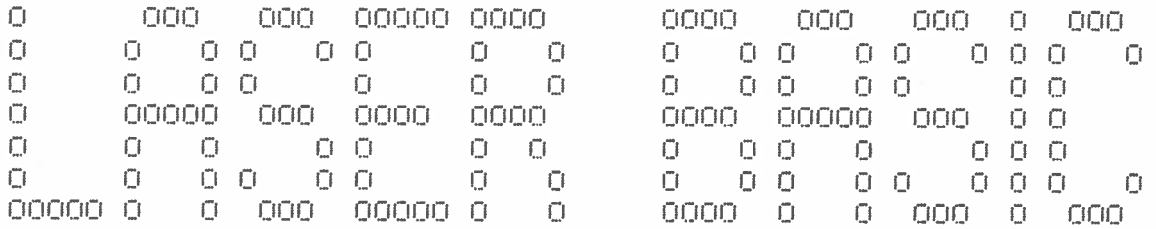

(Kevin Hambleton)

Na uvod

Laser BASIC je rozsiřenim interpretru BASIC v pameti ROM ZX Spectra. Ackoli je Sinclairuv BASIC vykonnou a pruznou implementaci tohoto casem provereneho jazyka, bylo nutne pri jeho navrhu vychazet z co nejobecnejsich a nejsirsich pozadavku. BASIC ma nejruznejsi pole aplikaci, ale zvlastnim polem naseho zajmu je grafika a animace. Laser BASIC byl navrzen proto, aby zvysil snadnost a zejmena rychlost, se kterou mohou byt vytvareny komplexni animovane grafiky. Tomu slouzi pres 100 prikazu a funkci, pridanych ke standartnimu BASICu. Technikou castecneho prekladu do strojoveho kodu se dosahlo zvyseni rychlosti techto novych prikazu.

Uzivatele predchozich programu serie Lightning rozpoznaji vetsinu prikazu, i kdyz pro zvyseni prehlednosti bylo mnoho nazvu prikazu zmeneno. Na tomto miste pripomenme, ze Laser BASIC neni schopen vytvorit samostatne fungujici program (v pameti pocitace musi byt zaroven Laser BASIC), ale byl vyvinut prekladac, ktery dale zrychli vase programy a neni pak nutna pritomnost Laser BASICu v pameti. To znamena, ze muzete vase programy komercne vyuzit.

Laser BASIC muze byt vyuzit i profesionalnim programatorem, ktery se seznamil s programi Lightning, jako rychly a jednoduchy nastroj k vytvareni programu. Sady povelu jsou velmi podobne, takze Laser BASIC muzete pouzit pro rychle seznameni se s animovanymi sekvencemi predtim, nez zacnete pracovat s programem White Lightning nebo Machine Lightning. Interaktivni povaha BASICu je pro tento druh nacviku idealni.

Pouzivani Laser BASICu s microdrivem

Laser BASIC muzete automaticky nahrat na microdrive stiskem vhodne volby v uvodnim menu. Nahrajte Laser BASIC z kazety do pocitace, pretocte kazetu zpet a zvolte volbu 5. Jednotlive soubory se budou postupne z kazety a prehravat na microdrive. Musite spouctet a zastavovat magnetofon podle instrukci na obrazovce.

Pote muzete nahrat Laser BASIC z microdrivu prikazem:

```
LOAD * "M";1;"LASER"
```

Byl-li Laser BASIC takto jednou nahran na microdrive, muzete s nim pracovat stejnym zpusobem, jako by byl nahran pasku.

S pouzivanim microdrivu je ovsem spjat jeden nevyhnutelny problem. Jestlize se projevi chyba spojena s provozem microdrivu (jako napr. "File not found"), kontrola se vymkne z interpretru Laser a prejde do Sinclair BASICu. Poznate to tak, ze jakykoliv pokus o vlozeni prikazu Laser BASICu se ohlasi blikajicim otaznikem. Podobne pokus o rozbehnuti jiz napsaneho programu, obsahujiciho rozsirene prikazy, skonci chybovym hlasenim "Nonsence in BASIC". K navratu do rozsireneho BASICu musite stisknout "#" a ENTER. Jestlize jiz samotne "#" vyvolava chybove hlaseni (?), potom vymazte "#" a napiste:

```
RANDOMIZE USR 58830 a ENTER
```

Omlouvame se za tento nevyhnutelny nedostatek.

Ve vyjimecne nepravdepodobnem pripade, ze by vypadl dokonce i interpretr nahrany z kazety, pouzijte:

```
RANDOMIZE USR 58820.
```

Slovnik terminu uzivanych v teto prirucce

Sprity

Sprite je graficky znak, ovladany programem. Laser BASIC vam umoznuje definovat az 255 techto spritu, kazdy se svymi vlastnimi rozmery. Omezeni mnozstvi spritu a jejich velikost je dano objemem pameti.
Zaroven s Laser BASICem je dodavan program "Sprite Generator" (generator spritu), ktery je uzivan k vytvareni spritu. Jestlize byly sprity jiz jednou vytvoreny, muzete je nahrat na kazetu nebo na microdrive volbou z menu.
Sprity z "OPTION 1" jsou pouzivany vylucne s programem Sprite Generator, zatimco sprity z "OPTION 2" pouzivame pouze s Laser BASICem.
Na kazete jsou nahrany dve sady spritu z OPTION 2, ktere muzete primo nahrat do Laser BASICu. Jsou oznaceny "SPRITE2A" a "SPRITE2B" (viz prilohy 2 a 3).

Okenka obrazovky

Okenko obrazovky je cast obrazovky, definovana ctyrmi promennymi COL, ROW, HGT a LEN. COL ma rozmer 0 az 31, ROW je mezi 0 az 23, HGT mezi 1 az 24 a LEN mezi 1 az 32. Jednotkou pro tyto rozmery je jeden znak. COL a ROW urcuji sloupec a radek, ve kterych je umisten horni levy roh okenka. HGT a LEN urcuji velikost okenka. Chcete-li videt priklad okenka, napiste nasledujici radek a stisknete ENTER:

```
.ROW=5:.COL=6:.HGT=4:.LEN=3:.INVV
```

Okenka spritq

Okenko spritu je cast spritu, urcena promennymi SPN, SCL, SRW, HGT a LEN. SPN urcuje sprite, SCL a SRW definuji sloupec a radek uvnitr spritu a HGT a LEN urcuji velikost okenka. Jestlize okenko definovane pomoci techto promennych lezi mimo sprite, nebo presahuje jeho hranice, potom se prikaz neprovede, ale neobjevi se zadne chybove hlaseni.

Prostor spritu

Prostor spritu je oblast pameti, ktera obsahuje vsechny dosud definovane sprity. Konec teto oblasti je 56575 (DCFFh) a dolni konec se od teto adresy postupne snizuje. Jak daleko se snizil v prubehu definovani zacatek teto oblasti muzeme zjistit jednou z nasledujicich procedur:

```
PRINT PEEK(62464)+256*PEEK(62465)
```

nebo

```
LET X=?PEEK (62464): PRINT X
```

Ramtop

Je dulezite, aby vami definovane sprity nepresahly nikdy svou dolni hranici pres RAMTOP. Ke zjisteni okamzite hodnoty RAMTOP

pouzijte:

```
PRINT PEEK(23730)+256*PEEK(23731)
```

nebo

```
LET X=?PEK(23730): PRINT X
```

Kazdy definovany sprite zabira 9*HGT*LEN+5 bytu.
Ve vetsine pripadu se nemusime obavat, ze by oblast spritu zasahla pod RAMTOP. Vyjimkou jsou pripady, kdy vytvarite sprity v prubehu programu prikazy .ISPR nebo .SPRT. I tehdy je vsak mozne za pomoci vyse uvedenych postupu vypocitat, zda vyhrazena oblast pameti nebude prekrocena. Zacatecnikum nedoporucujeme definovani spritu primo z Laser BASICu. Radeji pouzijte program Sprite Generator a takto definovane sprity nahrajte do vyhrazene oblasti pameti jednim z uvedenych postupu:
1) Pouzitim jedne ze tri moznosti v menu Laser BASICu
s) Po nahrani Laser BASICu piste:

```
CLEAR (pocatecni adresa spritu) - 1
LOAD "(jmeno souboru)" CODE (pocatecni adresa spritu)
.POKE 62464, (pocatecni adresa spritu)
```

Jmeno souboru je nazev, kterym byl pojmenovan soubor spritu, kdyz byl nahran na pasku za pouziti programu Sprite Generator. Pocatecni adresa spritu je nejnizsi byte pouzivany souborem spritu. Jeho hodnotu sdeli rovnez program Sprite Generator.

Data obrazovkovych bodu (pixelu)

Na obrazovce Spectra je kazdy znak tvoren polem 64 (8x8) bodu - pixelu. Kazdy pixel je v pameti reprezentovan jednim bitem. Pixel je bod, ktery muze mit bud barvu inkoustu, nebo barvu papiru. Body, ktere dohromady definuji znak nebo blok znaku, oznacujeme jako data pixelu.

Data atributu

Uvnitr kazdeho znaku je barva inkoustu, barva papiru, jas a pripadne blikani kontrolovano zvlastnim bytem. Byty, ktere definuji atributy bloku znaku, oznacujeme jako data atributu. Data pixelu a data atributu vystupuji v Laser BASICu casto jako dva samostatne utvary.

Operace s obrazovkou

Jsou to operace, ktere jsou provadeny na vyhrazene casti obrazovky. Tuto cast obrazovky nazyvame okenkem. Operace zahrnuji scrolovani ruznymi smery, inverze, zrcadleni atd. Vsechny prikazy teto kategorie jsou zakonceny pismenem "V". napr. .SR1V, .INVV, .MIRV atd. Pokud by okenko prekrocilo okraj obrazovky, bude automaticky upraveno tak, aby lezelo cele na obrazovce.

Operace obrazovka/sprite

Jsou to operace mezi obrazovkou a spritem. Rozmery spritu jsou pouzity jako rozmery okenka obrazovky a COL i ROW jsou pouzity k

definovani leveho horniho rohu okenka, takze operace potrebuje promenne SPN, COL a ROW. Jestlize okenko lezi mimo obrazovku, nebo sprite presahuje okraj obrazovky, pak pouze cast spritu vyuzije prikazu PUT a GOT. Prikazy teto kategorie maji jako prvni pismena "PT" nebo "GT", napr. .GTBL, .PTXR, .PTND atd.

Operace se sprity

Vicemene odpovidaji operacim s okenky obrazovky, ale tentokrat je misto okenka na obrazovce manipulovano se spritem v pameti. Je pouzivana jedina promenna SPN a vsechny prikazy teto kategorie jsou zakonceny pismenem "M".

Okenkove operace obrazovka/sprite

Jsou to operace mezi obrazovkovym okenkem a okenkem spritu. Jako predtim je obrazovkove okenko definovano pomoci ROW, COL, HGT a LEN. Tentokrat jsou vsak pouzity promenne SCL a SRW k urceni pozice okenka uvnitr spritu. SCL a SRW jsou mereny ve znacich, SCL zleva a SRW zhora. V pripade, ze SRW+HGT je vetsi nez vyska spritu, SCL+LEN je vetsi nez sirka spritu, LEN+COL je vetsi nez 32, nebo HGT+ROW je vetsi nez 24, se prikaz neprovede. Prikazy teto skupiny zacinaji pismeny "GW" nebo "PW".

Okenkove operace sprite/sprite

Jsou to operace mezi celym spritem a okenkem uvnitr druheho spritu. Obe cisla spritu jsou obsazena v SP1 (sprite bez okenka) a SP2 (sprite s okenkem). Rozmery okenka jsou rozmery spritu neobsahujiciho okenko. Pozice okenka uvnitr spritu SP2 je urcena pomoci SCL a SRW. Jestlize sprite SP1 prekroci hranici spritu SP2, prikaz se neprovede. Prikazy teto skupiny zacinaji pismeny "PM" nebo "GM".

Operace sprite/sprite

Temito operacemi je sprite menen a vysledek je ulozen do druheho spritu. V teto skupine jsou pouze dva prikazy: .SPNM a .DSPM.

Prazdny sprite

Prazdny sprite je sprite, ktery neobsahuje data pro displej. Muze byt pouzit napriklad pro uchovani podprogramu ve strojovem kodu, pro uchovani pole, nebo pouzit jako cast podprogramu pro zjis'tovani kolizi.

Editace a beh programu

Editor Spectra byl rozsiren, takze dalsi prikazy Laser BASICu pecuji o syntaktickou spravnost vkladanych prikazu. Vsechny prikazy Laser BASICu zacinaji teckou ".". Pak nasleduji ctyri velka pismena. Prikazy Sinclair BASICu jsou vkladany jako obvykle. Prirazeni (napr. .COL= nebo .ROW=) nesmi obsahovat mezi nazvem promenne a rovnitkem mezeru. Tak napr. .COL =4 by se ohlasilo chybovym hlasenim. Vsechny prikazy Laser BASICu vlozene mimo program se okamzite provedou.

Rozsirene funkce jsou oznaceny otaznikem "?", nasledovanym tremi znaky, napr. ?COL, ?ROW atd. Mohou byt pouzity pouze k prirazeni zjistene hodnoty k promenne. Nemohou byt pouzity jako soucast vyrazu nebo retezce PRINT.

```
LET X=?COL: LET Y=?KBF: LET Z=?SET      je spravne
LET X = ?COL*3+Y                        je nespravne
PRINT ?COL                              je nespravne
```

Je-li Laser BASIC pouzivan s pripojenym Interfacem 1, pak chyby spojene s provozem microdrivu zpusobi vypadnuti Laser BASICu a predani kontroly do Sinclair BASICu. Potom uz editor nebude prijimat rozsirene prikazy a pri pokusu o jejich vlozeni se ohlasi Syntax error. Abyste znovu vstoupili do Laser BASICu, vymazte radek, ktery se pokousite vlozit a stisknete "#" a ENTER. Nyni byste meli byt schopni pokracovat. Pokud ne, postupujte podle kapitoly "Pouzivani Laser BASICu s microdrivem" na str.2.
K rozbehnuti programu v Laser BASICu napiste povel RUN nasledovany ENTER. Nasleduje pauza, v prubehu ktere jsou prekladany rozsirene prikazy a pak zacne provadeni programu od prvniho radku. Prejete-li si zacit provadeni programu od urciteho radku, musite pouzit RUN nasledovane cislem radku. Tak napr. RUN 1000 rozbehne vas program od radku 1000. Na konci provadeni programu nasleduje opet pauza, behem ktere jsou rozsirene prikazy opet prelozeny do ctyrpismenne podoby. Taz pauza nasleduje po stisku tlacitka BREAK. Rozbehnout program muzete rovnez prikazy GO TO, GO SUB a CONTINUE.

Zaciname s vasim prvnim programem v Laser BASICu

Bylo by dobre, kdybyste na tomto miste nahrali Laser BASIC do pocitace zaroven se sprity, pokud jste tak dosud neucinily. Vetsina textu vysvetlujiciho jednotlive prikazy Laser BASICu obsahuje vypisy programu, ktere (jak doufame) vlozite do pocitace a vyzkousite. Tak se zacatecnikovi nejlepe objasni pouziti techto prikazu. Proto vam nalehave doporucujeme pomale prochazeni timto manualem a prubezne odskouseni vsech prikladu.

Nahrani Laser BASICu

Laser BASIC nahrajeme z kazety prikazem LOAD "". Pote, co se na obrazovce objevi vyzva, zastavte pasek. Laser BASIC bude pripraven k pouziti, jakmile se na obrazovce objevi menu. Pak byste meli nahrat soubor "SPRITE2A".

Nahrani spritu z OPTION 2

Tyto sprity pod nazvem "SPRITE2A" nahrajeme stiskem tlacitka "3" z menu a pak stiskem tlacitka PLAY na vasem magnetofonu. Po nahrani spritu zvolte volbu "1" z menu - tim spustite Laser BASIC.
Pouzijte sprity nahrane na puvodni kazete Laser BASIC, protoze nase priklady pracuji s temito sprity. Pozdeji, az budete psat vlastni programy, pouzijte sprity vami vytvorene v programu Sprite Generator.
Nyni muzete pracovat s Laser BASICem.

****
LASER BASIC
****

Jak jiz bylo uvedeno, vsechna klicova slova Sinclair BASICu se vkladaji jako obvykle. Tak napr. piseme:

```
10 REM TOTO JE VAS PRVNI PROGRAM V LASER BASICU
```

uplne stejne jako kdyz pouzivame Spectrum bez Laser BASICu, tzn. ze klicove slovo REM napiseme stisknutim klavesy E.
Nyni napiste dalsi radek, ktery jiz obsahuje prikaz Laser BASICu:

```
20 .COL=1
```

Samozrejme neexistuje klicove slovo .COL=, takze musite napsat byt zadne mezery mezi pismeny, ani po tecce, ani pred rovnitkem. Nemuzeme napsat .COL =.
Laser BASIC zaradi radek do programu po stisku ENTER. Jestlize jste radek napsali chybne, objevi se chybove hlaseni stejne jako v Sinclair BASICu.
Nyni dopiste zbytek programu:

```
30 .ROW=1
40 .HGT=20
50 .LEN=30
60 .INVV
```

Muzete ostartovat program prikazem RUN nebo GO TO 10. Pustime se do podrobneho vysvetleni prikazu Laser BASICu. Nejlepsim zpusobem je napsat do pocitace priklady a nechat je probehnout.

Prikazy pro psani programu

Aby se usnadnila tvorba programu, byly pridany ctyri nove prikazy .RNUM, .REMK, .TRON a .TROF.

Je to prikaz znamy z editacnich programu (toolkit), ktery precisluje radky a zaroven zmeni odpovidajicim zpusobem cisla nasledujici po prikazech GOTO, GOSUB a RESTORE. V porovnani s obdobnymi programi probiha precislovani velmi pomalu, ale zato neni potreba v pameti zadny prostor pro tabulky. Prikaz ovsem neprecisluje vypocitavana GO TO, tj. napr. GO TO 10 * y nebo GO TO 100 * 5, protoze hodnota vyrazu neni znama pred rozbehnutim programu. Spravna syntaxe je

```
.RNUM prvni radek, nova hodnota 1.radku, interval
```

Tak prikaz .RNUM 102, 1000, 5 ponecha beze zmeny radky s nizsim cislem nez 102 (a odkazy na ne), radek 102 precisluje na 1000 a vsechny nasledujici radky precisluje s intervalem 5. Napr. program:

```
   10 .COL=1
   32 .ROW=1
  102 .LEN=9
  118 .HGT=9
 3000 .INVV
```

se zmeni takto:

```
   10 .COL=1
   32 .ROW=1
 1000 .LEN=9
 1005 .HGT=9
 1010 .INVV
```

Neurcite-li velikost intervalu, pocitac pouzije interval 10. Proto napisete-li pouze .RNUM 102, 1000 bude program vypadat takto:

```
   10 .COL=1
   32 .ROW=1
 1000 .LEN=9
 1010 .HGT=9
 1020 .INVV
```

Pokud nenapisete ani cislo, kterym ma zacinat nove cislovani radku, pocitac pouzije cislo radku, od ktereho zacal precislovavat. Prikaz .RNUM 32 tedy da

```
   10 .COL=1
   32 .ROW=1
   42 .LEN=9
   52 .HGT=9
   62 .INVV
```

Jestlize nevlozite ani cislo radku, od ktereho se ma zacit precislovavat, pouzije pocitac prvni radek programu (s vyjimkou radku c.0). Napisete-li pouze .RNUM bez parametru, bude vysledek takovyto:

```
   10 .COL=1
   20 .ROW=1
   30 .LEN=9
   40 .HGT=9
   50 .INVV
```

Chybova hlaseni se objevi, jestlize
- nova hodnota cisla pocatecniho radku je nizsi nez puvodni hodnota predchazejiciho radku.
- precislovani by melo za vysledek cislo radku vetsi nez 10000.

Tento prikaz je pouzivan k odstraneni vsech prikazu REM z programu a tim ke zvetseni objemu volne pameti v pripade, ze program se stava prilis dlouhym. Prikaz nema zadne parametry a odstrani REM z celeho programu.

```
10 REM TOTO JE PRIKLAD
20 PRINT "AHOJ": REM TENTO RADEK TISKNE AHOJ
30 REM TENTO RADEK NETISKNE AHOJ: PRINT "AHOJ"
```

Zadejte ted .REMK a vsimnete si, co se stane s radky 20 a 30. Z programu vam ted zbyde pouze

```
20 PRINT "AHOJ"
```

Jedna z potizi pri odladovani BASICoveho programu je, ze nevime, jak program ve skutecnosti probiha. Moznost krokovani programem (.TRON) vam umozni krokovat programem radek po radku. Kazdy radek se pred provedenim vypise na obrazovku a Laser BASIC pocka az stisknete tlacitko a tim date prikaz k provedeni vypsaneho radku. Stiskem BREAK z krokovani vyjdete, ale muzete pouzit CONTINUE k pokracovani. Prikaz .TRON zajisti krokovani od nasledujiciho radku, nez ve kterem je obsazen. Prikaz musi byt obsazen primo v programu, protoze jinak by se zrusil prikazem RUN. Krokovani je opet vypnuto, jakmile pocitac narazi na prikaz

Graficke promenne

Zpusob, jakym jsou parametry predavany grafickym podprogramum je ponekud neobvykly. Je to v zajmu urychleni behu programu. Kazdy graficky prikaz pouziva zvlastni sadu deseti grafickych promennych. Nektere prikazy pozaduji az 5 parametru a vetsinou by tak bylo potreba vice casu na vyhodnoceni peti promennych, nez na provedeni prikazu sameho. Take je casto nutne, aby v prubehu vykonavani prikazu byly parametry prehodnoceny a vzdy je to nutne pri novem provadeni prikazu. Proto pouzivaji rozsirene prikazy Laser BASICu jako promenne jen ty parametry, ktere skutecne potrebuji byt meneny. Dalsi vyhodou je to, ze podprogramy maji tak presne urcene adresy promennych, ktere potrebuji, takze nemusi hledat ve vsech BASICovych promennych.

Existuje tedy 16 sad promennych po 10. Ty muzeme zvolit pouzitim prikazu .SET=(exp). EXP v tomto prikazu znamena cislo v rozsahu od 0 do 15 a muze byt vyjadreno jakymkoliv vyrazem pripustnym v BASICu, napr.

```
.SET=5*y+PEEK (58471)
```

Techto 10 promennych jsou:

Promenna	Pouziti
ROW	Obsahuje radek (koordinat Y) ve znacich, mereno od horniho okraje obrazovky (0-23). Nejhorejsi radek ma cislo 0.
COL	Obsahuje sloupec (koordinat X) ve znacich, mereno od leveho okraje obrazovky (0-31). Sloupec uplne vlevo ma hodnotu COL=0.
HGT	Obsahuje vysku platneho okenka obrazovky ve znacich (1-24).

LEN — Obsahuje sirku platneho okenka obrazovky ve znacich (1-32).

SRW — Urcuje radek (koordinat Y) uvnitr spritu mereno shora (0 az vyska-1). Jednotkou je znak.

SCL — Obsahuje sloupec (koordinat X) uvnitr spritu mereno zleva (0 az sirka-1). Jednotkou je znak.

NPX — Obsahuje velikost a smer vertikalniho scrolovani. Kladna cisla zajist~uji scroll nahoru a zaporna dolu. Jednotkou jsou pixely. Promenna muze nabyvat hodnot pouze od +127 do -128.

SPN — Obsahuje cislo spritu pro ty prikazy, ktere pracuji pouze s jedinym spritem. Nabyva hodnot 1 az 255.

SP1 — Tam, kde operace se provadi mezi spritem a okenkem ve spritu, obsahuje SP1 cislo spritu bez okenka (1-255).

SP2 — V operacich mezi spritem a okenkem ve spritu obsahuje SP2 cislo spritu, ktery nema okenko (1-255).

Prirazovani promennych

.COL= .ROW= .LEN= .HGT= .SP1= .SP2=
.SPN= .NPX= .SCL= .SRW= .SET=

Pomoci techto funkci muzete priradit hodnoty grafickym promennym Napr.

```
.COL=5:.ROW=2:.SPN=2
```

Zkuste napsat nasledujici program, ktery provadi se spritem na obrazovce funkci EXCLUSIVE OR tim, ze meni promennou COL pomoci smycky FOR-NEXT.

```
 1 REM PRIKLAD 1
 5 REM POUZITI JEDNE SADY GRAFICKYCH PROMENNYCH
10 .SET=0:.SPN=4:.ROW=0:.COL=-4
20 BORDER 1:BRIGHT 1:INK 6:PAPER 1: CLS:.ATOF
30 FOR I=-4 TO 32
40 .PTXR:.COL=I+1:.PTXR
50 PAUSE 4
60 NEXT I
70 STOP
```

K rozbehnuti pouzijte RUN nebo GO TO 1 a pak stisknete ENTER.

r.10 Je zvolena sada grafickych promennych c.1, cislo spritu nastaveno na 4, radek 0 (nejhorejsi radek) a sloupec -4.

r.20 Nastaveni atributu obrazovky, mazani obrazovky a navesti atributu je vypnuto.

r.30 Je jednoducha smycka, ktera pohybuje spritem c.4 (kachna) od sloupce -4 po sloupec 32.

r.40 Provadi XOR na starem spritu (viz .PTXR), pricita jednicku k promenne COL a pak provadi XOR na novem spritu.

r.50 Vytvari pausu v dobe, kdy je sprite na obrazovce.

Ted muzeme tento jednoduchy program rozsirit tak, aby se po obrazovce pohybovaly soucasne 2 sprity (ve skutecnosti zrcadlovy

obraz tehoz spritu) opacnymi smery. Opet budeme menit hodnoty COL, tentokrate vsak pouzijeme dve sady grafickych promennych. (V pripade, ze chcete v programu pouzit pouze jednu sadu techto promennych, nemusite pouzit vubec promennou SET ani funkci

```
 1 REM PRIKLAD 2
10 REM POUZITI DVOU SAD GRAFICKYCH PROMENNYCH
20 DEF FN A#(X,Y):.SET=X:.SPN=4:.ROW=Y
30 .RETN
35 INK 6: PAPER 0: BORDER 0: BRIGHT 1: CLS: .ATOF
40 .PROCFN A#(0,0):.PROCFN A#(1,1)
50 .COL=32:.SET=0:.COL=-4
60 FOR I=-4 TO 32
65 .SET=0:.PTXR:.COL=I+1:.PTXR
70 .SET=1:.MIRM:.PTXR:.COL=28-I:.PTXR:.MIRM
75 PAUSE 2
80 NEXT I
90 GO TO 40
```

Pozor na to, ze cast procedury na r.40 nejsou samostatna pismena FN, nybrz klicove slovo (Symbol shift +2)!
Tento program bude pohybovat dvema kachnami proti sobe z obou okraju obrazovky. K rozbehnuti pouzijte RUN nebo GO TO 1 a pak ENTER.

Zjist¯ovani obsahu promennych

Prave tak, jak je nutne prirazovat promennym hodnoty, je nutne zkoumat jejich soucasnou hodnotu. K tomu slouzi jedenact funkci:

?COL	?ROW	?LEN	?HGT	?SP1	?SP2
?SPN	?NPX	?SCL	?SRW	?SET	

Pri pouziti techto funkci muzete priradit platnou hodnotu promennych obvykle promenne ze Sinclair BASICu, napr.:

```
LET X=?COL: LET ROW=?ROW: LET ST=?SET
```

Vsimnete si, ze promenne ROW=?ROW jsou navzajem prirazeny, ale nezamenujte normalni promennou ROW a grafickou promennou ROW. Uvedenych 11 funkci nemuze byt pouzito jako soucast normalniho vyrazu, tj.:

```
LET X=?COL*3+?ROW/8    je neplatne
```

a nemohou byt pouzity jako parametry PRINT, napr.:

```
PRINT X,Y,?ROW         je rovnez neplatne
```

Stejneho vysledku ovsem muzeme dosahnout pri pouziti

```
LET COL=?COL: LET Y=?ROW: LET X=COL*3+Y/8
```

v prvnim pripade a

```
LET R=?ROW: PRINT X,Y,R
```

ve druhem pripade.

Muzete vlozit nasledujici radky do vyse uvedeneho prikladu 2, aby se vytiskly hodnoty COL pak, jak se sprity pohybuji:

```
75.SET=0: LET X0=?COL:.SET=1: LET X1=?COL
76 PRINT AT 10,16;" ";AT 11,16;" "
77 PRINT AT 10,12;"COL=";X0;AT 11,12;"COL=";X1
```

r.75 priradi promenne X0 hodnotu COL sady 0 a promenne X1 hodnotu COL sady 1
r.76 vymaze stare hodnoty na obrazovce
r.77 vytiskne nove hodnoty

Detailni popis rozsirenych grafickych prikazu

Prikazy spritu

Vsechny prikazy spritu pouzite v teto casti mohou byt pouzity pri provadeni programu pri zachovani nutne opatrnosti. Nicmene nalehave doporucujeme nezkusenemu uzivateli, aby sve sprity vytvarel v programu Sprite Generator.

vyhrazeny pro sprity je rozsiren, takze pocatek spritu zustava nezmenen, ale konec se zvysi o 9*HGT*LEN+5 bytu. Jestlize cislo spritu jiz bylo predtim prirazeno, potom je "stary sprite" nejdrive vymazan nezli je novy sprite prirazen. Je-li sprite definovan poprve, bude pravdepodobne obsahovat nesmysly a data musi byt ziskana prikazem "GOT" za pouziti prikazu jako napr. .GTBL.

Parametr	Pouziti
SPN	cislo vytvareneho spritu (1-255)
LEN	delka spritu ve znacich (1-255)
HGT	vyska spritu ve znacich (1-255)

Pozn: Protoze prikaz rozsiruje prostor spritu v pameti smerem vzhuru, bude pouzivan velmi zridka pro specialni ucely. Vetsina ucelu bude pozadovat nasledujici prikaz .ISPR. Neucelne pouziti prikazu .SPTR muze zpusobit zhrouceni systemu.

pameti se rozsiruje smerem dolu, takze konec spritu zustava nezmenen, ale pocatek spritu se posune dolu o 9*HGT*LEN+5 bytu. Pokud byl jiz predtim sprite prirazen, je ohlasena chyba. Jestlize provedeni prikazu .ISPR zpusobi posunuti zacatku spritu pod RAMTOP, system se zhrouti. Pamatujte, ze podobne jako u .SPRT muze sprite na zacatku obsahovat nesmyslna data a prava data jsou dodana za pouziti prikazu tridy "GOT", napr. .GTBL.

Parametr	Pouziti
SPN	cislo vytvareniho spritu (1-255)
LEN	delka spritu ve znacich (1-255)
HGTzl	vyska spritu ve znacich (1-255)

spritu je zmensen, takze zacatek spritu zustava nezmenen, ale konec spritu je snizen. Tento prikaz je rovnez uzivan pouze ve slozitejsich aplikacich a normalne se k vymazani spritu pouzije prikaz .DSPR. Je-li uskutecnen pokus o vymazani neexistujiciho spritu, objevi se chybove hlaseni 0. Tak napr. k vymazani spritu 1 musime napsat

```
.SPN=1:.WSPR
```

Parametr	Pouziti
SPN	cislo vymazavaneho spritu (1-255)

pameti je zmensen tak, ze konec spritu zustava nezmenen, ale pocatek spritu se posune vzhuru. V pripade, ze se pokusite vymazat neexistujici sprite, bude vyvolano chybove hlaseni 0. Tento prikaz se obvykle pouzije k vymazani spritu a prikaz .WSPR se pouzije pouze v narocnejsich aplikacich. K vymazani spritu 2 napr. pouzijete

```
.SPN=2:.DSPR
```

Parametr	Pouziti
SPN	cislo vymazavaneho spritu (1-255)

V nize uvedenem prikladu 3 zkoumame pamet¯, zda je dostatek prostoru pro .ISPR:

```
 5 REM PRIKLAD 3
10 REM PROCEDURA K PREZKOUSENI ZDA JE DOSTATEK PAMETI PRO
   .ISPR
20 REM X=LENGTH, Y=HEIGTH
30 DEF FN T#(X,Y)
40 LET SIZE=9*X*Y+5
50 LET SPACE=PEEK(62464)+256*(PEEK(23731)-PEEK(62465))
60 IF SIZE>SPACE THEN PRINT "NO ROOM AVAILABLE"
70 IF SIZE<SPACE THEN PRINT "ROOM AVAILABLE"
80 .RETN
```

Procedure musime predat parametry X a Y udavajici delku a vysku spritu. Napr. .PROCFN T#(5,5) pro sprite velikosti 5x5. Velikost spritu je vypocitana a porovnana s velikosti volne pameti (SPST-RAMTOP) a je generovano prislusne sdeleni.

```
 5 REM PRIKLAD 4
10 REM PROCEDURA KE ZJISTENI, ZDA JE DOSTATEK MISTA PRO
       DEFINOVANI SPRITU N
20 DEF FN S#(N,X,Y)
30 LET SIZE=9*X*Y+5
40 LET SPACE=PEEK(62464)-PEEK(23730)+256*(PEEK(23731)-PEEK
   (62465))
50 IF SIZE<=SPACE THEN .SPN=N:.LEN=X:.HGT=Y:.ISPR
60 IF SIZE>SPACE THEN PRINT "NENI MISTO"
70 .RETN
```

Procedura v priklade 4 prezkouma, zda je dostatek mista. Jestli ano, provede .ISPR. Jestli ne, vyvola prislusne hlaseni.
Tak napr. muzete pro vytvoreni spritu c.1 o velikosti 2x3 znaky Psat:

```
.PROCFN S#(1,2,3)

 5 REM PRIKLAD 5
10 REM PROCEDURA K VYMAZANI EXISTUJICIHO SPRITU
       A K VYTISKNUTI MNOZSTVI UVOLNENYCH BYTU
20 DEF FN D#(N)
30 .SPN=N: LET START=?TST: LET X=?HGT: LET Y=?LEN
40 .DSPR: PRINT 9*X*Y+5;"BYTU UVOLNENO"
50 .RETN
```

Procedura v prikladu 5 zmeri rozmery spritu N (viz ?TST) a vymaze jej, pokud existoval. Pak vypise pocet uvolnenych bytu. Napr. muzete vymazat sprite 1 napsanim .PROCFN D#(1).

Vodorovne scrolovani obrazovky

Obrazovku muzeme scrolovat vodorovne o 1, 4 nebo 8 pixelu, vlevo nebo vpravo, s prechodem znovu na zacatek obrazovky nebo bez nej. K tomu slouzi tyto prikazy:

Prikaz ucinek

Parametr	Pouziti
COL	sloupec leveho okraje (0-31)
ROW	radek od horniho okraje (0-23)
LEN	delka okenka (1-32)
HGT	vyska okenka (1-24)

Uvedene prikazy pracuji na obrazovce. Scroluji okenko, jehoz rozmery jsou obsazeny v promennych LEN a HGT a ktere je umisteno podle souradnic obsazenych v promennych COL a ROW.

Priklad 6 predvadi scrolovani o 1, 4 a 8 pixelu s prechodem znovu na zacatek obrazovky.
r.10 zaplni obrazovku
r.20 definuje okenko 20x10 umistene ve sloupci 6 a radku 2
r.30 az 80 scroluje okenko (300x pro kazdy priklad).

```
 5 REM PRIKLAD 6
10 FOR N=1 TO 55: PRINT "LASER BASIC";:NEXT N
20 .LEN=20:.HGT=10:.ROW=2:.COL=6
30 REM SCROLL O 1 PIXEL
40 FOR N=1 TO 300:.WL1V: NEXT N
50 REM SCROLL O 4 PIXELY
60 FOR N=1 TO 300:.WL4V: NEXT N
70 REM SCROLL O 8 PIXELU
80 FOR N=1 TO 300:.WL8V: NEXT N
```

Nahradte .WL1V prikazem .SL1V (bez prechodu) a pozorujte vysledek. Musite pocitat s tim, ze tyto prikazy scroluji pouze pixely a ne atributy, ktere jsou umisteny v pameti jinde.

Priklad 7 ukazuje rozdil mezi pixely a atributy.
r.10 zapina pohyb atributu

r.20 umisti sprite 34 (mouchu) na pozici 15,2 (viz .PTBL)
r.30 definuje okenko o sirce cele obrazovky okolo spritu
r.50 scroluje pixely 256x vpravo s prechodem, ponecha atributy za sebou

```
 5 REM PRIKLAD 7
10 .ATON
20 .SPN=34:.ROW=2:.COL=15:.PTBL
30 .LEN=32:.HGT=2:.COL=0:.ROW=2
40 PAUSE 50
50 FOR N=1 TO 256:.WR1V: NEXT N
```

Nejste samozrejme omezeni na scrolovani pouze o 1, 4 nebo 8 pixelu.
Tak napr. scrolovani o 3 pixely dosahnete kombinaci prikazu napr. scrol o 9 pixelu docilite devaterym opakovanim prikazu
r.10 potiskne obrazovku
r.20 definuje rozmery okenka (32x1)
r.30 zacatek smycky
r.40 vypocet ROW a poctu scrolovani - na teto radce zacatek smycky
r.50 nastavuje promennou ROW pro scrolovani okenka
r.60 1x scroluje okenko

```
 5 REM PRIKLAD 8
10 FOR N=1 TO 20: PRINT AT N,0;"SCROLL 0 ";N;" PIXELU":
   NEXT N
20 .LEN=32:.HGT=1:.COL=0
30 FOR N=1 TO 256
40 FOR I=1 TO 20
50 .ROW=I
60 FOR M=1 TO I:.WR1V: NEXT M
70 NEXT I
80 NEXT N
```

Vertikalni scrolovani obrazovky

Tyto prikazy pracuji podobne jako prikazy pro horizontalni scrol, ale pro jejich funkci je krome okenkovych promennych COL, ROW, HGT a LEN potrebna jeste promenna NPX, ktera udava smer a velikost scrolovani v pixelech. Kladna hodnota NPX zajisti scrolovani nahoru a zaporna dolu.
Napr. .NPX=-1 o 1 pixel dolu
.NPX=1 o 1 pixel nahoru

Prikaz Vysledek

Parametr	Pouziti
COL	levy krajni sloupec (0-31)
ROW	radek odzhora (0-23)
LEN	sirka okenka (1-32)
HGT	vyska okenka (1-24)
NPX	velikost a smer scrolovani (-128 az +128)

Pozn.: Vsechny scroly pixelu nebo atributu vyzaduji prostor v pameti (plati pouze pro vertikalni scrolovani). Urceny prostor vypocitame nasobenim NPX a LEN. Vysledek nesmi presahnout 256 bytu, protoze je k dispozici pouze prostor mezi adresami 23296 a 23551.

Pojistit se muzeme nasledujici sadou prikazu:

```
LET X=?NPX: LET Y=?LEN: IF ABS(X)*Y<256 THEN .WCRV
```

Priklad 9 je podobny prikladu 6, ale predvadi scrolovani o 1 pixel nahoru a dolu.
r.40 provede 300x scrolovani o 1 pixel nahoru
r.60 provede 300x scrolovani o 1 pixel dolu

```
 5 REM PRIKLAD 9
10 FOR N=1 TO 55: PRINT "LASER BASIC";: NEXT N
20 .LEN=20:.HGT=10:.ROW=2:.COL=6
30 REM SCROLL NAHORU O 1 PIXEL
40 .NPX=1: FOR N=1 TO 300:.WCRV: NEXT N
50 REM SCROLL DOLU O 1 PIXEL
60 .NPX=-1: FOR N=1 TO 300:.WCRV: NEXT N
```

V prikladu 10 budeme scrolovat nahodne vybrane sloupce siroke 1 znak o 1 pixel.
r.10 vytiskne radu "A" u spodniho okraje obrazovky
r.20 nastavi parametry okenka
r.30 vybere nahodne sloupec a ulozi jeho cislo do promenne COL
r.40 scroluje sloupec o 1 pixel nahoru bez prechodu
r.50 uzavira smycku

```
 5 REM PRIKLAD 10
10 FOR N=0 TO 31: PRINT AT 21,N;"A": NEXT N
20 .LEN=1:.HGT=22:.ROW=0:.NPX=1
30 LET X=INT(RND*33)-1:.COL=X
40 .WCRV
50 GO TO 20
```

Scrolovani atributu

Scrolovani atributu je podobne scrolovani pixelu, ale vzdy se scroluje o 1 cely znak s prechodem na zacatek obrazovky. Prostor uzivane pameti je rovny hodnote LEN a je proto vzdy mensi nez 33 bytu.

Prikaz Vysledek

Parametr	Pouziti
COL	levy krajni sloupec (0-31)
ROW	horni radek (0-23)
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)

Priklad 11 predvadi pouziti prikazu .ATRV. Na obrazovce jsou umisteny sloupce atributu. Za pouziti smycek FOR-NEXT je scrolo-

van horni radek o 1 znak, druhy radek o 2 znaky atd. Opakovanim jsou vytvareny vzorce.

```
  5 REM PRIKLAD 11
 10 INK 7: PAPER 0: BRIGHT 1: CLS
 20 FOR N=0 TO 703: PRINT CHR$ (129): NEXT N
 30 .LEN=1:.HGT=24:.ROW=0: FOR N=0 TO 31:.COL=N:
    INK INT(RND*7)+1:.SETV: NEXT N
 40 .LEN=32:.COL=0:.HGT=1
 50 FOR I=0 TO 31
 60 FOR Y=1 TO 22
 70 .ROW=Y-1
 80 FOR X=1 TO Y
 90 .ATRV
100 NEXT X
110 NEXT Y
120 NEXT I
```

```
r.10  nastavi atributy
r.20  zaplni obrazovku CHR$ 129
r.30  zaplni obrazovku nahodne zbarvenymi sloupci (viz .SETV)
r.40  nastavi parametry pro .ATRV
r.50  I je pocet scrolovani nutnych pro znovusestaveni obrazovky
r.60  Y je radek
r.80  je pocet znaku scrolovanych v radku
r.90  provadi scrolovani
```

Zpusob jak jsou atributy scrolovany oddelene od pixelu je ukazan v prikladu 12, ktery je podobny prikladu 7. V tomto pripade jsou ale ponechany bez pohybu pixely.

```
  5 REM PRIKLAD 12
 10 .ATON
 20 .SPN=34:.ROW=2:.COL=15:.PTBL
 30 .LEN=32:.HGT=2:.COL=0:.ROW=2
 40 PAUSE 50
 50 FOR N=1 TO 32:.ATLV: NEXT N
```

Vzhledem k omezenim, danym hardwarem Spectra, muzete scrolovat atributy pouze po 8 pixelech, cili po jednom znaku. Jestlize zmenite r.50 v prikladu 12 takto:

```
 50 FOR N=1 TO 32:.ATLV:.WL8V: NEXT N
```

budou scrolovat zaroven pixely i atributy.

Vodorovne scrolovani spritu

Muzeme scrolovat o 1, 4 nebo 8 pixelu, vlevo i vpravo, s prechodem na zacatek obrazovky nebo bez nej. Jestlize pozadovany sprite neexistuje, vyvola se chybove hlaseni. Seznam prikazu:

Prikaz Vysledek

Parametr Pouziti
SPN cislo scrolovaneho spritu (1-255)

Vetsina drive uvadenych prikazu koncila na "V", coz znamena "video". Prikazy koncici na "V" se tykaji pouze obrazovky a ponechavaji sprity v pameti beze zmeny.
Vsechny nyni uvedene prikazy konci na "M", coz znamena "memory, pamet". Pri provedeni tohoto prikazu se meni sprite v pameti. Zmenu neuvidite na obrazovce, dokud neumistite sprite na obrazovce.

Priklad 13 je podobny prikladu 6, ale zde scrolujeme sprite, ktery pak umistime na obrazovce, to vse 300x. Podobne jako u scrolovani obrazovky se pohybuji pouze pixely. Pred pouzitim prikladu musite nahrat blok spritu "SPRITE2A".

```
 5 REM PRIKLAD 13
10 .ROW=10:.COL=14:.SPN=49
20 REM SCROLOVANI SPRITU O 1 PIXEL
30 FOR N=1 TO 300:.WL1M:.PTBL: NEXT N
40 REM SCROLOVANI SPRITU O 4 PIXELY
50 FOR N=1 TO 300:.WL4M:.PTBL: NEXT N
60 REM SCROLOVANI SPRITU O 8 PIXELU
70 FOR N=1 TO 300:.WL8M:.PTBL: NEXT N
```

r.10 nastavi ROW a COL pro sprite
r.30 300x scroluje sprite o 1 pixel a pak ho umisti na stinitko
r.50 300x scroluje sprite o 4 pixely a pak ho umisti
r.70 300x scroluje sprite o 8 pixelu a pak ho umisti
      (to se deje tak rychle, ze sprite se rozmazne)

Vertikalni scrolovani spritu

Prikazy funguji stejne jako u vertikalniho scrolovani obrazovky. Promenna NPX urcuje velikost a smer scrolovani. Jestlize sprite neexistuje, vyvola se chybove hlaseni.

Prikaz Vysledek

Parametr Pouziti
SPN cislo scrolovaneho spritu (1-255)
NPX o kolik pixelu se bude scrolovat (-128 az +127)

Pozn.: Podobne jako u vertikalniho scrolovani obrazovky nesmi vyuzivana cast pameti presahout 256 bytu. Potrebny rozsah pameti je dan NPX delkou spritu. Delku spritu ziskame pouzitim funkce ?TST.

Priklad 14 predvadi vertikalni scrolovani spritu tak, ze scroluje sprite 5 (tanecnik) a vyplni obrazovku scrolovanymi sprity.

```
 5 REM PRIKLAD 14
10 BORDER 0:.SPN=5:.NPX=8
20 FOR Y=0 TO 20 STEP 4
30 FOR X=0 TO 31 STEP 2
40 .COL=X:.ROW=Y:.PTBL
50 .WCRM
60 NEXT X
70 NEXT Y
80 GO TO 80
```

r.10 nastavi parametry a barvu BORDERu
r.20 a 30 nastavi souradnice X a Y pro umisteni spritu
r.40 umisti sprite c.5 na souradnice X a Y
r.50 scroluje sprite nahoru o 8 pixelu

Scrolovani atributu spritu

Podobne jako u scrolovani atributu obrazovky existuji 4 prikazy pro scrolovani atributu spritu ve 4 ruznych smerech, vzdy o 1 znak s prechodem na zacatek obrazovky.

Prikaz Vysledek

Parametr	Pouziti
SPN	cislo scrolovaneho spritu (1-255)

Prikazy PUT a GET skupiny 1

Operace PUT premisti sprite na obrazovku nebo do jineho spritu, zatimco GET premisti opacne data z obrazovky nebo spritu do jineho spritu. Jsou tri skupiny techto prikazu. Prvni skupina (nejrychlejsi) provadi operace mezi spritem a predem definovanym okenkem obrazovky. Prikazy v teto skupine zacinaji pismeny "GT" nebo "PT". Prvni skupina nema oddelene prikazy pro premisteni pixelu a atributu, misto toho pouziva prepinani atributu (viz V pripade, ze sprite neexistuje, vyvola se chybove hlaseni.

Prikaz Vysledek

Parametr	Pouziti
COL	levy sloupec vysledne pozice na obrazovce (0-31)
ROW	horni radek vysledne pozice na obrazovce (0-23)
SPN	cislo spritu, se kterym je operace provadena (1-255)

Pozn.: Rozmery okenka obrazovky urcuji rozmery spritu. Jestlize COL+delka spritu je vetsi nez 32 nebo ROW+vyska spritu je vetsi nez 24, pak prikaz premisti pouze cast spritu.

Jestlize nyni napisete:

```
.SPN=39:.ROW=1:.COL=1:.PTBL
```

pak umistite sprite c.39 na obrazovce v pozici AT 1,1. Je to nejrychlejsi a nejjednodussi zpusob umisteni spritu na obrazovce

Napsanim prikazu .ATOF zastavite pohyb atributu. Napisete-li nyni .ROW=5:.PTBL (nemusite definovat SPN ani COL, ty zustavaji beze zmeny), pak se sprite c.39 objevi znovu na obrazovce, ale tentokrat bez svych atributu. Napiste .ATON:.PTBL; sprite se opet objevi s atributy.
Prikaz .PTBL premisti veskera data spritu na obrazovku. Mate k dispozici tri dalsi operace - .PTOR,.PTXR a .PTND, ktere provadi logicke operace OR, XOR a AND mezi spritem a obrazovkou.
Napisete-li:

```
.ATOF:.SPN=18:.ROW=10:.COL=10:.PTBL
```

umistite sprite c.18 (tank) na obrazovku. Napsanim prikazu obrazovce. Vlozite-li opet .PTXR a ENTER, sprite se sam vymaze a obnovi puvodni data obrazovky.

V prikladu 15 se provadi XOR mezi spritem 18 (tank) a polem mysi tak, ze se nic neznici:

```
 5 REM PRIKLAD 15
10 .ATOF:.SPN=6
20 FOR N=1 TO 50
30 LET X=INT(RND*32): LET Y=INT(RND*20):.COL=X:.ROW=Y:.PTBL
40 NEXT N
50 .SPN=16:.ROW=10
60 FOR X=-4 TO 32
70 .COL=X:.PTXR: PAUSE 4:.PTXR
80 NEXT X
90 .ATON
```

r.10 vypina pohyb atributu a nastavi SPN na 6 (mysi)
r.20 az 40 naplni obrazovku padesati myskami
r.50 nastavi parametry pro sprite 16
r.60 jednoducha smycka pro vypocet COL
r.70 sprite c.16 je podroben oparaci XOR s obrazovkou, zatimco je sprite na obrazovce, zaradi se pauza. Pak je se spritem provedeno XOR na teze pozici, takze se sam vymaze a obnovi se puvodni data na obrazovce.

Vlozite-li nyni .SPN=18:.ROW=10:.COL=10:.PTBL, objevi se podle ocekavani na obrazovce sprite c.18. Pozorujte, co se stane, kdyz napisete .COL=29:.PTBL (pozor sprite je siroky 6 znaku). Pocitac umistil ze spritu na obrazovku tolik, kolik mohl.
Zkuste ted .COL=3:.PTBL.
Prikazy GET (zacinajici "GT") berou data z obrazovky na pozici urcene promennymi ROW a COL z okenka, jehoz rozmery jsou urceny rozmery spritu. Ziskana data se ulozi do spritu.
V prikladu 16 je nakreslen bod v pravem dolnim rohu obrazovky, okolo nej je definovano okenko, to je premisteno do spritu 1 (ktery je tim znicen) a sprite je pak umisten na obrazovce. Tento proces se pak opakuje.

```
  5 REM PRIKLAD 16
 10 .SET=0:.SPN=1:.ROW=20:.COL=28
 20 .SET=1:.SPN=1:.ROW=0:.COL=0
100 DEF FN A#()
110 LET PX=INT(RND*32): LET PY=INT(RND*16)
120 PLOT PX+223,PY
130 .RETN
200 FOR Y=0 TO 20 STEP 2
220 FOR X=0 TO 32 STEP 4
240 .PROCFN A#()
250 .SET=0:.GTBL
260 .SET=1:.ROW=Y:.COL=X:.PTBL
270 NEXT X
280 NEXT Y
```

r.10 nastavi SET na nulu (pravy dolni roh obrazovky)
r.20 nastavi SET 1 (sprite)
r.100 definuje proceduru A
r.110 a 120 nahodne umisti pixel na obrazovce
r.200 a 210 vypocitaji hodnoty COLa ROW pro umisteni spritu 1
r.240 vyvola kreslici proceduru
r.250 uchova sprite 1
r.260 umisti na obrazovku sprite 1

Prikazy GET a PUT akupiny 2

Tyto prikazy umoznuji operace mezi okenkem spritu a okenkem obrazovky. Na rozdil od skupiny 1 existuji oddelene prikazy pro pohyb pixelu a atributu, i kdyz prikazy .ATON a .ATOF maji vliv na tyto operace. Jsou zavedeny 4 nove parametry k definovani sloupce a radku ve spritu a vysky i sirky okenka uvnitr spritu. Jestlize okenko presahuje hranice spritu nabo obrazovky, prikaz se naprovede. Jestlize sprite neexistuje, je generovano chybove hlaseni. Vsechny prikazy v teto skupine zacinaji pismeny "GW" nebo "PW".

Prikaz Vysledek

Parametr	Pouziti
COL	levy krajni sloupec ciloveho okenka obrazovky (0-31)
ROW	horni radek ciloveho okenka obrazovky (0-23)
SCL	levy krajni sloupec okenka spritu; 0 az delka spritu-1
SRW	horni radek okenka spritu; 0 az vyska spritu-1
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)
SPN	cislo spritu (1-255)

V prikladu 17 jsou nahodne vybirana okenka velikosti 1x1 znak za spritu 49 (medvidek) a umistena na obrazovce.

```
  5 REM PRIKLAD 17
 10 .SPN=49:.HGT=1:.LEN=1
200 FOR Y=0 TO 20
220 FOR X=0 TO 32
230 LET GX=INT(RND*4)
240 LET GY=INT(RND*5)
250 .COL=X.ROW=Y:.SCL=GX:.SRW=GY
260 .PWBL
270 .PWAT
280 NEXT X
290 NEXT Y
```

r.10 nastavi SPN (cislo spritu), HGT a LEN (rozmery okenka)
r.200 a 220 vypocitaji pozice COL a ROW pro umisteni okenek
r.230 a 240 vypocitaji nahodne COL a ROW uvnitr spritu
r.250 nastavi tyto hodnoty
r.260 vybere pixely z okenka a umisti je na obrazovce
r.270 vybere atributy

Prikazy GET a PUT skupiny 3

Tato skupina, pravdepodobne nejuzitecnejsi ze vsech, zahrnuje prikazy, ktere umoznuji operace mezi spritem a okenkem ve druhem spritu. Vsechny prikazy treti skupiny zacinaji pismeny "PM" nebo "GM". Jestlize okenko spritu presahuje hranice spritu, prikaz se neprovede. Neexistuje-li nektery ze spritu, generuje se chybove hlaseni 0.

Prikaz Vysledek

Parametr	Pouziti
SP1	cislo prvniho spritu (1-255)
SP2	cislo druheho spritu s okenkem (1-255)
SCL	levy sloupec okenka spritu (1-delka spritu)
SRW	horni radek okenka spritu (1-vyska spritu)

Prikaz pouzivany k vytvoreni jednoducheho a efektivniho animovaneho nebo neanimovaneho pohybu spritu. Prikaz pouziva operaci XOR k vytvoreni nedestruktivniho pohybu spritu. Jetlize tedy sprite zacina na obrazovce, musite pouzit .PTXR pred pouzitim prikazu .MOVE. Neni to nutne v pripade, ze se sprite umisti na obrazovku z pozice mimo obrazovku. Logicka operace XOR pusobi stejne jako tisk s prikazem OVER 1.

Parametr	Pouziti
COL	COL spritu, kterym budeme pohybovat (0-31)
ROW	ROW spritu, kterym budeme pohybovat (0-23)
HGT	HGT pohybu ve znacich (-24 az +24)
LEN	LEN pohybu ve znacich (-32 az +32)
SP1	cislo pohybovaneho spritu (1-255)
SP2	cislo spritu po pohybu (1-255)

obrazovce na pozici urcene COL a ROW a pak umisti sprite SP2 na

obrazovku na pozici COL+LEN, HGT+ROW. Promenne COL a ROW jsou pak zvetseny o hodnoty HGT a LEN a vymeni se cisla spritu SP1 a SP2.

Priklad 18 pohybuje nedestruktivne po obrazovce spritem zleva shora vpravo dolu. V promennych SP1 a SP2 uchovava hodnoty stareho a noveho spritu (v obou pripadech c.6 - mys).
Protoze sprite je v levem hornim rohu, jsou puvodni hodnoty ROW i COL nula.
Ted chceme spritem pohybovat dolu a doprava o 1 znak. HGT a LEN musi byt proto nastaveny na 1. Prikaz .MOVE automaticky inkrementuje ROW i COL o hodnoty uchovane v HGT a LEN. Tak vam skutecne zbyva pouze zadat .MOVE abyste spritem pohybovali.

```
  5 REM PRIKLAD 18
 10 FOR N=1 TO 100
 20 LET X=INT(RND*32)
 30 LET Y=INT(RND*21)
 40 PRINT AT Y,X;"A"
 50 NEXT N
 60 .ATOF:.COL:.ROW:.SPN=6:.PTXR
 70 .HGT=1:.LEN=1
 80 .SP1=6:.SP2=6
 90 FOR N=1 TO 25
100 .MOVE
110 PAUSE 3
120 NEXT N
```

r.10 az 50 zaplni obrazovku pismeny "A" v nahodnych pozicich
r.60 umisti sprite c.6 (mys) do leveho horniho rohu obrazovky
r.70 nastavi hodnoty HGT a LEN na 1
r.80 urci, ze oba sprity, stary i novy budou c.6
r.90 je jednoducha smycka
r.100 provede .MOVE

Samozrejme HGT a LEN mohou byt zaporne.
Pri pouziti vhodnych SP1 a SP2 muzete zaroven animovat i pohybovat spritem.

V prikladu 19 jsou HGT i LEN zaporne, aby se dosahlo pohybu v opacnem smeru, nez v prikladu 18.

```
  5 REM PRIKLAD 19
 10 FOR N=1 TO 100
 20 LET X=INT(RND*32)
 30 LET Y=INT(RND*21)
 40 PRINT AT Y,X;"A"
 50 NEXT N
 60 .ATOF:.COL=0:.ROW=22:.SPN=6:.PTXR
 70 .HGT=-1:.LEN=-1
 80 .SP1=6:.SP2=6
 90 FOR N=-1 TO 25
100 .MOVE
110 PAUSE 3
120 NEXT N
```

V prikladu 20 pohybujeme spritem c.6 do kruhu tim, ze stale menime hodnoty prirustku HGT a LEN.

```
  5 REM PRIKLAD 20
 10 FOR N=1 TO 150
 20 LET X=INT(RND*32)
 30 LET Y=INT(RND*22)
 40 PRINT AT Y,X;"A"
 50 NEXT N:.ATOF
 60 .ROW=10:.COL=4:.SPN=6:.PTXR
 70 .SP1=6:.SP2=6
 80 .HGT=0:.LEN=0
 90 LET C=4: LET R=10
100 FOR N=1 TO 30
110 .COL=C:.ROW=R
120 LET C=INT(14-10*COS(N/15*PI))
130 LET R=INT(10+10*SIN(N/15*PI))
140 LET OC=?COL: LET OR=?ROW
150 .LEN=(OC-C)*-1:.HGT=(OR-R)*-1
160 .MOVE
170 NEXT N
180 GO TO 100
```

r.10 az 50 naplni obrazovku znaky "A"
r.60 presune sprite c.6 na obrazovku
r.70 az 90 nastavi parametry
r.100, 120 a 130 vypocitaji hodnoty COL a ROW tak, ze vypocitaji body na okraji kruhu
r.140 a 150 vypocitaji hodnoty, ktere budou uchovany v HGT a LEN
r.160 provede .MOVE

Pohyb atributu je rizen dvema prikazy, ktere urcuji, zda se atributy budou pohybovat zaroven s pixely v prikazech PUT a GET.

Prikaz Vysledek

Parametr Pouziti
zadny zadne

Transformace
============

Ke zvyseni uzitne hodnoty programu byla do nej vclenena ctyri slova, ktera invertuji, otaceji, zrcadli a zvetsuji. Inverze a zrcadleni funguji pro obrazovku i sprity, ale rotace a zvetsovani jsou urceny pouze pro sprity.

LEN je invertovano, tzn. vsechny pixely, ktere byly zobrazeny, se vymazou, a vysledkem je vymena barev "papiru" a "inkoustu". Jestlize okenko presahuje okraje obrazovky, pak je prislusne zmenseno tak, aby cele lezelo na obrazovce.

Parametr	Pouziti
COL	levy krajni sloupec okenka (0-31)
ROW	horni radek okenka (0-23)
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)

Kdybyste napriklad napsali .ROW=1:.COL=1:.HGT=5:.LEN=5:.INVV, pak okenko o rozmerech 5x5 by se invertovalo. Jiny priklad uziti

```
  5 REM PRIKLAD 21
 10 BORDER 0
 20 FOR N=1 TO 58: PRINT "LASER BASIC";: NEXT N
 30 FOR N=1 TO 10
 40 .HGT=22-(N*2)
 50 .LEN=32-(N*2)
 60 .ROW=N
 70 .COL=N
 80 .INVV
 90 PAUSE 10
100 NEXT N
110 GO TO 30
```

priklad 21 vytvari dojem tunelu menenim pozice a rozmeru okenka.

Prikaz	Vysledek
	na celem spritu, jehoz cislo je ulozeno v SPN.

Parametr	Pouziti
SPN	cislo invertovaneho spritu (1-255)

K inverzi spritu c.10 postaci napsat .SPN=10:.INVM

Prikaz	Vysledek
	ROW, HGT a LEN jsou zrcadleny podel svisle osy, lezici uprostred okenka. Jestlize okenko presahuje okraje obrazovky, pak je zmenseno.

Parametr	Pouziti
COL	levy krajni sloupec okenka (0-31)
ROW	horni radka okenka (0-23)
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)

Priklad 22 predvadi .MIRV tak, ze zrcadli text na polovine obrazovky.

```
  5 REM PRIKLAD 22
 10 LET A$="TOTO JE PRIKLAD POUZITI .MIRV V LASER BASICU NA
    POCITACI SPECTRUM PODLE MANUALU, KTERY PRELOZIL M.MANEK"
 20 LET Y=0: LET X=0
 30 FOR S=1 TO LEN A$
 40 LET C=CODE A$(S TO S)
 50 IF C=32 THEN LET Y=Y+1: LET X=0: GO TO 80
 60 PRINT AT Y,X; CHR$(C); AT Y,X+16; CHR$(C)
 70 LET X=X+1
```

```
     80 NEXT S
     90 .ROW=0:.COL=0:.LEN=16:.HGT=22:.MIRV
```

```
r.10  vytvori textovy retezec
r.20  az 80 vytisknou retezec s jednim slovem na kazdem radku
r.90  zrcadli polovinu obrazovky
```

Prikaz	Vysledek
	zrcadleny podel svisle osy lezici ve stredu spritu. Jestlize sprite neexistuje, je generovano chybove hlaseni.

Parametr	Pouziti
SPN	cislo zrcadleneho spritu (1-255)

Prikaz	Vysledek
	ROW, HGT a LEN jsou zrcadleny podel svisle osy okenka. Jestlize okenko presahuje hranice obrazovky, pak je prislusne zmenseno.

Parametr	Pouziti
COL	levy krajni sloupec okenka (0-31)
ROW	horni radek okenka (0-23)
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)

Prikaz	Vysledek
	jsou zrcadleny podel svisle osy spritu. Jestlize sprite neexistuje, je generovano chybove hlaseni.

Parametr	Pouziti
SPN	cislo spritu, jehoz atributy budou zrcadleny (1-255)

Prikaz	Vysledek
	90 stupnu po smeru hodinovych rucicek. Tato operace vyzaduje, aby oba sprity mely odpovidajici rozmery. Jestlize napriklad mame otocit a prenest sprite o rozmerech 8x3, tak v tom pripade musi mit SP1 rozmery 3x8, jinak se prikaz proste neprovede. Otaci se zaroven pixely i atributy.

Parametr	Pouziti
SP1	cislo otaceneho spritu (1-255)
SP2	cislo spritu, kam bude otoceny SP1 umisten (1-255)

Pozn.: Sprite SP2 by mel byt pred tim, nez jsou do nej ulozena data otaceneho spritu, vyprazdnen prikazem .CLSM.

V prikladu 23 je otocen sprite c.5 (tanecnik) do spritu c.1 a pak umisten na obrazovce. Sprite 1 (coz je nyni o 90 stupnu pootocena verze spritu 5) je pak otocen zpet do vyprazdneneho spritu c.5 cimz vznikne o 180 stupnu pootocena verze puvodniho spritu. Tento proces se opakuje.

```
 5 REM PRIKLAD 23
10 .ATOF:.COL=15:.ROW=10:.SPN=5:.PTBL
20 FOR N=50 TO 1 STEP -1
30 .SPN=1:.CLSM:.SP2=1:.SP1=5:.SPNM
40 .ROW=10:.COL=15:.HGT=1:.LEN=-1:.SP1=5:.SP2=1
50 .MOVE: PAUSE N
60 .SPN=5:.CLSM:.SP2=5:.SP1=1:.SPNM
70 .ROW=11:.COL=14:.HGT=-1:.LEN=1:.SP1=1:.SP2=5
80 .MOVE: PAUSE N
90 NEXT N
```

r.10 umisti sprite 5 na obrazovce
r.20 je smycka FOR-NEXT, ktera ridi prodlevu po dobu, po kterou je sprite na obrazovce
r.30 vyprazdni sprite 1 a otoci do nej sprite 5
r.40 nastavi parametry pro .MOVE
r.50 provede .MOVE s
r.60 vyprazdni sprite 5 a otoci do nej sprite 1
r.70 nastavi parametry pro .MOVE
r.80 provede .MOVE s pauzou

Prikaz	Vysledek
	presne dvojnasobne rozmery, nez SP2, jinak se prikaz neprovede. Zaroven jsou zvetseny pixely i atributy.

Parametr	Pouziti
SP1	cislo ciloveho spritu (1-255)
SP2	cislo zvetsovaneho spritu (1-255)

V prikladu 24 je sprite c.51 zvetsen do spritu c.32, ktery ma rozmery 2x2.

```
 5 REM PRIKLAD 24
10 .ROW=0:.COL=0:.SPN=51:.PTBL
20 .SP1=32:.SP2=51:.DSPM
30 .COL=2:.SPN=32:.PTBL
```

r.10 premisti sprite c.51 na obrazovku
r.20 zvetsi sprite c.51 do spritu c.32
r.30 premisti sprite c.32 na obrazovku

Ruzne prikazy

Prikaz	Vysledek
	ku, definovanem pomoci HGT, LEN, COL a ROW.

Parametr	Pouziti
COL	levy krajni sloupec okenka (0-31)
ROW	horni radek okenka (0-23)
HGT	vyska okenka (1-24)
LEN	sirka okenka (1-32)

V prikladu 25 je okenko o velikosti 5x5 naplneno na nahodne vybranych pozicich nahodne urcenymi atributy.

```
 5 REM PRIKLAD 25
10 .LEN=5:.HGT=5
20 .COL=INT(RND*28)
30 .ROW=INT(RND*18)
40 PAPER INT(RND*7)
45 BRIGHT INT(RND*1)
50 .SETV
60 GO TO 20
```

r.10 nastavi LEN a HGT
r.20 a 30 nastavi ROW a COL
r.40 vybere nahodne barvu PAPER
r.50 provede .SETV
r.60 uzavira smycku

L A S E R B A S I C C O M P I L E R

Chris Melling

Na uvod

Prekladac "Laser BASIC Compiler" je urcen uzivatelum, kteri napsali program v Laser BASICu a chteji vytvorit samostatny program, ktery by mohl byt spusten bez pritomnosti prekladace Laser BASIC. Programy prelozene timto prekladacem mohou byt sireny jakymkoliv zpusobem bez licenci a poplatku.

Prekladac muze byt pouzit k prekladani normalnich programu pro Spectrum do strojoveho kodu. Beh programu se tim vsak zrychli pouze dvojnasobne, nebo i mene. Vysledny strojovy kod zabere vzdy mene mista nez zdrojovy text v BASICu, ale je treba pocitat s naslednym pripojenim programu run-time ve strojovem kodu. Programy, ktere byly napsany za pomoci rozsireneho interpreteru "Laser BASIC" budou svym rozsahem vzdy vyhovovat pro preklad do strojoveho kodu pomoci "Laser BASIC Compileru".

Obsah kazet

	Nazev souboru:	Povel k prehrani:
Strana A:	"COMPCODE"	LOAD "" CODE
	"LOADER"	LOAD ""
	"RTCODE"	LOAD "RTCODE" CODE
Strana B:	"DEMO"	LOAD ""

Prekladani programu

Pri prekladu je nutne, aby v pameti pocitace byl pouze prekladac a prekladany program. Je tedy nutne zajistit, aby v pameti nebyl soucasne Laser BASIC. Byl-li vytvoren program za pomoci Laser BASICu, musime tento program nahrat na kazetu povelem SAVE, provedeme na pocitaci reset a program znovu nahrajeme do pameti.

Poznamka: Bez pritomnosti Laser BASICu nelze provadet zadne zmeny v programu, ktery obsahuje povely a klicova slova Laser BASICu.

Pred nahranim prekladace do pameti pocitace je treba snizit RAMTOP pomoci povelu

```
CLEAR 59799
```

Prekladac potom nahrajeme povely

```
LOAD "" CODE nebo
LOAD "COMPCODE" CODE
```

Jestlize program, ktery ma byt prelozen do strojoveho kodu neni

dosud v pameti, pak jej nyni nahrajeme beznym zpusobem.
Preklad potom zahajime povelem

```
RANDOMIZE USR 59800
```

V dobe, kdy probiha preklad, se na obrazovce objevuji zmatene cary a barvy. Nelekejte se, to pouze prekladac vyuziva obrazovku jako docasny pracovni prostor. Po ukonceni prekladu se obrazovka vycisti a objevi se znama zprava "OK" zaroven s cislem posledni radky a posledniho prikazu, ktere byly prelozeny.

Forma prelozeneho programu

V prubehu prekladu je vas program v BASICu prepisovan strojovym kodem. Po prelozeni je vysledny strojovy kod chapan pocitacem jako BASIC. Muze proto byt nahran a prehran povely SAVE a LOAD jako jakykoliv obycejny program v BASICu.

Nepokousejte se zmenit prelozeny program pripsanim novych radku, vypoustenim radku nebo jejich editaci. Nejde take spojit dva prelozene programy povelem "MERGE". Muzete vsak spojit prikazem "MERGE" dva neprelozene programy a pak prelozit celek.

Pokusite-li se po prekladu zadat "LIST", uvidite pouze nekolik radek BASICu. Prvni dva jsou vlozeny prekladacem nezavisle na puvodnim programu. Jako prvni se objevi radek

```
0 PRINT 0
```

Tento prikaz neni provaden, obsahuje vsak skrytou informaci, nutnou pro spravnou funkci podprogramu run-time. Druhy radek

```
1 RANDOMIZE USR 59800
```

zajisti automaticke rozbehnuti prelozeneho programu, jak o tom bude pojednano nize.
Obsahoval-li vas program uzivatelske funkce, uvidite jejich seznam ve tretim radku, cislovanem rovnez 0. Vsechny funkce budou smestnany do jednoho radku. To se ovsem nevstahuje na definice procedur Laser BASICu, ktere jsou prekladany samostatne.
Jestlize vas program obsahoval DATA, budou zahrnuta do dalsiho radku 0. Vsechny radky DATA z puvodniho programu budou spojeny do jednoho dlouheho radku, ale to se nijak nedotkne funkce prikazu RESTORE, ktere budou fungovat jako obvykle.

Spusteni prelozeneho programu

Prelozeny program zustava v pameti pocitace (nebo muze byt nahran v tomto okamziku ci pozdeji). Nyni musime nahrat podprogram run-time. Predpokladame, ze prikaz CLEAR 59799 stale plati (v pochybnostech jej nyni zopakujte). Podprogram nahrajeme

```
LOAD "RTCODE" CODE
```

V tomto stadiu musime nahrat i SPRITy stejnym zpusobem, jakym bychom je nahrali pro Laser BASIC (samozrejme jen tehdy, pouziva-li program SPRITy).
Jestlize dosud neni v pameti prelozeny program, nahrajeme jej nyni (presne stejne jako normalni program v BASICu).
Program je nyni pripraven k rozbehnuti. To zajistime prikazem

```
RANDOMIZE USR 59800 nebo
GOTO 1
```

s ohledem na radek 1, o kterem byla rec drive.
Program se nyni rozbehne, ale nemuzeme ho zastavit stiskem klavesy BREAK (s vyjimkou vystupu nebo vstupu na pasek).

Vytvoreni samostatneho programu

Prejete-li si publikovat program, ktery jste napsali v Laser BASICu, musite ho prelozit. Obvykle si budete prat vytvoreni samostatneho programu, pro jehoz nahrani i rozbehnuti postaci prikaz LOAD ""*
Toho dosahneta pouzitim zaradeciho programu (LOADERu), ktery se sam spusti, nahraje podprogram run-time, sprity a konecne prelozeny program, ktery muze byt rovnez spusten automaticky. Takovy zavadeci program, ktery muzete upravovat a rozsirovat, je obsazen na prvni strane kazety. Vypada takto:

```
10 CLEAR 59799
20 LET spst =
30 LOAD "RTCODE" CODE
40 LOAD "SPRITES" CODE spst
50 LET t=INT (spst/256): POKE 62464,spst-256*t: POKE 62465,t
60 POKE 62466,255: POKE 62467,220
70 POKE 56575,0
80 LOAD ""
```

r. 10 provede nastaveni RAMTOPu nutne pro podprogram run-time
r. 20 musite doplnit o hodnotu pocatku spritu, jak je popsano v generatoru spritu
r. 30 nahraje podprogramy run-time
r. 40 nahraje sprity
r. 50 vepise pocatek spritu
r. 60 vepise konec spritu viz manual Laser BASIC
r. 70 vepise konec charakteru spritu
r. 80 nahraje vysledny prelozeny program

Poznamka: Nejsou-li sprity pouzity, muzete vypustit radky 20, 40, 50, 60 a 70.

Zbyva vam tedy vyplnit spravnou hodnotu spst a nahrat na pasek ve spravnem poradi vsechny soubory. Pritom zajistite, aby zavadeci program i prelozeny program se automaticky rozbihaly.
U zavadeciho programu to znamena:

```
SAVE "loader" LINE 10
```

a pro prelozeny program:

```
SAVE "name" LINE 1
```

Priklad tohoto postupu je uveden dale s navodem vytvoreni A prelozene verze hry.

Omezeni tykajici se prekladaneho textu v BASICu

Pri vyvoji tohoto prekladace se muselo sahnout k jistym omezenim:

a) Nepouzivejte prikazy GO TO, GO SUB, RUN a RESTORE, ve kterych je parametr vypocitavan, protoze v dobe prekladu maji vsechny promenne hodnotu 0 a vsechny retezce delku 0.

b) Prekladac obsahuje prikazy DIM s vypocitavanymi parametry. Tyto prikazy, pokud obsahuji rozmery vyhodnocene jako 0 (ve standartnim Sinclair BASICu nesmyslna hodnota), nebudou provedeny az do rozbehnuti programu. To ovsem znamena, ze prikazy jako

```
DIM a (n+12)
```

nebudou provedeny v teto podobe, protoze pri prekladu bude rozmer vyhodnocen jako 12 a tak take bude pole deklarovano.

c) Pocet postupnych deklaraci pole neni omezen, ale zjistite, ze prvni deklarace je provedena v prubehu prekladu a proto neni obsazena v prelozenem programu. Je-li v programu skok pred misto, kde byl prvni prikaz DIM, pri druhem prubehu se prikaz neprovede. To lze obejit deklaraci pole hned na zacatku programu, pred vsemi skoky.

d) Nejsou povoleny prikazy CLEAR, vysledkem by bylo chybove hlaseni (viz nize).

e) Prikazy RUN funguji obdobne jako v BASICu, ale nemazou promenne.

f) Prikazy MERGE, CONTINUE, LIST a LLIST nejsou povoleny a vyvolavaji chybova hlaseni (viz nize).

g) Nepokousejte se pouzit pri zhrouceni nebo zastaveni programu prikaz CONTINUE.

h) OPEN# a CLOSE# funguji pouze ve vztahu k proudum (streams) "S", "K" a "P". Operace s microdrive nejsou zahrnuty.

i) Prikazy LOAD, SAVE a VERIFY funguji jako obvykle s vyjimkou, ze BASICovy program nahrany prikazem LOAD je povazovan za program prelozeny prekladacem a podprogram run-time se jej bude snazit rozbehnout od zacatku.

Pouzivani prekladace s microdrivem

Prekladac i podprogram run-time jsou pouzitelne na microdrivu. Operace s microdrivem vsak nejsou pripustne v ramci behu prelozeneho programu. Vsechny priklady uvedene v teto prirucce mohou byt pouzity na microdrivu pouze pridanim *"m";<d>; mezi prikaz a jmeno souboru.

Chybova hlaseni

Illegal statement found
Prekladac nalezl nektery z prikazu CLEAR, MERGE, LIST, LLIST nebo CONTINUE.

Procedure definition nesting error
Byly nalezeny dve procedury bez toho, ze by byly oddeleny prikazem .RETN. Procedury nelze vkladat do sebe.

Byl nalezen prikaz .RETN bez odpovidajici definice procedury.

Procedure not defined
Je nalezeno volani procedury, ktera nebyla definovana. Vsimnete si, ze nemusite deklarovat proceduru pred jejim pouzitim.

Program not compiled
Podprogram run-time zjistil, ze zacatek programu neodpovida zacatku programu, jaky by vytvoril prekladac.

Provadeni prelozeneho programu podleha stejnemu riziku vyskytu chyb, jako u jakehokoliv jineho programu v BASICu. Bohuzel vam vsak podprogram run-time nemuze zdelit, kde k chybe doslo. Meli byste take ignorovat cisla, uvadena v chybovych hlasenich Spectra. Meli byste proto program dukladne vyzkouset a odladit pred jeho prekladem.

Po vydani chyboveho hlaseni je program castecne nebo uplne prelozen, takze musite znovu nahrat puvodni program v BASICu a provest nutne zmeny pred novym pokusem o preklad.

Mapa pameti

Prekladac i podprogram run-time se oba nahravaji na adresu 59800. Podprogram run-time zabira pamet od teto adresy az ke konci pameti a po nahrani jej muzete opet nahrat na pasek prikazem:

```
SAVE "RTCODE" CODE 59800, 65536-59800
```

Prekladac konci na 62856, zabira 3057 bytu a muzete ho nahrat na pasek prikazem:

```
SAVE "COMPCODE" CODE 59800, 3057
```

Znamena to tedy, ze at pouzijete jakykoliv z techto programu, musite provest CLEAR 59799.
Prelozeny program z Laser BASICu zabira pamet takto:

Program	Variables \	\ SPRITy	m/c zasobnik	RTCODE
PROG	VARS	ELINE	spst spnd=56575	59800 65535

Poznamka pro programatory ve strojovem kodu

Jednoduchou aritmetikou zjistite, ze mezera mezi koncem spritu a podprogramem run-time je asi 3k nebo vice. Za normalnich okolnosti zaujima zasobnikova pamet mene nez 256 bytu, takze zbytek tohoto prostoru muze byt vyuzit pro programy ve strojovem kodu, bud tak, ze zacnete od vrcholu spritu, nebo provedete CLEAR pobliz spritu a vyuzijete prostor za timto bodem az k podprogramu run-time. Zrejme vam take nebude nic branit v provedeni CLEAR pod sprity a tak vyuzit celeho prostoru, atd.

Navod k vytvoreni samostatneho prelozeneho programu - hry "INVADER CUBE"

Na druhe strane druhe kazety Laser BASICu naleznete hru nazvanou INVADER CUBE. Podle tohoto navodu muzete vytvorit samostatnou verzi teto hry, ktera bude fungovat i bez Laser BASICu.

U puvodni hry jsou soubory na pasku serazeny takto:

- Zavadeci program (pro automaticky start)
- Obrazovka
- Tri programy obsahujici Laser BASIC
- Sprity pro hru
- Samotna hra (s automatickym startem).

Prelozena verze musi byt usporadana takto:

- Zavadeci program (pro automaticky start)
- Obrazovka
- Podprogram run-time
- Sprity
- Prelozena hra (s automatickym startem).

Abyste tuto verzi vytvorili, potrebujete prazdnou stranu kazety. Nejprve musite napsat zavadeci program. Pouzijeme zavadeci program dodavany jako soucast kompletu prekladace (viz vyse). Hodnotu spst musime nastavit na zacatek spritu pro hru, tj. 53780. Musime vlozit take novy radek, ktery zajisti nahrani obrazovky:

```
25 LOAD "scrn" CODE
```

Dale musime zabranit zniceni obrazovky v prubehu nahravani dalsich bloku a pridame tedy radek

```
5 INK 0: PAPER 0: BORDER 0: CLS
```

a vlozime prikazy PRINT AT 0,0; pred kazdy prikaz LOAD. Pri zmene posledniho radku (ktery nahraje vlastni hru) muzeme vlozit nazev hry, napr. "cgame":

```
80 PRINT AT 0,0;: LOAD "cgame"
```

Konecny vysledek tedy bude:

```
  5 INK 0: PAPER 0: BORDER 0: CLS
 10 CLEAR 59799
 20 LET spst=53780
 25 PRINT AT 0,0;: LOAD "scrn" CODE
 30 PRINT AT 0,0;: LOAD "RTCODE" CODE
 40 PRINT AT 0,0;: LOAD "SPRITES" CODE spst
 50 LET t=INT (spst/256): POKE 62464, spst-256*t: POKE 62465, t
 60 POKE 62466, 255: POKE 62467, 220
 70 POKE 56575, 0
 80 PRINT AT 0,0;: LOAD "cgame"
```

Tento zavadeci program nahrajte nyni na kazetu tak, aby se sam po nahrani do pocitace rozbehl:

```
SAVE "loader" LINE 5
```

Prohledneme-li si seznam souboru, jak maji za sebou nasledovat a prave dokonceny zavadeci program, je jasne, ze ted musime nahrat na pasek obrazovku. Tu ziskame z puvodni hry na kazete

Laser BASIC. Udelame to tak, ze napiseme prikaz

```
INK 0: PAPER 0: CLS
LOAD "SCREEN" CODE
```

Kdyz je obrazovka nahrana a objevi se zprava OK, napiseme:

```
SAVE "scrn" CODE 16384, 6872
```

a tim nahrajeme obrazovku na nas cisty pasek hned za novy zavadeci program. Po nahrani pasek zastavime a provedeme reset.

Po nahlednuti do seznamu souboru zjistime, ze nyni musime nahrat podprogramy run-time. Nahrajeme je tedy z kazety prekladace, ale nesmime zapomenout na CLEAR:

```
CLEAR 59799
LOAD "RTCODE" CODE
```

Ted nahrajem na kazetu za obrazovku prikazem

```
SAVE "RTCODE" CODE 59800, 65536-59800
```

a po nahrani magnetofon zastavime.

Podle seznamu souboru musime nyni nahrat sprity pro hru. Opet je ziskame z puvodni hry, kde jsou umisteny za tremi soubory strojoveho kodu, ktere obsahuji Laser BASIC. Prikaz

```
LOAD "SPRITES" CODE 53780
```

zajisti preskoceni techto tri souboru a nahrani spritu. Pak je umistime na kazetu za podprogram run-time prikazem

```
SAVE "SPRITES" CODE 53780, 56576-53780
```

Konecne musime prelozit samotnou hru a umistit ji na kazetu za sprity. Pcesvedcime se, ze pamet~ pocitace je prazdna a nahrajeme hru

```
LOAD "INVADER"
```

Hra se pak okamzite rozbehne, ale brzy se zhrouti, kdyz pocitac narazi na povel Laser BASICu, kteremu samotne Spectrum nerozumi. Jakmile k tomu dojde, nahrajte prekladac obvyklym zpusobem:

```
CLEAR 59799
LOAD "COMPCODE" CODE
```

Prelozte program prikazem

```
RANDOMIZE USR 59800
```

a nehrajte vysledek na kazetu

```
SAVE "cgame" LINE 1
```

tak, aby byl zajisten automaticky start.

Nyni muzete previnout pasek na zacatek, provest na pocitaci reset a napsat:

```
LOAD ""
```

Prelozena verze hry "Invader Cube" se nahraje a rozbehne.

Uzivatele microdrivu uvitaji, ze stejna procedura plati i pro toto medium. Zmenime pouze vsechny prikazy SAVE na SAVE *"m";1; a vsechny prikazy LOAD v zavadecim programu na LOAD *"m";1; s.