funkce ERR$(ER) dává chybové hlášení odpovídající této chybové podmínce. Příkaz RESUME může být použit, aby se pokračovalo v provádění programu. TRAP bez čísla řádku deaktivuje rozlišování chyb. Chyba v programu TRAP nemůže být identifikována.

Příklady:

100 TRAP 1000 Jestliže se narazí na chybu, přeskoč na řádek 1000.
1000 ?ERR$(ER);EL Vytiskne chybové hlášení a číslo řádku, kde došlo k chybě.
1010 RESUME Obnoví provadění programu.

TROFF

- Deaktivuje trasovací mod.

TROFF

Tetno příkaz deaktivuje trasovací mod.

TRON

- Aktivuje trasovací mod.

TRON

TRON se používá v řadě míst programu. Tento příkaz aktivuje trasovací mod. Jestliže se provádí progra, čísla řádků programu v uvozovkách se objeví na obrazovce dříve, než se provedou operace požadované v tomto řádku. Jestliže řádek obsahuje více příkazů, číslo řádku se vytiskne před provedením každého příkazu.

VERIFY

- Porovnává program v paměti počítače s programem uloženým na disku nebo na pásku.

VERIFY "jméno fajlu"[,číslo zařízení][,flag přeadresování]

Tento přípaz přikazuje Commodoru 128 porovnat program na kazetě nebo na disku s programem v paměti počítače, aby se zjistilo, zda byl program z paměti správně nahrán. Tento příkaz se mimoto často používá k uložení programu bezprostředně za posledním programem již uloženým na kazetě.
VERIFY bez argumentů přikazuje Commodoru 128 porovnat první program uložený na kazetě, nezávisle na jeho jméně, s programem v paměti počítače. VERIFY následované jménem programu v uvozovkách nebo stringovou proměnnou, vyhledá na kazetě určený program a porovná jej s programem v paměti počítače. VERIFY následované jménem, čárkou a číslem porovná program na odpovídajícím zařízení (1 pro kazetu, 8 pro disk). Flag přeadresování je týž jeko v příkazu LOAD (prověří program v tom níste paměti, odkud byl uložen). Viz rovněž DVERIFY.

Příklady:

VERIFY Porovná první program, na nějž na kazetě narazí.

VERIFY "AHOJ" Vyhledá program AHOJ na kazetě a porovná jej s programem v paměti.

VERIFY "AHOJ",8,1 Vyhledá program AHOJ na disku a porovná jej s programem v paměti.

VERIFY "Poslední fajl" Vyhledá na kazetě poslední fajl, zkontroluje a vypíše chybové hlášení, jestliže nalezne neshody. V takovém případě je možné uložit nový program za tímto programem, aniž bychom riskovali vymazání předešlých programů.

Poznámka: Jestliže bude přenesena nějaká grafická oblast, pak použití VERIFY nabo DVERIFY po jejím uložení vyvolá chybové hlášení. S technického hlediska je tato operace korektní. BASICový text je v tomto případě přemístěn z původního místa do jiné množiny adres. Z tohoto důvodu VERIFY, který porovnává BYTE/BYTE dá negativní výsledek, přestože program je v pořádku.

VOL

- Určuje výstupní sílu zvuku.

VOL síla zvuku

Tento příkaz stanoví sílu zvuku pro příkazy SOUND a PLAY. Hlasitost může být od 0 do 15, kde 15 je maximální a 0 je minimální síla. VOL ovlivňuje všechny hlasy.

Příklady:

VOL 0 Zcela ztlumí hlasitost.

VOL 1 Stanoví minimální hlasitost.

VOL 15 Stanoví maximální hlasitost pro příkazy SOUND a PLAY.

WAIT

- Přeruší provádění programu až do prověření určité podmínky.

WAIT místo, maska 1[,maska 2]

Příkaz WAIT vyvolá krátkodobé přerušení provádění programu až do prověření specifikovaného bitu nebo hodnoty na dané paměťové adrese. WAIT se může rovněž použít k přerušení programu, až do vnějšího zásahu. Tato operace může být ovlivněna kontrolou stavu bitu registru vstup/výstup. Datové prvky používané v příkazu WAIT mohou mít libovolnou hodnotu. Většině programátorů připadá tento příkaz neužitečný, protože způsobí zastavení programu až do provedení změny jediného bitu v zadaném místě paměti. Tento typ příkazů

se používá pouze pro speciální operace I/O. Příkaz WAIT přečte hodnotu z určitého místa paměti a provede logické AND s hodnotami v masce 1. Jestliže je uvedena i maska 2, provede se s výsledkem první operace operace OR. Jinými slovy, 1. maska vyloučí bit, který se neshoduje. Kde je v 1. masce bit 0, odpovídající bit výsledku bude vždy 0. Hodnota v 2. masce zamění všechny bity tak, aby se mohla prověřit jak podmínka ON tak podmínka OFF. Každý kontrolovaný bit 0 musí mít 1 v poloze odpovídající druhé masce. Jestliže se bity odpovídající operandům v 1. a 2. masce liší, operace OR dá výsledek 1. Jestliže odpovídající bity dosáhnou téhož výsledku, bit bude 0. Pomocé příkazu WAIT je možné zavést nekonečnou přestávku; v tomto případu tlačítka RUN/STOP a RESTORE mohou být použita pro obnovení provádění programu. WAIT může vyžadovat příkaz BANK, jestliže paměť, do níž je vyžadován přístup není v bloku zvoleném v daném okamžiku.

Následující příklady jsou použitelné pouze v modu C 128. V prvním příkladu se čeká na stisknutí tlačítka, aby program mohl pokračovat. V druhém příkladu se čeká na stisknutí a poté puštění tlačítka SHIFT. V třetím

WIDTH

příkladu se čeká dokud bit 7 (128) je aktivován nebo bit 4 (16) je deaktivován.

Příklady:

WAIT 1,32,32

WAIT 211,1: WAIT 211,1,1

WAIT 36868,144,16

(144 a 16 jsou binární masky. 144=10010000 binárně a 16=10000 binárně).

WIDTH

- Stanoví tloušťku kreslené čáry.

WIDTH n

Tento příkaz stanoví tloušťku kreslené čáry při použití grafických příkazů BASICu - jednoduchá nebo dvojnásobná tloušťka. Hodnotě n=1 odpovídá jednoduchá tloušťka, hodnotě n=2 odpovídá dvojitá tloušťka.

Příklady:

WIDTH 1 Stanoví jednoduchou tloušťku pro grafické příkazy.

WIDTH 2 Stanoví dvojnásobnou tloušťku pro grafické příkazy.

WINDOW

- Vymezuje okno na obrazovce.

WINDOW lhs,lhr,pds,pdr[,vymazání]

Tento příkaz vymezí logické okno na textové obrazovce se 40 nebo 80 sloupci. Souřadnice musí být v rozmezí 0-39/79 pro sloupce (s) a 0-24 pro řádky (r). Flag mazání 1 vyvolá vymazání obrazovky (pouze v hranicích definovaného okna).

Příklady:

WINDOW 5,5,35,20 Vymezí okno se souřadnicemi levéhoho horního rohu (5,5 a souřadnicemi pravého dolního rohu (35,20).

WINDOW 10,2,33,24,1 Vymezí okno se souřadnicemi levého horního rohu (10,2) a souřadnicemi pravého dolního rohu (33,24). Mimoto vymaže část obrazovky zaujímanou oknem, jak t stanoví podlední parametr

Poznámka: Jestliže se v modu se 40 sloupci specifikuje sloupec větší než 39, objeví se hlášení ILLEGAL QUANTITY ERROR (chyba - nepřípustná hodnota).

ČÁST 18

Funkce BASICu

Funkce jsou popsány v následujícím tvaru:

FUNKCE(argument)

kde argumentem může být číselná hodnota, proměnná nebo string.
Každý popis funkce je následován příkladem. Řádky zobrazené polotučně v příkladech jso operace zaváděné uživatelem. Řádek normálnic znaků je reakce počíteče.

ABS

- Vrací absolutní hodnotu argumentu.

ABS(X)

Funkce absolutní hodnota vrací kladnou hodnotu argumentu.

Příklad:

```
PRINT ABS(7*(-5))        35
```

ASC

- Vrací ASCII CBM kód znaků.

ASC(X$)

Tato funkce vrací ASCII kód znaků X$. V modu C 128 není nutné přidávat CHR$(0) do

prázdného stringu. K chybovému hlášení ILLEGAL QUANTITY ERROR nedojde.

Příklad:

```
X$="C128": PRINT ASC(X$)        67
```

ATN

- Vrací arkustangentu argumentu v radiánech.

ATN(X)

Tato funkce vrací úhel vyjádřený v radiánech, jehož tangenta je X.

Příklad:

```
PRINT ATN(3)        1.24904577
```

BUMP

- Vrací informace o srážkách sprajtů.

BUMP(n)

Tato funkce se používá, aby se určilo, které sprajty vstoupily dá srážky od poslední provedené kontroly. BUMP(1) registruje, které sprajty se srazily s jinými sprajty, zatímco BUMP(2) registruje, které sprajty se srazily s předměty na obrazovce. Aby bylo možné použít BUMP nemusí být aktivován příkaz

COLLISION. Bity (od 0 do 7) hodnoty BUMP odpovídají příslušně sprajtům od 1. do 8. BUMP(n) je vždy znovu přiřazena hodnota 0 po každém volání.
Hodnota vrácená příkazem BUMP(n) je výsledk umocnění dvojky na polohy nenulových bitů. Například jestliže BUMP vrátí hodnotu 16 znamená to, že srážky se zúčastnil sprajt č. 4, protože $2^4=16$.

Příklady:

PRINT BUMP (1) 12	Ukazuje, že se srazily spraj ty č. 2 a 3 (12=4+8)
PRINT BUMP (2) 32	Ukazuje, že sprajt č. 5 se srazil s předmětem na obrazovce.

CHR$

- Vrátí ASCII znak odpovídající zadané hodnotě ASCII CBM kódu.

CHR$(X)

Tato funkce je inverzní funkcí k ASC a vrací znak, jehož ASCII CBM kódem je X. Viz Dodatek E - tabulka kódů CHR$.

Příklady:

PRINT CHR$(65) a PRINT CHR$(147)	Vytiskne znak a Vymaže textovou obrazovk

COS

- Vrátí kosinus úhlu zadaného v radiánech.

COS(X)

Tato funkce vrací hodnotu kosinu X, kde X je úhel vyjádřený v radiánech.

Příklad:

```
PRINT COS(π)    -1
```

DEC

- Vrátí desítkové vyjádření hexadecimálního čísla.

DEC(hexadecimální řetězec)

Tato funkce vrací desítkovou hodnotu řetězce - hexadecimálního čísla.

Příklad:

```
PRINT DEC("D020")     53280
F$="F": PRINT DEC(F$)  15
```

ERR$

- Vrátí řetězec - chybové hlášení.

ERR$(n)

Tato funkce vrací řetězec reprezentující chybové hlášení. Viz rovněž systémové pro-

měnné EL a ER a Dodatek A se seznamem chybo
vých hlášení BASICu.

Příklad:

```
PRINT ERR$(10)    NEXT WITHOUT FOR
```

EXP

- Vrátí hodnotu e (2.7182813 přibližně) umocněnou na mocnitel X.

EXP(X)

Tato funkce vrací hodnotu e (2.7182813) umocněnou na X.

Příklad:

```
PRINT EXP(1)    2.7182813
```

FNxx

- Vrátí hodnotu funkce definované uživatelem

FNxx(X)

Tato funkce vrací hodnotu funkce xx definované uživatelem pomocí příkazu DEF FNxx.

Příklad:

```
10 DEF FNAA(X)=(X-32)*5/9
20 INPUT X
30 PRINT FNAA(X)              4.44444445
RUN     ?40     (? je žádost o vstup)
```


FRE

- Vrátí počet volných bytů v paměti počítače.

FRE(X)

kde X je číslo banky. X=0 pro ukládání programů BASIC a X=1 pro kontrolu volného prostoru pro proměnné BASICu.

Příklady:

```
PRINT FRE(0)   48893 Vrací počet bytů, které
                     jsou k dispozici pro
                     program BASIC.
PRINT FRE(1)   64256 Vrací počet bytů, které
                     jsou k dispozici pro ulo-
                     žení proměnných BASIC.
```

HEX$

- Vrací hexadecimální hodnotu (řetězec) desítkového čísla.

HEX$(X)

Tato funkce vrací řetězec čtyř znaků, reprezentující hexadecimální hodnotu čísla X ($0 \leq X < 65535$). Funkce inverzní k DEC.

Příklad:

```
PRINT HEX$(53280)    D020
```

Poznámka: HEX$(0) = 0000

INSTR

- Vrací polohu stringu 1 ve stringu 2.

INSTR(string 1, string 2[,počáteční poloha

Funkce INSTR vyhledá nejprve bod, v němž porovná string 2 s vnitřkem stringu 1 a vrátí tu polohu ve stringu, v níž byla nalezena korespondence. Nepovinný parametr "počáteční poloha" určuje tu pozici ve stringu kde začíná hledání. Hodnota "počáteční polohy" musí být v rozmezí od 1 do 255. Jestliže se nenalezne žádná korespondence, nebo jestliže "počáteční poloha" má hodnotu větší než je délka stringu 1, nebo jestliže st string 1 je prázdný, INSTR vrátí hodnotu 0. Jestliže je prázdný string 2, INSTR vrítí (

Příklad:

```
PRINT INSTR("COMMODORE 128","128")    11
```

INT

- Vrátí celou část čísla v pohyblivé desetinné tečce.

INT(X)

Tato funkce vrací celočíselnou část výrazu.

Jestliže je výraz kladný, odřízne se desetinná část. Jestliže je výraz záporný, vrátí INT sousední menší celé číslo.

Příklady:

```
PRINT INT(3.14)       3
PRINT INT(-3.14)     -4
```

JOY

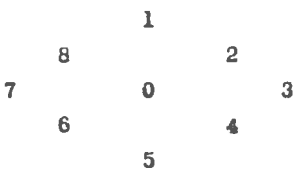
- Vrací polohu joysticku a stav jeho tlačítka.

JOY(n)

kde hodnota n značí:

1 JOY vrací polohu joysticku č. 1
2 JOY vrací polohu joysticku č. 2

Výsledek větší než 128 znamená, že je současně stisknuto tlačítko. Abyste určili polohu joysticku, odečtěte od hodnoty větší než 128 číslo 128. Směry jsou označeny následovně:



Příklady:

JOY(2) je 135 Joystick 2 je vlevo, tlačítko je stisknuto

IF(JOY(1) AND 128)=128 THEN PRINT "VYSTREL"

kontroluje, zda je palební tlačítko stisknuto.

LEFT$

- Vrací určený počet krajních levých znaků stringu.

LEFT$(string, celé číslo)

Tato funkce vrátí string obsahující daný počet krajních levých znaků stringu. Číselný argument musí být celé číslo od 1 do 255. Jestliže je toto celé číslo větší než délka stringu, bude vrácen celý string. Jestliže se použije hodnota 0, vrátí se prázdný string (nulové délky).

Příklad:

PRINT LEFT$("COMMODORE",5) COMMO

LEN

- Vrátí délku stringu.

LEN(string)

Tato funkce vrací počet znaků ve stringu. Zahrne rovněž znaky, které nelze vytisknout, a mezery.

Příklad:

PRINT LEN("COMMODORE 128") 13

LOG

- Vrátí přirozený logaritmus X.

LOG(X)

Tato funkce vrací přirozený logaritmus X. Přirozený logaritmus je logaritmus se základem e (viz EXP(X)). Abyste dostali dekadický logaritmus, dělte LOG(10).

Příklad:

PRINT LOG(37/5) 2.00148

MID$

- Vrátí část stringu nebo naloží string na širší string.

MID$(string,počáteční poloha[,délka])

Tato funkce vrací část stringu určené délky počínajíc znakem v dané počáteční poloze. Počáteční poloha určuje první znak, jímž začíná substring. Délka substringu je určena argumentem "délka". Oba číselné argu-

menty musí mít hodnotu od 0 do 255. Jestliže je "počáteční poloha" větší než délka stringu, nebo jestliže "délka" je nula, MID$ vrátí prázdný string. Jestliže argument "délka" není uveden, vrátí MID$ všechny znaky napravo od "počáteční plohy".

Příklad:

```
PRINT MID$("COMMODORE 128",8,5)     MMODO
```

Příklad "naložení" stringů:

```
A$="123456"
MID$(A$,3,2)="ABCD": PRINT A$    12AB56
```

Poznámka: Funkci "naložení" lze použít pro rozšíření délky stringu, pro nějž předchozí příklad MID$(A$,3,5) není možný.

PEEK

- Vrací obsah určeného místa v paměti.

PEEK(X)

Tato funkce vrátí obsah paměťové buňka s číslem X, kde X leží v rozmezí od 0 do 65535, a vrácená hodnota bude ležet v rozmezí od 0 do 255. Je to opak příkazu POKE. Data se vrací z banky, určené předchozím příkazem BANK. Viz příkaz BANK.

Příklad:

```
10 BANK 15: VIC=DEC("D000")
20 FOR I=1 TO 47
30 PRINT PEEK(VIC+I)
40 NEXT
```

V této ukázce se zobrazí obsah registrů čipu VIC.

PEN

- Vrátí souřadnice Y a X optického pera.

PEN(n)

kde

n=0 PEN vrátí souřadnici X polohy optického pera

n=1 PEN vrátí souřadnici Y polohy optického pera

n=2 PEN vrátí souřadnici X v zobrazení s 80 sloupci

n=3 PEN vrátí souřadnici Y v zobrazení s 80 sloupci

n=4 PEN vrátí hodnotu doutnavky optického pera

Všimněte si, že při pohybu podél sprajtu PEN používá reálné souřadnice, ne grafické souřadnice bodové matrice. Poloha X je dána jako číslo zhruba od 60 do 320, zatímco poloha Y může být číslem uzavřeným mezi 50 a 250. Je to množina souřadnic viditelných na obrazovce, zatímco všechny ostatní hodnoty

nejsou vidět. Nulová hodnota obou souřadnic značí, že pero je mimo obrazovku a že nemá přiložen doutnavku k přerušení posledního znaku. Povšimněte si, že pro použití PEN není nutné, aby byl aktivován příkaz COLLISION. Aby optické pero mělo výraznější efekt, používejte bílé pozadí. Hodnoty PEN jsou různé pro různé systémy.
Na rozdíl od obrazovky se 40 sloupci (VIC), souřadnice pro 80 sloupců (8563) jsou ztotožněny se znaky řádek/sloupec a ne se souřadnicemi pixelů, jako tomu bylo pro obrazovku VIC. Obě hodnoty souřadnic pro obrazovku se 40 a 80 sloupci jsou přibližné a závisí na vlastnostech optického pera. Hodnoty psaní nejsou platné, pokud není PEN(4) nula.

Příklady:

```
10 PRINT PEN(0);PEN(1) Zobrazí souřadnice X
                       a Y optického pera.
10 DO UNTIL PEN(4):LOOP Zajišťuje platnost
                        hodnot čtení.
20 X=PEN(2)
30 Y=PEN(3)
40 REM ZBYTEK PROGRAMU
```

π

- Vrací hodnotu π(3.14159265)

Příklad:

```
PRINT π      3.14159265
```

POINTER

- Vrací adresu proměnné.

POINTER(jméno proměnné)

Příklad:

```
PRINT POINTER(Z) V tomto příkladu bude vrácena adresa proměnné Z.
```

POS

- Vrací běžnou polohu sloupce kursoru v oknu obrazovky, v němž je kursor umístěn.

POS(X)

Funkce POS dává polohu kursoru v definovaném oknu obrazovky. X je fiktivní argument, který musí být uveden, ale jehož hodnota se ignoruje.

Příklad:

```
PRINT "0123456789" POS(1)  012345678910
```

Zobrazí se běžná poloha kursoru v určeném okně obrazovky, v daném případě 10.

POT

- Vrací hodnotu potenciometru "vesla" barvy.

POT(n)

kde:

n=1 POT vrací polohu "pádla" #1

n=2 POT vrací polohu "pádla" #2

n=3 POT vrací polohu "pádla" #3

n=4 POT vrací polohu "pádla" #4

Hodnoty POT leží v rozmezí od 1 do 255. Všechny hodnoty nad 255 indikují, že je rovněž stisknuto tlačítko.

Příklad:

```
10 PRINT POT(1)
20 IF POT(1)>=256 THEN PRINT "PALBA"
```

Tento příklad zobrazí hodnotu "padla" hry 1.

Poznámka: vrátí hodnotu 255 jestliže není zapojené žádné "pádlo".

RCLR

- Vrací barvu barevného zdroje.

RCLR(n)

Tato funkce vrací barvu (od 1 do 16) přiřazenou zdrojové barvě n (0≤n≤6), kde byly použity následující hodnoty n:

0 = pozadí, 40 sloupců

1 = popředí bodové matrice

2 = multicolor 1

3= multicolor 2

4 = rámeček 40 sloupců

5 = barva znaků, 40 a 80 sloupců

6 = barva pozadí, 80 sloupců

Opakem funkce RCLR je příkaz COLOR.

Příklad:

```
10 FOR I=1 TO 6
20 PRINT "ZDROJ";I;"MA BAREVNY KOD";RCLR(I)
30 NEXT
```

V této ukázce se vytisknou kódy barev pro úplnou řadu zdrojů barvy.

RDOT

- Vrací běžnou polohu nebo zdrojovou barvu pixelového kursoru.

RDOT(n)

kde:

n=0 vrací souřadnici X kursoru

n=1 vrací souřadnici Y kursoru

n=2 vrací zdrojovou barvu pixelového kursoru

Tato fenkce vrací běžnou polohu pixelového kursoru (PK) nebo jeho zdrojovou barvu.

Příklady:

PRINT RDOT(0) vrátí polohu X PK
PRINT RDOT(1) vrátí polohu Y PK
PRINT RDOT(2) vrátí zdrojovou barvu PK (od 0 do 3)

RGR

- Vrací číslo běžného grafického modu.

RGR(X)

Tato funkce vrací číslo běžného grafického modu. Argument X je fiktivní, ale musí být uveden. Opakem funkce RGR je příkaz GRAPHIC. Hodnotám vráceným funkcí RGR(X) přísluší následující mody:

Hodnota	Grafický mod
0	Text 40 sloupců (VIC)
1	Standartní bodová matrice
2	Bodová matrice s dělenou obrazovkou
3	Multibarevná bodová matrice
4	Multibarevná bodová matrice s dělenou obrazovkou
5	Text 80 sloupců (8563)

Příklad:

PRINT RGR(0) 1 zobrazí současný grafický mod, v tomto případě mod standartní bodové matrice

RIGHT$

- Vrátí pravou krajní část řetězce.

RIGHT$(<řetězec> ,<číslo>)

Tato funkce vrací zvolenou část stringu z krajních pravých znaků řetězcového argumentu. Délka této části řetězce je určena délkovým argumentem, jímž může být libovolné celé číslo od 0 do 255. Jestliže je hodnota číselného výrazu nula, bude vrácen prázdný řetězec. Jestliže specifikovaná hodnta argumentu bude větší než délka řetězce, bude vrácen celý řetězec. Viz rovněž funkce LEFT$ a MID$.

Příklady:

PRINT RIGHT$("BASEBALL",5) EBALL

RND

- Vrátí náhodné číslo.

RND (X)

Tato funkce vrátí náhodné číslo x, 0=x<1). Může být použita ve hrách, při simulování vrhů kostkou a v jiných náhodných hrách. Mimoto se používá ve statistice.

X=0 RND vrátí náhodné číslo založené na hardvérových hodinách

X 1 RND generuje reprodukovatelné pseudonáhodné číslo, založené na rodové hodnotě

X 0 produkije náhodné číslo používající jako základ rod

Vrh kostkou simulujeme pomocí vzorce INT(RND(1)*6+1). Nejdříve se náhodné číslo násobí 6, čímž se rozšíří rozmezí na 0-6 (přesněji menší než šest). Jestliže přidáme 1, dostaneme rozmezí od 1 do méně než 7. Funkce INT poté "uřízne" všechna desetinná místa a zbyde celočíselný výstředek od 1 do

Příklady:

```
PRINT RND(0)   .507824123  zobrazí náhodné číslo
PRINT INT(RND(1)*100+1)  89  zobrazí kladné
                             celé náhodné číslo
                             menší než 100
```

RSPCOLOR

- Vrátí charakteristiky multibarevného sprajtu.

RSPCOLOR(registr)

kde:

X=1 RSPCOLOR vrátí sprajtový multicolor 1

X=2 RSPCOLOR vrátí sprajtový multicolor 2

Vrácená barva má hodnotu od 1 do 16. Opakem funkce RSPCOLOR je příkaz SPRCOLOR. Viz rovněž příkaz SPRCOLOR.

Příklad:

```
10 SPRITE 1,1,2,0,1,1,1
20 SPRCOLOR 5,7
30 PRINT "MULTICOLOR 1 SPRAJTU JE";RSPCOLOR(1)
40 PRINT "MULTICOLOR 2 SPRAJTU JE";RSPCOLOR(2)
RUN
zobrazí
MULTICOLOR 1 SPRAJTU JE 5
MULTICOLOR 2 SPRAJTU JE 7
```

V této ukázce se na řádku 10 aktivuje sprajt 1, bílé barvy, roztažený ve směrech X a Y a zobrazený v multicolorovém modu. Řádek 20. multicolor 1 a 2 sprajtu. Řádky 30 a 40 tisknou hodnoty RSPCOLOR multicoloru 1 a 2.

RSPPOS

- Vrátí hodnotu rychlosti a polohu sprajtu.

RSPPOS(číslo sprajtu, X)

kde "číslo sprajtu" stanoví, o který sprajt jde a X určuje, zda jde o souřadnice X a Y, nebo o rychlost sprajtu.

Jestliže X je:

0 RSPPOS vrátí současnou polohu X daného sprajtu

1 RSPPOS vrátí současnou polohu Y daného sprajtu

2 RSPPOS vrátí rychlost (od 0 do 15) daného sprajtu

Příklad:

```
10 SPRITE 1,1,2
20 MOVSPR 1,45#13
30 PRINT RSPPOS(1,0);RSPPOS(1,1);RSPPOS(1,2)
```

V této ukázce se vytisknou souřadnice X a Y sprajtu a jeho rychlost (13).

RSPRITE

- Vrátí charakteristiky sprajtu.

RSPRITE(číslo sprajtu, charakteristika)

RSPRITE vrací charakteristiky sprajtu, specifikované v příkazu SPRITE, "Číslo sprajtu" určuje, o který sprajt jde a "charakteristika" určuje následující údaje o sprajtu:

charakteristika	RSPRITE vrátí hodnotu	
0	Aktivován(1)/deaktivován(0)	
1	Barva sprajtu (od 1 do 16)	
2	Sprajt zobrazen před (0) nebo za (1 předměty na obrazovce	
3	Rozšíření ve směru X	ano=1 ne=0
4	Rozšíření ve směru Y	ano=1 ne=0
5	Multicolor	ano=1 ne=0

Příklad:

```
10 FOR I=0 TO 5
20 PRINT RSPRITE(1,I)
30 NEXT
```

V této ukázce se vytiskne všech 6 charakteristik sprajtu č. 1.

RWINDOW

- Vrací rozměry definovaného okna.

RWINDOW(n)

n odpovídá:

0 RWINDOW vrátí počet řádků běžného okna

1 RWINDOW vrátí počet sloupců běžného okna

2 RWINDOW vrátí 40 nebo 80, podle současně používaného formátu výstupu

Opakem funkce RWINDOW je příkaz WINDOW.

Příklad:

```
10 WINDOW 1,1,10,10
20 PRINT RWINDOW(0);RWINDOW(1);RWINDOW(2)
RUN
vytiskne  9940
```

V této ukázce se vytiskl počet řádků (9), sloupců (9) bežného okna a rozumí se, že byl použit formát 40 sloupců.

Argument SPC uvedený jako X určuje počet znaků nahrazených mezerami na obrazovce nebo ve fajlu. Pro fajl na obrazovce nebo na kazetě hodnota argumentu musí ležet v mezích od 0 do 255 a pro fajl na disku je maximem hodnota 254. Pro tištěné fajly, jestliže bude zavedena mezera pomocí SPC na místě posledního znaku řádku, uskuteční se návrat vozíku a přejde se na nový řádek při tisku. Na následujícím řádku nebude mezera vytištěna.

Příklad:

```
PRINT "COMMODORE";SPC(3);"128"  COMMODORE   12
```

SQR

- Vrátí odmocninu argumentu.

SQR(X)

Tato funkce vrací odmocninu X, kde X je nezáporné číslo. Argument nesmí být záporný, nebo se objeví chybové hlášení BASICu: ?ILLEGAL QUANTITY (nepřípustná hodnota).

Příklad:

```
PRINT SQR(25)    5
```

STR$

- Vrátí stringovou reprezentaci čísla.

STR$(X)

Tato funkce vrací stringovou reprezentaci čísla X. Jestliže hodnota STR$ bude převedena, před a za zobrazenými číslicemi bude mezera, s vyjímkou záporných čísel, kterým bude předcházet znak minus. Opakem funkce STR$ je funkce VAL.

Příklad:

```
PRINT STR$(123.45)      123.45
PRINT STR$(-89.03)     -89.03
PRINT STR$(1E20)        1E+20
```

TAB

- Přemístí kursor do polohy tabulační uvedené v daném příkazu.

TAB(X)

Tato funkce přemístí kursor vpřed, pokud to je možné, do polohy na textové obrazovce určené argumentem X, počínajíc krajní levou polohou běžného řádku. Hodnota argumentu musí být v rozmezí 0 až 255. Jestliže běžná poloha tisku je za polohou X, příkaz TAB

se ignoruje. Funkce TAB se používá pouze s příkazem PRINT, protože má různý vliv, v závislosti na použitém zařízení, pokud se použije v PRINT# v logickém fajlu.

Příklad:

```
10 PRINT "COMMODORE"TAB(10)"128"
COMMODORE 128
```

TAN

- Vrátí hodnotu tangenty argumentu.

TAN(X)

Tato funkce vrací tangentu X, kde X je úhel vyjádřený v radiánech.

Příklad:

```
PRINT TAN(.785398163)        1
```

USR

- Volá funkci definovanou uživatelem.

USR(X)

Jestliže se zavolá tato funkce, program přejde na program ve strojovém kódu, jehož počáteční bod je uložen v paměťová buňce 4633 ($1219) a 4634 ($121A) (nebo 785 ($0311) a 786 ($0312) pro mod C64). Parametr X bude přepsán do strojového programu do akumulá-

toru pohyblivé čárky.Vrátí se hodnota progra mu BASIC obráceně volané hodnotě. Aby bylo možné získat hodnotu akumulátoru pohyblivé čárky, musíme odebrat hodnctu proměnné programu. Jestliže proměnna nebyla specifikována, objeví se hlášení ILLEGAL QUANTITY ERROR. Umožní to uživateli změnit proměnnou z věcného do BASICového kódu.

Příklad (pouze pro C 128)

```
10 POKE 4633,0
20 POKE 4634,192
30 A=USR(X)
40 PRINT A
```

Poznámka: Banka není uvedena, takže se rozumí 15. Umístí počáteční adresy ($C000=49152 $00=0:$C0=192) rutiny ve strojovém kódu do buněk 4633 a 4634. Řádek 30 uloží vrácenou hodnotu akumulátoru pohyblivé čárky.

VAL

- Vrátí číselnou hodnotu řetězce z číslic.

VAL(X$)

Tato funkce konvertuje řetězec X$ v číslo (inverzní operace k STR$). Řetězec bude kontrolován od levého krajního až po pravý krajní znak (všechny znaky musí být rozpoznány

jako číselné). Jestliže Commodore 128 narazí na nepřípustné znaky, pak konvertuje pouze část řetězce až do tohoto místa. Jestliže X$ neobsahuje číselné znaky, VAL vrátí 0.

Příklad:

```
10 A$="120"
20 B$="365"
30 PRINT VAL(A$)+VAL(B$)          485
RUN
```

XOR

- Vrátí výlučné OR.

XOR(n1,n2)

Tato funkce dává výlučné OR hodnot n1 a n2.

```
x = XOR(n1,n2)
```

kde n1 a n2 jsou dvě čísla bez znaků (od 0 do 65535).

Příklad:

```
PRINT XOR(128,64)        192
```

Poznámka: není nutné, aby n1 a n2 byla celá čísla.

ČÁST 19

Proměnné

Commodore 128 používá v BASICu tři typy proměnných: reálná čísla, celá čísla a stringy (alfanumerické proměnné).

Normální číselné proměnné, nazývané rovněž proměnné v pohyblivé desetinné tečce, mohou mít libovolnou hodnotu exponentu od -10 do +10 a maximálně 9 desetinných míst. Jestliže je nějaké číslo delší než 9 číslic, počítač je zobrazí ve vědecké notaci jako číslo s jednou číslicí pro jednotky a osmi desetinnými místy, následované písmenem E a mocninou 10, jíž se číslo násobí. Například číslo 12345678901 bude zobrazeno jako 1.23456789E+10.

Celočíselné proměnné jsou čísla od +32767 do -32768, bez desetinné části. Celočíselné proměnné jsou čísla jako 5, 10 nebo -100. Celá čísla zabírají méně místa v paměti než čísla s pohyblivou desetinnou tečkou, obzvlášť jestliže se jedná o matice.

Stringové proměnné jsou proměnné používané v znakových datech a mohou obsahovat číslice, písmena a jakékoliv další znaky, zobrazitelné Commodorem 128. Příkladem stringové proměnné může být "COMMODORE 128".

Jméno proměnné může obsahovat jedno písmeno, písmeno následované číslicí nebo dvě

písmena. Jména proměnných mohou být i delší než dva znaky, ale pro počítač jsou důležité pouze prvné dva. Celé číslo je určeno znakem procent (%) za jménem proměnné. Stringová proměnná je charakterizována znakem dolaru ($), který následuje za jejím jménem.

Příklady:

Jména číselných proměnných: A, A5, BZ
Jména celočíselných proměnných: A%,A5%,BZ%
Jména stringových proměnných: A$,A5$,BZ$

Matice jsou skupiny proměnných téhož jména, které používají dodatečné číslo (nebo čísla) pro specifikování prvků této matice. Matice se definují pomocí příkazu DIM a mohou to být matice v pohyblivé desetinné tečce, celočíselné nebo stringové matice. Jméno matice je následováno dvojicí závorek (), obsahujících počet proměnných v matici.

Příklad:

A(7), BZ%(11), A$(87)

Matice může mít více než jeden rozměr. Může být vytvořena dvouměrná matice se řádky a sloupci, přičemž první číslo označuje řádek a druhé číslo sloupec.(podobně jako se určuje síť na mapě).

Příklad:

A(7,2), BZ%(2,3,4), Z$(3,2)

Rezervovaná jména proměnnných jsou jména rezervovaná pro použití Commodorem 128 a nemohou být použita pro jiné účely. Jsou to proměnné DS, DS$, ER, ERR$, EL, ST, TI a TI$. Klíčová slova jako například TO a IF nebo jiná jména patřící ke klíčovým slovům jako RUN, NEW nebo LOAD rovněž nelze používat.

ST je stavová proměnná pro vstupy a výstupy (s vyjímkou normálních operací na obrazovce a klávesnici). Hodnota ST závisí na výsledku poslední operace I/O. Obecně, jestliže je hodnota ST=0, operace byla korektní. TI a TI$ jsou proměnné, které souvisí s hodinami reálného času, zabudovanými do Commodoru 128. Tyto systémové hodiny se posouvají po každé šedesátině vteřiny. Po zapnutí Commodoru 128 hodiny začnou jít od 0 a mohou být "seřízeny" pouze změnou hodnoty TI$. Proměnná TI udává momentální čas v šedesátinách vteřiny. TI$ je řetězec obsahující reálný čas v běžně používané formě na hodinkách, tj. 24 hodin. Prvé dva znaky v TI$ udávají hodiny, třetí a čtvrtý znak - minuty a pátý a čestý - vteřiny. Této proměnné lze přiřadit libovolnou hodnotu (s pod-

mínkou, že všechny znaky jsou číslice) a bude se měnit dále automaticky jako hodinky s 24 hodinami.

Příklad:

TI$="101530" Nařídí hodiny na $10^h 15^m 30^{vt}$.

Po vypnutí Commodoru 128 se hodiny vynulují a začnou opět na nule při příštím zapnutí Commodoru 128. Hodiny se rovněž vrátí na nulu, jestliže čas překročí 235959 (23 hod. 59 minut a 59 vteřin.

Proměnná DS čte obsah příkazového kanálu diskdrajvu a vrací momentální stav diskdrajvu. Pro vytištění této informace použijte příkaz PRINT DS$. Tato stavová proměnná se používá po diskových operacích jako DLOAD, DSAVE, aby se lokalizovala chyba, indikovaná blikajícím okénkem na diskdrajvu.

ER, EL a ERR$ jsou proměnné používané v pod programech pro identifikaci chyb a používají se obecně pouze v programech. ER vrací poslední chybu na niž se narazilo, jestliže se začal provádět program. EL je číslo řádku, v němž se vyskytla chyba. ERR$ je funkce umožňující tisknout chybová hlášení BASICu. PRINT ERR$(ER) vytiskne příslušné chybové hlášení.

Operátory

Operátory BASICu zahrnují aritmetické, relační a logické operátory. Aritmetické operátory zahrnují následující symboly:

+ sčítání
- odečítání
* násobení
/ dělení
↑ umocňování

Jestliže se v jednom řádku vyskytuje více operátorů, provádějí se různé operace v určitém pořadí. Jestliže se použije více operátorů v téže operaci, počítač určí jejich prioritu následovně: nejprve se provede umocňování, dále násobení a dělení a nakonec sčítání a odčítání. Jestliže mají dvě operace stejnou prioritu, výpočet se provede v pořadí zleva doprava. Jestliže je nutné provést operace v jiném pořadí, BASIC Commodoru 128 nabízí možnost změnit prioritu pomocí závorek. Operace v závorkách se provádějí nejdříve. Přesvědčte se, že jste ve výrazu použili sudý počet závorek, otevírací a zavírací po dvojicích, jinak se objeví chybové hlášení SYNTAX ERROR při provádění programu. Mimoto existují další operátory pro porovnávání, nazývané relační. Aritmetické operátory mají prioritu vyšší než relační.

=	rovná se
<	menší než
>	větší než
<= nebo =<	menší než nebo rovno
>= nebo =>	vetší než nebo rovno
<> nebo ><	nerovná se

Nakonec máme tři logické operátory s nejnižší prioritou (vzhledem k aritmetickým a relačním operátorům):

AND
OR
NOT

Tyto operátory se často používají pro vytváření násobných vzorců v příkazu IF...THEN. Jestliže se použijí s aritmetickými operátory, provedou se až nakonec (až po + a -). Jestliže relace vyjádřená vzorcem platí, výsledku bude přiřazena hodnota -1. Jestliže naopak neplatí, bude mu přiřazena hodnota 0.

Příklady:

```
IF A=B AND C=D THEN 100   Vyžaduje splnění
                          podmínek A=B a C=D

IF A=B OR C=D THEN 100    Vyžaduje splnění
                          alespon jedné z pod
                          mínekA=B,C=D nebo
                          obou.

A=5:B=4:PRINT A=B  Vytiskne 0
A=5:B=4:PRINT A 3  Vytiskne 1
PRINT 123 AND 15: PRINT 5 OR 7 Vytiskne 11a
```


ČÁST 20

Rezervovaná systémová slova (klíčová slova)

Tato část manuálu obsahuje seznam slov a symbolů, používaných v jazyce BASIC.7.0. Tato slova a symboly nelze použít v programu jinak, než jako součást jazyka BASIC. Jedinou výjimkou je, že mohou být použita v xáxmxx uvozovkách v příkazu PRINT.

ABS	DEC	GOTO	PAINT
AND	DEF	GO TO	PEEK
APPEND	DELETE	GRAPHIC	PEN
ASC	DIM	GSHAPE	PLAY
ATN	DIRECTORY	HEADER	POINTER
AUTO	DLOAD	HELP	POKE
BACKUP	DO	HEX$	POS
BANK	DOPEN	IF	POT
BEGIN	DRAW	INPUT	PRINT
BEND	DS	INPUT#	PRINT#
BLOAD	DSAVE	INSTR	PUDEF
BOOT	DS$	INT	QUIT *
BOX	DVERIFY	JOY	RCLR
BSAVE	EL	KEY	RDOT
BUMP	ELSE	LEFT$	READ
CATALOG	END	LEN	RECORD
CHAR	ENVELOPE	LET	REM
CHR$	ER	LIST	RENAME
CIRCLE	ERR$	LOAD	RENUMBER
CLOSE	EXIT	LOCATE	RESTORE
CLR	EXP	LOG	RESUME
CMD	FAST	LOOP	RETURN
COLLECT	FETCH	MID$	RGR
COLLISION	FILTER	MONITOR	RIGHT$
COLOR	FN	MOVSPR	RND
CONCAT	FOR	NEW	PREG
CONT	FRE	NEXT	RSPCOLOR
COPY	GET	NOT	RSPPOS
COS	GETKEY	OFF*	RSPRITE
DATA	GET#	ON	RUN
DCLEAR	GO64	OPEN	RWINDOW
DCLOSE	GOSUB	OR	SAVE
SCNCLR	SQR	THEN	SCALE
SCRATCH	SSHAPE	TI	VOL
SGN	ST	TI$	WAIT
SIN	STASH	TO	WHILE
SLEEP	STEP	TRAP	WIDTH
SLOW	STOP	TRON	WINDOW
SOUND	STR$	TROFF	XOR
SPC	SWAP	UNTIL	
SPRCOLOR	SYS	USING	
SPRDEF	TAB	USR	
SPRITE	TAN	VAL	
SPRSAV	TEMPO	VERIFY	

* OFF a QUIT jsou ve stadiu rozpracování.

Rezervované systémové symboly

Následující znaky jsou rezervovanými systémovými symboly.

Symbol	Použití
+ znak plus	Aritmetické sčítání; spojování řetězců; relativní posun sprajtů; zadání desítkového čísla v monitoru strojového kodu.
- minus	Aritmetické odčítání; záporné číslo; spojující pomlčka; relativní posun sprajtů.
* hvězdička	Násobení
/	Dělení
↑ šipka	Umocňování
Mezera	Odděluje klíčová slova a jména proměnných.
= rovnítko	Přiřazuje hodnotu; relace
< menší	Relační operátor.
> větší	Relační operátor.

,	čárka	Formátuje výstupy v seznamu proměnných; funkční parametry v příkazech
.	tečka	Desetinná tečka v číslech s pohyblivou desetinnou tečkou.
;	středník	Formátuje výstupy v seznamu proměnných
:	dvojtečka	Odděluje příkazy BASICu na stejném programovém řádku.
""	uvozovky	Označují stringové konstanty
?	otazník	Zkratka klíčového slova PRINT
(	závorka	Výrazy a funkce
)	závorka	Výrazy a funkce
%		Označuje jména celočíselných proměnných; označuje binární číslo v monitoru strojového kodu.
#	číslo	Předchází číslologického fajlu v příkazech vstu/výstup
$	dolar	Označuje jména stringových proměnných; hexadecimální čísla v monitoru strojového kodu.
&	komerční a	Označuje oktálová čísla v monitoru strojového kodu
π	řecké	π = 3.14159265

DODATKY

DODATKY

DODATEK A

Chybová hlášení jazyka BASIC

BASICem jsou zobrazována následující chybová hlášení. Chybová hlášení mohou být rovněž zobrazována pomocí funkce ERR$(). Čísla chyb uvedená níže jsou čísla přiřazená chybám pro použití v příkazu ERR$().

Chyba=	Jméno chyby	Popis
1	TOO MANY FILES	(Příliš mnoho fajlů) Počet současně otevřených fajlů je omezen na 10.
2	FILE OPEN	(Fajl je otevřen) Došlo k pokusu otevřít fajl s číslem, které má již otevřený fajl.
3	FILE NOT OPEN	(Fajl není otevřen) Fjal, jehož číslo bylo použito v I/O příkazu, musí být otevřen dříve, než se použije.
4	FILE NOT FOUND	(Fajl nebyl nalezen) Neexistuje žádný fajl tohoto jména (na disku) nebo byla načtena značka konec pásku (kazeta).
5	DEVICE NOT PRESENT	(Zařízení není k dispozici) Volané zařízení I/O není k dispozici nebo bufer nebyl lokalizován (kazeta). Prověřte, zda zařízení je připojeno a zapnuto).

6	NOT INPUT FILE	(Není to výstupní fajl). Došlo k pokusu obdržet data z fajlu, který je určen pouze pro ukládání výstupu.
7	NOT OUTPUT FILE	(Není to vstupní fajl). Došlo k pokusu přenést data z fajlu, který byl určen k záznamu.
8	MISSING FILE NAME	(Chybí jméno fajlu). V příkazu chybí jméno fajlu.
9	ILLEGAL DEVICE NUMBER	(Nepřípustné číslo zařízení). Došlo k pokusu použít nesprávné zařízení (příkaz SAVE použit pro obrazovku atp).
10	NEXT WITHOUT FOR	Smyčka nebyla správně sestavena nebo jméno proměnné v příkazu NEXT neodpovídá proměnné v příkazu FOR.
11	SYNTAX	(Syntaktická). BASIC nezná příkaz. Může se stát, že chybí závorky, klíčové slovo je špatně napsáno atp.
12	RETURN WITHOUT GOSUB	(RETURN bez GOSUB). Narazilo se na příkaz RETURN aniž by předtím byl aktivován příkaz GOSUB.
13	OUT OF DATA	(Data vyčerpana). Narazilo se na příkaz READ, ale všechna data se již načetla.

14	ILLEGAL QUANTITY	(Nepřípustná hodnota) Číslo použité jako argument funkce nebo příkazu je mimo povolené meze.
15	OVERFLOW	Výsledek výpočtu je větší než největší dovolené číslo (1.701411 833E+38).
16	OUT OF MEMORY	(Paměť vyčerpána). Není již místo pro program a/nebo proměnné nebobbylo aktivováno příliš mnoho opakujících se příkazů (DO, FOR nebo GOSUB).
17	UNDEF'D STATEMENT	(Nedefinovaný příkaz). Požadované číslo řádku není v programu.
18	BAD SUBSCRIPT	(Špatný index). Program hledá prvek matice mimo meze specifikované v příkazu DIM.
19	REDIM'D ARRAY	(Předimenzovaná matice) Matice může mít určeny rozměry pouze jednou.
20	DIVISION BY ZERO	(Dělení nulou). Dělit nulou je zakázáno.
21	ILLEGAL DIRECT	(Nelegální přímý mod). Příkazy INPUT nebo GET či INPUT# nebo GET# je dovolené použít pouze v programu.
22	TYPE MISMATCH	(Chyba tisku) Číselná hodnota byla použita na místě stringové nebo naopak.

23	STRING TOO LONG	(Řetězec je příliš dlouhý) Řetězec smí obsahovat nanejvýš 255 znaků.
24	FILE DATA	(Fajlová data) Data nebyla správně načtena z kazety nebo z disku.
25	FORMULA TOO COMPLEX	(Vzorec je příliš složitý). Počítač není schopen porozumět tomuto výrazu. Zjednodušte výraz (rozdělte jej na dvě části nebo použijte méně závorek).
26	CAN´T CONTINUE	(Nemohu pokračovat). Příkaz CONT nefunguje jestliže program nemůže být proveden, jestliže byla nalezena chyba nebo jestliže byl změněn řádek.
27	UNDEF´D FUNCTION	(Nedefinovaná funkce). Volá se funkce uživatele, která nebyla definována.
28	VERIFY	(Prověř). Program na kazetě nebo disku neodpovídá programu v paměti počítače.
29	LOAD	(Načti). Narazilo se na problémy během načítání. Zkus to znovu.
30	BREAK	(Přerušení). Po stisknutí tlačítka STOP pro přerušení běhu programu
31	CAN´T RESUME	(Nelze znovu začít). Narazilo se na příkaz RESUME bez aktivování příkazu TRAP.

32	LOOP NOT FOUND	(Nenalezen LOOP) Program narazil na příkaz DO, ale nenalezl odpovídající LOOP.
33	LOOP WITHOUT DO	(LOOP bez DO). Program narazil na LOOP, aniž by předtím byl aktivován příkaz DO.
34	DIRECT MODE ONLY	(Pouze přímý mod). Daný příkaz je povolen pouze v přímém modu, ne v programu.
35	NO GRAPHIC AREA	(Není místo pro grafiku). Narazilo se na příkaz pro grafiku (DRAW, BOX atp) dřívo, než byl proveden příkaz GRAPHIC.
36	BAD DISK	(Vadný disk). Pokus o přeformátování neformátované diskety nebo vadné diskety zrychleným formátováním (bez ID).
37	BEND NOT FOUND	(BEND nenalezen). Program narazil na IF...THEN BEGIN nebo na IF...THEN..... ELSE BEGIN a nenalezl klíčové slovo BEND doplnující toto BEGIN.
38	LINE = TOO LARGE	(Číslo řádku příliš velké). Chyba při přečíslování programu BASIC. Použité parametry vedou k číslu řádku 63999 a přečíslování tedy nelze provést.
39	UNRESOLVED REFERENCE	(Chybné odvolání). Chyba při přečíslování programu BASIC. Číslo řádku uvedené v příkazu (např. GOTO 999) se v programu nevyskytuje. Přečíslování se neprovede.

UNIMPLEMENTED COMMAND (Neproveditelný příkaz)
Narazilo se na příkaz, který BASIC 7.0 nezná.

FILE READ (Chyba při čtení fajlu)
Narazilo se na chybu během nahrávání do paměti nebo čtení programu nebo datového fajlu z diskdrajvu (například otevřelo se okénko drajvu během načítání programu).

D O D A T E K B

Chybová hlášení DOS

Následující chybová hlášení DOS jsou vydávána pomocí proměnných DS a DS$. Proměnná DS obsahuje pouze číslo chyby, zatímco proměnná DS$ obsahuje číslo chyby, chybové hlášení a číslo stopy a odpovídajícího sektoru.

Poznámka: Čísla chybových hlášení menší než 20 musí být ignorována s vyjímkou 01, která dodává informace k číslu fajlu zrušeného příkazem SCRATCH.

Číslo	Chybové hlášení - popis
20	READ ERROR (chyba čtení) (nenalezena hlavička bloku) Kontrolor blo disku nedokázal lokalizovat hlavičku bloku požadovaných dat. Může to být způsobeno nelegálním číslem sektoru, nebo tím, že hlavička byla zničena.
21	READ ERROR (chybí synchronizační znak) Diskový kontrolor nenalezl synchronizační indikátor na požadované stopě. Může to být způsobeno vychýlením čtecí/záznamové hlavy, nepřítomností diskety, nebo jejím špatným založením. Může to rovněž svědčit o hardvérové poruše.
22	READ ERROR (chybí datový blok) Po kontroloru disku se požaduje čtení nebo kontrola bloku dat, který není

Číslo Chybové hlášení - popis

správně zaznamenán. Toto chybové hlášení může následovat za příkazem BLOCK a může indikovat stopu a/nebo volání sektoru, které není legální.

23 READ ERROR

(chybný kontrolní součet v datovém bloku)
Toto chybové hlášení indikuje, že se narazilo na chybu v jednom z několika datových bytů. Data jsou zaznamenána do paměti DOS, ale kontrolní součet těchto dat není správný. Toto hlášení může rovněž svědčit o problémech s uzemněním hardvéru.

24 READ ERROR

(chybné dekódování bytu)
Data nebo hlavička jsou zaznamenána do paměti DOS, ale byla zjištěna hardvérová chyba, způsobená neplatnou konfigurací bitů v datovém bytu. Toto hlášení může rovněž svědčit o problémech s hardvérovým uzemněním.

25 WRITE ERROR (chyba při záznamu)

(chyba kontroly záznamu)
Toto hlášení se objeví, jestliže kontrolor zjistí kontraversi mezi zaznamenanými daty a daty v paměti DOS.

26 WRITE PROTECT ON (ochrana proti záznamu)

Toto hlášení se objeví, jestliže od kontroloru požadujeme zaznamenat datový blok, přestože přepínač ochrany proti záznamu dat je zapnut, nebo jestliže se použije disketa chráněná proti záznamu příslušnou nálepkou.

Číslo Chybové hlášení - popis

27 READ ERROR

(chybný kontrolní součet v hlavičce)
Toto hlášení se objeví, jestliže se narazí na chybný kontrolní součet v hlavičce požadovaného datového bloku. Blok nebude načten do paměti DOS.

28 WRITE ERROR

(datový blok je příliš dlouhý)
Toto hlášení se objeví, jestliže datový blok je příliš dlouhý a překryl by indikátor synchronizace následující hlavičky.

29 DISK ID MISMATCH (chybná identifikace disku)

Toto hlášení se objeví, jestliže požadujeme, aby kontrolor vstoupil do neinicializované diskety. Hlášení se může rovněž objevit, jestliže disketa má nesprávnou hlavičku.

30 SYNTAX ERROR (syntaktická chyba)

(obecná syntaxe)
DOS nedokázal interpretovat příkaz zavedený do příkazového kanálu. Obecně je to způsobeno nelegálním počtem jmen fajlů, nebo chybným použitím konfigurací, například když se dvě jména fajlů sdruží vlevo od příkazu COPY.

31 SYNTAX ERROR

(neplatný příkaz)
DOS neporozuměl příkazu. Příkaz musí být zaveden znovu v příkazovém řádku.

32 SYNTAX ERROR

(řádek příliš dlouhý)
Příkaz je delší než 58 znaků. Použijte zkratky diskových příkazů.

slo Chybové hlášení - popis

)3 SYNTAX ERROR

(neplatné jméno fajlu)
Chybna korespondence konfigurací v příkazech OPEN a SAVE. Přepište jméno fajlu správně.

)4 SYNTAX ERROR

(chybí datový fajl)
Jméno fajlu není uvedeno v příkazu nebo je DOS neidentifikoval jako takové. Nejčastěji se opomene uvést v příkazu dvojtečku (:).

)9 SYNTAX ERROR

(neplatný příkaz)
Toto hlášení se objeví, jestliže příkazový kanál (druhotná adresa 15) nebyl DOSem identifikován.

50 RECORD NOT PRESENT (záznam chybí)

Výsledek záznamu z disku na druhé straně od posledního záznamu pomocí příkazu INPUT# nebo GET#. Toto hlášení se objeví po umístění záznamu za koncem fajlu v relativním fajlu. Jestliže však chceme rozšířit fajl přidáním nového záznamu (příkazem PRINT#), ignurujte toto chybové hlášení. Nesmí se používat příkazy INPUT# a GET# po objevení této chyby, dokud se nepřemístí pointer

51 OVERFLOW IN RECORD (přetečení záznamu)

Příkaz PRINT# překročil hranice záznamu. Informace se odřízne. Za předpokladu, že návrat vozíku, který je vysílán jako indikátor konce záznamu, se započítává do rozměru záznamu, se toto hlášení objeví, jestliže souhrn znaků záznamu (včetně posledního návratu vozíku) přesáhne určené rozměry.

Číslo Chybové hlášení - popis

52 FILE TOO LARGE (příliš velký fajl)

Poloha záznamu uvnitř relativního fajlu svědčí o tom, že došlo k přetečení disku.

60 WRITE FILE OPEN (fajl otevřen pro záznam)

Toto hlášení se objeví, jestliže se otevře pro čtrní fajl pro zaznamenávání, který ještě nebyl uzavřen.

61 FILE NOT OPEN (fajl nebyl otevřen)

Toto hlášení se objeví, jestliže se pokoušíme o přístup do fajlu, který dosud nebyl v DOS otevřen. Někdy se v tomto případě neobjeví hlášení, ale požadavek bude prostě ignorován.

62 FILE NOT FOUND (fajl nenalezen)

Požadovaný fajl se na určeném disku nenalézá.

63 FILE EXISTS (fajl existuje)

Fajl tohoto jména již na disku existuje

64 FILE TYPE MISMATCH (chybná specifikace typu fajlu)

Přístup do požadovaného fajlu není možný, jestliže se použije fajl specifikovaného typu. Viz kapitolu pojednávající o typech fajlů.

65 NO BLOCK (žádný blok)

Toto hlášení se objeví spolu s lokalizací bloku. Sektor, který se pokoušíme lokalizovat, je již lokalizován.

Číslo Chybové hlášení - popis

Číslo stopy a sektoru zobrazené počítačem jsou nejvyšší dosažitelné. Jestliže počítač uvede číslo stopy 0, všechny zbývající sektory jsou již obsazené. Jestliže disk ještě není zcela obsazen, zkuste nižší stopu a sektor.

66 ILLEGAL TRACK AND SECTOR (nelegální stopa a sektor)

DOS se snažil vstoupit do stopy nebo bloku, které v použitém formátu neexistují. Toto hlášení může svědčit o problémech se čtením pointeru v následujícím bloku.

67 ILLEGAL SYSTEM T OR S (nelegální systémový T nebo S)

Toto specifické chybové hlášení ukazuje na nelegální systémovou stopu nebo sektor.

70 NO CHANNEL (žádný kanal k dispozici)

Požadovaný kanal není k dispozici nebo všechny kanály byly použity. Uživatel má k dispozici maximálně pět bufferů. Sekvenční fajl vyžaduje dva buffery; relativní fajl - tři a chybový/příkazový kanál - jeden. Je možné použít je v libovolné kombinaci, pokud nepřekročíme pět bufferů.

71 DIRECTORY ERROR (chyba v direktorech)

BAM (Block Availability Map - mapa disponsibilních bloků) diskety neodpovídá přesně diskové paměti. Abyste napravili chybu, inicializujte disk.

Číslo Chybové hlášení - popis

72 DISK FULL (disk zaplněn)

Všechny bloky na disketě byly použity, nebo je vyčerpán disponsibilní prostor v seznamu. Toto hlášení se objeví, jestliže na disketě zbývají k dispozici ještě dva bloky, umožnující uzavřít běžný fajl.

73 DOS MISMATCH (VERSION NUMBER)

(rozpor na DOS - číslo verse)
DOS 1 a 2 jsou kompatibilní z hlediska čtení, ale ne z hlediska záznamu. Disky mohou být čteny stejně dobře s oběma DOS, ale disketa formátovaná v jedné určité versi nemůže být použita pro záznam v jiné versi, protože se liší formáty. Chybové hlášení se objeví, jestliže se pokusíte zapisovat na disk, který byl formátován v nekompatibilním formátu. Toto hlášení se objeví rovněž po zapnutí, ale v tomto případě neukazuje na chybu.

74 DRIVE NOT REDY (drajv není připraven)

Stává se to při pokusu o přístup do diskdrajvu, do něhož jsme nevložili disketu, nebo když páčka nebo okénko uzávěru drajvu je otevřené.

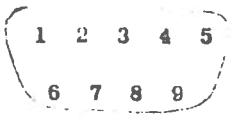
D O D A T E K C

Konektory/vstupy pro periferiální zařízení

Výstupy na bočním panelu

1. Proudová zdířka - s touto zdířkou bude spojen kabel se zástrčkou s 5 kolíky.
2. Vypínač napájení - vypíná počítač.
3. Tlačítko RESET - uvádí počítač do základního výchozího stavu.
4. Řídící vstupy - dva řídící vstupy číslované 1 a 2. Do každého z nich může být zapojen joystick nebo "paddle" pro řízení her. Optické pero může být připojeno pouze na vstup č. 1 - vstup, který se nachází blíže přední části počítače. Používejte tyto vstupy tak, jek je to popsáno v dokumentaci příslušného softvéru.

Řídící vstup č. 1			Řídící vstup č. 2		
kolík	Typ	Pozn.	Kolík	Typ	Pozn.
1	JOY A0		1	JOY B0	
2	JOY A1		2	JOY B1	
3	JOY A2		3	JOY B2	
4	JOY A3		4	JOY B3	
5	POT AY		5	POT BY	
6	Tlač. A/LP		6	Tlač. B	
7	+5 V	max. 50 mA	7	+5 V	max. 50 mA
8	kostra		8	kostra	
9	POT AX		9	POT BX	

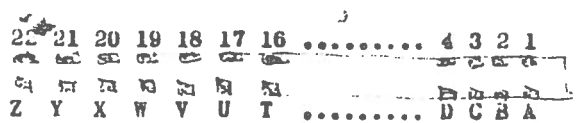


Zadní vstupy

5. Expanzní vstup - tento obdélníkový vstup je paralelním vstupem, do něhož se zapojují programové či herní "patrony" nebo specialní interfejsy.

Expanzní vstupy pro patrony

kolík	typ	kolík	typ	kolík	typ	kolík	typ
1	kostra	12	BA	A	kostra	N	A9
2	+5 V	13	DMA	B	ROMH	P	A8
3	+5 V	14	D7	C	RESET	R	A7
4	IRQ	15	D6	D	NMI	S	A6
5	R/W	16	D5	E	S 02	T	A5
6	CLOCK	17	D4	F	A15	U	A4
7	I/O 1	18	D3	H	A14	V	A3
8	HRA	19	D2	J	A13	W	A2
9	EXROM	20	D1	K	A12	X	A1
10	I/O 2	21	D0	L	A11	Y	A0
11	ROML	22	kostra	M	A10	Z	kostra



6. Kazetový vstup - do tohoto vstupu se připojí přehrávač Dataset 1530 pro záznam programů a informací.

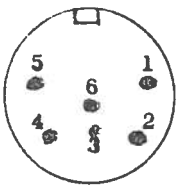
Kazetový vstup

kolík	typ
A-1	kostra
B-2	+5 V
C-3	motor přehrávače
D-4	čtení kazety
E-5	záznam na kazetu
F-6	směr kazety

7. Sériový vstup - na tento vstup lze připojit sériovou tiskárnu nebo diskdrajv Commodoru 128.

I/O sériový

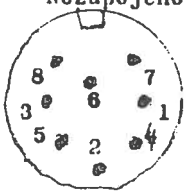
kolík	typ
1	SRQIN
2	kostra
3	ATN sériový IN/OUT
4	CLK sériový IN/OUT
5	DATA sériový IN/OUT
6	RESET



Poznámka: sériový vstup Commodoru 128 není kompatibilní s RS-232. Úrovně RS-232 TTL lze obdržet z uživatelského vstupu.

8. Videokonektor 40 sloupců - konektor DIN dává přímé audio a kompozitní videosignály. Signály mohou být přenášeny na vhodný monitor nebo na audioaparaturu. Signály lze přenášet na monitor Commodoru nebo nebo je použit s oddělenými komponentami.

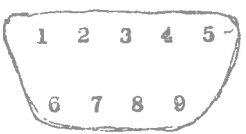
kolík	typ	poznámka
1	LUM/SYNC	Výstup světelnosti/sync
2	kostra	
3	audiovýstup	
4	videovýstup	Výstup kompozitního signálu
5	vstup audio	
6	výstup barev	Výstup osminového signálu
7	NC	Nezapojeno
8	NC	Nezapojeno



9. Konektor RF - tento konektor umožňuje uživateli obdržet zobrazení a zvuk na televizním přijímači (televizor může zobrazit na obrazovce pouze 40 sloupců).

10. Konektor RGBI 80 sloupců - tento 9-kolíkový konektor dává přímý zvukový signál a signál RGBI (červená/zelená/modrá/intensita).
11. Uživatelský vstup - tento vstup lze připojit pomocí interfejsu k různým zařízením.

kolík	signál
1	kostra
2	kostra
3	červená
4	zelená
5	modrá
6	intensita
7	monochromatický
8	synchron. vodorovná
9	synchron. svislá



I/O uživatele

kolík	typ	pozn.	kolík	typ	pozn.
1	kostra		A	kostra	
2	+5 V	max. 100 mA	B	FLAG2	
3	RESET		C	PB0	
4	CNT1		D	PB1	
5	SP1		E	PB2	
6	CNT2		F	PB3	
7	SP2		G	PB4	
8	PC2		H	PB5	
9	ATN SR. IN		J	PB6	
10	9 VAC	max. 100 mA	K	PB7	
11	9 VAC	max 100 mA	L	PA2	
12	kostra		M	kostra	

D O D A T E K D

Zobrazení kódů na obrazovce se 40 sloupci

V tabulce uvedené níže jsou uvedeny všechny znaky, které tvoří soubor znaků pro obrazovku Commodoru. Tabulka uvádí číselné hodnoty (POKE), které je třeba uložit do obrazovkové paměti (buňky 1024 až 2023), abychom na obrazovce o 40 sloupcích obdrželi daný znak. Pro zadání barvy použijte paměťové buňky od čísla 55296 do čísla 56295. Tabulka rovněž samozřejmě ukazuje korespondenci mezi znaky a čísly, kterámůžeme obdržet z obrazovkové paměti pomocí PEEK. K dispozici jsou dva soubory znaků. V modu s 80 sloupci jsou k dispozici současně oba soubory, zatímco pro 40 sloupců může být použit vždy pouze jeden. První nebo druhý soubor se volí stisknutím tlačítka SHIFT současně s tlačítkem Cx (Commodore).

V BASICu příkaz PRINT CHR$(142) aktivuje mod velká písmena/grafika, zatímco PRINT CHR$(14) aktivuje mod velká/malá písmena.

Všechna čísla uvedená v tabulce mohou být zobrazena v inverzním videu. Kódy znaků pro inverzní zobrazení obdržíme přičtením 128 k uvedeným hodnotám.

řada 1	řada 2	POKE
@		0
A	a	1
B	b	2
C	c	3
D	d	4
E	e	5
F	f	6
G	g	7
H	h	8
I	i	9
J	j	10
	k	11
L	l	12
M	m	13
N	n	14
O	o	15
P	p	16
Q	q	17
R	r	18
S	s	19
T	t	20
U	u	21
V	v	22
W	w	23

řada 1	řada 2	POKE
X	x	24
Y	y	25
Z	z	26
[		27
£		28
]		29
↑		30
←		31
SPACE		32
!		33
"		34
#		35
$		36
%		37
&		38
'		39
(		40
)		41
*		42
+		43
,		44
-		45
.		46
/		47

řada 1	řada 2	POKE
0		48
1		49
2		50
3		51
4		52
5		53
6		54
7		55
8		56
9		57
:		58
;		59
<		60
=		61
>		62
?		63
		64
	A	65
	B	66
	C	67
	D	68
	E	69
	F	70
	G	71
	H	72

řada 1	řada 2	POKE
	I	73
	J	74
	K	75
	L	76
	M	77
	N	78
	O	79
	P	80
	Q	81
	R	82
	S	83
	T	84
	U	85
	V	86
	W	87
	X	88
	Y	89
	Z	90
		91
		92
		93
		94
		95
SPACE		96
		97

řada 1	řada 2	POKE
		98
		99
		100
		101
		102
		103
		104
		105
		106
		107
		108
		109
		110
		111
		112

řada 1	řada 2	POKE
		113
		114
		115
		116
		117
		118
		119
		120
		121
		122
		123
		124
		125
		126
		127

Kódy od 128 do 255 jsou stejné jako kódy od 0 do 127 v inverzním videu.