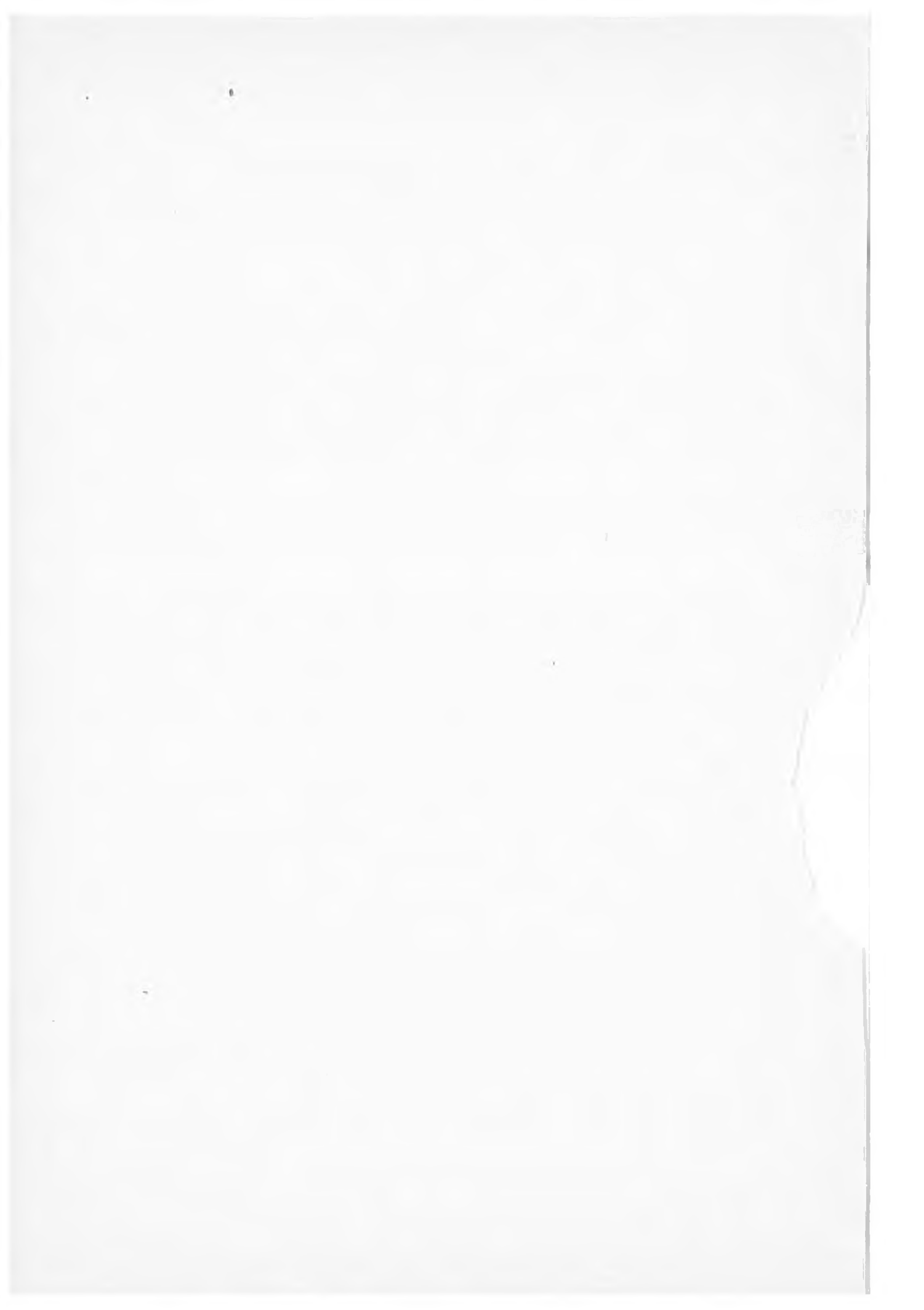


Vít Libovický

Grafický editor s elektronickou myší

Vstupní a výstupní zařízení mikropočítačů.

602.ZO Svazarmu Praha 6



Obsah

ÚVODNÍ SLOVO AUTORA	6
1. HARDWARE pro připojení myši	10
1.1 Konektor pro myš	10
1.2 Zapojení vstupní brány	12
1.3 Rozhraní pro ZX-Spectrum	13
1.4 Rozhraní pro PMD-85	13
1.5 Inicializace hardware	15
1.6 Testovací program v BASICu	16
1.7 Hledání chyby v zapojení myši	20
1.8 Mlžení displeje PMD-85 po připojení myši	22
1.9 Vypínač přerušování pro ZX-Spectrum	23
2. GREDITOR - interaktivní grafický editor	26
2.1 Na kterých počítačích pracuje GREDITOR	26
2.2 Načtení GREDITORu z kazety do PMD-85-1	26
2.3 Načtení GREDITORu z kazety do ZX-Spectra	27
2.4 Co je to interaktivní grafický editor s myší	28
2.5 Typy grafických editorů	29
2.6 Funkce GREDITORu	30
2.6.1 Práce s myší - pohyb	31
2.6.2 Práce s myší - tlačítka	32
2.6.3 Funkce GREDITORu	34
2.6.3.1 "změna módu" (36)	
2.6.3.2 "barva" - PMD-85 (43)	
2.6.3.3 "barva" - ZX-Spectrum (50)	
2.6.3.4 "bod" (58)	
2.6.3.5 "přímka" (61)	
2.6.3.6 "přímka 45°" (63)	
2.6.3.7 "rámeček" (67)	
2.6.3.8 "obdélník" (69)	
602. ZO Svazarmu - Grafický editor s elektronickou myší	1

2.6.3.9 "kružnice" (71)	
2.6.3.10 "text" (73)	
2.6.3.11 "vyplnění" (76)	
2.6.3.13 "přesun" a "kopie" (80)	
2.6.3.14 "výmaz" (86)	
2.6.3.15 "vrať" (87)	
2.6.3.16 "záznam" (89)	
2.6.4 Metodika práce s GREDITORem	94
2.7 Modularita, přenositelnost	99
2.8 Umístění GREDITORu v paměti	102
3. SOFTWARE PRO MYŠ	106
3.1 K čemu slouží myš	106
3.2 Generace souřadnic pomocí myši	107
3.2.1 Čekání na změnu	108
3.2.2 Vyhodnocení změny a posun souřadnic	108
3.2.3 Test tlačítek	110
3.2.4 Potlačení zákmitů tlačítek	111
3.2.5 Celkové vyhodnocení signálů z myši	113
3.2.6 Program pro kreslení	115
3.3 Test myši při přerušení	117
3.3.1 Uložení informací z myši do tabulky	118
3.3.2 Výběr informací o myši z tabulky	121
3.3.3 Čekání na informace z myši	124
3.3.4 Inicializace cyklického časového přerušení	124
3.4 Princip dynamického kreslení	128
3.5 Kurzor - tvary, algoritmy	133
3.5.1 Šipka - kurzor GREDITORu	134
3.5.2 Princip kreslení kurzoru	135
3.5.3 Výpočet polohy kurzoru	138
3.5.4 Jiný princip kreslení kurzoru	144
3.6 MENU - volba z nabídky	151
3.6.1 Postup při volbě z nabídky	151

3.6.2 Hlavní smyčka podprogramu pro volbu z nabídky	153
3.6.3 Úschůva a obnova pozadí pod textem nabídky	155
3.6.4 Určení řádky nabídky s kurzorem	158
3.7 Základní grafické algoritmy	160
3.7.1 Adresace displeje PMD-85	160
3.7.2 Adresace displeje ZX-Spectrum	162
3.7.3 Podprogramy pro adresaci displeje	163
3.7.4 Posun adresy displeje	167
3.7.5 Operace s bodem	172
3.8 Vrácení o krok	174
3.9 Hlavní smyčka GREDITORu	177
3.10 Příklad další aplikace elektronické myši . . .	181
Závěr	188

ÚVODNÍ SLOVO AUTORA

ÚVODNÍ SLOVO AUTORA

Vážený čtenáři!

V ruce držíte knihu, která je součástí stavebnice elektronické myši vyrobené 602.Z0 Svazarmu. V této knize bude řeč o třech různých oblastech, které ovšem všechny dohromady mají jedno společné - **elektronickou myš**.

1. HARDWARE pro připojení myši
2. GREDITOR - interaktivní grafický editor
3. SOFTWARE pro myš

Tomuto rozdělení odpovídá i členění knihy do kapitol. V **první kapitole** se dočtete o připojení sestavené myši k počítači a to konkrétně - na jaký konektor a kontakty. Další pohled je už z hlediska programátora: na které vstupní bráně lze číst informace z myši, který bit co znamená, jak je třeba inicializovat hardware počítače. A nakonec si vyzkoušíte jednoduchý program v BASICu, který Vám pomůže zkontrolovat správné připojení myši.

Ve **druhé kapitole** se dočtete o grafickém editoru, který máte nahrán na magnetofonové kazetě. Získáte informace o tom, co to je grafický editor, jak program načíst do paměti počítače a spustit a hlavně o možnostech tohoto editoru, o jeho funkcích a jejich ovládání.

Třetí kapitola je věnována zejména programátorům. Podrobně popisuje řešení problémů, které musí řešit programátor, který chce ve svém programu využít ovládání pomocí elektronické myši. Dozvíte se o zpracování informací z myši pomocí cyklického časového přerušení, o volbě z nabídky, o principu základních grafických operací, o kreslení kurzoru, dynamickém kreslení grafických objektů a dalších oblastech programování. Součástí textu jsou i příklady programů v jazyce symbolických adres (assembleru)

mikroprocesoru Z-80 řešící daný problém nebo využívající popsaný podprogram a také podprogramy použité v programu GREDITOR.

Několik slov o GREDITORu

GREDITOR je interaktivní grafický editor. Co to je a k čemu je takový program dobrý, to se dočtete v samostatné kapitole, zde se zmíním o okolnostech vzniku GREDITORu a jeho účelu.

Program GREDITOR vznikl v prosinci 1985 jako aplikační program pro demonstraci možností elektronické myši a jejího uplatnění u osmibitových mikropočítačů. Jedná se o poměrně jednoduchý program realizující základní funkce grafického editoru a je určen pro uvedení uživatele do problematiky využívání grafických editorů. Má sloužit k pochopení principu ovládnutí programu pomocí elektronické myši a možností grafického zobrazení Vašeho počítače. Přesto lze použít GREDITOR i k tvorbě komerčně využitelných grafických obrázků - např. ilustrace do textu, úvodní obrazovky programů apod.

Pro uživatele ZX-Spectrum, kteří znají program ARTSTUDIO a chtějí s ním srovnávat GREDITOR, dovolte malou poznámku. ARTSTUDIO jistě poskytuje uživateli více funkcí než GREDITOR. Ale jak již bylo řečeno dříve, nebylo účelem vytvořit dokonalý grafický editor, ale umožnit uvedení do dané problematiky. GREDITOR má délku 10 KB, což je pouze **třetinový** rozsah oproti ARTSTUDIU a vznikl koncem roku 1985, tedy v době, kdy ARTSTUDIO ještě neexistovalo.

Na závěr tohoto úvodu mi ještě dovolte pochlubit se tím, že program GREDITOR zvítězil v roce 1986 v celostátním kole SVOČ (Soutěž vysokoškolské odborné činnosti) a umístil se na 4. místě mezinárodního kola.

1. HARDWARE

pro připojení myši

2. GREDITOR

- grafický editor

3. SOFTWARE

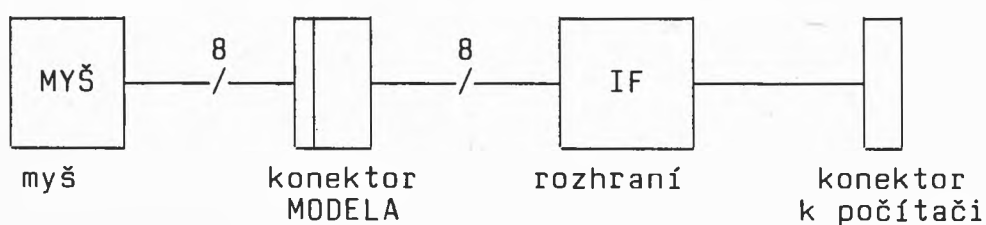
pro myš a editor

1. HARDWARE pro připojení myši

Tato kapitola pojednává o připojení sestavené myši k počítači, o čtení informací z myši přes vstupní bránu a testování správného připojení myši.

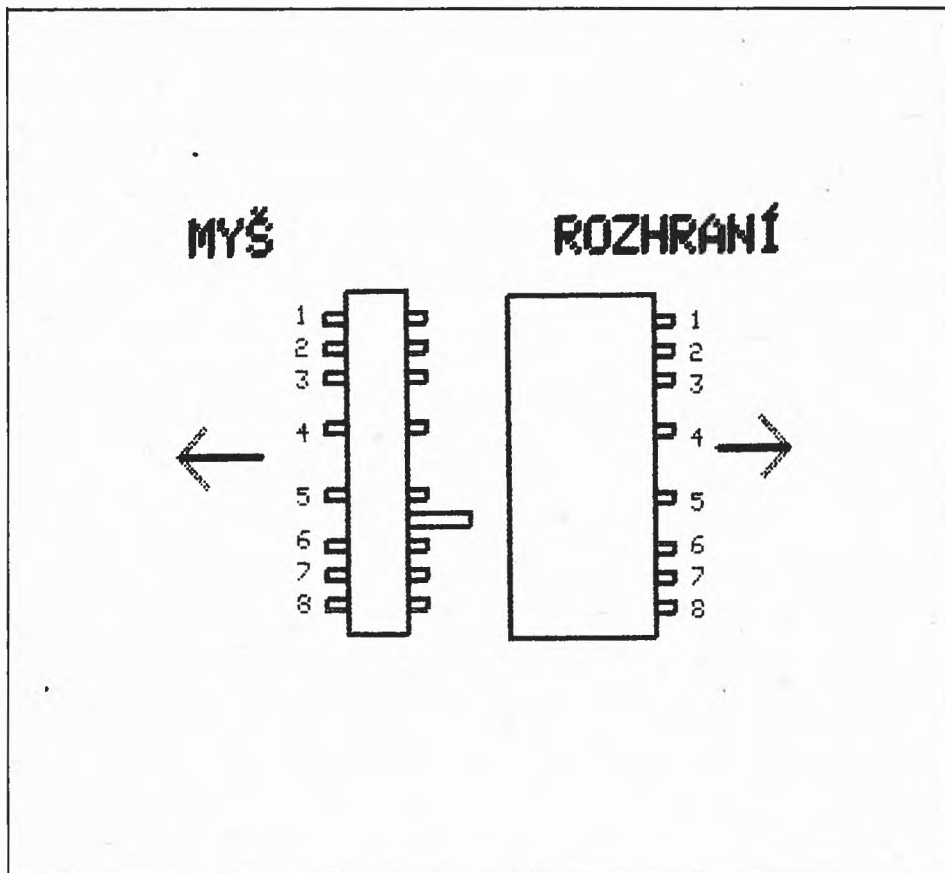
1.1 Konektor pro myš

Elektronická myš, kterou jste si sestavili ze stavebnice, je univerzální, to znamená, že je určena k libovolnému počítači, nikoli pouze k jednomu konkrétnímu typu. Teprve konektor pro připojení k počítači a případné rozhraní (interface) je závislé na použitém typu počítače. Pro možnost využití myši u více typů počítačů je výhodné, aby myš a rozhraní byly dva nezávislé díly, které se dohromady spojují pomocí konektoru. První díl - univerzální - pak je tvořen myší, kabelem a jedním dílem konektoru, druhý díl - závislý na typu počítače - je tvořen druhým dílem konektoru, případným rozhraním a konektorem pro připojení k počítači:



Pro propojení myši a rozhraní je třeba konektor s osmi kontakty. Nejsnáze dostupný pro amatéry je osmikolíkový konektor MODELA, který je i velice levný (10,50 Kčs). Tento konektor je běžně dostupný v modelářských prodejnách. Použití tohoto typu konektoru není podmínkou, ale vzhledem ke kompatibilitě s ostatními vlastníky myši z této stavebnice (a těch je 4999!) Vám jeho použití vřele
10 602. ZO Svazarmu - Grafický editor s elektronickou myší

doporučuji. K myši patří díl s noži (samec) a k rozhraní díl s otvory (samice), číslování kontaktů:



Zapojení kontaktů konektoru MODELA je toto:

- 1 +5V - napájení
- 2 PT - pravé tlačítko
- 3 LT - levé tlačítko
- 4 XA - posun v X, fáze 1
- 5 XB - posun v X, fáze 2
- 6 YA - posun v Y, fáze 1
- 7 YB - posun v Y, fáze 2
- 8 0V - napájecí zem

1.2 Zapojení vstupní brány

Šest signálů z myši je přivedeno na vstupní bránu počítače. Jedná se o dva signály od tlačítek myši a po dvou signálech pro každou osu pohybu myši. Tyto informace můžeme načíst ze vstupní brány v BASICu pomocí funkce INP, v assembleru instrukcí IN A,(brána). Jednou instrukcí načteme stav celé vstupní brány, tedy 8 vodičů - bitů informace. 6 bitů odpovídá šesti výše popsaným signálům z myši, 2 bity nemají žádný význam. Přiřazení signálů z myši jednotlivým bitům je toto:

- bit 7 - PT pravé tlačítko
- bit 6 - LT levé tlačítko
- bit 5 - nemá význam
- bit 4 - nemá význam
- bit 3 - XA - posun v X, fáze 1
- bit 2 - XB - posun v X, fáze 2
- bit 1 - YA - posun v Y, fáze 1
- bit 0 - YB - posun v Y, fáze 2

Logika zapojení tlačítek je taková, že při stisknutém tlačítku má příslušný bit hodnotu log. "1", při puštění log. "0".

Význam signálů XA, XB, YA a YB je podrobně popsán v knize o periferních zařízeních, zde proto jen stručně uvedu správné fázování při pohybu myši. V levém sloupci je druh pohybu a v pravém sloupci hodnoty příslušných bitů.

pohyb	hodnota bitů
dolu	0, 1, 3, 2, 0 ...
nahoru	0, 2, 3, 1, 0 ...
vpravo	0, 4, 12, 8, 0 ...

vlevo 0, 8, 12, 4, 0 ...

Ten, komu to není jasné, si nemusí dělat starosti, přehození signálů XA s XB a YA s YB se jednoznačně projeví pohybem kurzoru na opačnou stranu než se pohybuje myš. V tomto případě stačí přehodit příslušné vodiče na konektoru MODELA u myši (samec) a je po problému.

1.3 Rozhraní pro ZX-Spectrum

Protože počítač ZX-Spectrum má na konektor vyvedenu pouze systémovou sběrnici a žádný paralelní port není k dispozici, je nutné pro připojení myši k počítači mít vyrobeno rozhraní. Toto rozhraní je popsáno v knize o periferních zařízeních, kam čtenáře odkazují. Pouze upozorňuji na nutnost zachování přiřazení signálů z myši jednotlivým bitům tak, jak to bylo dříve popsáno.

1.4 Rozhraní pro PMD-85

U počítače PMD-85 je k dispozici vstupní brána vyvedená na tzv. aplikační konektor. Rozhraní mezi myší a počítačem může být díky tomu velice jednoduché - je tvořeno pouze dvěma propojenými konektory. Na jedné straně je konektor MODELA (samice, zapojení už bylo popsáno dříve) a na druhé straně konektor FRB - samec s 30 kontakty. Do PMD-85 se tento konektor zapojuje do konektoru, který je při pohledu zepředu úplně vlevo. V následující tabulce je uvedeno propojení obou konektorů:

samice MODELA-8	samec FRB-30	signál

1	1	+5V

2	10	pravé tlačítko
3	12	levé tlačítko
4	8	XA
5	6	XB
6	3	YA
7	5	YB
8	29	OV

Upozornění:

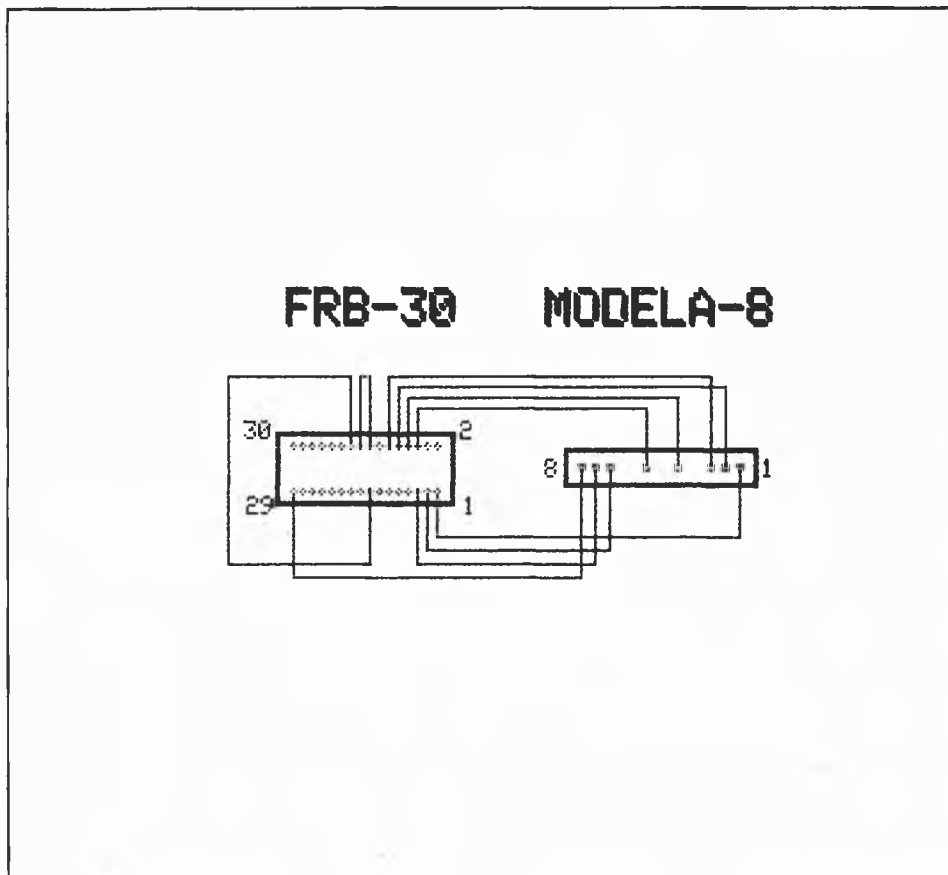
Čísla pinů v prostředním sloupci odpovídají značení na volném konektoru FRB. Značení konektoru (samice) v PMD-85 je většinou stranově převrácené! Při pohledu ze zadu na konektor v PMD-85, je pin číslo 1 vlevo nahoře.

Na konektoru FRB je dále nutné propojit dvě dvojice pinů:

- 1) pin 15 - pin 20
- 2) pin 16 - pin 18

Tyto propojky umožňují využití vnitřního časovače 8253 jako generátoru časového přerušení.

Na obrázku je pohled na konektor FRB-30 a MODELA-8 ze zadu, ze strany vodičů:



1.5 Inicializace hardware

Před čtením informací z myši bývá potřebné inicializovat hardware rozhraní a počítače.

Tato inicializace u ZX-Spectrum závisí na použitém rozhraní, hardware počítače není třeba inicializovat.

U PMD-85 před čtením informací z myši není třeba inicializovat ani počítač ani rozhraní. Pro využití časového přerušení je třeba inicializovat časovač 8253, ale to bude popsáno v příslušné části textu.

1.6 Testovací program v BASICu

Pro další etapu práce na oživení myši předpokládám, že již máte sestavenou celou myš, přiletovány potřebné konektory a máte i realizováno rozhraní pro připojení myši k počítači.

Takže jste zasunuli konektor myši do rozhraní, rozhraní do počítače, počítač jste zapnuli a čekáte na první projevy života Vaší myši. Jistě jste zvědaví, zda jste vše sestavili a proletovali správně, zda se někde při práci nevloudila chybička.

Pro další postup použijeme programovací jazyk BASIC, ve kterém si naprogramujeme jednoduchý testovací program.

Vložte do svého počítače tento program:

Testovací program pro PMD-85

```
100 GCLEAR :REM smazat displej
110 SCALE 0,255,0,255 :REM nastavit měřítko
120 DIM T(15) :REM deklarovat pole
130 X=128: Y=128 :REM počáteční souřadnice
140 FOR I=0 TO 15 :REM načíst data do pole
    T(i)
150 READ T(I)
160 NEXT
170 I=0 :REM nulovat minulý stav
180 PLOT X,Y,1 :REM rozsvítit bod na
    displeji
190 F=4 :REM volba směru
200 B=INP(140) :REM načtení stavu myši
210 J=(B/F) AND 3 :REM ponechat pouze bity
    směru
220 R=X :REM uschovat minulou
    souřadnici
230 X=X+T(I*4+J) :REM změna souřadnice X
240 I=J :REM uschovat nový stav
```

```

250 IF X=R THEN 200           :REM byla změna? NE = do
                               cyklu
260 PLOT R,Y,1                :REM smazat minulý bod
270 PLOT X,Y,1                :REM rozsvítit nový bod
280 GOTO 200                  :REM opět do cyklu
290 DATA 0,1,-1,0, -1,0,0,1 :REM data pro inicializaci
                               pole
300 DATA 1,0,0,-1, 0,-1,1,0

```

Testovací program pro ZX-Spectrum

```

110 CLS                       :REM smazat displej
120 DIM t(15)                 :REM deklarovat pole
130 LET x=128                  :REM počáteční souřadnice X
140 LET y=96                   :REM a Y
150 FOR i=0 TO 15             :REM načíst data do pole
160 READ t(i)
170 NEXT i
180 LET i=0                    :REM nulovat minulý stav
190 LET f=4                    :REM volba směru
200 LET b=INP 0                :REM načtení stavu myši
210 LET j=INT (b/f)           :REM ponechat pouze bity
                               směru
220 LET j=j-4* INT (j/4)
230 LET r=x                    :REM uschovat minulou
                               souřadnici
240 LET x=x+t(i*4+j)          :REM změna souřadnice X
250 LET i=j                    :REM uschovat novou
                               souřadnici
260 IF x=r THEN 200           :REM byla změna? NE = do
                               cyklu
270 PLOT r,y                  :REM smazat minulý bod
280 PLOT x,y                  :REM rozsvítit nový bod
290 GOTO 200                  :REM opět do cyklu

```

```
300 DATA 0,1,-1,0, -1,0,0,1 :REM data pro inicializaci
      pole
310 DATA 1,0,0,-1, 0,-1,1,0
```

Pokud u ZX-Spectrum je nutné inicializovat rozhraní, učiňte tak.

Po spuštění testovacího programu se smaže obrazovka a uprostřed se vykreslí bod. Zkuste nyní velice pomalu pohnout myš doprava. Pokud je myš bezchybně připojena, bod na obrazovce se bude pohybovat rovněž doprava. Při pohybu myši doleva se bude pohybovat bod doleva. Je tomu tak? Jestliže ano, gratuluji! Vaše myš žije!

Myš je nutno pohybovat opravdu velice pomalu, BASIC je velmi pomalý a rychlejší pohyb myši nestihne zpracovat. Při rychlejším pohybu myši bod na obrazovce "tančí" skoro na místě.

Nyní pro ty, co byli méně šťastní. Je možné, že se bod pohybuje v opačném směru než posunujete myš. Jak již bylo dříve řečeno, to není žádná tragedie, stačí přehodit dráty na pinech 4 a 5 konektoru MODELA u myši a vše bude v pořádku.

Je-li vše v pořádku, vyzkoušíme pohyb v předozadní ose. K tomuto účelu změňte jeden řádek programu takto:

PMD-85
190 F=1

ZX-Spectrum
190 LET f=1

Další postup je obdobný jako u pravo-levé osy, pouze pohyb myši změníme. Pohybujeme-li jí směrem nahoru (od sebe) - bod se musí pohybovat doleva, při pohybu směrem dolů (k sobě) - bod míří doprava. Opačný směr pohybu vyřešíme přehozením drátů na pinech 6 a 7 konektoru MODELA u myši.

Nyní víme, že programové sledování pohybu myši probíhá bez závad, myš i rozhraní je v pořádku. Zbývá nám ještě zkontrolovat funkci tlačítek na myši. Smažte první testovací program a zadejte do počítače tento:

PMD-85

```
100 A=0 :REM nulovat minulý stav
110 B=INP(140)/64 AND 3 :REM načtení stavu myši
120 C=A*4+B :REM sečíst minulý a nový
stav
130 A=B :REM uschovat nový stav
140 IF C=1 THEN PRINT"STISKNUT LEVY"
150 IF C=2 THEN PRINT"STISKNUT PRAVY"
160 IF C=4 THEN PRINT"PUSTEN LEVY"
170 IF C=8 THEN PRINT"PUSTEN PRAVY"
180 GOTO 110 :REM opět do cyklu
```

ZX-Spectrum

```
100 LET a=0 :REM nulovat minul stav
110 LET b=(( INP 0)/64) :REM načíst stav myši
120 LET b=b-4*INT (b/4) :REM ponechat pouze bity
tlačítek
130 LET c=a*4+b :REM sečíst minulý a nový
stav
140 LET a=b :REM uschovat nový stav
150 IF c=1 THEN PRINT "STISKNUT LEVY"
160 IF c=2 THEN PRINT "STISKNUT PRAVY"
170 IF c=4 THEN PRINT "PUSTEN LEVY"
180 IF c=8 THEN PRINT "PUSTEN PRAVY"
190 GOTO 110 :REM opět do cyklu
```

Spusťte program a pak zkuste stisknout levé tlačítko na myši a po chvíli ho pustit. Totéž pak proveďte i s pravým tlačítkem. Je-li vše v pořádku, vypíše se na obrazovku:

```
STISKNUTO LEVE  
PUSTENO LEVE  
STISKNUTO PRAVE  
PUSTENO PRAVE
```

Pokud se vypíše opačně PRAVE = LEVE, stačí na konektoru MODELA přehodit signály od tlačítek, tedy dráty na pinech 2 a 3.

Bylo při testech vše v pořádku? Věřím, že vaše myš ožila napoprvé, případně že jste drobné chybičky snadno a rychle opravili. To znamená, že jste zdárně zvládli i poslední úskalí stavby myši - připojení k počítači. Jako odměna za vynaložené úsilí na vás čeká na kazetě nahraný grafický editor GREDITOR. Neváhejte a přistupte k jeho nahrání a spuštění, tak jak je to popsáno v příslušné kapitole!

1.7 Hledání chyby v zapojení myši

Nejhorší případ při programovém testování myši pomocí výše popsaného programu v BASICu nastane tehdy, když při žádném pohybu myši se bod na obrazovce ani nepohne nebo naopak "rejdí" sám, bez pohybu myši. To jste se někde dopustili chyby při zapojování elektrické části myši nebo u ZX-Spectrum také třeba při zapojování rozhraní. Nemůžu Vám přesně popsat postup, ten se bude lišit podle druhu závady, ale základní postupy při hledání závady jsou tyto:

1. Zkontrolovat správnost programu v paměti.

2. Zkontrolovat správnost zapojení všech konektorů.
3. Zkontrolovat průchodnost rozhraní.
4. Zkontrolovat vnitřní elektrické zapojení myši.

Průchodnost rozhraní nejlépe zkontrolujeme pomocí kusu drátu a jednoduchého programu:

PMD-85

```
100 PRINT INP(140) AND 207;  
110 GOTO 100
```

ZX-Spectrum

```
100 PRINT INP 0,  
110 GOTO 100
```

K počítači necháme připojeno rozhraní, ale odepneme od něho myš. Zadáme uvedený prográmeček a spustíme ho. Na obrazovku se začne vypisovat řada shodných čísel. Nyní vezmeme kus drátu a na konektoru MODELA budeme zkoušet propojovat kontakty, ovšem ne leccjaké! Jeden konec zasuneme do otvoru 1 (+5V) a druhý do otvoru 7 (YB). Zapamatujeme si číslo, které se po propojení vypisuje na displej. Přemístíme konec z 1 do 8 (0V). Srovnáme číslo, které se vypisuje nyní, s původním. Předchozí číslo by mělo být o 1 větší. Pokud se nevypisují shodná čísla, mělo by alespoň první číslo být liché a druhé sudé. Není-li tomu tak, je chyba v rozhraní. Pokud sami nemáte dostatek zkušeností s ožíváním elektronických zařízení, poraďte se s odborníkem.

Obdobným způsobem můžeme zkontrolovat průchodnost ostatních signálů. Jeden konec drátu zasouváme postupně do otvorů 7, 6, 5, 4, 3, a 2 (datové bity), druhý konec střídavě zasouváme do otvorů 1 a 8 (+5V a 0V). Tím na datové signály střídavě přivádíme hodnotu log. "1" a log. "0".

Přitom sledujeme vypisovaná čísla. Rozdíl obou zjištěných hodnot musí odpovídat této tabulce:

konektor MODELA	rozdíl hodnot	datový bit
7	1	0
6	2	1
5	4	2
4	8	3
3	64	6
2	128	7

Odpovídají-li rozdíly při kontrole všech šesti signálů této tabulce, je rozhraní průchodné a chyba je v myši. V opačném případě je chyba v rozhraní.

1.8 Mlžení displeje PMD-85 po připojení myši

Po připojení rozhraní s myší k mikropočítači PMD-85 a jeho zapnutí se může stát, že při pohybu myši po obrazovce přebíhají bílé vodorovné čárky. Tento jev je způsoben nedokonalým ošetřením signálu IOR uvnitř počítače. Jedná se sice pouze o kosmetickou závadu, ale mlžení displeje může být při delším sledování nepříjemné. Závadu lze bezesbytku odstranit přiletováním jednoho odporu dovnitř počítače.

Odpor by měl mít hodnotu v rozmezí 2-5000 ohmů, např. 2k7, 3k3, 4k7. Jedním koncem přijde přiletovat na pin 14 (první shora vpravo - u vykousnutí) obvodu B7 (7403) a druhým na pin 5 (pátý shora vlevo) obvodu C7 (8255A). Signál IOR se přes odpor přivede na +5V, čímž je definována jeho klidová hodnota na log. "1", a mlžení přestane.

1.9 Vypínač přerušení pro ZX-Spectrum

Pro ošetření signálů z myši je nutné použití cyklického časového přerušení. Vnitřní přerušení ZX-Spectrum má frekvenci 50 Hz, což je nepoužitelně pomalé. Součástí rozhraní pro připojení myši k ZX-Spectrum proto musí být astabilní klopný obvod, který generuje přerušení s frekvencí 1-4 kHz. Zapnutí tohoto přerušení ale má jeden vedlejší efekt. Test klávesnice ZX-Spectrum a opakování kláves při podržení (autorepeat) je řízeno přerušením. Změnou frekvence přerušení z 50 na 1000 Hz se celý proces mnohonásobně zrychlí. Výsledkem je to, že při stisknutí klávesy se příslušný znak nevypíše jednou, ale mnohokrát. Tak je znemožněno normální psaní na klávesnici. Z uvedeného důvodu je nutné přivést "rychlé" přerušení na sběrnici ZX-Spectrum přes vypínač. Ze začátku je přerušení vypnuto, takže lze normálně zadat načtení programu z kazety a jeho spuštění. Teprve potom zapneme vypínač - zapneme rychlé přerušení, se kterým si už příslušný program - GREDITOR - poradí.

Když nezapnete rychlé přerušení a spustíte GREDITOR, bude správně reagovat pouze na velice pomalý pohyb myši. Při rychlejším pohybu nenastane očekávaný pohyb - kurzor zůstane na místě. Je to obdoba efektu jako při běhu testovacího programu v BASICu při ožívání myši.

1. HARDWARE

pro připojení myši

2. GREDITOR

- grafický editor

3. SOFTWARE

pro myš a editor

2. GREDITOR - interaktivní grafický editor

V této kapitole se dozvíte, co to je grafický editor a k čemu slouží. Dále se dozvíte, jak načíst a spustit program, který máte nahrán na kazetě - GREDITOR, a hlavně co tento program umí a jakým způsobem ho můžete ovládat. A nakonec ještě získáte trochu informací o složení GREDITORu a jeho uložení v paměti.

2.1 Na kterých počítačích pracuje GREDITOR

Na kazetě, která je součástí stavebnice, jsou nahrávky GREDITORu pro dva počítače:

1. PMD-85
2. ZX-Spectrum

Byly zvoleny právě tyto dva typy mikropočítačů, neboť jsou v Československu nejrozšířenější.

Program GREDITOR lze implementovat na libovolný mikropočítač s mikroprocesorem 8080 nebo Z-80 a bodovou grafikou. Existuje verze pro SAPI-1 s grafickou kartou, perspektivní jsou i IQ-151 s grafickou kartou, SHARP, SORD, PP-01. Pokud se najde dostatečný počet zájemců, může být vytvořena verze i pro tyto počítače.

2.2 Načtení GREDITORu z kazety do PMD-85-1

Načtení a spuštění GREDITORu je velice jednoduché. Postup je následující:

1. Připneme k PMD-85 myš.

2. Zapneme PMD-85.
3. Vložíme do magnetofonu kazetu přetočenou před začátek verze pro PMD-85.
4. Zadáme monitoru příkaz MGLD 01.
5. Stiskneme na magnetofonu tlačítko pro přehrávání.
6. Po nalezení hlavičky se v dialogové řádce vypíše text: 01/? GREDIT 1.6.
7. Po 2.5 sec. se na obrazovce začne kreslit obrázek. Nezastavujte magnetofon - nahrávání pokračuje!
8. Po načtení (trvá 110 sekund) se GREDITOR automaticky spustí a vykreslí se úvodní obrázek GREDITORu.
9. Zastavíme magnetofon.
10. GREDITOR je připraven k použití - uchopte myš a můžete začít!

2.3 Načtení GREDITORu z kazety do ZX-Spectra

Proces načtení a spuštění je obdobný jako v případě PMD-85. Postup je následující:

1. Připneme k ZX-Spectru rozhraní s myší.
2. Vypneme vypínačem "rychlé" přerušování.
3. Zapneme mikropočítač.
4. Vložíme do magnetofonu kazetu přetočenou před začátek verze pro ZX-Spectrum.
5. Zadáme BASICu příkaz LOAD "".
6. Stiskneme na magnetofonu tlačítko pro přehrávání.
7. Po nalezení hlavičky se vypíše název programu.
8. Nejdříve se načte úvodní obrázek.
9. Během nahrávání hlavního programu se na obrazovce vypisuje text. Nezastavujte magnetofon - nahrávání pokračuje!
10. Po načtení se GREDITOR automaticky spustí a vykreslí se úvodní obrázek GREDITORu.

11. Zastavíme magnetofon.
12. Zapneme "rychlé" přerušení.
13. GREDITOR je připraven k použití - uchopte myš a můžete začít!

Pro oba typy počítačů platí, že pokud dojde při nahrávání k chybě, je třeba přetočit kazetu na začátek nahrávky a zkusit načtení znovu.

U PMD-85-1 ještě může dojít při čtení k problému s fází magnetofonu. Nahrávka byla pořízena na magnetofonu SP-210. Bez problémů by měla jít načíst na magnetofonech typů SM-260, SP-210, M-710, GERACORD a dalších, které mají shodnou fázi. Naopak načíst nejde z magnetofonu K10, který má fázi opačnou!

Pro PMD-85-2 je nutná odlišná nahrávka než pro PMD-85-1 z důvodu jejich vzájemné nekompatibility. Proces načtení GREDITORu do PMD-85-2 je obdobný jako byl popsán pro PMD-85-1, pouze při čtení se na obrazovce nic nekreslí (odpadá bod 7 postupu pro nahrávání).

2.4 Co je to interaktivní grafický editor s myší

Jednoduše řečeno, grafický editor je program, který uživateli umožňuje tvorbu grafických obrázků na displeji počítače, tak jak by to dělal na papíře s pomocí barevných tužek, pravítka, kružítka a gumy. Hlavní výhody použití počítače k této činnosti jsou:

- rychlost a přesnost práce
- možnost zrušení posledního kroku
- snadnost opravy chybných tvarů
- neomezený počet "gumování" na jednom místě
- možnost zopakování shodných částí bez překreslování

- automatické vybarvení uzavřené oblasti
- nové možnosti grafického projevu, např. inverze, posunutí či přebarvení
- úschova obrázků na počítačovém paměťovém médiu - možnost vícenásobného použití a dalšího počítačového zpracování

Slovíčko **interaktivní** znamená, že kreslení probíhá v reálném čase. Zvolená funkce se ihned provede a na obrazovce se zobrazí aktuální stav obrázku. Uživatel vidí, co svým zásahem způsobil a může třeba poslední krok vrátit a zkusit něco trochu jiného, když to není ono; anebo je s výsledkem spokojen a pokračuje v práci dál.

Dovětek **s myší** je dán tím, že veškeré funkce grafického editoru se ovládají pomocí elektronické myši. Klávesnice počítače se využívá pouze k zápisu textů a názvu souboru při práci s magnetofonem. Tak je umožněn i takový zdánlivý paradox, že můžete několik hodin pracovat s počítačem a přitom se nedotknout klávesnice! Možná Vám to teď připadá divné, ale až si to sami zkusíte a okusíte tu jednoduchost, rychlost a pohodlí ovládání programu pomocí myši, určitě nedáte na myš dopustit a stanete se dalšími jejími propagátory, tak jako já. A jistě se budete snažit ji používat i pro ovládání svých vlastních programů.

2.5 Typy grafických editorů

Existují dva základní typy grafických editorů:

1. bitově orientovaný,
2. objektově orientovaný.

Bitově orientovaný grafický editor pracuje s obrázkem v takovém stavu, v jakém je na obrazovce, tzn. pracuje s

bitovou mapou displeje. Zvolená funkce provede příslušnou změnu v bitmapě a zapomene se na ni. Nepamatuje se žádná informace o tom, jak obraz vznikl.

Objektově orientovaný grafický editor pracuje s grafickými objekty (bod, úsečka, kružnice, písmeno atd.), nikoli s bitovou mapou. V paměti jsou uloženy informace o rozmístění grafických objektů na obrazovce.

Oba popsané přístupy se liší účelem a možnostmi. Výstupem obou může být obrázek, který lze vytisknout na tiskárně, vložit do textu pomocí textového editoru nebo použít jinak, např. jako úvodní obrazovku programu.

Výstupem objektově orientovaného grafického editoru může být i tabulka objektů, podle které se dá obrázek nakreslit, a to třeba i v jiné velikosti nebo pootočený. Tabulku s příslušným interpretem lze použít v programech pro kresbu ilustračních obrázků. Další výhodou je většinou podstatně menší délka příslušné tabulky oproti bitové mapě a možnost přímého vykreslení na souřadnicovém zapisovači.

Grafický editor objektového typu je ale podstatně složitější než bitově orientovaný a vyžaduje větší kapacitu paměti a vyšší rychlost procesoru. Z těchto důvodů byl jako aplikační program pro stavebnici myši vytvořen bitově orientovaný grafický editor - **GREDITOR**.

Je nesporné, že objektově orientovaný grafický editor má podstatně vyšší užitnou hodnotu a proto je zvažována jeho tvorba - pokud bude dostatek zájemců.

2.6 Funkce GREDITORu

V následujícím textu se dozvíte o konkrétních funkcích GREDITORu a jeho ovládání pomocí volby z nabídky.

2.6.1 Práce s myší - pohyb

Předmětem přímého řízení pomocí elektronické myši je tzv. **aktuální bod**. Je to bod na obrazovce, kterým pohybujeme pomocí myši. Poloha tohoto bodu je zviditelněna pomocí **kurzoru**. V případě GREDITORu je kurzor představován šipkou, která směřuje do aktuálního bodu. Směr této šipky se mění tak, aby vždy směřovala k nejbližšímu okraji. Tím je umožněn pohyb aktuálního bodu po celé obrazovce až k jejím okrajům.

Při pohybu myši po podložce se ve stejném směru a stejnou rychlostí pohybuje i kurzor na obrazovce. Pohybu myši vpravo a vlevo logicky odpovídá pohyb kurzoru doprava a doleva. Pohybu myši dozadu (od sebe) a dopředu (k sobě) odpovídá pohyb kurzoru nahoru a dolů.

Myš je nutné držet tak, aby strana s tlačítky směřovala od nás - dozadu - a přední strana byla rovnoběžná s přední hranou podložky. Pro mačkání tlačítek je nejlépe používat ukazovák a prostředník, které máme na tlačítkách stále lehce položeny.

Je třeba si také uvědomit, že kurzor se pohybuje pouze při posunu myši po podložce, nikoli při jejím přemístění vzduchem.

Důležitá je i volba podložky, po které budete s myší jezdit. Je nutné, aby povrch byl spíše hrubý, aby po něm ping-pongový míček neklouzal. Nevhodnými povrchy jsou například sklo a lakované dřevo. Lze použít třeba sešit s tuhými deskami, ale naprosto nejlepším řešením je použití obdélníku z linolea (PVC) obráceném stranou bez vzorku (rubem) nahoru. Tuto podložku používám již několik let ke své naprosté spokojenosti. Pro pohyb kurzoru po celé obrazovce stačí obdélník velikosti sešitu A4 naležato.

Rychlost pohybu myši po podložce není prakticky omezena. Pokud GREDITOR nestačí kreslit touto rychlostí (to je možné pouze u funkce "bod"), pohyb myši si zapamatuje (až

100 bodů) a při zpomalení či zastavení funkci správně dokončí.

Co se týče přesnosti myši, s trochou cviku lze nastavit kurzor s přesností na jeden bod!

2.6.2 Práce s myší - tlačítka

Na Vaší myši jsou dvě tlačítka. Pro potřebu ovládní GREDITORu budeme levé nazývat **SELECT** a pravé **MENU**.

Pomocí tlačítka **MENU** si na obrazovku vyvoláte nabídku možných funkcí, ze kterých si při práci budete vybírat.

Tlačítko **SELECT** slouží k vyvolání předem zvolené funkce.

Postup při volbě funkce z nabídky je tento:

1. Stisknete a podržíte tlačítko **MENU**. Na obrazovku se v blízkosti kurzoru (šipky) vypíše tabulka s texty funkcí, ze kterých můžete volit. Nebojte se o svůj obrázek, přes který se tabulka vypsala, po ukončení výběru z nabídky bude obnoven.
2. Stále držíte **MENU** a jezdíte šipkou po nabídce. Řádek s popisem funkce, na kterou v daném okamžiku ukazuje šipka, se vypisuje inverzně.
3. Po najetí šipkou na funkci, kterou chcete zvolit, pustíte tlačítko **MENU**. Nabídka zmizí a GREDITOR se přepne na zvolenou funkci. Pokud pustíte tlačítko **MENU** v okamžiku, kdy je šipka mimo text nabídky, nic se nestane - volba se neprovede.
4. Pokud jste zvolili funkci, kterou je třeba ještě přesněji definovat, vypíše se nová nabídka a ve volbě pokračujete opět od bodu 2.

Při volbě grafických funkcí (BOD, ÚSEČKA...) se nic neprovede, GREDITOR se pouze přepne do zvoleného módu a příslušná funkce se vyvolá až pomocí tlačítka SELECT, jak bude o kousek dál popsáno.

Některé funkce, které není třeba blíže specifikovat a nemá smysl je vyvolávat vícekrát, se provedou ihned po vybrání z nabídky pomocí tlačítka MENU. Jsou to například funkce "výmaz", "vrať".

Postup při vyvolání zvolené funkce:

1. Najedete kurzorem na počáteční bod, stisknete a podržíte tlačítko SELECT.
2. Stále držíte SELECT a jedete kurzorem do koncového bodu. Na obrazovce se přitom dynamicky kreslí daný objekt.
3. Po najetí do cílového bodu pustíte SELECT - nakreslí se grafický objekt nebo se provede zvolená funkce.
4. U funkcí "přesun" a "kopie" je třeba ještě pokračovat dál. Předchozím postupem jste určili obdélníkovou zdrojovou oblast, která se na displeji znázorní rámečkem. Shodně veliký rámeček se nyní pohybuje spolu s kurzorem. Najedete s ním na cílovou oblast a po krátkém stisknutí tlačítka SELECT se provede zvolená funkce.

Chcete-li vykreslit další objekt zvoleného typu, pokračujete znovu od bodu 1. Tímto způsobem nakreslíte žádaný počet grafických objektů (třeba úseček), pak přes MENU zvolíte další typ a pokračujete v práci.

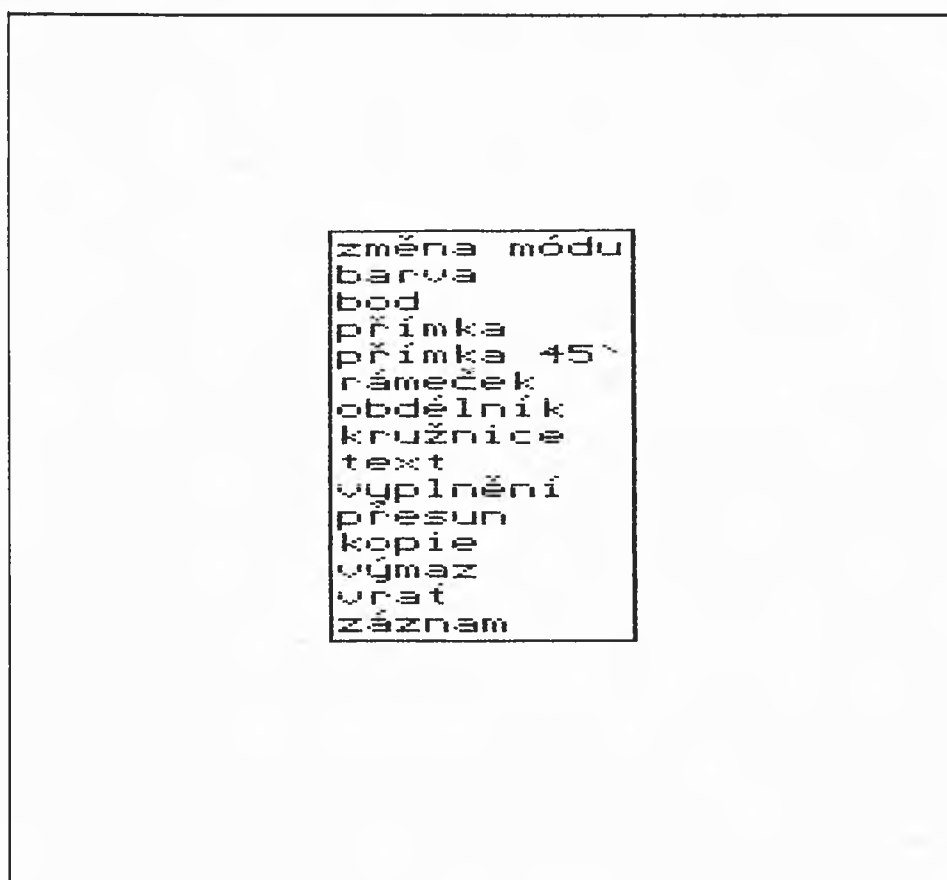
V bodu 2. byla řeč o **dynamickém kreslení objektu**. Význam tohoto pojmu si vysvětlíme na příkladu funkce "úsečka", podrobný popis najdete v kapitole o programování.

Zvolili jsme počáteční bod úsečky, držíme tlačítko SELECT a jedeme kurzorem po obrazovce. Mezi počátečním bodem

úsečky a aktuálním bodem, na který ukazuje kurzorová šipka, se neustále - dynamicky - vykresluje spojovací úsečka. Její směr a délka se při pohybu kurzoru průběžně mění tak, aby vždy spojovala počáteční a koncový bod. Tak je v každém okamžiku vidět, jak obrázek s úsečkou vypadá, což umožňuje její pohodlné a přesné umístění. Stejného principu dynamického kreslení objektu je použito i u většiny dalších grafických funkcí.

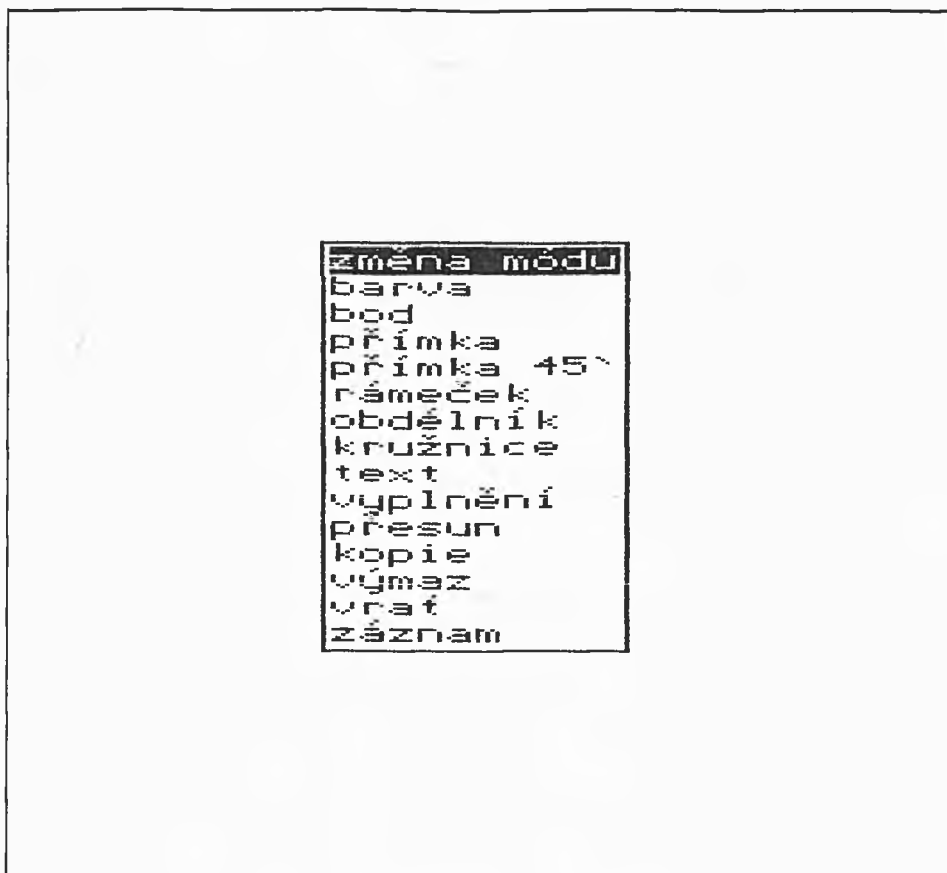
2.6.3 Funkce GREDITORu

Při stisknutí tlačítka MENU se objeví tato základní nabídka:

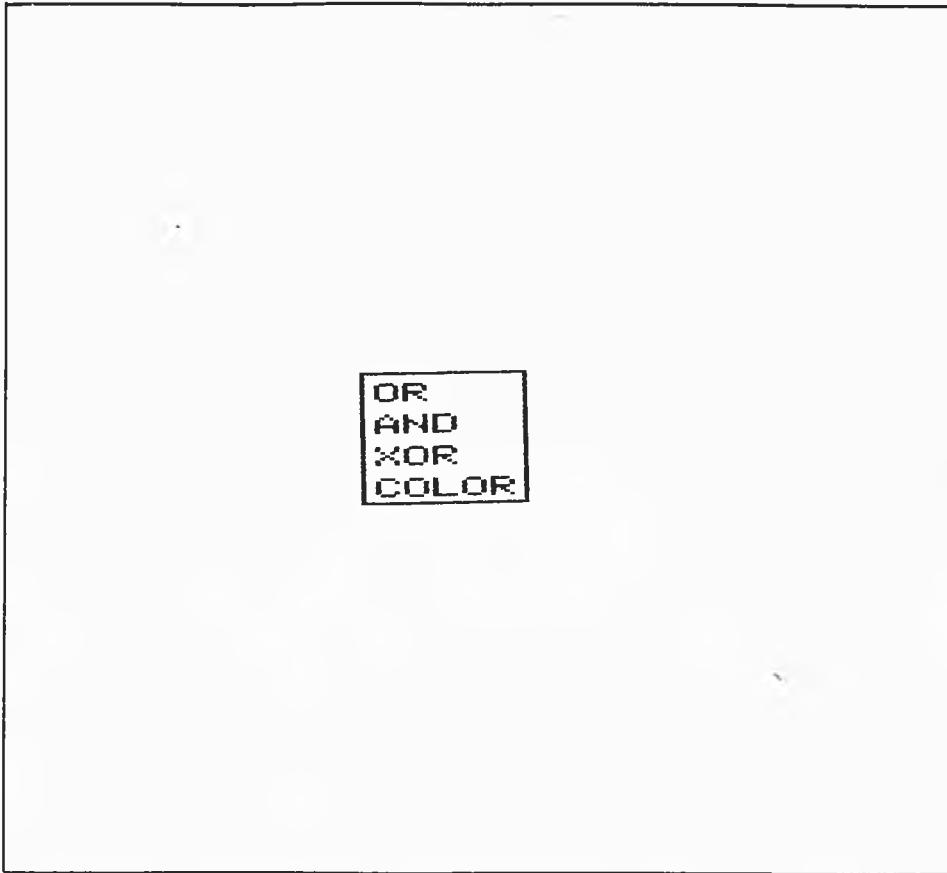


V dalším textu se dozvíte o významu jednotlivých funkcí.

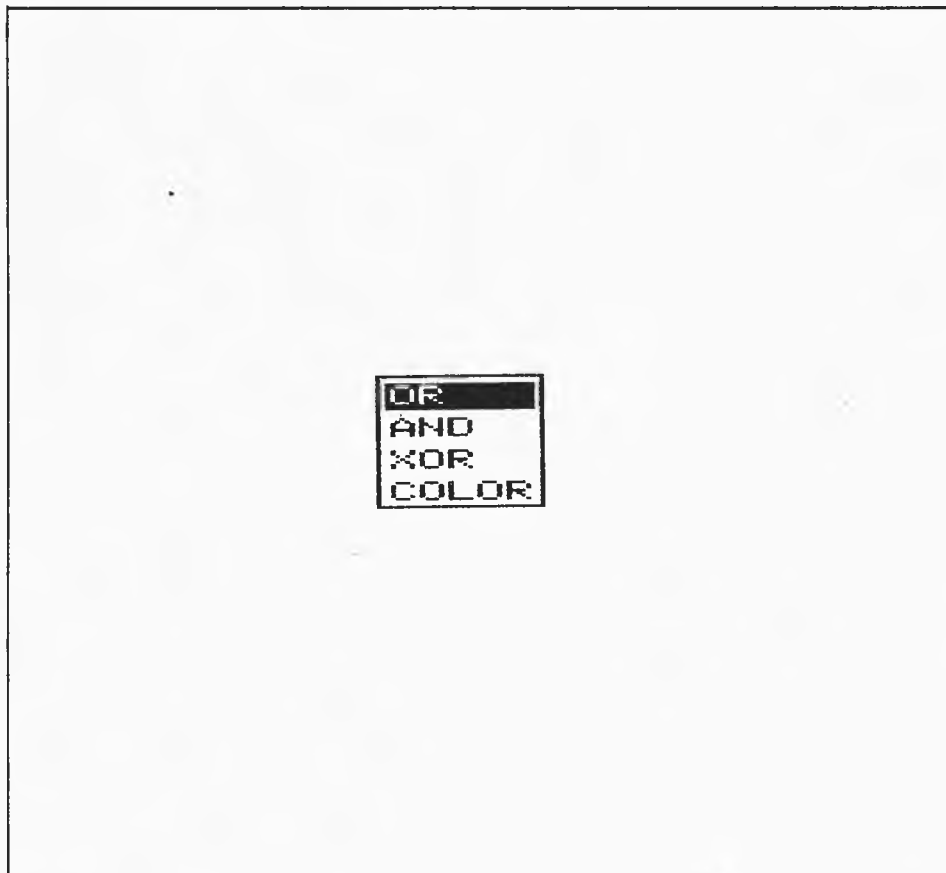
2.6.3.1 "změna módu"



Funkce "změna módu" slouží k nastavení grafického módu, ve kterém se budou vykreslovat grafické objekty. Protože je třeba zvolit jeden ze čtyř grafických módů, objeví se další nabídka:

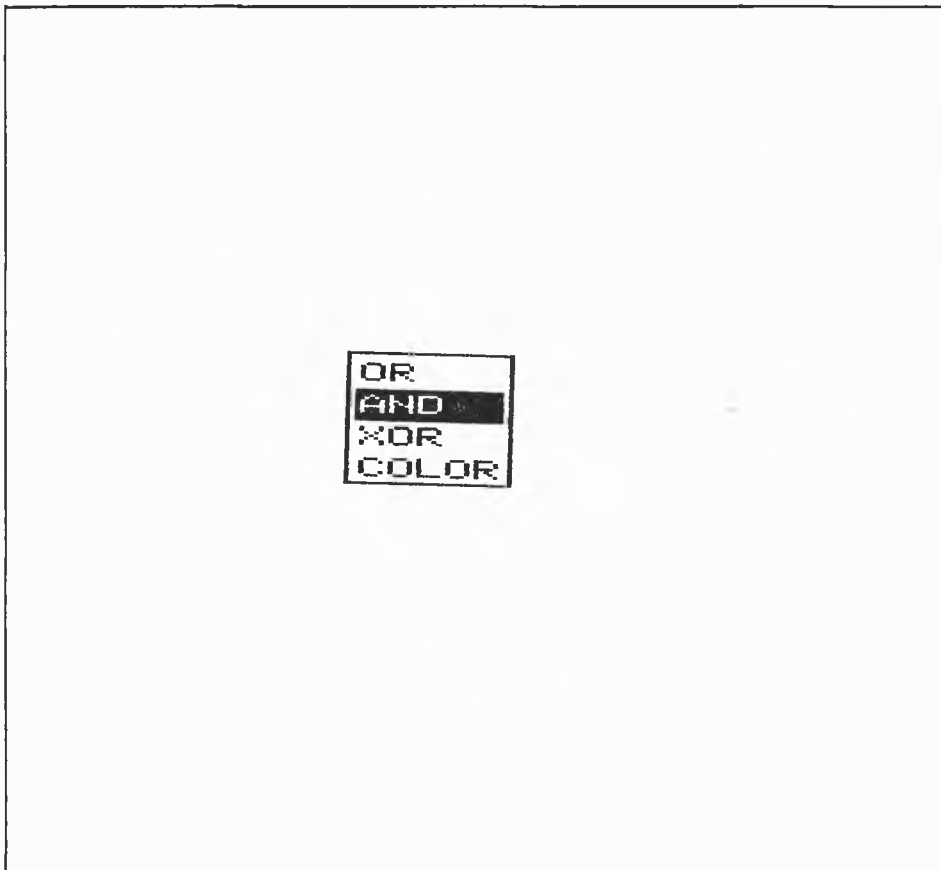


a) Mód OR



Tento grafický mód představuje kreslení barevnou tužkou. Grafické objekty se v tomto módu budou kreslit jako popředí ve zvolené barvě. Jedná se o základní a také nejpoužívanější mód. Pro jeho realizaci se používá logická funkce OR (NEBO - odtud název) - bity příslušející kreslenému objektu se nastaví do hodnoty log. "1".

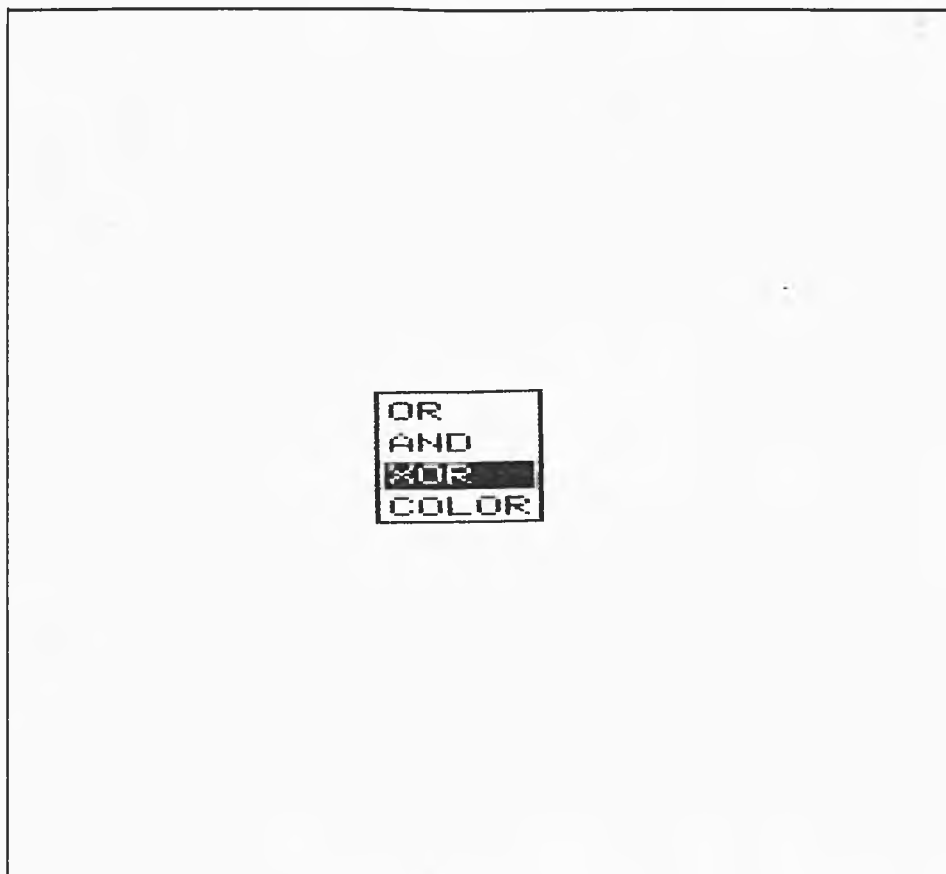
b) Mód AND



Tento mód představuje kreslení tužkou v barvě pozadí anebo jinak řečeno mazání gumou. Grafické objekty se v tomto módu budou kreslit v barvě pozadí. Mód AND se využívá hlavně pro mazání chybně zadaných grafických objektů.

Pro jeho realizaci se používá logická funkce AND (A ZÁROVEŇ) - bity příslušející kreslenému objektu se nulují - nastavují do hodnoty log. "0"

c) Mód XOR



U tohoto módu nejde použít žádné srovnání s tužkou či gumou, neboť se jedná o počítačovou specialitu, na papíře jednoduše nerealizovatelnou. Pro jednotlivé body grafického objektu se mění barva popředí za barvu pozadí a naopak. Mód XOR se využívá hlavně ve spojení s obdélníkem pro inverzi části obrázku, třeba pro zvýraznění textu.

Pro jeho realizaci se používá logická funkce XOR (VÝHRADNÍ NEBO) - bity příslušející kreslenému objektu se invertují - z log. "1" se udělá log. "0" a naopak.

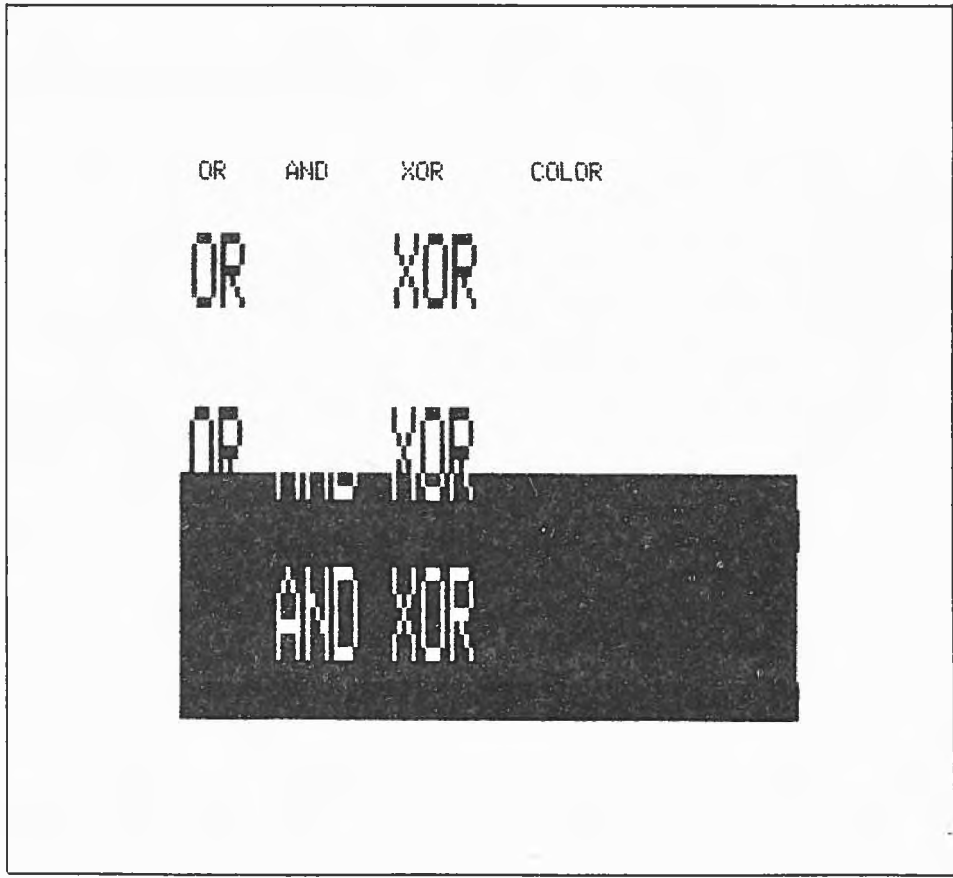
d) Mód COLOR



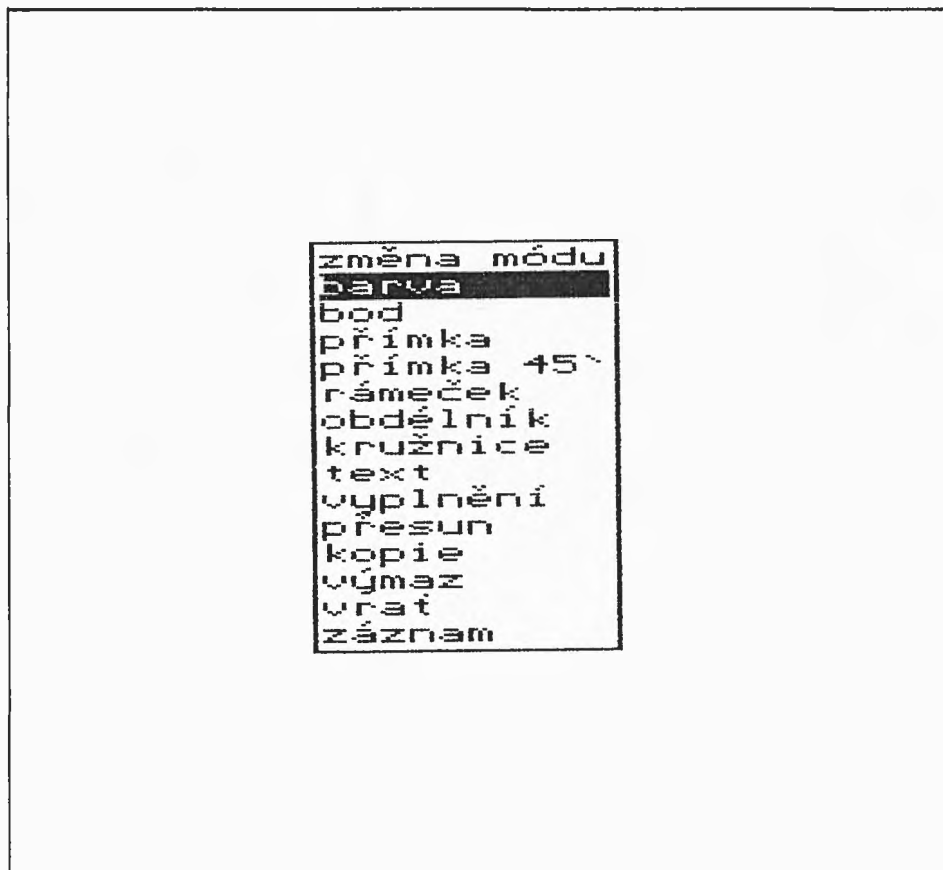
Tento mód představuje vybarvování obrázku. Při použití tohoto módu se nemění tvar nakresleného obrázku (nemění se bitmapa), pouze se mění barva - nastavují se barevné atributy.

Po výběru jednoho ze čtyř popsaných grafických módů z nabídky se nic nestane, změna se projeví až při následném kreslení grafických objektů.

Tomu, komu není z výkladu rozdíl mezi jednotlivými grafickými módy jasný, doporučuji empirické odzkoušení, při tom se vše nejlépe pochopí.



2.6.3.2 "barva" - PMD-85

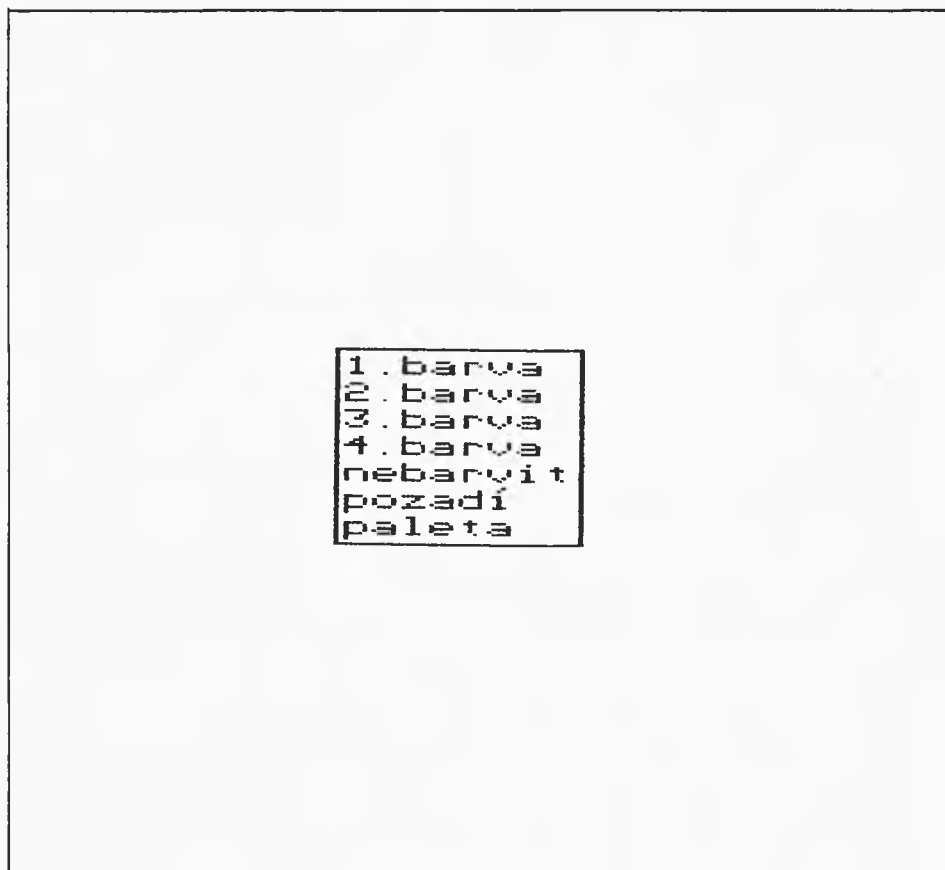


Funkce "barva" slouží k nastavení barvy, ve které se budou kreslit grafické objekty. Pro uživatele PMD-85 s úpravou pro barevný monitor slouží dále k volbě barvy pozadí a volbě barevné palety.

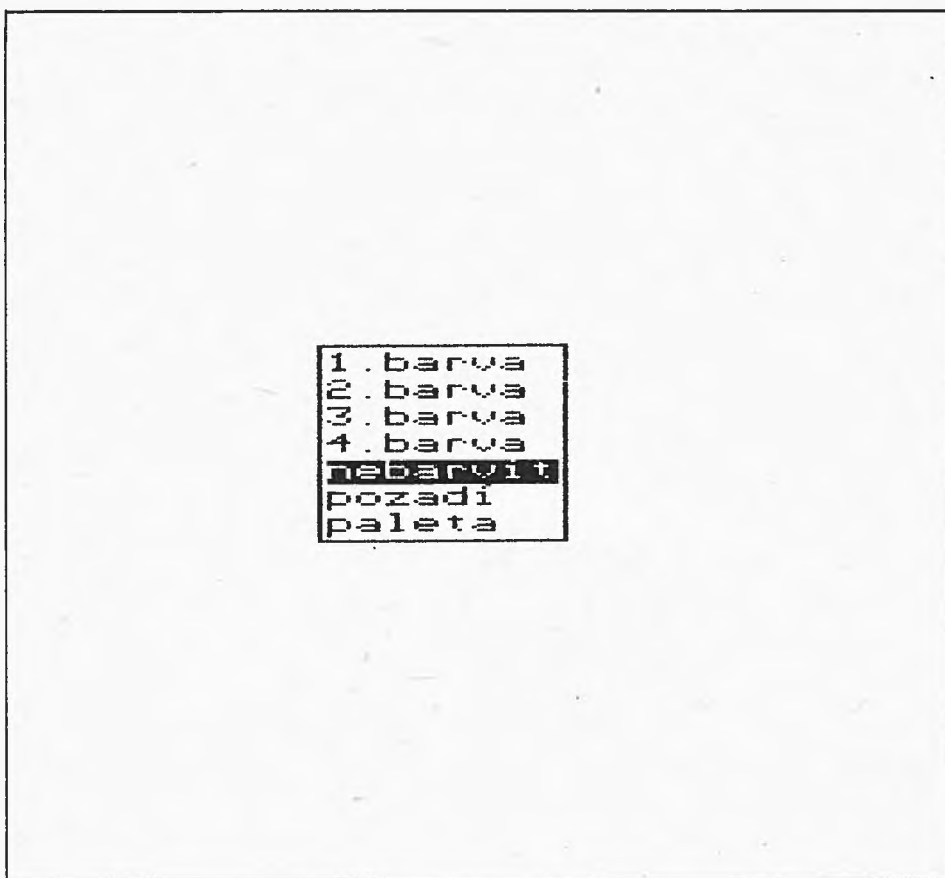
Pro uživatele PMD-85 bez barevné úpravy, v černobílé verzi, znamená volba barvy volbu atributu - pološedi a blikání, volba barvy pozadí a palety nemá žádný význam. Těmto uživatelům barevnou úpravu vřele doporučuji, neboť možností barevného zobrazení dostává PMD-85 další dimenzi! I jednoduchý obrázek při použití barevného rozlišení je mnohem přehlednější a názornější. A neopomenu dodat, že existuje i

barevná verze oblíbené hry MANIC-MINER! Schema barevné úpravy je ve **VTM 11/1986**.

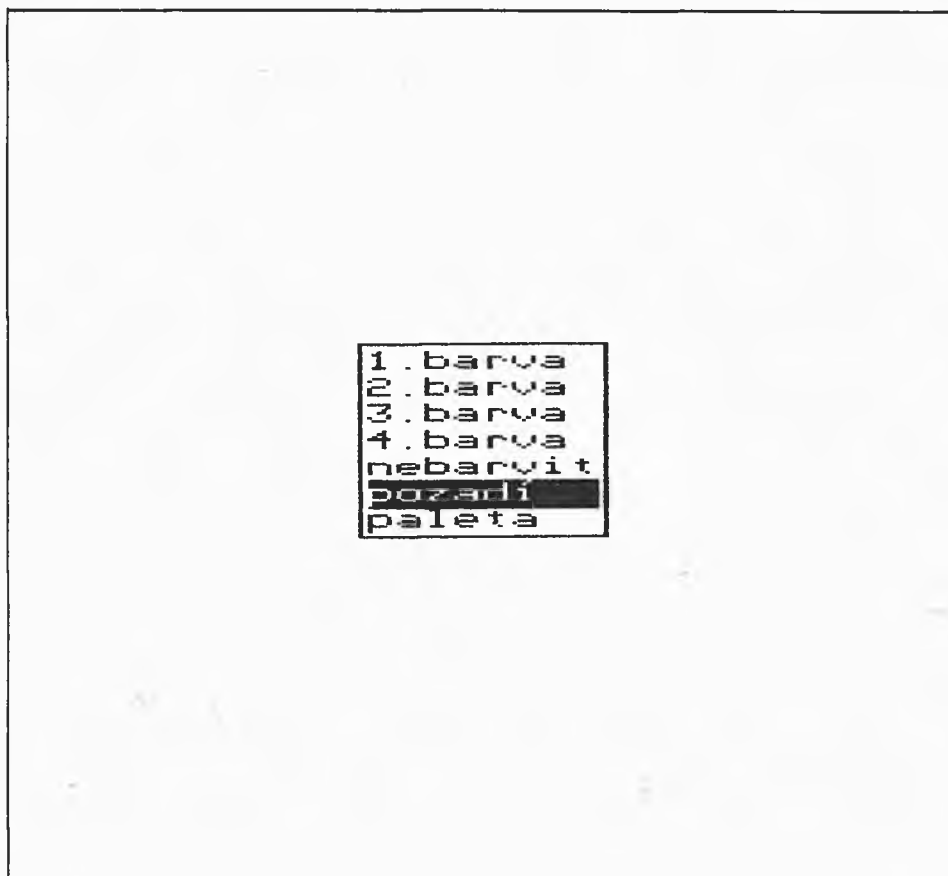
Po zvolení funkce "barva" z nabídky se vypíše tato další nabídka:



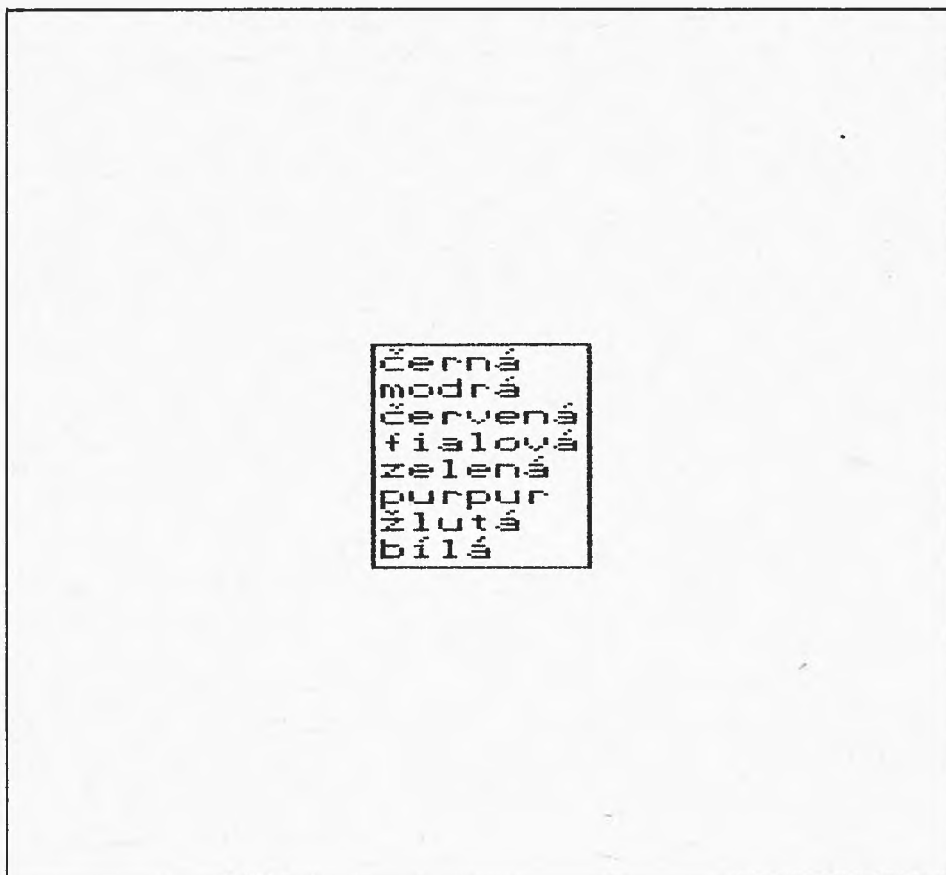
První čtyři řádky slouží ke zvolení jedné za čtyř barev, které mohou být současně zobrazeny na obrazovce. Text těchto čtyř řádek je vypsán vždy v příslušné barvě, takže volba je snadná a názorná. Tyto čtyři barvy jsou zobrazeny v pravé horní části obrazovky, v aktuálně zvolené barvě je vypsáno: *****.



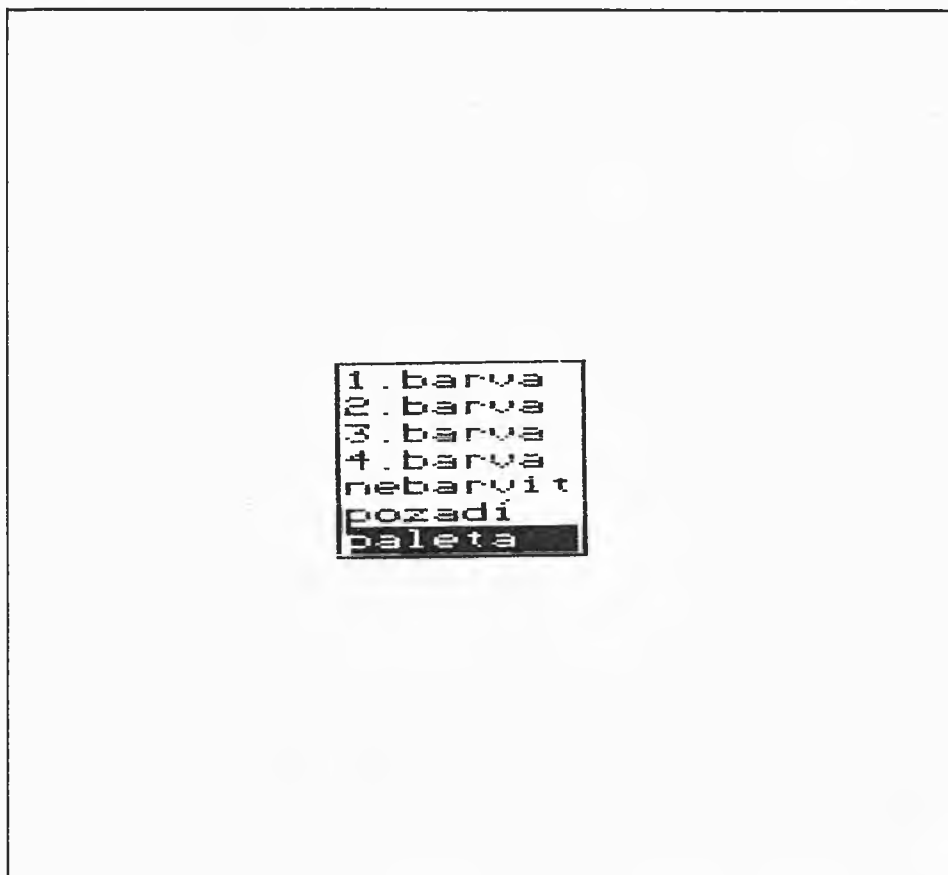
Výběrem řádku s textem "nebarvit" se při kreslení grafických objektů potlačí generování barevného atributu, tzn. změní se bitmapa ale nezmění se barva. Například vedeme-li přes žlutý obdélník červenou úsečku, část obdélníku se obarví do červena. Máme-li ale nastaven mód "nebarvit", celý obdélník zůstane žlutý.



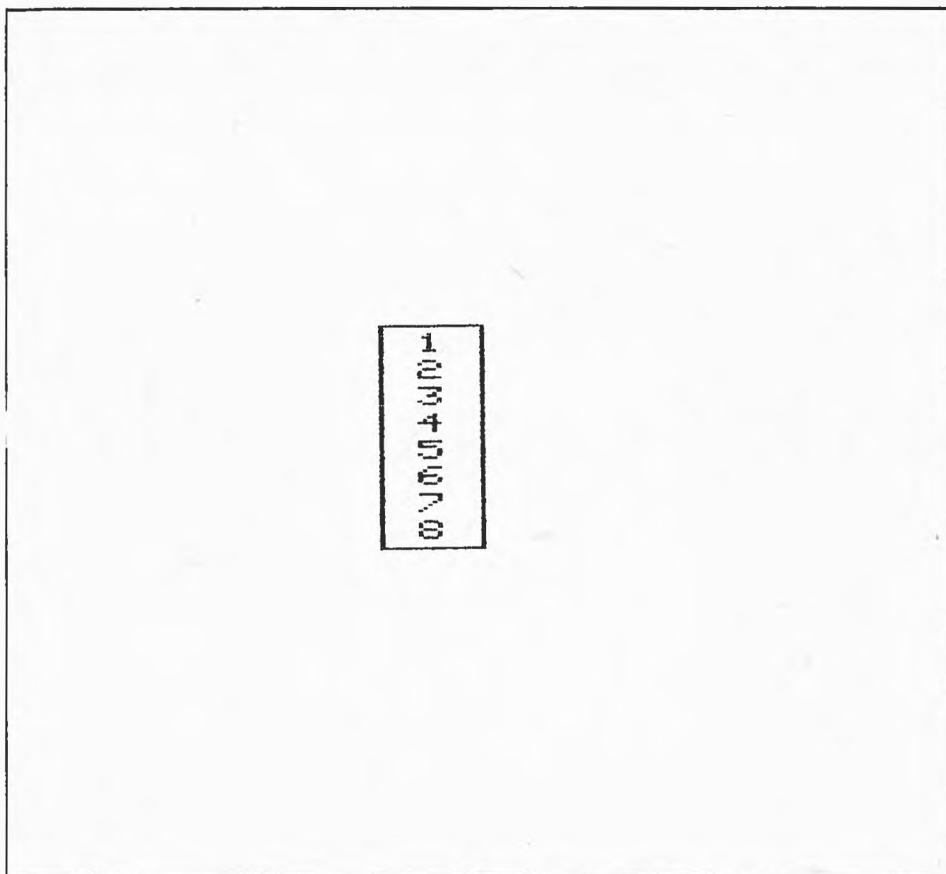
Řádek "pozadí" slouží k volbě barvy pozadí. Po jeho zvolení se objeví další nabídka:



Výběrem některé řádky nastavíme pozadí do zvolené barvy. Při pohybu kurzorové šipky po nabídce se průběžně nastavuje pozadí do barvy odpovídající řádce s kurzorem. Tak lze snadno a rychle zhodnotit barevný soulad pozadí s barvami popředí a vybrat optimální variantu. Při přemístění kurzoru mimo nabídku se nastaví původní barva pozadí

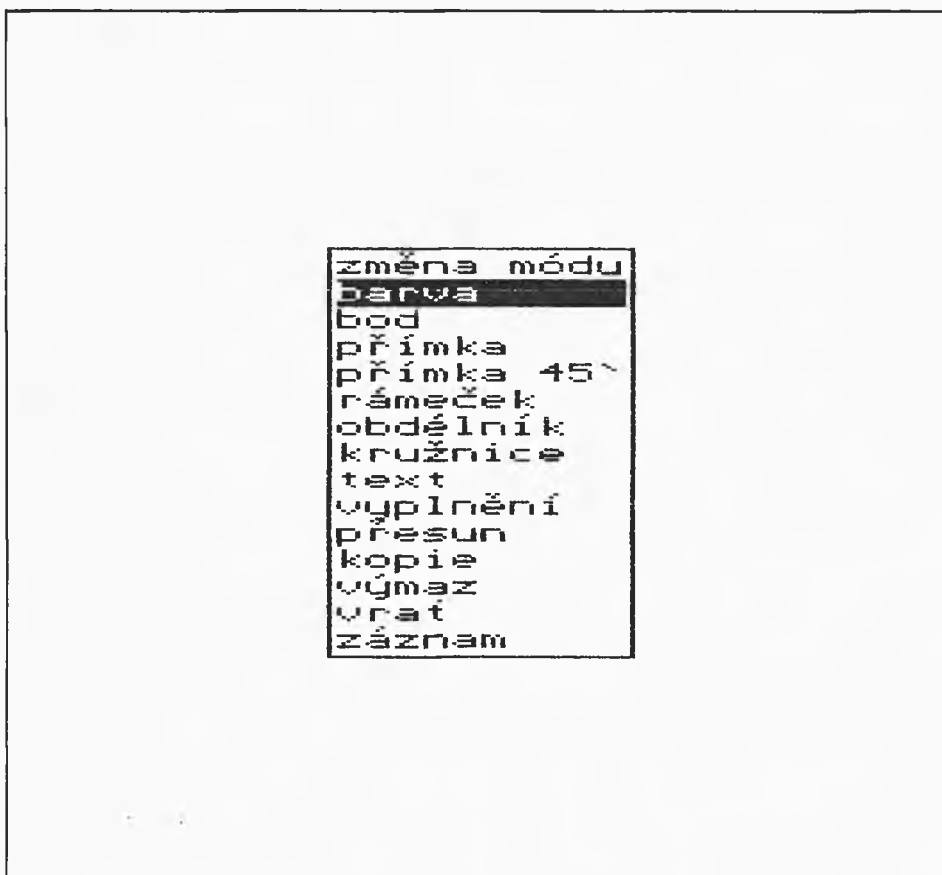


Poslední řádek "paleta" slouží k volbě barevné palety popředí. Po jeho zvolení se objeví další nabídka:



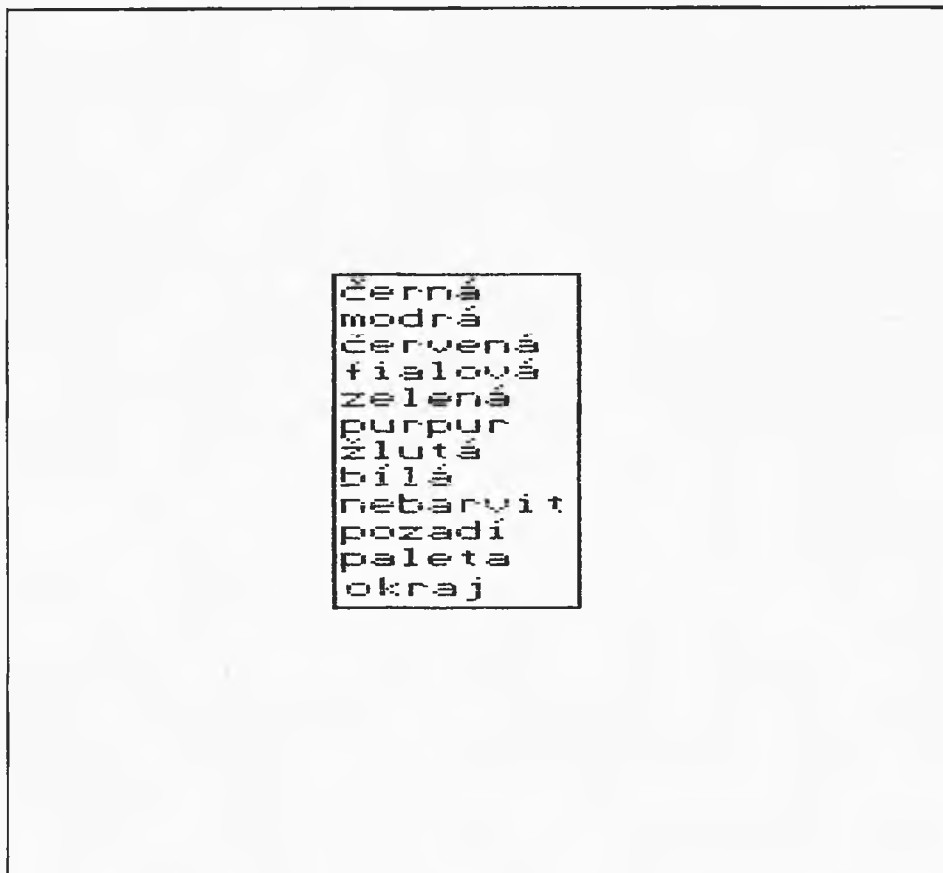
Lze zvolit jednu z osmi různých kombinací, kterými vybereme 4 barvy popředí z 8 možných. Konkrétní barvy jsou dány obsahem paměti PROM v úpravě PMD-85 pro barvu. Při pohybu kurzorové šipky po nabídce se příslušná paleta barev průběžně nastavuje, takže si lze snadno vizuálně vybrat vhodnou kombinaci. Při přemístění kurzoru mimo nabídku se opět nastaví původní barevná paleta.

2.6.3.3 "barva" - ZX-Spectrum

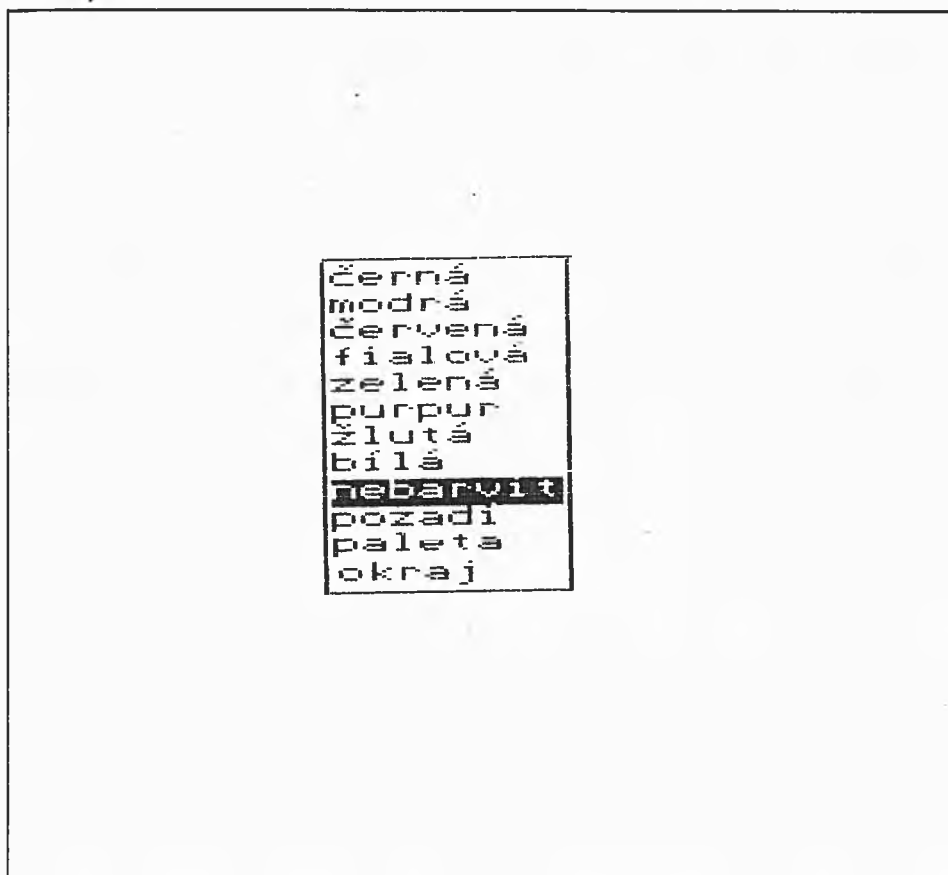


Obdobně jako u PMD-85, slouží i na ZX-Spectrum tato funkce pro nastavení barevných atributů, které se budou přiřazovat kresleným grafickým objektům.

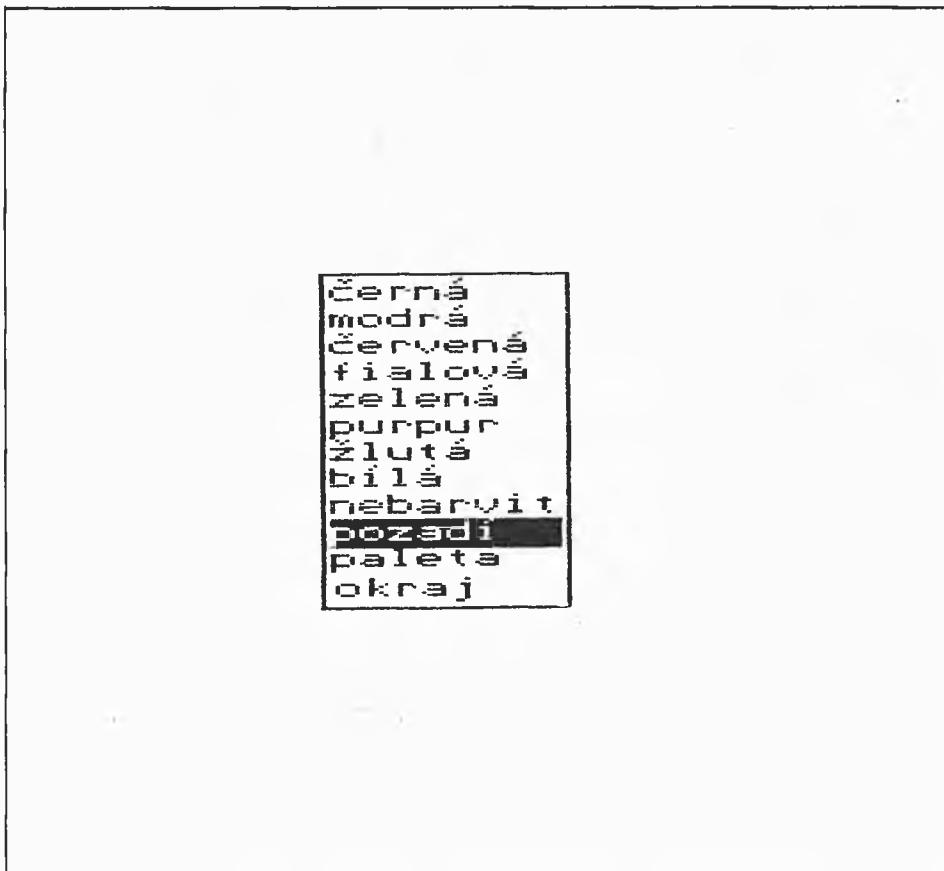
Po zvolení funkce se vypíše další nabídka:



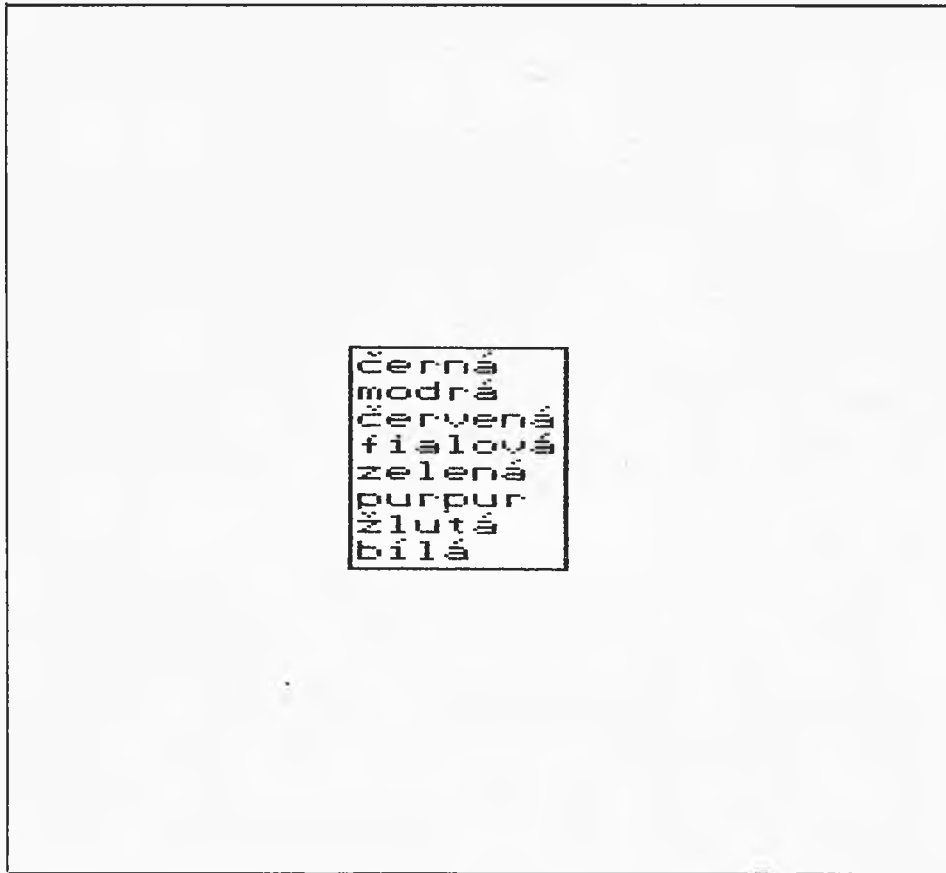
Prvních osm řádek slouží ke zvolení jedné z osmi barev, které mohou být zobrazeny na obrazovce.



Výběrem řádku "nebarvit" se při kreslení grafických objektů potlačí generování barevných atributů, tzn. mění se pouze bitmapa, nikoli i barevné atributy.



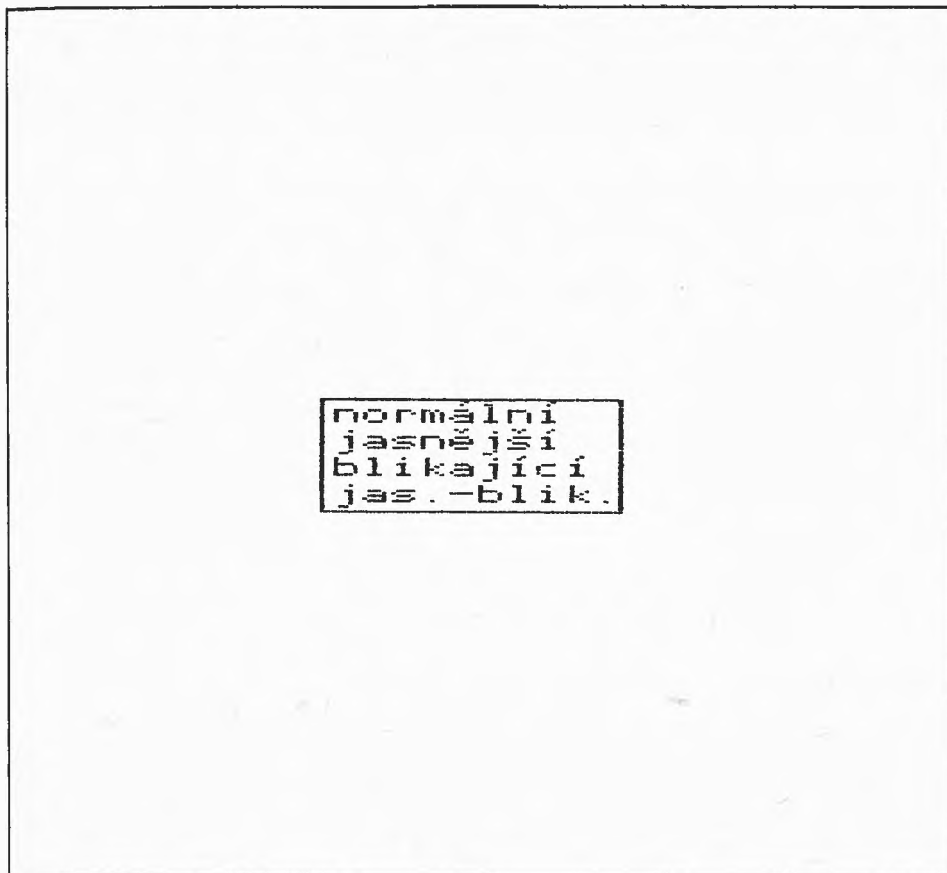
Řádek "pozadí" slouží k volbě barvy pozadí. Po jeho zvolení se objeví tato nabídka osmi barev:



Zvolená barva se použije při kreslení grafických objektů pro obarvení pozadí.

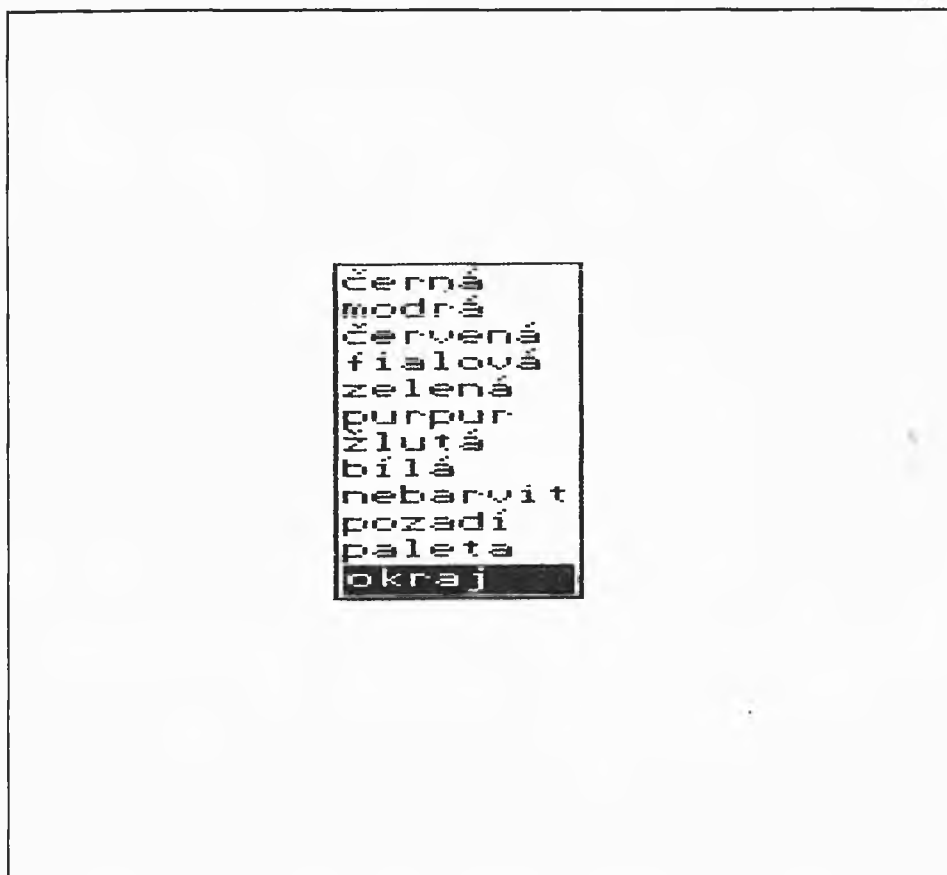


Další řádek "paleta" slouží k volbě dalších barevných atributů. Po jeho zvolení se objeví tato nabídka:



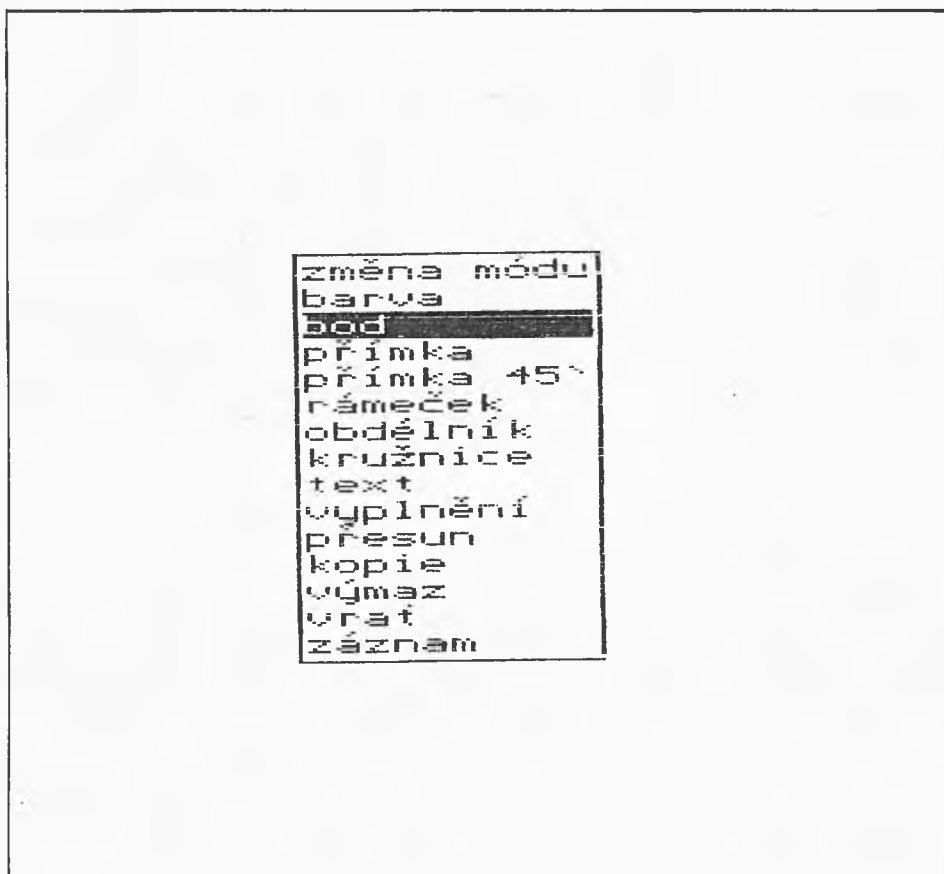
Význam těchto čtyř možností je shodný jako při použití příkazů BASICu BRIGHT a FLASH, přiřazení je toto:

normální	BRIGHT 0: FLASH 0
jasnější	BRIGHT 1: FLASH 0
blikající	BRIGHT 0: FLASH 1
jas.-blik.	BRIGHT 1: FLASH 1



Poslední řádek "okraj" slouží k nastavení barvy okraje obrazovky. Vypíše se další nabídka osmi barev, která je shodná jako pro volbu barvy pozadí. Při pohybu kurzorové šipky po nabídce se průběžně nastavuje okraj obrazovky do barvy odpovídající textu řádky nabídky, na kterou ukazuje šipka.

2.6.3.4 "bod"



Řádek "bod" v základní nabídce je první ze sedmi, které nastavují GREDITOR do módu kreslení zvoleného grafického objektu. Jsou to základní - nejdůležitější - funkce, protože slouží ke kreslení obrázku. Vykreslení grafického objektu se provede pomocí tlačítka SELECT, jak již bylo popsáno dříve. Na PMD-85 se název zvolené funkce vypisuje svisle v pravé části obrazovky.

Mějme zvolenu funkci "bod" mód "OR" a nějakou barvu. Co se stane, stiskneme-li krátce tlačítko SELECT?

První okamžik po stisknutí tlačítka se nestane nic, po malé chvilce (desetiny sekundy pro PMD-85, na ZX-Spectrum takřka nepostřehnutelné) se rozsvítí na obrazovce bod, na

který ukazuje kurzorová šipka. Čím je způsobena ona prodleva? GREDITOR umožňuje uživateli navrácení o jeden krok (viz. funkce "vrať" v dalším textu). Proto musí uložit stav obrázku před provedením zvolené funkce. A přenesení 12 KB na PMD-85 nebo 7 KB na ZX-Spectru trvá právě onu chvilíčku.

Při krátkém stisknutí tlačítka SELECT se tedy na obrazovce rozsvítí jeden bod. Takto lze kreslit jednotlivé body - ukážeme šipkou do požadovaného místa a stiskneme krátce SELECT.

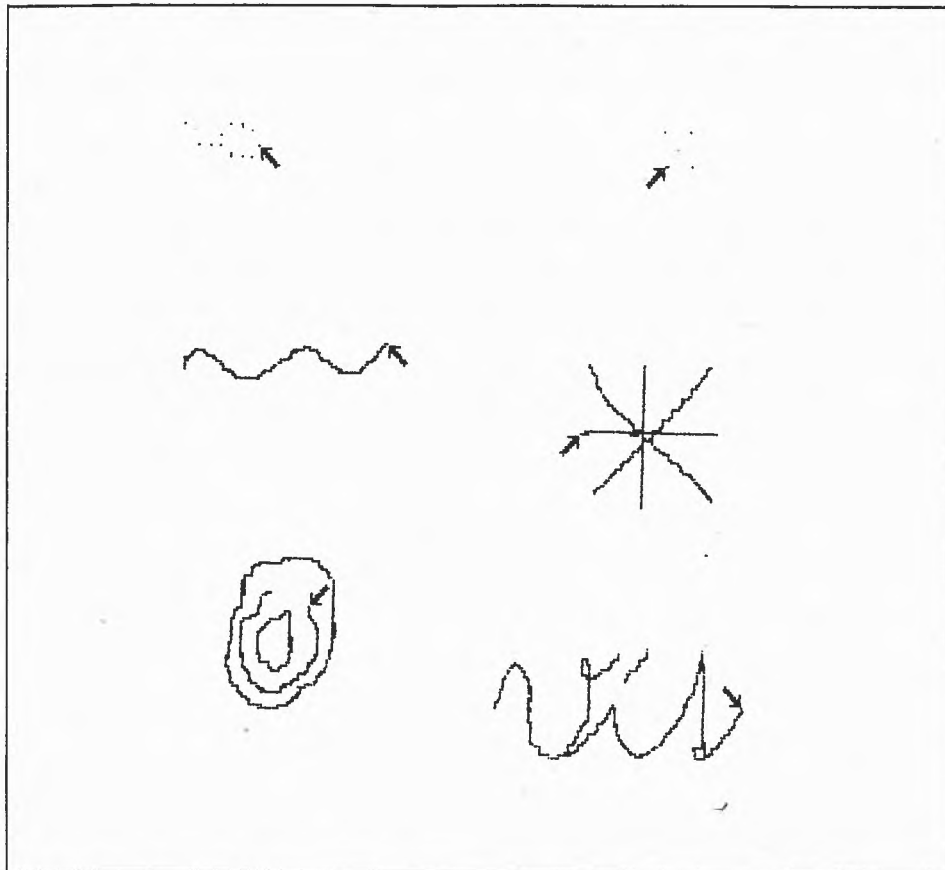
Co ale, když SELECT podržíte a pohnete myší? V tomto případě se vykreslí řada na sebe navazujících bodů, jejichž poloha odpovídá pohybu myši. Takto lze snadno kreslit obecné čáry - jako tužkou po papíře.

Mnoho hezkých obrázků lze nakreslit pouze s využitím této funkce.

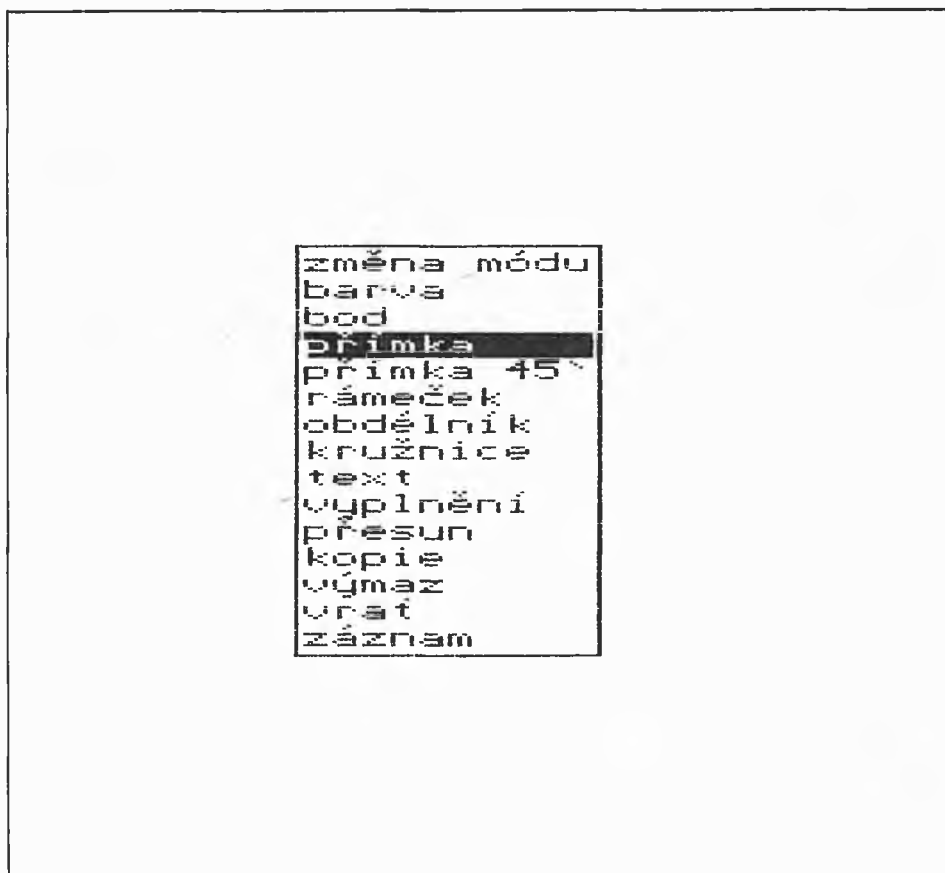
Funkce "bod" je rovněž vhodná pro procvičení jezdění myší po podložce. Zkuste si cvičně nakreslit třeba tyto objekty:

- vodorovnou čáru
- svislou čáru
- čáru vpravo nahoru pod úhlem 45 stupňů
- kružnici
- vlastní podpis

Jistě se Vám nepodaří hned napoprvé vše nakreslit tak, jak byste si to představovali - pokud nejste druhý Jiránek. Všechno chce cvik, po nějakém čase získáte praxi a opravdu nakreslíte i solidní kružnici a ne bramboroid.



2.6.3.5 "přímka"

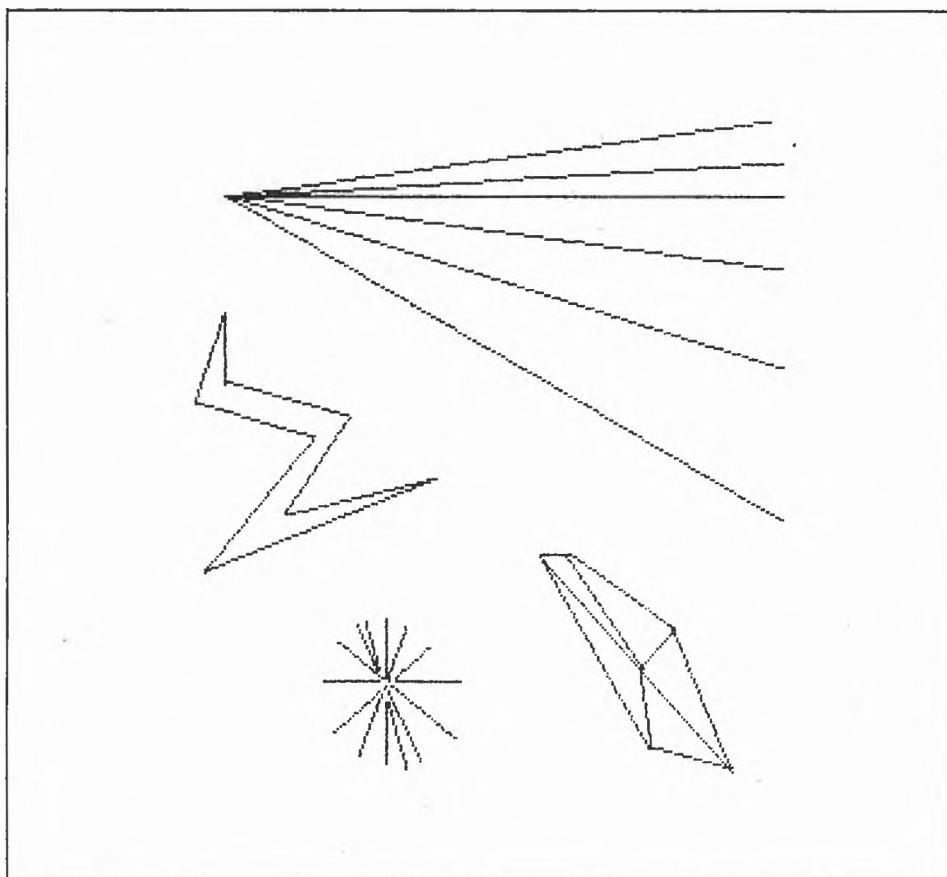


Správně by se tato funkce měla jmenovat "úsečka", ale to je drobnost. Důležité je to, že Vám umožní nakreslení rovné spojnice dvou bodů - úsečky.

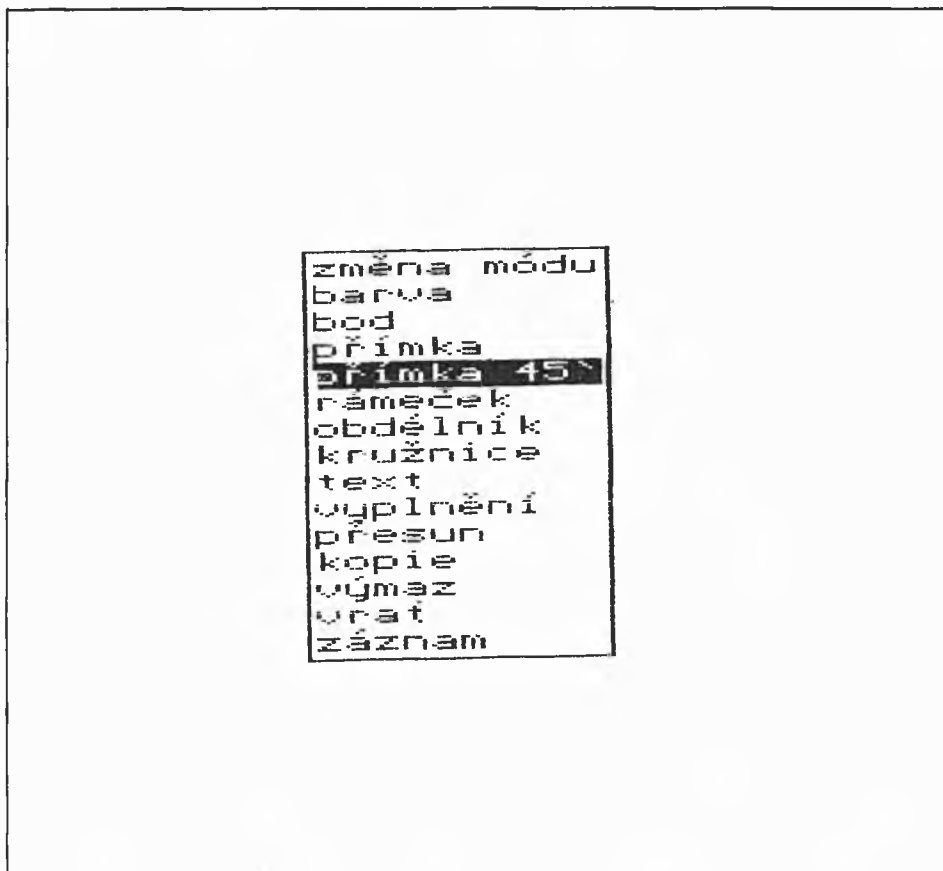
Postup je jednoduchý: v počátečním bodu stisknete SELECT, přejedete do cílového bodu a SELECT pustíte. Jak již bylo řečeno v odstavci o dynamickém kreslení, v každém okamžiku se mezi počátečním bodem a kurzorem kreslí spojnice, takže přesně vidíte, jak úsečka bude vypadat.

Jediný rozdíl je ten, že se kreslí v grafickém módu XOR (aby ji bylo možné smazat) a nebarví se. Teprve po puštění tlačítka SELECT se vykreslí ve zvoleném grafickém módu a barvě. Ovšem její poloha je shodná, ta už se nezmění.

Shodný princip je použit i u dalších grafických objektů, takže již se o něm nebudu zmiňovat.

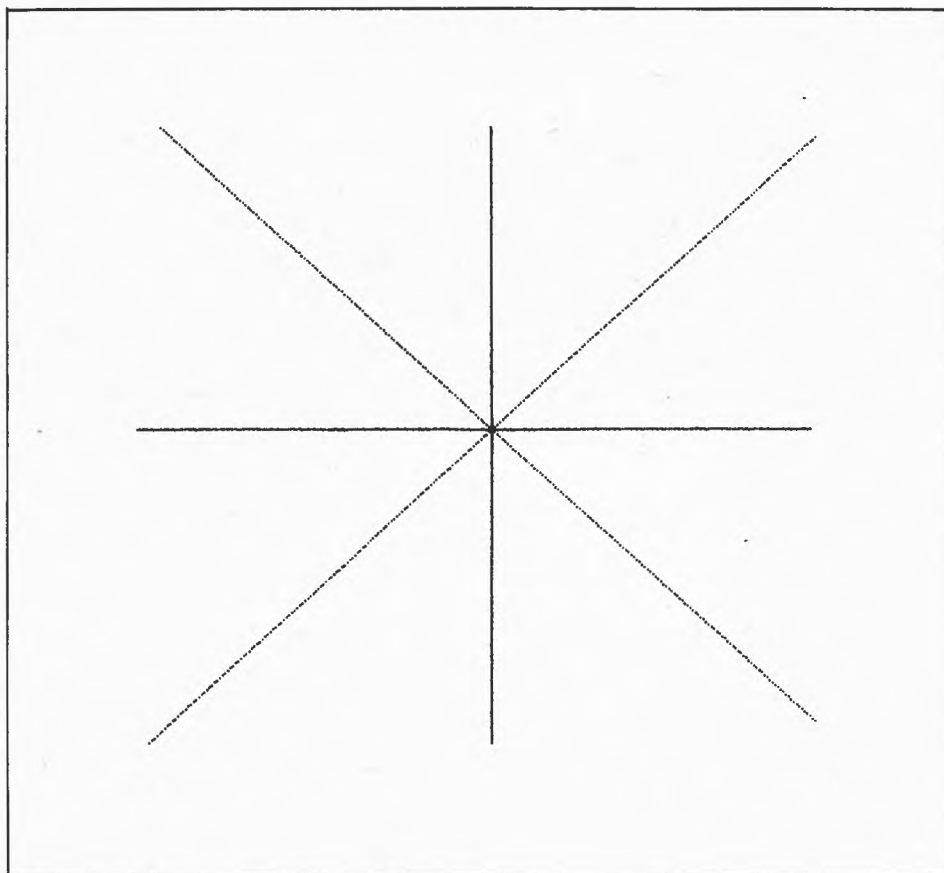


2.6.3.6 "přímka 45°"



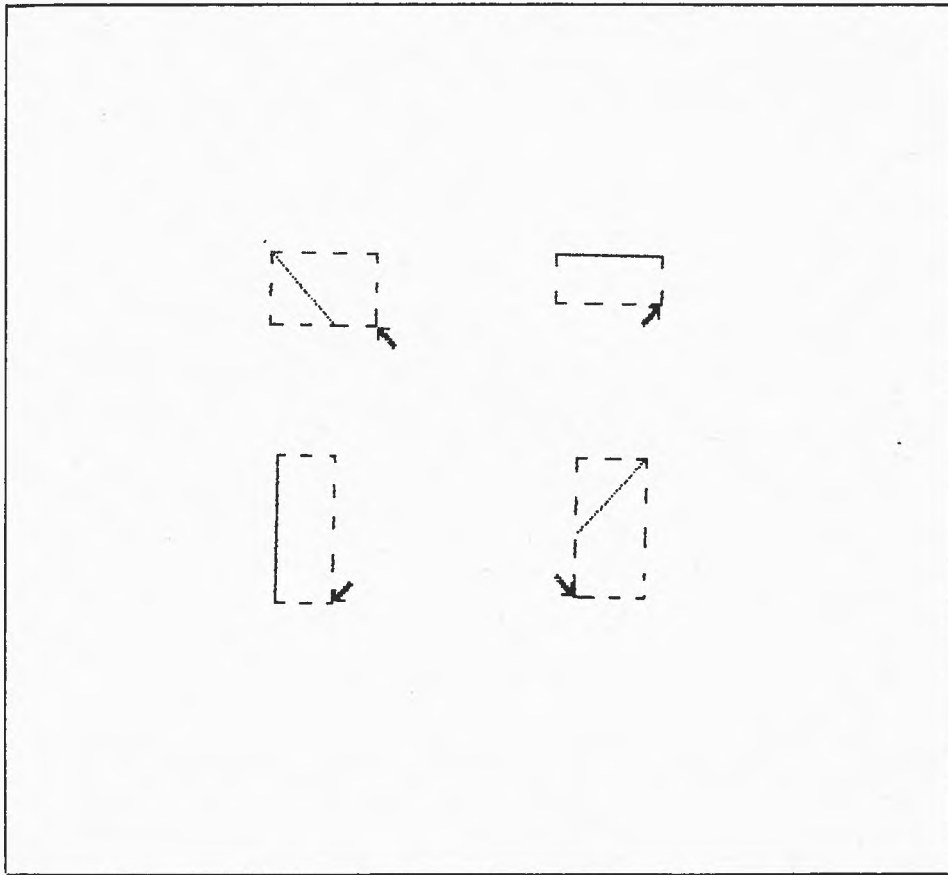
Za číslem 45 by správně měla být značka pro stupeň, ale na počítačích tato značka bohužel není obvyklá.

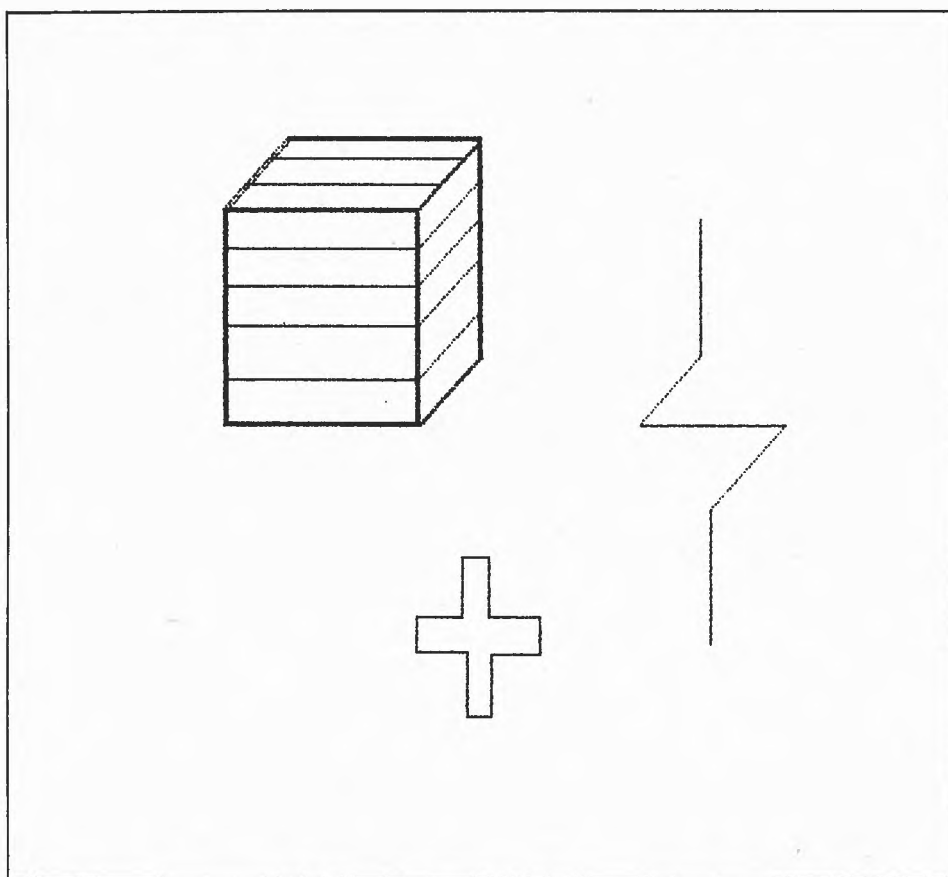
Tato funkce slouží ke kreslení úseček vedených z počátečního bodu pod úhlem 45 stupňů, což představuje osm různých směrů.



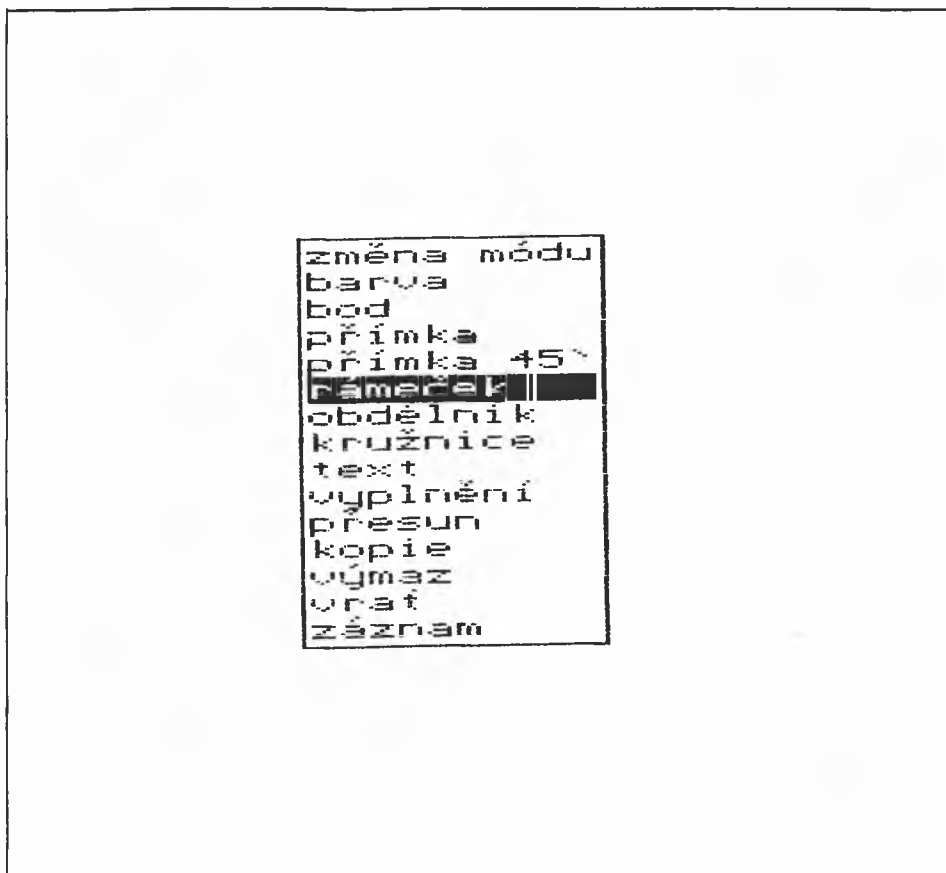
Výhody této funkce vyniknou při kreslení pravoúhlých objektů - odpadnou problémy s rovnoběžnostmi.

Po zvolení počátečního bodu se z něho dynamicky kreslí úsečka směrem ke kurzoru - ale ne přímo k němu! Jeden z osmi možných směrů se volí automaticky tak, aby úsečka směřovala co nejbližší ke kurzoru. Její délka je volena tak, aby nevyčuhovala z myšleného obdélníku daného dvěma rohy - počátečním bodem a aktuálním bodem (kurzorem).

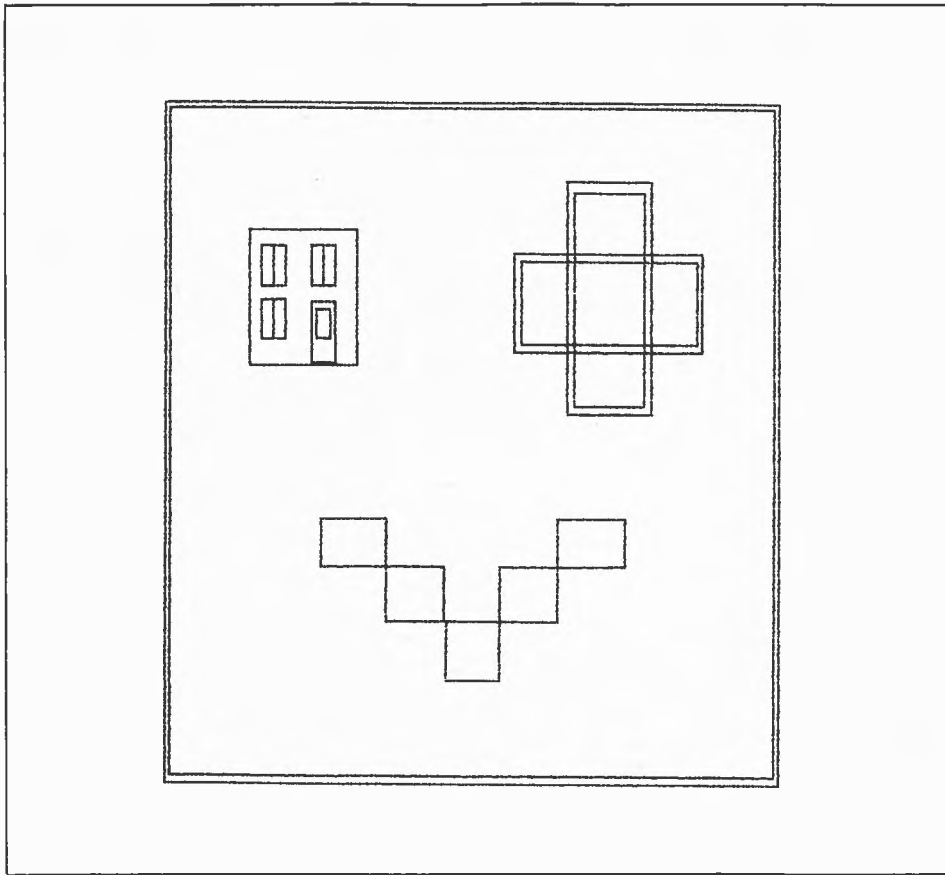




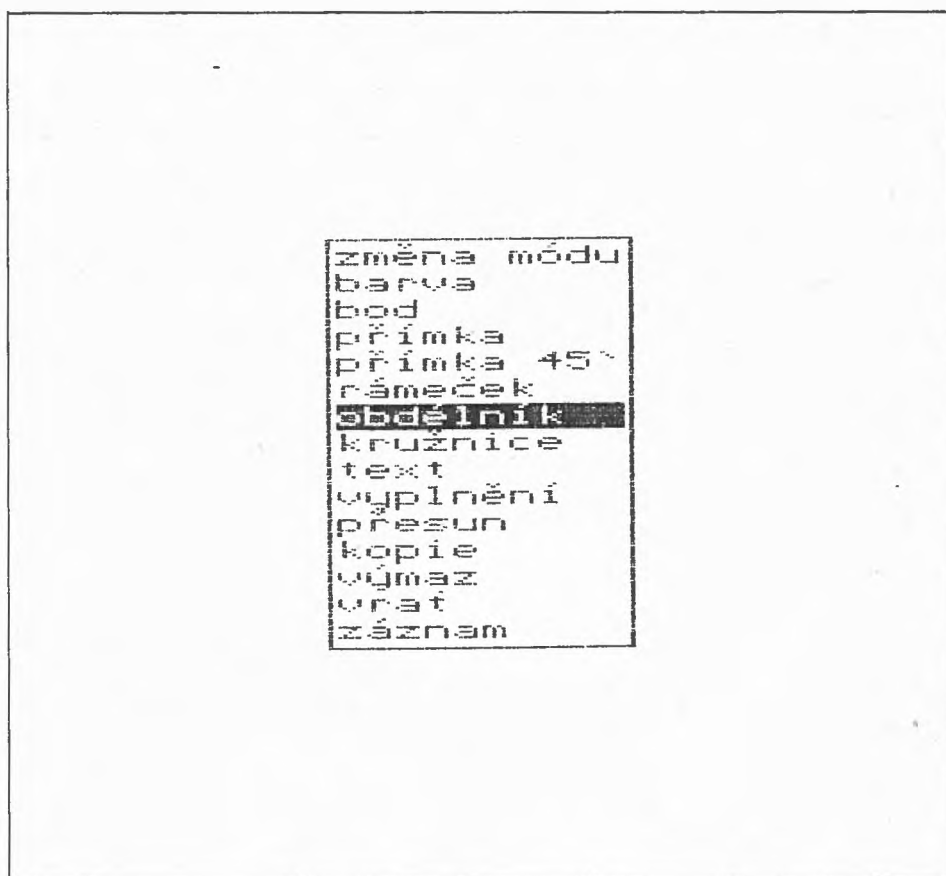
2.6.3.7 "rámeček"



Funkce "rámeček" slouží k nakreslení pravoúhlého rámečku. Poloha rámečku je dána jeho dvěma vrcholy - v jednom leží počáteční bod a do druhého ukazuje kurzorová šipka. Použití rámečku je celkem zřejmé - k zarámování části obrázku, zdůraznění textu nebo při kreslení objektu složeného z pravoúhlých čar.



2.6.3.8 "obdélník"

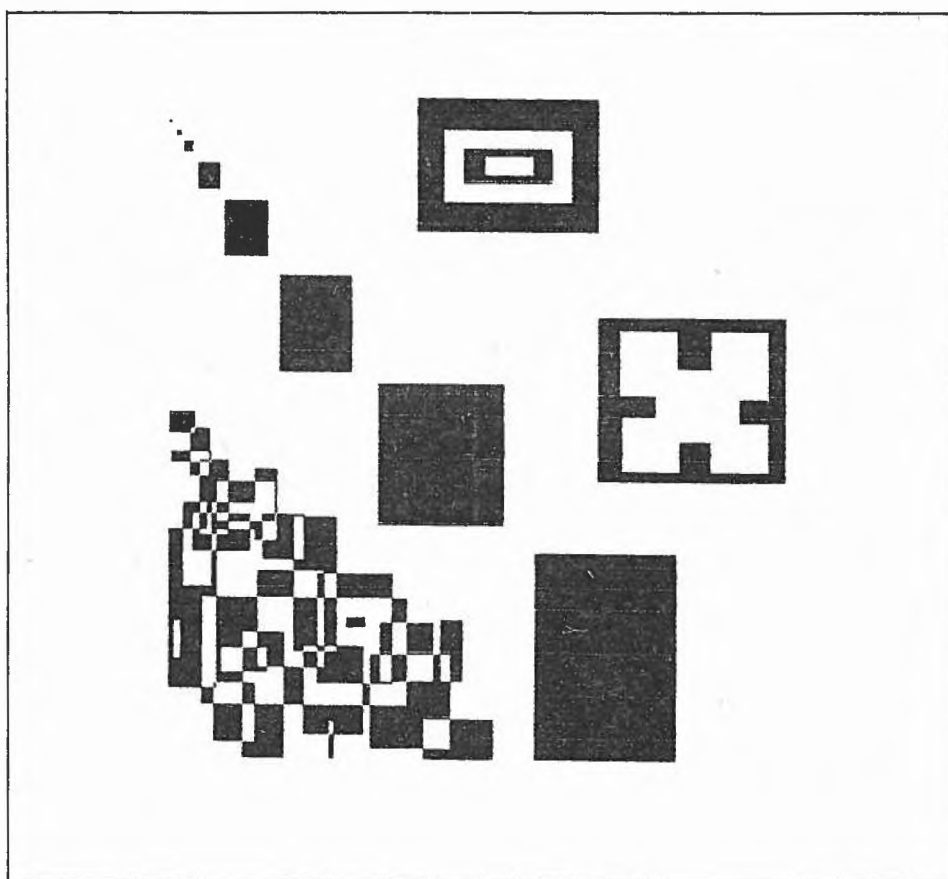


Průběh kreslení obdélníku se zpočátku neliší od kreslení rámečku, teprve při puštění tlačítka SELECT se vykreslí plný obdélník, nejenom rámeček.

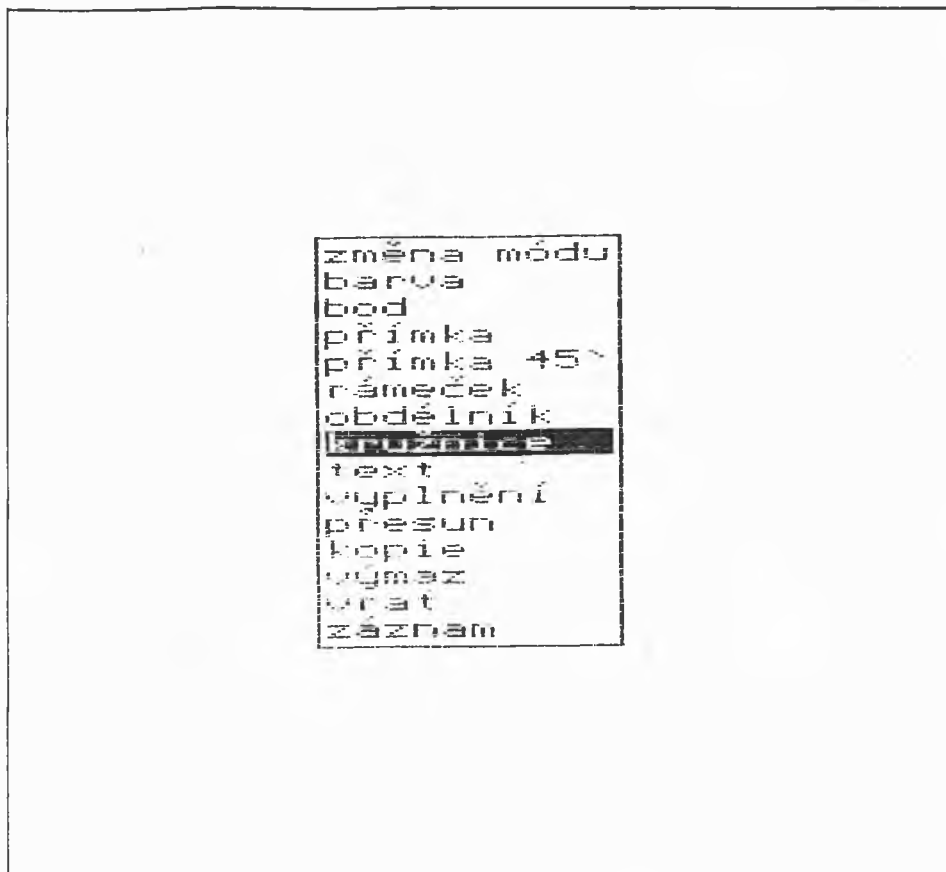
Za zmínku stojí použití obdélníku v různých grafických módech. Při zvolení módu XOR lze obdélník použít k inverzi - třeba celého obrázku, jeho části či textu. Obdélník se také velice často používá v módu AND - k mazání. Potřebujete smazat část obrázku? Nastavíte si mód AND a funkci "obdélník" a poté zarámujete nežádoucí část obrázku, pustíte SELECT a zarámovaná oblast se smaže. Pro smazání členitější oblasti použijeme obdélník vícekrát po sobě.

A nakonec obdélník se výhodně použije i v módu \COLOR pro vybarvení obrázku. Máme třeba na obrazovce víceřádkový text a chceme ho barevně rozlišit? Zvolíte si barvu, mód COLOR, funkci "obdélník" a pak již jen zarámujete požadovanou oblast, řádek textu nebo jednotlivé slovo či písmeno a to se Vám vybarví do zvolené barvy. Můžete volit jiné barvy a obarvit další části až k Vaší úplné spokojenosti.

Před kreslením dalších grafických objektů nezapomeňte nastavit vhodný grafický mód - třeba OR!



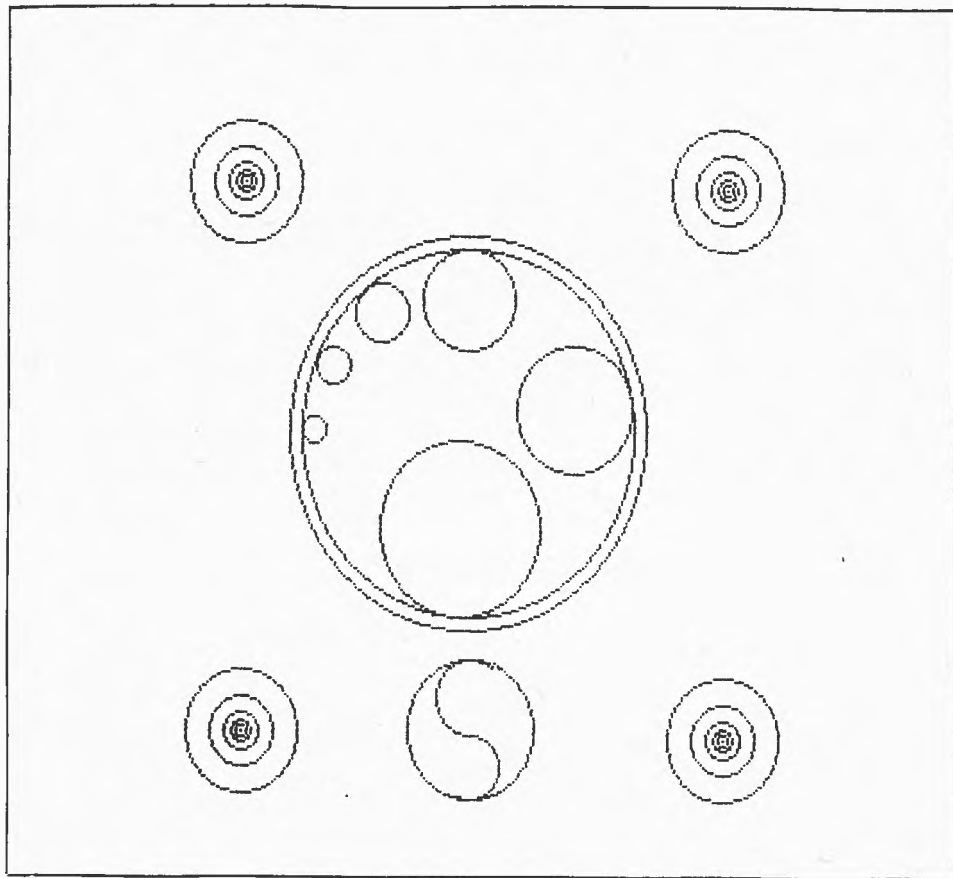
2.6.3.9 "kružnice"



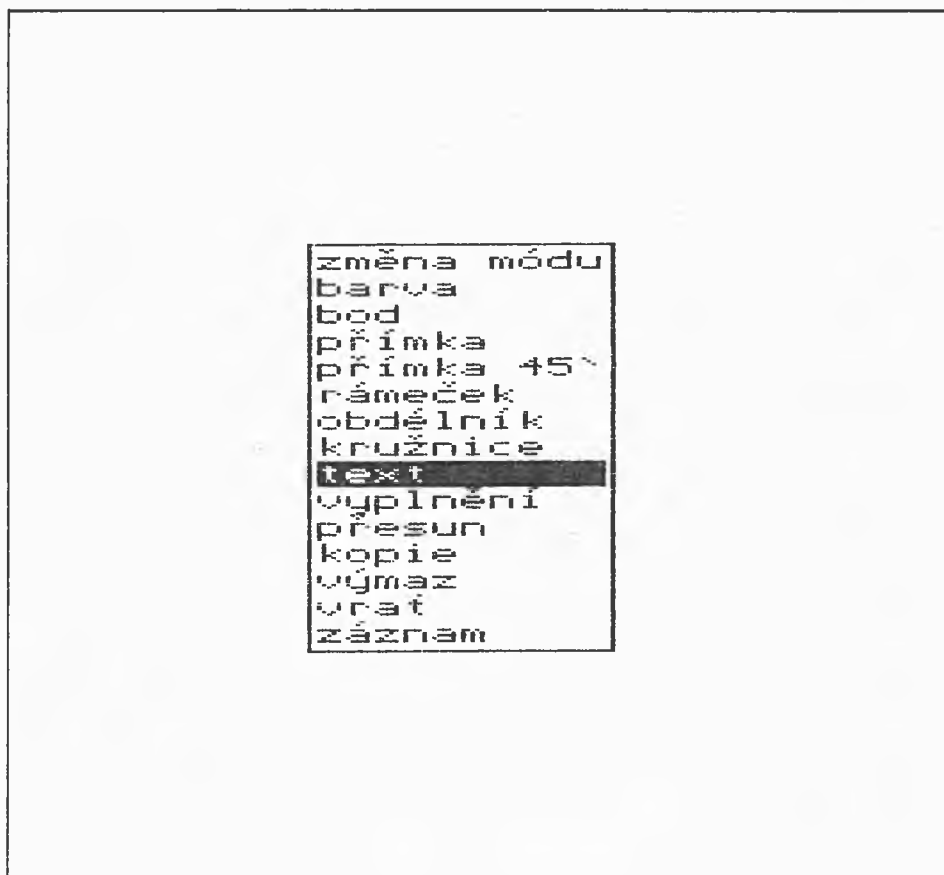
Jak je dobře zřejmé z názvu, tato funkce slouží ke kreslení kružnic.

Nejdříve zadáme počáteční bod, který bude tvořit **střed** kružnice. Poté najedeme kurzorem do cílového bodu, čímž zadáme poloměr kružnice. Celá kružnice musí být na obrazovce, tzn. nejde zadat kružnici tak, aby na obrazovce byla zobrazena pouze její část.

Volba poloměru kružnice opět probíhá dynamicky, kružnice se průběžně vykresluje. Protože kreslení kružnice je časově náročnější, není její zvětšování a zmenšování plynulé.



2.6.3.10 "text"



Funkce "text" slouží k vykreslení textu do obrázku. Lze psát text libovolné velikosti, od normální velikosti až k písmenům přes celý displej. Poměr zvětšení může být pouze celé číslo (nejde třeba zvětšit text 1,5x), ale zvětšení v osách X a Y je na sobě nezávislé.

Před zápisem textu nejdříve musíme určit polohu a velikost písma. Počáteční bod (kde stiskneme SELECT) určuje levý dolní roh prvního písmene. Držíme SELECT a jedeme kurzorem doprava nahoru. Z počátečního bodu se průběžně vykresluje rámeček, který ukazuje velikost písma. Puštěním tlačítka SELECT při vhodné velikosti rámečku zvolíme velikost písma.

Vlastní text již zadáváme na klávesnici počítače. Vždy po stisknutí klávesy se vypíše příslušný znak ve zvolené velikosti, grafickém módu a barvě. Rámeček nám průběžně ukazuje místo, do kterého se vypíše následující znak.

Poslední vypsaný znak lze zrušit klávesou BACKSPACE a celý řádek zrušíte klávesou CLR (RUBOUT). Pomocí klávesy EOL (ENTER) lze přejít na nový řádek, tedy na pozici pod prvním znakem předchozí řádky.



Pokud by se další znak již nevešel na obrazovku, rámeček zmizí a další znaky se ignorují. V tomto případě je nutné pomocí myši zvolit novou polohu a velikost textu nebo zvolit jinou grafickou funkci.

Každý znak textu může být vypsán v jiné barvě a grafickém módu. K tomu stačí před jeho stisknutím na klávesnici pomocí myši a příslušného řádku nabídky zvolit novou barvu nebo grafický mód. Tyto informace si GREDITOR pamatuje o každém znaku zvlášť.

□

Píšu text□

Přechod na nový řádek.

□

První řádek.

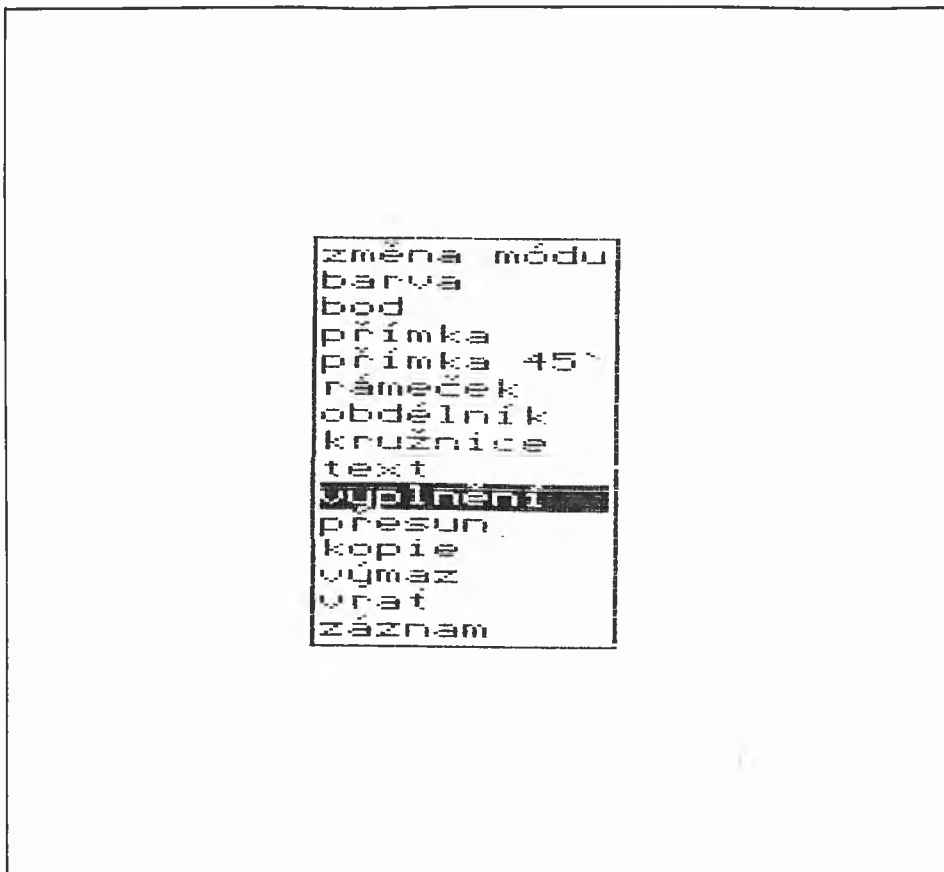
Druhý řádek.

Třetí řádek.

Čtvrtý řádek.

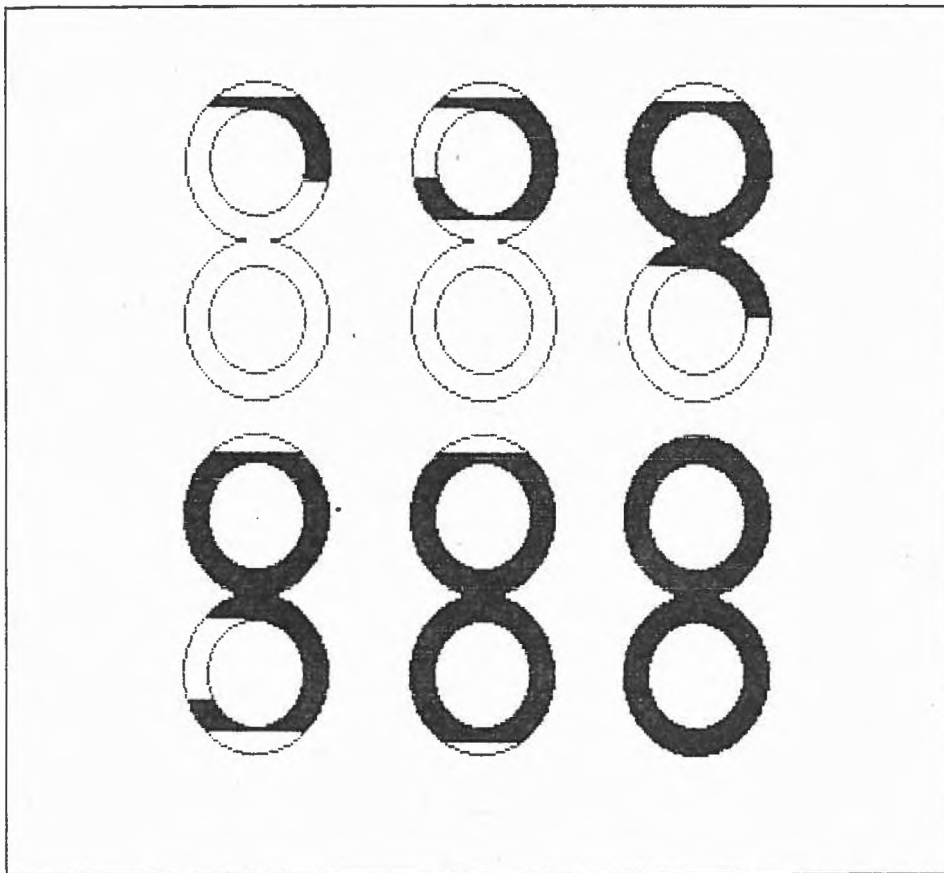
atd.□

2.6.3.11 "vyplnění"



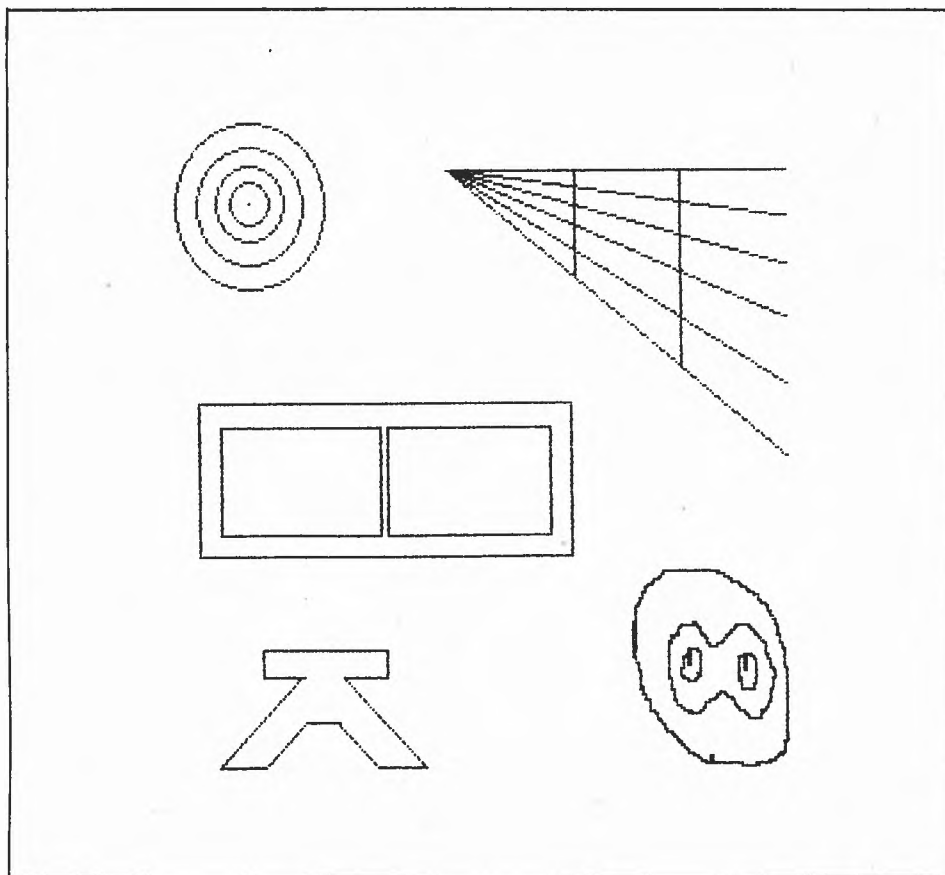
Tato funkce Vám umožní snadné vyplnění souvislé uzavřené oblasti. Oblast, kterou chcete vyplnit, musí být ohraničena souvislou čarou. Za hranici se považuje i okraj obrazovky. Pokud by v hranici někde zůstala štěrbinka, barva by vytekla z oblasti ven. To ovšem není žádná tragedie, stačí se vrátit o krok funkcí "vrat", uzavřít štěrbinu a vyplnění spustit znovu.

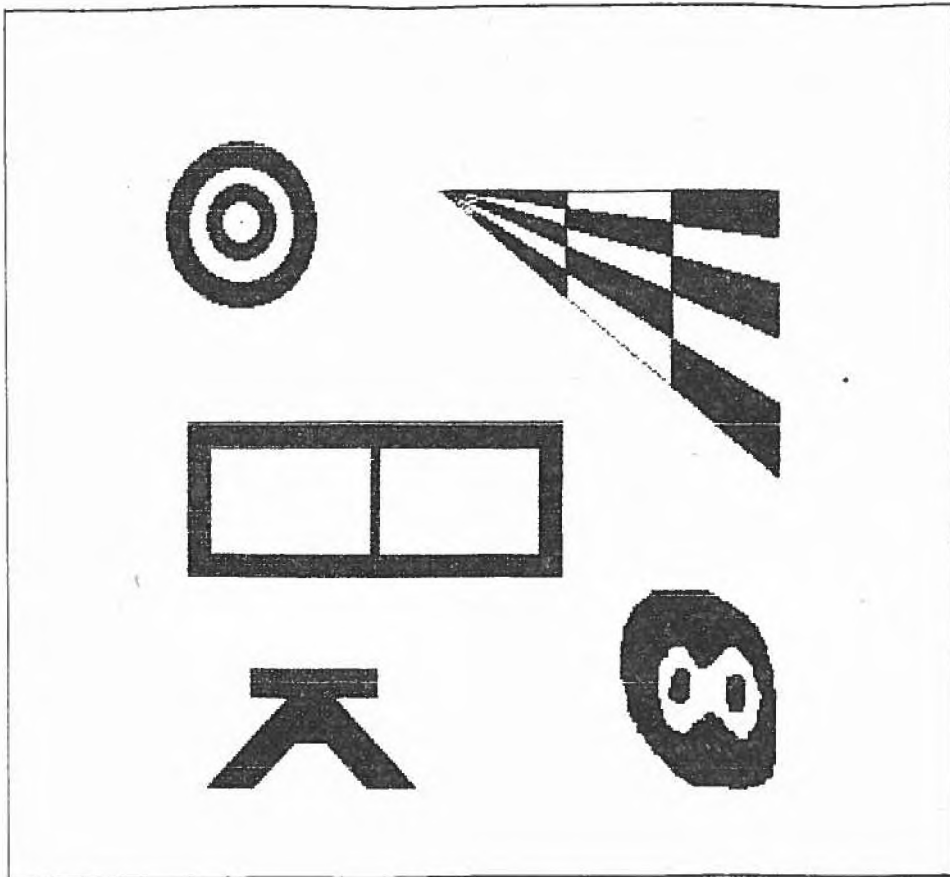
Pokud již v průběhu vyplňování, které není příliš rychlé, zjistíte, že se děje něco nechtěného, můžete vyplňování zastavit stisknutím klávesy STOP (BREAK).



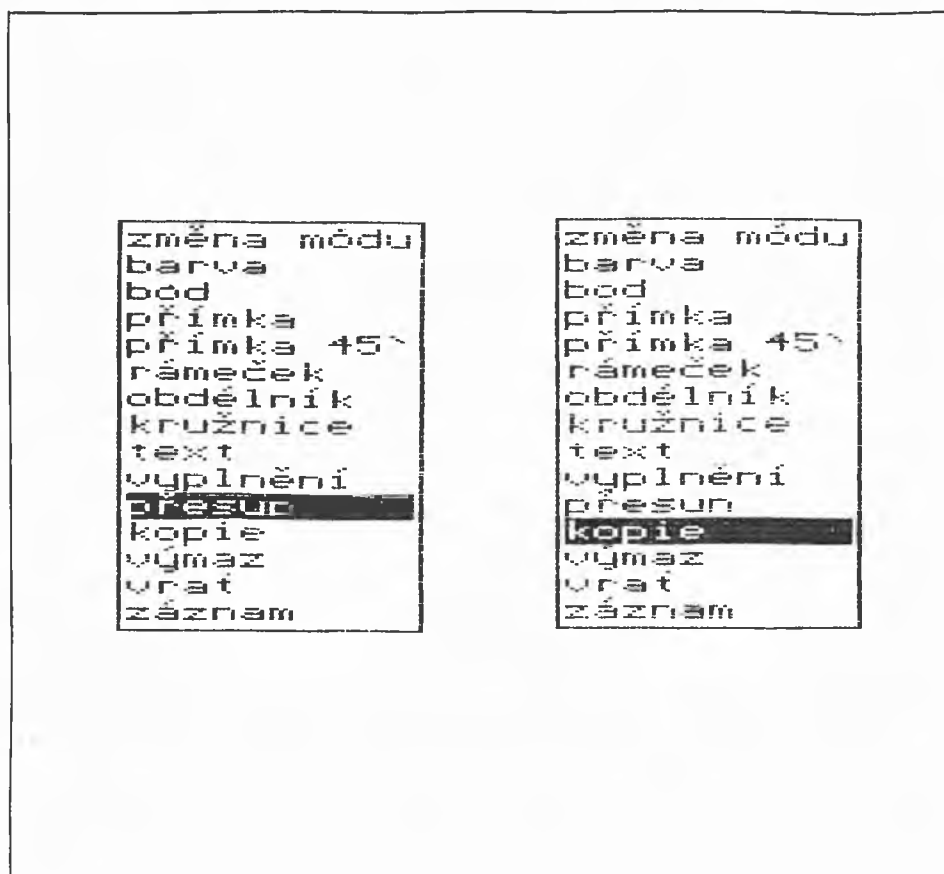
Vyplnění se spouští tak, že najedete šipkou na libovolný vnitřní bod oblasti, kterou chcete vyplnit, a krátce stisknete tlačítko SELECT.

Na proces vyplňování nemá vliv volba grafického módu, vyplnění se vždy provede do barvy popředí (log. "1" do bitmapy) nastavené funkcí "barva".





2.6.3.13 "přesun" a "kopie"

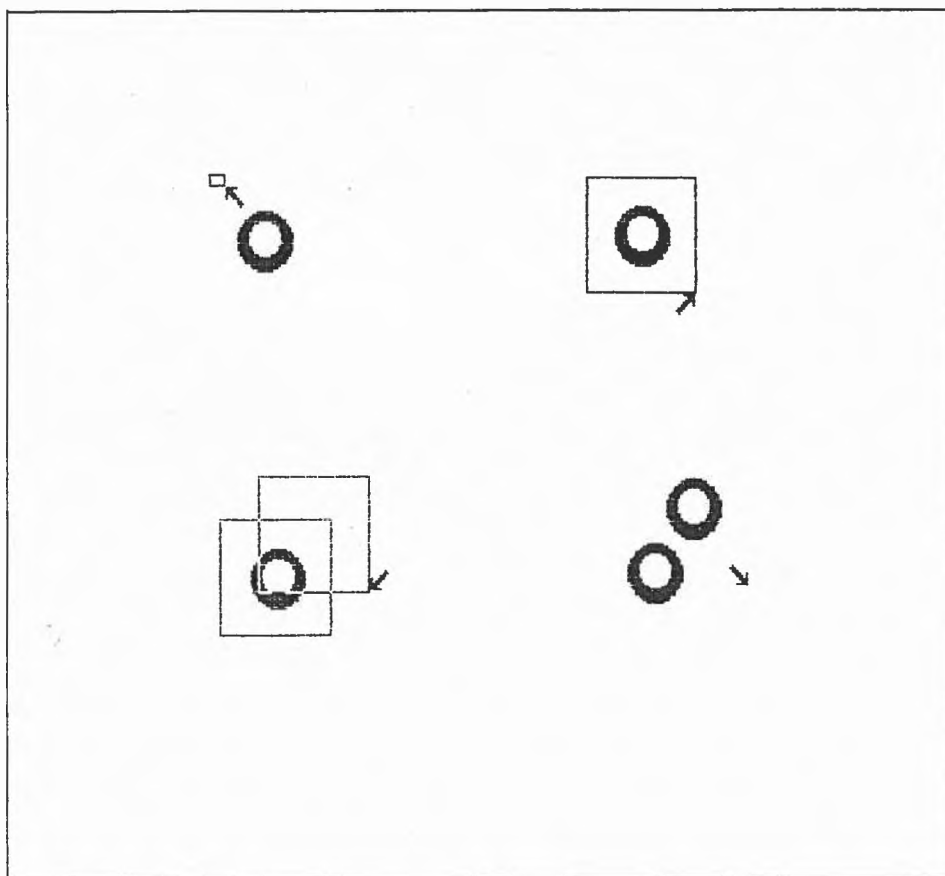


Funkce "přesun" a "kopie" neslouží ke kreslení grafického objektu, ale k přemístění či překopírování obdélníkové oblasti na jiné místo.

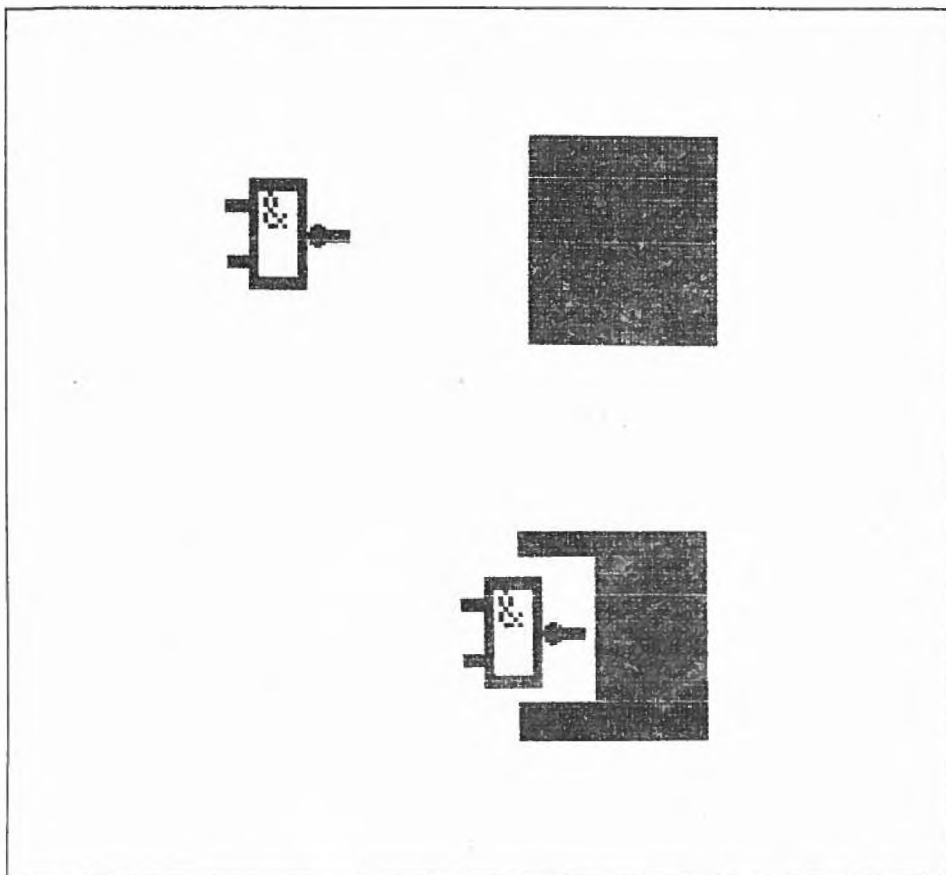
Volba zdrojové a cílové oblasti je pro obě funkce stejná a probíhá následovně:

1. Nejdříve zvolíme standardním způsobem obdélníkovou zdrojovou oblast. Počáteční bod tvoří jeden roh obdélníku, aktuální bod při puštění tlačítka SELECT tvoří druhý roh. Zdrojová oblast se pro zviditelnění zarámuje.

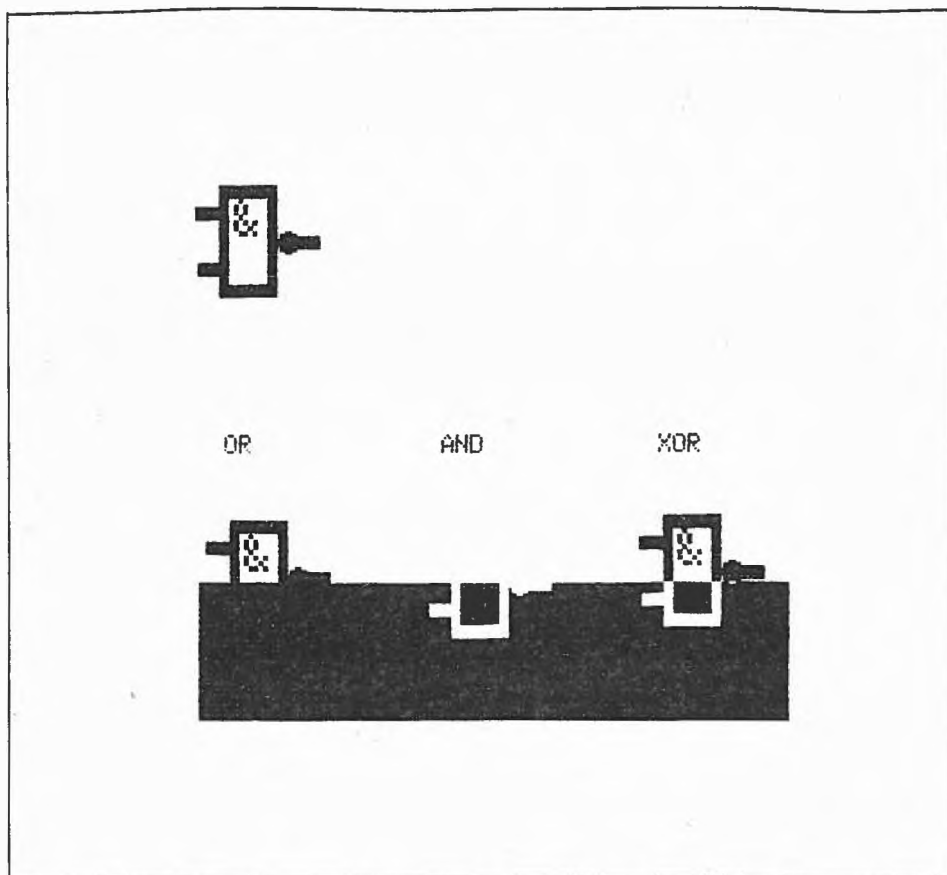
2. Druhý rámeček, stejně veliký jako je zdrojová oblast, se současně vykreslí u kurzoru a pohybem kurzoru po obrazovce pohybujeme i tímto rámečkem. Nastavíme rámeček na cílovou oblast a krátce stiskneme SELECT.
3. Podle typu zvolené funkce se zdrojová oblast přenesne nebo překopíruje do cílové oblasti.



Při funkci "přesun" se nejdříve smaže (mód "AND", funkce "obdélník", barva "nebarvit") zdrojová oblast a pak se zapíše původní obsah do cílové oblasti. Na funkci nemá vliv nastavení grafického módu, přenos je tzv. "natvrdo", cílové bity se přepíší zdrojovými bity.



Při funkci "kopie" zůstává zdrojová oblast beze změny, nic se s ní nedělá. Do cílové oblasti se data ze zdrojové oblasti zapíší ve zvoleném grafickém módu. Například přes sebe (mód "OR"), vymazat ("AND"), pouze obarvit ("COLOR"). Rozdíly při použití jednotlivých grafických módů se Vám nejlépe ozřejmí při vyzkoušení.



Je také dobré vědět, že zdrojová a cílová oblast se **mohou** překrývat! Tak je umožněno i posunutí oblasti o malý kousek, třeba jen o jeden bod v libovolném směru.



SLOVO

SLOVO

SLOVO

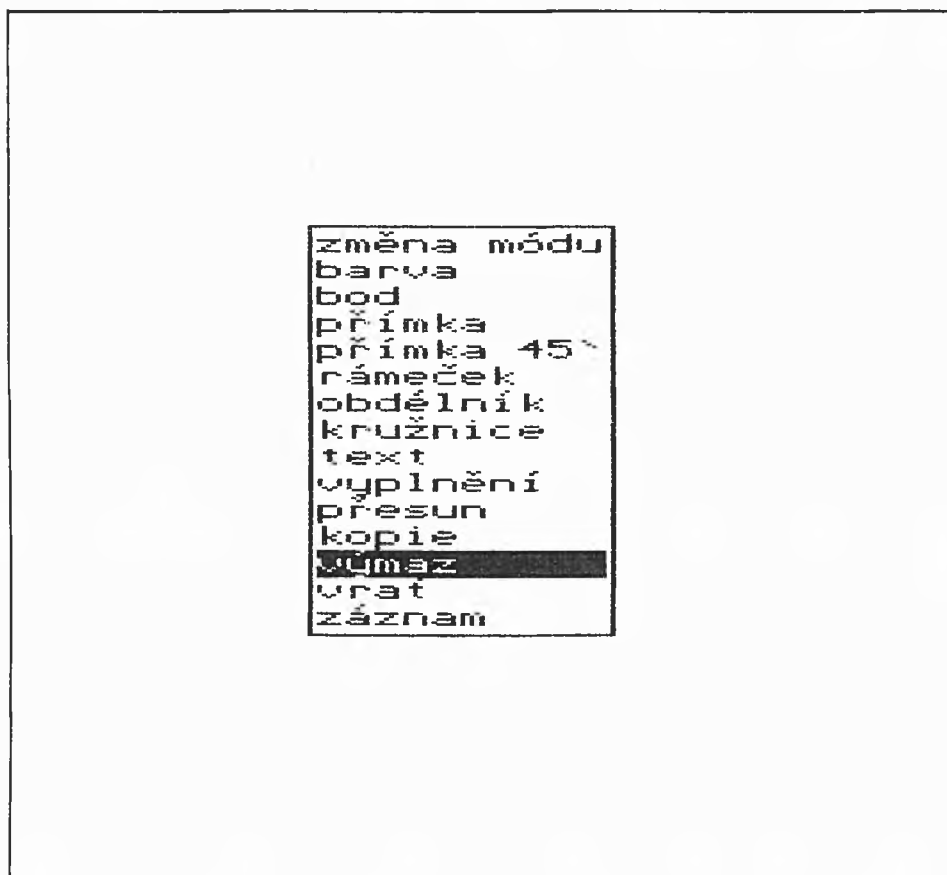
U obou funkcí mohou nastat nečekané efekty při přenášení několika barevných oblastí. To je dáno konstrukcí počítače. Jak u PMD-85, tak i u ZX-Spectra nemůže mít každý bod libovolnou barvu nezávisle na ostatních. Tímto konstrukčním řešením se zmenší potřebná kapacita obrazové paměti, ovšem za cenu omezení barevných možností.

Tak u PMD-85 vždy 6 bodů vedle sebe může mít buď barvu pozadí nebo jednu ze čtyř barev popředí, ovšem pro všech 6 bodů stejnou.

U ZX-Spectra dokonce čtvereček 8x8 bodů může mít pouze dvě barvy - jednu pro popředí a jednu pro pozadí. Podrobný popis těchto barevných atributů je v samostatné části textu u popisu adresování obrazové paměti.

Zde jen upozorňuji, že uvedené technické vlastnosti mohou u funkcí "přesun" a "kopie", stejně jako u ostatních grafických funkcí, způsobit "podivné" prolínání barev, které ale není chybou GREDITORu!

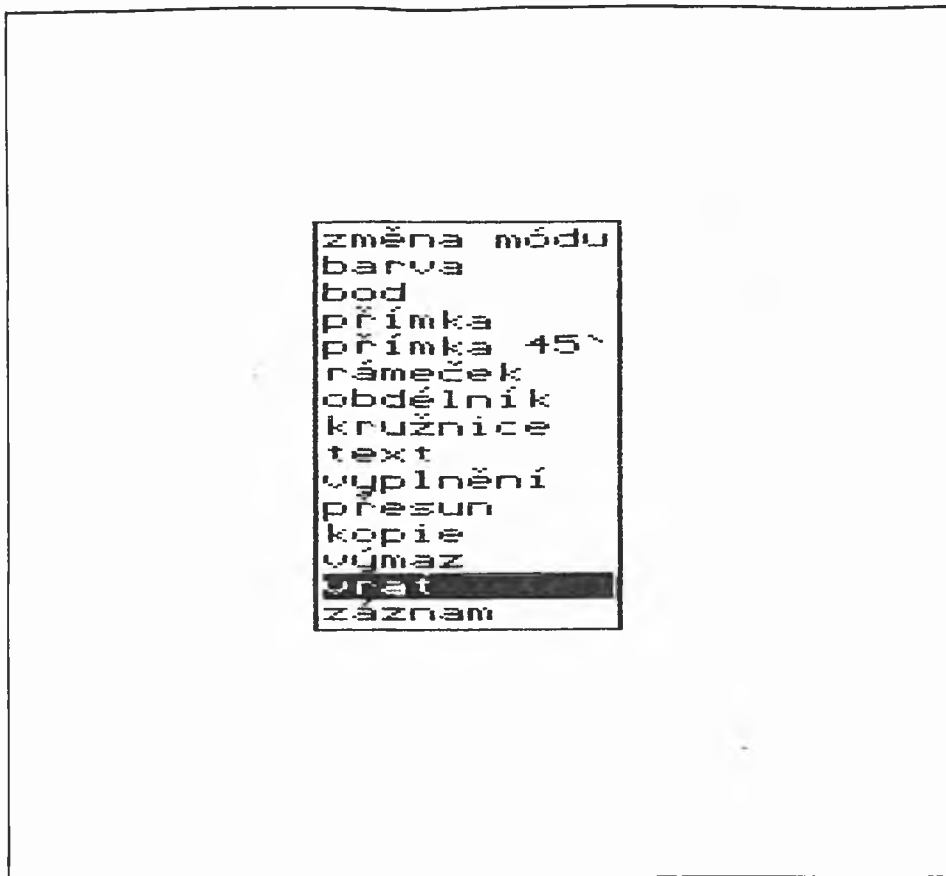
2.6.3.14 "výmaz"



Tato funkce je velice jednoduchá - smaže celý displej a nastaví barevné atributy celého displeje na zvolenou hodnotu.

Pokud vyvoláte funkci "výmaz" omylem a smažete si třeba hodinu tvořené veledílo, není třeba zoufat, od toho je funkce "vrat", pomocí které si obrázek obnovíte.

2.6.3.15 "vrať"

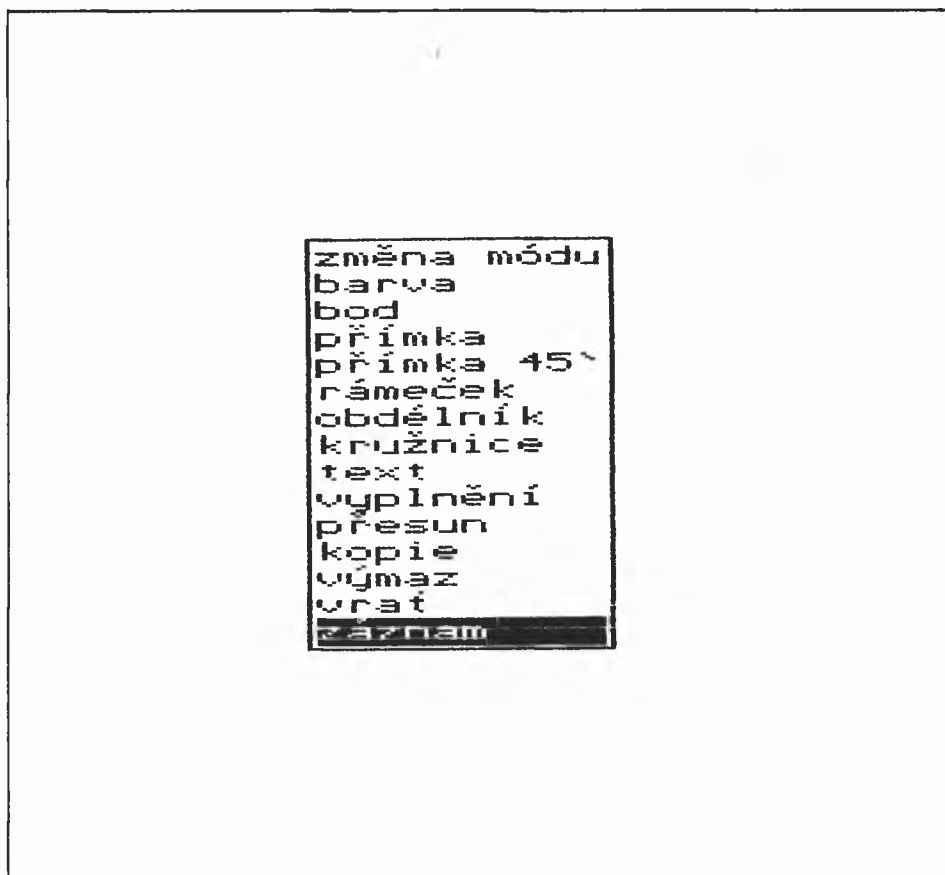


V paměti počítače je vyhražena oblast o velikosti obrazové paměti - záložní obrazovka. Do této oblasti se uloží obraz vždy při stisknutí tlačítka SELECT, jak jsem se již zmínil u popisu funkce "bod". Následující grafické funkce pak změní pouze obsah displeje. V záložní obrazovce je tedy vždy uložen stav obrázku před provedením poslední funkce.

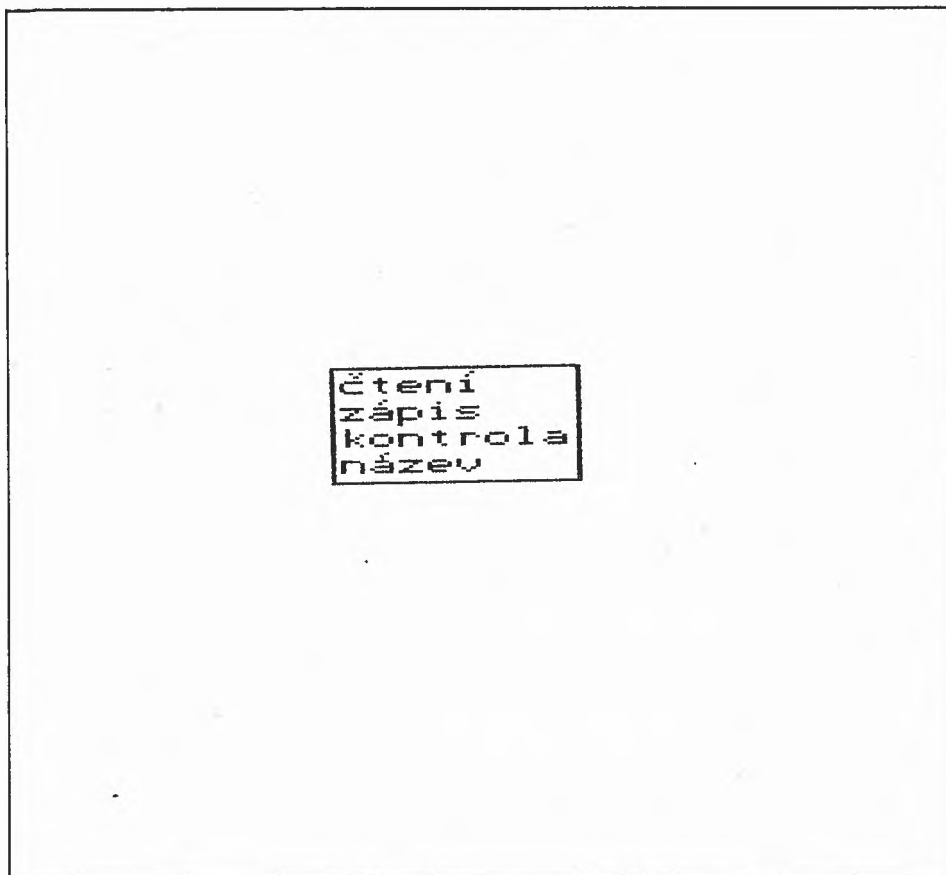
Vyvolání funkce "vrať" pouze přenese obsah záložní obrazovky do displeje - tím se obnoví stav před chybně vyvolanou funkcí. Vrácení je možné pouze o jeden krok, více záložních obrazovek se do paměti nevejde.

U funkce "text" se ukládá stav před vypsáním prvního znaku na řádce. Tím je umožněno postupné mazání znaků v řádku nebo celého textu řádky. Nový stav se ukládá při přechodu na nový řádek nebo při vyvolání jiné grafické funkce.

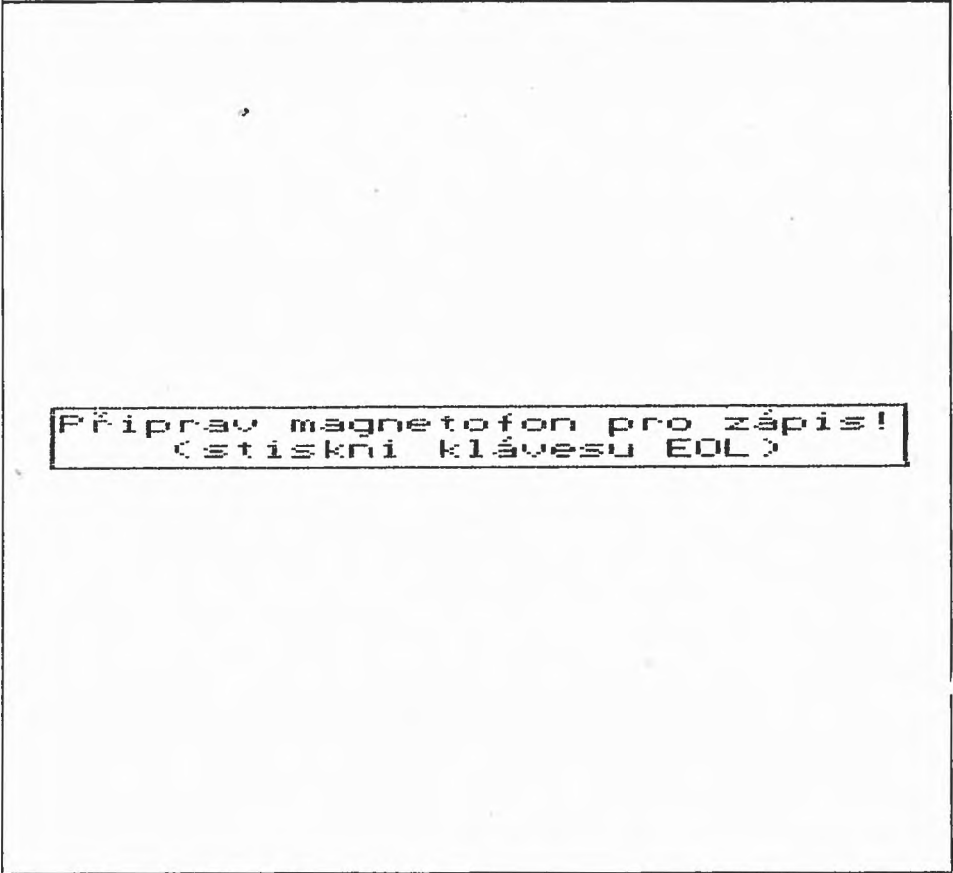
2.6.3.16 "záznam"



Poslední funkcí základní nabídky je přístup k operacím s magnetofonem. Po jejím vyvolání se vypíše další nabídka:



Po zvolení funkce "zápis" se vypíše text:



```
Připrav magnetofon pro zápis!  
(stiskni klávesu EOL)
```

Na ZX-Spectru je místo EOL napsáno ENTER.

Vložíme do magnetofonu kazetu přetočenou na volné místo, zapneme záznam a potvrdíme to stisknutím požadované klávesy.

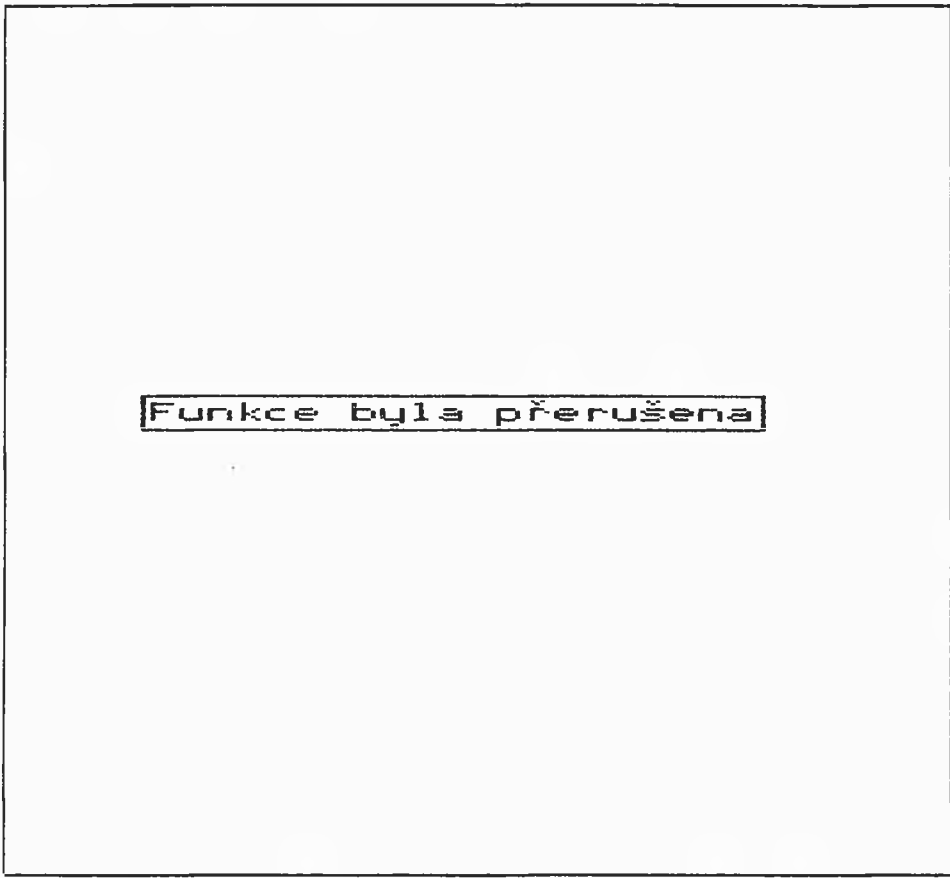
Na kazetu se ve standardním formátu zapíše obsah celé obrazovky pod předem zadaným názvem. Je vhodné nahrávku překontrolovat pomocí funkce "kontrola".

Vyvolání funkcí "čtení" a "kontrola" způsobí výpis stejného textu:

Připrav magnetofon pro čtení!
(stiskni klávesu EOL)

Nyní vložíme do magnetofonu kazetu přetočenou před nahrávku obrazovky, pustíme přehrávání a stiskneme na klávesnici požadovanou klávesu. Při "čtení" se načte obsah souboru do obrazovky, při "kontrola" se jeho obsah porovnává s obsahem obrazovky. Nakonec se vypíše hlášení o bezchybnosti načtení nebo o chybě.

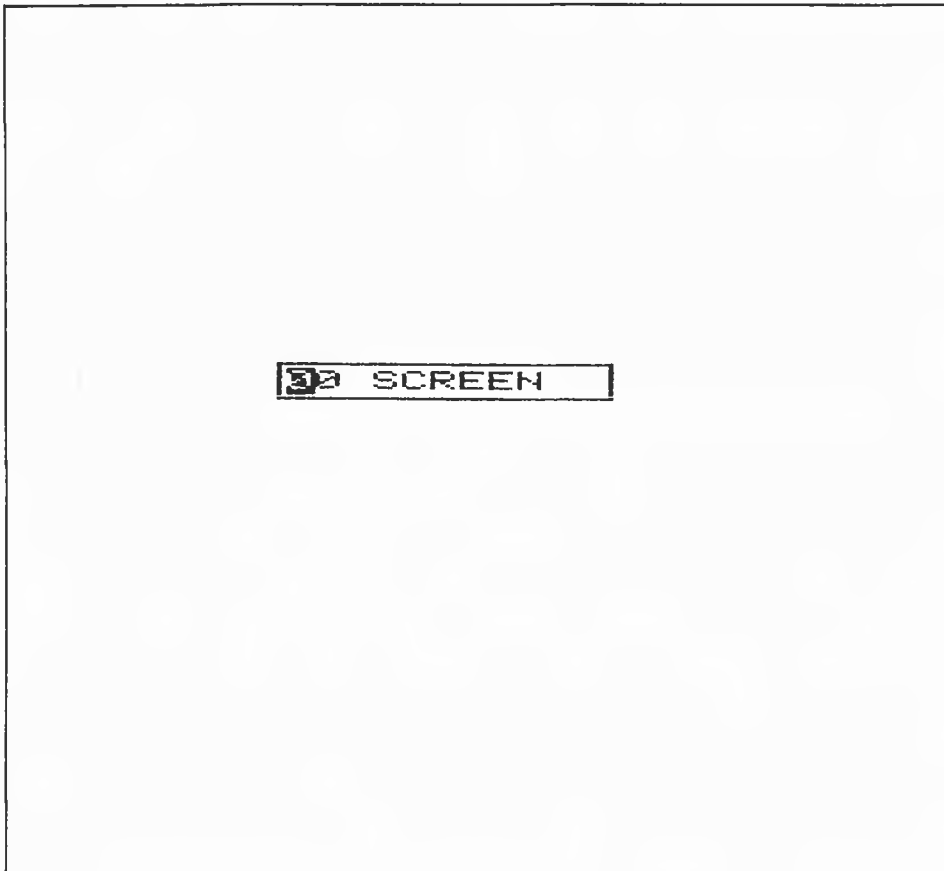
Proces čtení nebo zápisu lze kdykoliv přerušit stisknutím klávesy STOP (BREAK). V tomto případě se vypíše:



Funkce byla přerušena

Po stisknutí klávesy EOL (ENTER) můžete pokračovat v práci s GREDITOREm.

Poslední funkce magnetofonové nabídky - "název" - slouží k zadání názvu souboru pro čtení, kontrolu nebo zápis. Vypíše se aktuální název, který pomocí klávesnice můžete změnit. Zadání se ukončí klávesou EOL (ENTER).



U PMD-85 je součástí názvu i číslo souboru, pomocí kterého ho můžete načíst standardní funkcí monitoru MGLD.

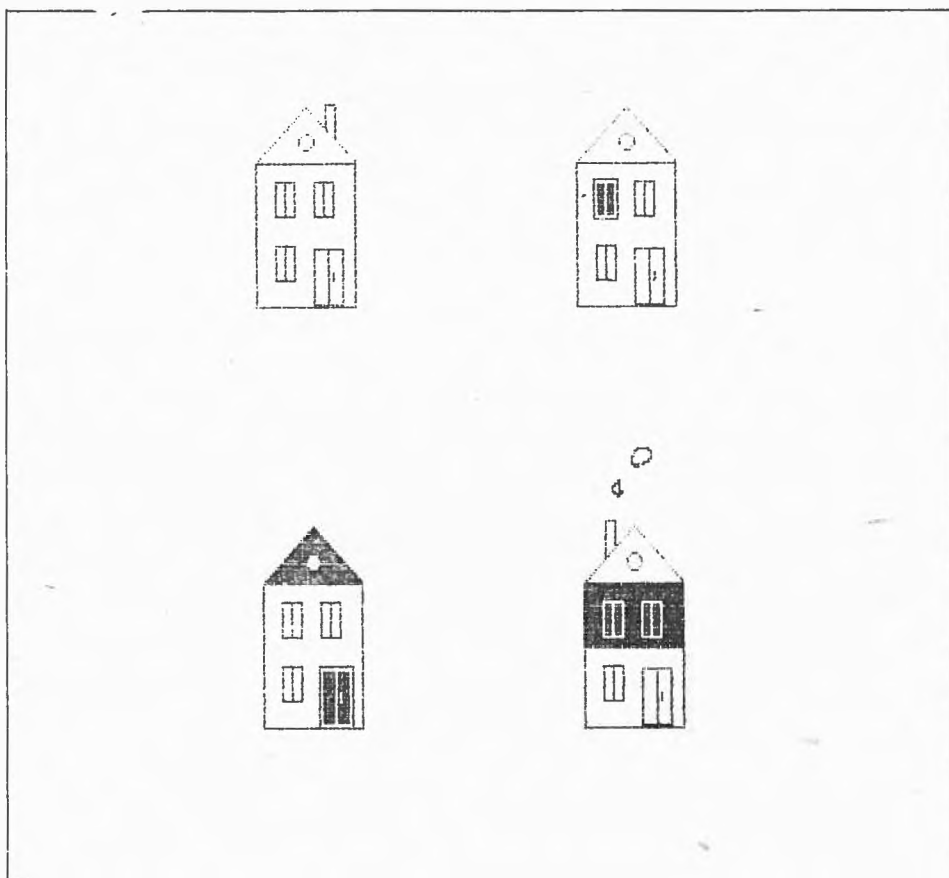
Na ZX-Spectrum lze s nahrávkou pracovat pomocí funkcí LOAD "" SCREEN\$ a SAVE "" SCREEN\$.

2.6.4 Metodika práce s GREDITORem

Pro úspěšnou práci s GREDITORem není třeba znát žádné speciální postupy. Pouze pro urychlení a usnadnění práce je výhodné využívat všechny možnosti, které Vám GREDITOR poskytuje. V dalším textu Vás upozorním na několik základních možností.

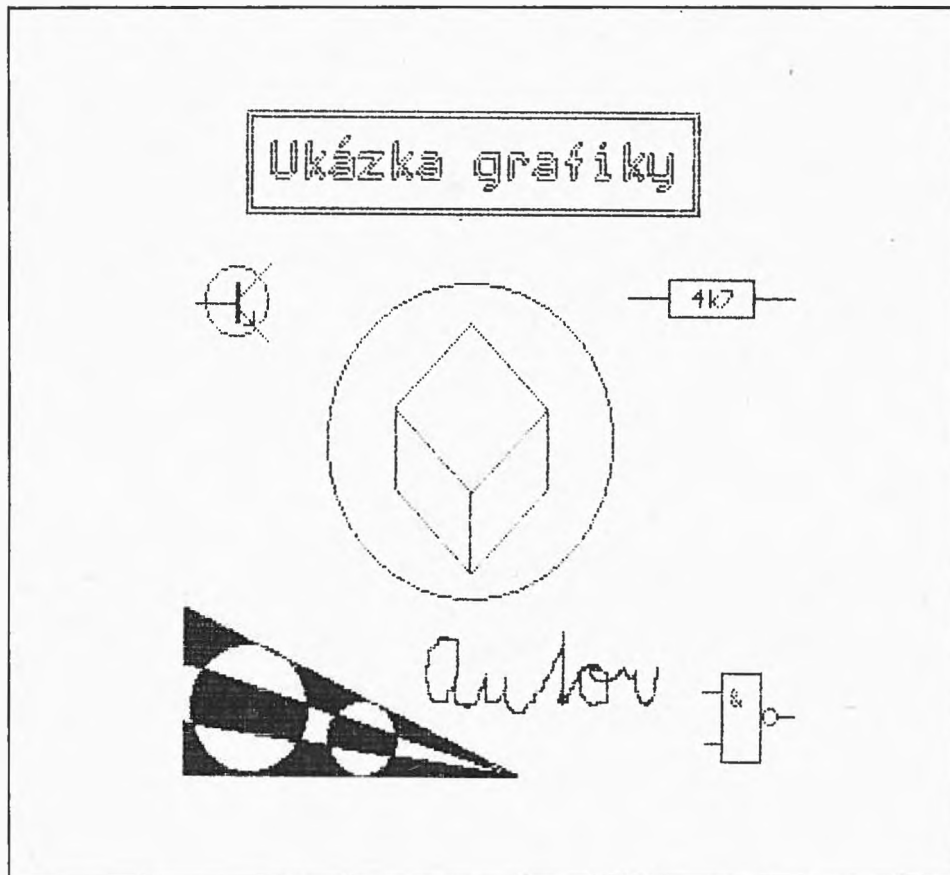
Díky funkci "přesun" není nutné kreslit daný grafický objekt přímo na přesné místo. Většinou zpočátku ani nevíme, jak velký bude výsledný obrázek a o jeho rozložení na displeji nemáme přesnou představu. Nevadí, nakreslíme vše tak, jak to vyjde, a teprve hotový objekt přemístíme na cílové místo. Stejně tak si můžeme někde na straně připravit textové popisky grafu a hotové je přesunout na přesné místo.

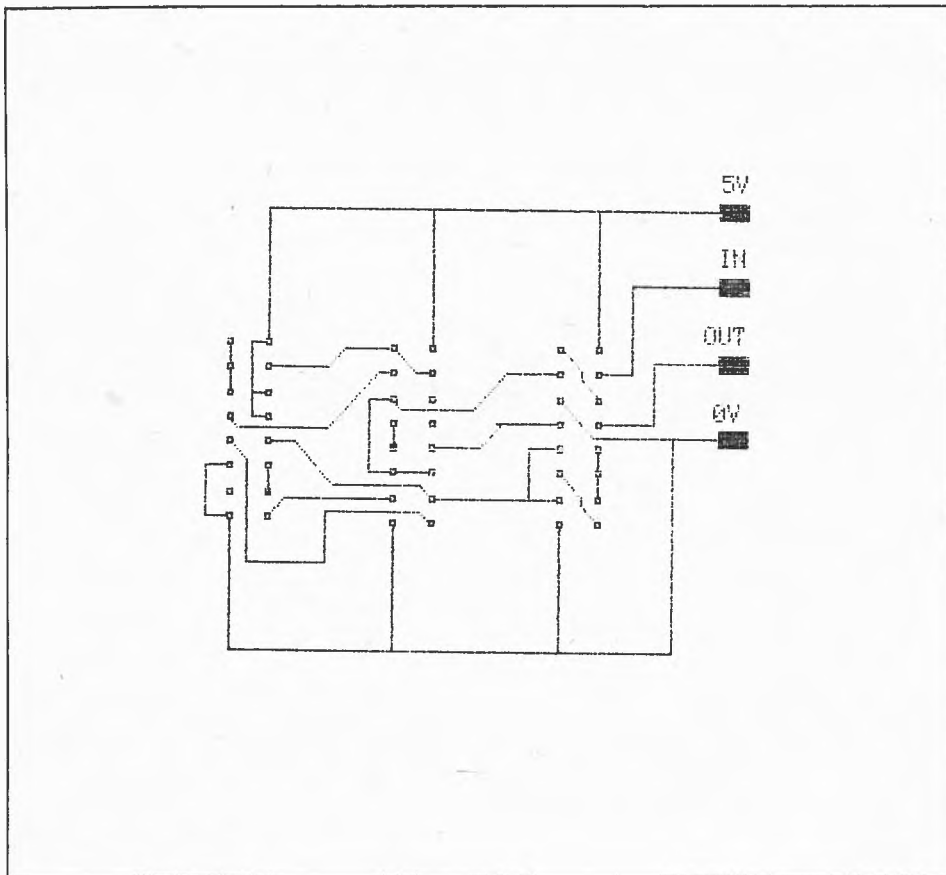
Velmi výhodné je využití funkce "kopie", pokud se v obrázku opakuje vícekrát stejný motiv. Připravíme si požadovaný motiv a ten překopírujeme na všechna další místa. Výhodou je nejen urychlení práce, ale i to, že všechny kopie jsou přesně stejné, což by se nám při "ručním" kreslení těžko povedlo. Po překopírování lze také jednotlivé kopie "dozdobit" detaily.

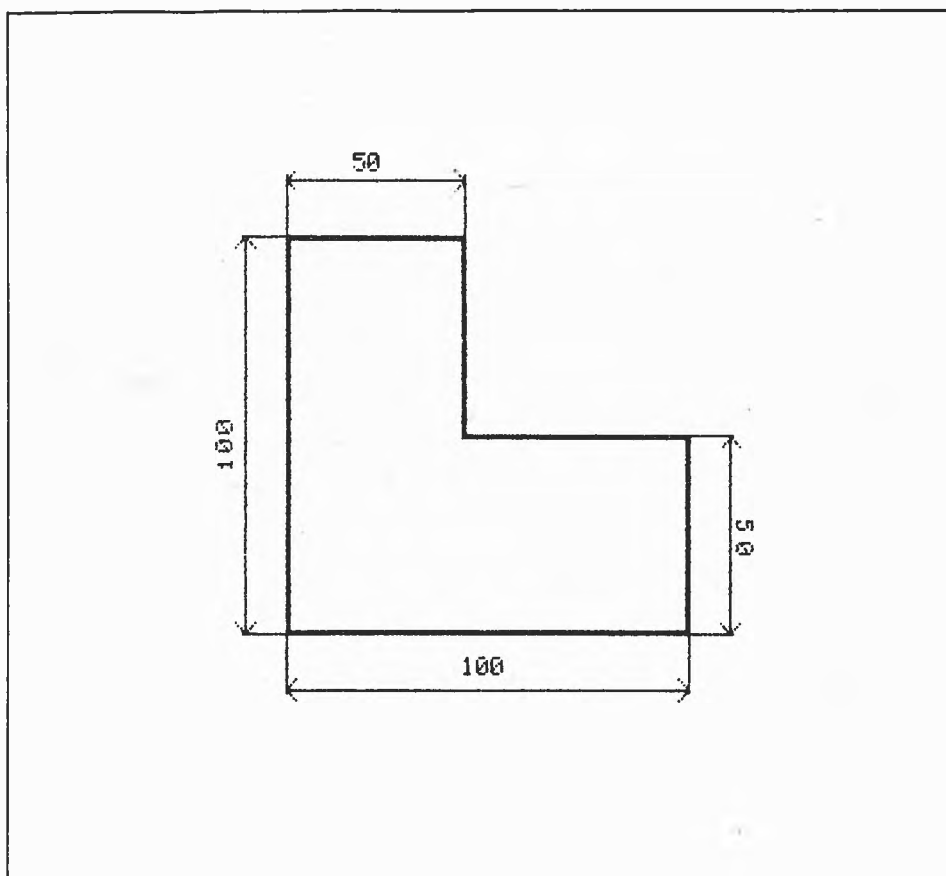


Při kreslení obrázku s využitím barev je také užitečné využití grafického módu "COLOR" pro vybarvení hotového obrázku. Nejdříve nakreslíme celý obrázek jednou barvou, teprve když je hotov, nastavíme mód "COLOR", volíme vhodné barvy a obrázek vybarvujeme. Pro větší stejnobarevné plochy je vhodné využití funkce "obdélník". Pro barvení menších nebo nepravidelných tvarů lze použít funkci "bod". Krátkým stiskem tlačítka SELECT změním jeden barevný atribut, při držení tl. SELECT můžeme pohybem myši vybarvovat obrázek jako štětcem.

2.6.5 Ukázky obrázků vytvořených pomocí GREDITORu





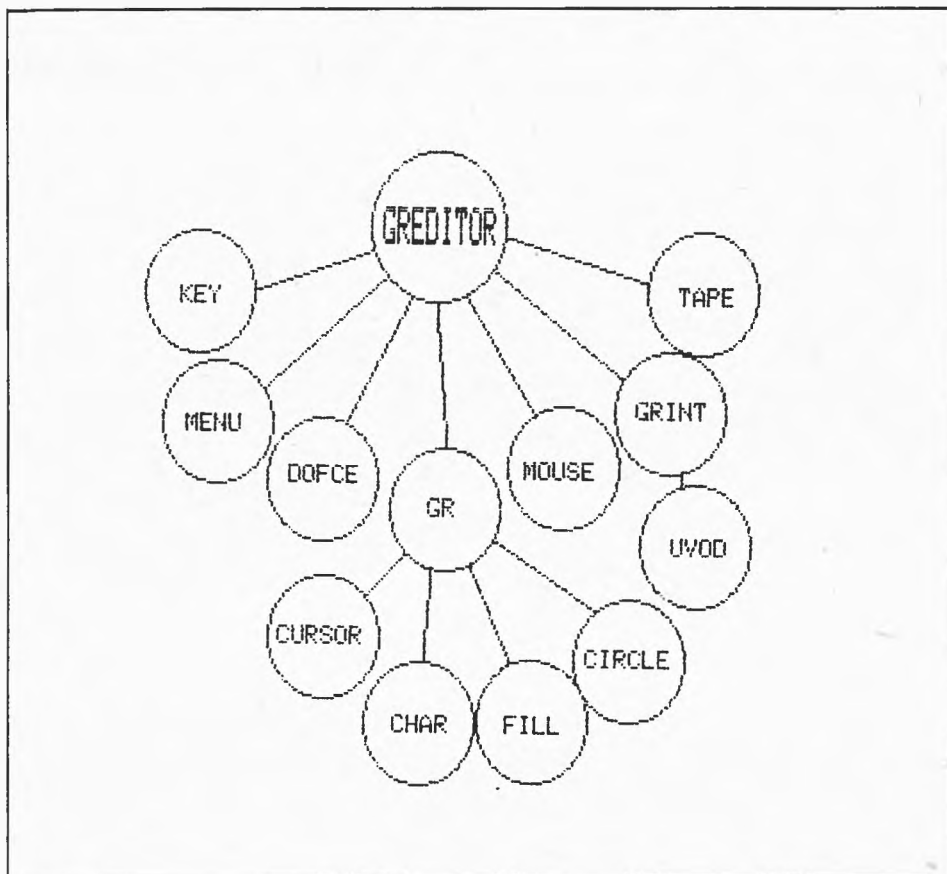


2.7 Modularita, přenositelnost

Celý grafický editor je složen z několika samostatných modulů. Kromě hlavního lze všechny moduly využít i v jiném aplikačním programu - ať už komunikaci s myší nebo grafické podprogramy. Některé moduly jsou nezávislé na hardware počítače (DOFCE, FILL, CIRCLE), jiné jsou určeny pro konkrétní počítač (MOUSE, MENU, KEY). Všechny tyto programové moduly jsou napsány v jazyce symbolických adres (assembleru) a jsou samostatně překládány. Dohromady jsou spojeny sestavovacím programem (linkrem).

Grafický editor GREDITOR je sestaven z těchto modulů:

GREDITOR	- hlavní modul definující grafický editor
MENU	- volba z nabídky
DOFCE	- volba grafických funkcí
CURSOR	- kreslení kurzorové šipky
MOUSE	- komunikace s myší
FILL	- vyplnění plochy
CIRCLE	- kreslení kružnice
GRINT	- grafický interpret
UVOD	- úvodní obrázek
GR	- soubor základních grafických podprogramů
KEY	- čtení znaků z klávesnice
TAPE	- komunikace s magnetofonem
CHAR	- tvary znaků



U grafických programů vystupuje do popředí problém přenositelnosti. Je možné napsat všechny grafické podprogramy tak, že závislý na hardware počítače je pouze podprogram kreslící jeden bod a všechny ostatní podprogramy pak využívají tuto funkci a jsou nezávislé. Toto řešení je velice jednoduché, ale podprogramy jsou pak nepoužitelně pomalé. Všechny základní grafické podprogramy je proto nutné "ušít" pro daný mikropočítač. Pro jiný typ počítače se musí napsat znovu. Naštěstí nikoli úplně znovu - princip je přece stále stejný. Zde poskytuje neocenitelné služby MACRO - assembler, pro vytvoření podprogramů pro jiný počítač stačí napsat několik jednoduchých MACRO definic, přeložit a je hotovo. Podprogramy na každém počítači jsou maximálně rychlé a nebylo je nutné psát znovu!

V dané verzi GREDITORu jsou použity tyto grafické podprogramy závislé na hardware:

bod, úsečka, vodorovná úsečka, svislá úsečka, přesun, kopie.

Všechny další funkce (rámeček, obdélník, přímka 45°, vyplnění, kružnice, text) jsou již na hardware nezávislé a přesto jsou dostatečně rychlé.

2.8 Umístění GREDITORu v paměti

Pro laického uživatele není důležité, kde co v paměti počítače je uloženo, ale jak to celé funguje. Přesto pro zajímavost a úplnost informací zde uvedu využití paměti počítačů PMD-85 a ZX-Spectrum GREDITORem.

V prvním sloupci za názvem modulu je adresa v paměti PMD-85 a ve druhém sloupci adresa v paměti ZX-Spectra.

GREDITOR	0100H	8000H	Hlavní programový modul. Definuje celý vnější projev grafického editoru.
MENU	0703H	877BH	Volba z nabídky. Provádí uložení pozadí, výpis textu nabídky, volbu funkce a obnovení pozadí.
DOFCE	091EH	8987H	Volba souřadnic pro vykreslení grafického objektu, dynamické kreslení.
CURSOR	0A6CH	8AD7H	Kreslení kurzorové šipky.

MOUSE	0B80H	8C62H	Komunikace s myší. Inicializace hardware, ošetření cyklického časového přerušení, test signálů z myši, tvorba souřadnic, realizace tabulky typu FIFO.
FILL	0D43H	8DFAH	Vyplnění uzavřené oblasti.
CIRCLE	0E32H	8EEFH	Vykreslení kružnice.
GRINT	0EB6H	8F78H	Grafický interpret - kreslí obrázek podle vektorové tabulky. Používá se k vykreslení úvodního obrázku.
UVOD	0F68H	902AH	Vektorová tabulka úvodního obrázku.
GR	1076H	9138H	Základní grafické funkce - adresace do displeje, volba grafického módu, volba barevných atributů, bod, úsečka, rámeček, obdélník, text, přesun, kopie, úschova a vrácení obrazové paměti...
KEY	15FEH	95E7H	Čtení znaků z klávesnice.
TAPE	18A8H	9673H	Komunikace s magnetofonem - zápis, čtení a kontrola souborů, zadání názvu souboru.

CHAR	1C15H	979CH	Znakový generátor - tabulka tvarů písmen a dalších grafických znaků.
DSEG	2800H	A800H	Datový segment. Prostor pro systémové proměnné a pracovní tabulky.
SCRBUF	4500H	C000H	Místo pro uložení obrazové paměti pro funkci "vrať".
BFMENU	7000H	DD00H	Místo pro uložení pozadí pod textem nabídky.
SPLow	7E00H	DB00H	Dolní okraj místa pro procesorový zásobník.
SPBAZE	8000H	DD00H	Horní okraj místa pro procesorový zásobník.

1. HARDWARE

pro připojení myši

2. GREDITOR

- grafický editor

3. SOFTWARE

pro myš a editor

3. SOFTWARE PRO MYŠ

Tato kapitola je určena zejména programátorům. Dozvíte se v ní o základních podprogramech, které jsou potřebné k realizaci řízení programu pomocí elektronické myši a vytvoření grafického editoru. Nejdříve bude nanesen problém, který je třeba programově vyřešit, potom princip a algoritmy jeho řešení a nakonec, pokud to délka umožní, výpis podprogramu, který diskutovanou funkci využívá nebo přímo realizuje.

V ukázkách podprogramů je použito mnemoniky mikroprocesoru Z-80, ale v podprogramech pro PMD-85 jsou použity pouze ty instrukce, které "umí" i mikroprocesor 8080.

3.1 K čemu slouží myš

Základní princip myši spočívá v tom, že počítač může snímat pohyb myši po podložce a podle něj odpovídajícím způsobem provádět změny na obrazovce. Tyto změny mohou být rozličných principů. Nejznámější je pohyb kurzoru po obrazovce, ale smysluplné jsou třeba i tyto:

- řízení pohybu zobrazovaného okénka po textu,
- řízení rychlosti výpisu textu,
- řízení pohybu postavičky ve hře.

My zůstaneme u toho nejrozšířenějšího principu - řízení pohybu kurzoru po obrazovce, vyvolání nabídky a výběru z ní. Co to znamená pro programátora?

1. Z impulsů snímaných z myši průběžně zjišťovat směr a rychlost jejího pohybu a podle těchto veličin měnit aktuální souřadnice kurzoru.

2. Na aktuálních souřadnicích vykreslit kurzor (třeba šipku), při změně souřadnic jej smazat a vykreslit na novém místě, přitom zachovat beze změny podklad, po kterém se kurzor pohybuje.
3. Při stisknutí tlačítka na myši vyvolat nabídku, umožnit výběr funkce z této nabídky a pak nabídku smazat, opět bez porušení podkladu.

Při realizaci grafického editoru přibudou další úkoly:

4. Grafické funkce - kreslení grafických objektů.
5. Dynamické kreslení - volba souřadnic.
6. Zrušení naposledy vyvolané funkce.
7. Hlavní smyčka grafického editoru.

V dalším textu postupně probereme všechny tyto úkoly a nakonec si ještě stručně povíme o jedné další konkrétní aplikaci, kde jsou všechny popsány podprogramy použity - Simulátoru logických obvodů.

3.2 Generace souřadnic pomocí myši

O tom, co znamenají signály z myši, již řeč byla dříve. Nyní si povíme o tom, jak tyto signály zpracovat a převést na souřadnice. Budu popisovat zpracování signálů YA a YB, zpracování XA a XB je naprosto shodné neboť svislý a vodorovný pohyb myši se odečítá stejným způsobem.

3.2.1 Čekání na změnu

Při svislém pohybu myši po podložce se střídavě mění hodnota dvou příslušných signálů: YA a YB. Změna logické hodnoty libovolného z nich znamená posun myši o vzdálenost asi 1 mm, což představuje změnu souřadnice o 1. Dokud se signály nemění - myš stojí anebo se ještě nepohnula o potřebnou vzdálenost. Při programovém sledování signálů z myši tedy čekáme na jejich změnu. To znamená, že musíme mít v paměti uložen minulý stav signálů a s ním srovnávat nový stav, načtený z vstupní brány. Porovnáním obou vzorků zároveň zjistíme směr pohybu.

3.2.2 Vyhodnocení změny a posun souřadnic

Test změny a vyhodnocení směru pohybu lze realizovat současně pomocí dvourozměrné tabulky. Prvním indexem je minulý stav (0..3) a druhým indexem je nový stav (0..3). Tabulka má 16 položek, které mají tři různé hodnoty: -1, 0 nebo 1.

Při změně souřadnice je ještě třeba sledovat překročení mezí - okrajů obrazovky.

Následující podprogram realizuje test dvou vzorků signálů, změnu souřadnic a test překročení mezí 0..255.

```
; Vstupní parametry:  
;      B = nový stav signálů (bity 0 a 1)  
;      C = starý stav signálů (bity 0 a 1)  
;      A = stará souřadnice
```

```
; Výstupní parametry:  
;      A = nová souřadnice
```

```
minl::
```

```

push    de           ;uschovat registr DE
push    af           ;uschovat starou souřadnici
ld      a,b          ;nový stav signálů
rlca                    ;vynásobit 4 pro indexaci
rlca
and     00001100b    ;vynulovat ostatní bity
ld      d,a          ;uložit 1. index

ld      a,c          ;starý stav signálů
and     00000011b    ;vynulovat ostatní bity
add     a,d          ;sečíst oba indexy
ld      e,a          ;létí bitový index do DE
ld      d,0
ld      hl,tab       ;adresa tabulky do HL
add     hl,de        ;indexace do tabulky
pop     af           ;stará souřadnice do A
ld      e,a          ; do DE
ld      a,(hl)       ;vybrat položku z tabulky
ld      l,a          ; do L
rlca                    ;rozšířit znaménko
sbc     a,a
ld      h,a          ;offset z tabulky do HL
add     hl,de        ;stará souřadnice + offset =
                    ; nová souřadnice

ld      a,h          ;test překročení mezí 0..255
or      a            ;mimo rozsah?
ld      a,e          ;původní souřadnice
pop     de           ;obnovit původní obsah DE
ret     nz           ;AND, mimo rozsah =
                    ; vrátit původní souřadnici

ld      a,l          ;nová souřadnice
ret                    ;návrat z podprogramu

```

```

tab:    db      0,-1,1,0 ;tabulka offsetů
        db      1,0,0,-1

```

```
db      -1,0,0,1
db      0,1,-1,0
```

3.2.3 Test tlačítek

U tlačítek na myši nás zajímá změna jejich stavu, stisknutí a puštění, ale i jejich aktuální stav. Pro zjištění změny musíme mít v paměti uložen jejich minulý stav, který porovnáme s novým stavem.

Následující podprogram realizuje tento test a vrací aktuální stav tlačítek a příznak změny jejich stavu:

```
; Vstupní parametry:
;      B = nový status myši (stav vstupní brány)
;      C = starý status myši

; Výstupní parametry:
;      C = aktuální stav tlačítek:
;          bit 7 = pravé, bit 6 = levé
;          "1" = stisknuto, "0" = puštěno
;      C = příznak změny stavu tlačítek:
;          bit 5 = pravé, bit 4 = levé
;          "1" = změna stavu, "0" = beze změny

tlacitka::
    ld      a,b          ;nový status myši
    xor     c            ;změna stavu?
    and     11000000b    ;ponechat pouze tlačítkové
                        ;bity
    jp     z,tlc2        ;beze změny

; Nahodit čítač pro potlačení zákmitů.

    push   af           ;uschovat příznaky změn
```



```

ld      a,150      ;časová konstanta 0,15 sec.
ld      (tlcitac),a ;zapsat do čítače
pop     af         ;obnovit příznaky změn

tlc2:   rrca              ;přesunout bity 7 a 6 do 5 a 4
        rrca
ld      c,a        ;uschovat do C
ld      a,b        ;nový status myši
and     l1000000b  ;stav tlačítek
or      c          ;připojit příznaky změny
ld      c,a        ;uložit výsledek do C
ret     ;návrat z podprogramu

```

3.2.4 Potlačení zákmitů tlačítek

Jako u každého mechanického kontaktu, i u tlačítek myši dochází při jejich stisknutí a puštění k přechodovému jevu - zákmitům. To se projeví tak, že logická hodnota výstupního signálu se nezmění pouze jednou z jedné hodnoty na opačnou, ale několikrát se velmi rychle změní sem - tam. Nepřipravený program by tento jev vyhodnotil jako několikeré stisknutí a puštění tlačítka, což by vedlo k nechtěným a nepříjemným efektům. Proto je nutné s tím počítat a zákmity programově ošetřit. Princip je velmi jednoduchý - při zjištění změny stavu tlačítka tuto změnu zpracovat, ale pro následující chvíli test změny stavu tlačítek myši potlačit. Délka tohoto potlačení se osvědčila 150 msec.

Popsaný princip lze realizovat třeba tak, že po stisknutí tlačítka nějaký čas (měřeno čítačem přerušení) předáváme k vyhodnocení minulý stav tlačítek, ne aktuální - tak to realizuje následující podprogram:

```

; Vstupní parametry:
;      žádné

```

; Výstupní parametry:

; B = nový status myši
; C = starý status myši

odskoky::

```
ld      a,(last)    ;vybrat minulý status myši
ld      c,a         ; do C
in      a,(pin)     ;načíst nový status myši ze
                    ;vstupní brány
ld      b,a         ; do B

ld      hl,tlcitac  ;adresa čítače do HL
ld      a,(hl)     ;je už čítač nulový?
or      a
jp      z,odskl    ;ANO = normální test
dec     (hl)       ;NE = snížit čítač
ld      a,c        ;předat starý status
and     11000000b  ;bity tlačítek
ld      d,a        ; do D
ld      a,b        ;nový status
and     00111111b  ;vynulovat tlačítkové bity
or      d          ;přidat starý stav tlačítek
ld      b,a        ;výsledný status do B

odskl:  ld      a,b    ;uložit nový status do paměti
        ld      (last),a ;pro příští test
        ret         ;návrat z podprogramu

tlcitac: db      0    ;čítač časového intervalu pro
                    ;potlačení odskoků tlačítek
last:   db      0    ;status myši
```

3.2.5 Celkové vyhodnocení signálů z myši

S využitím předchozích podprogramů si nakonec ukážeme realizaci hlavního podprogramu, který provede vyhodnocení signálů z myši a příslušnou změnu souřadnic. Tento podprogram je zcela funkční a je použit v GREDITORu prakticky beze změny.

```
; Vstupní parametry:
;      žádné

; Výstupní parametry:
;      L = nová souřadnice X (0..255)
;      H = nová souřadnice Y (0..255)
;      A = status myši:
;          bit 7 = stav pravého tlačítka
;          bit 6 = stav levého tlačítka
;          "1" = stisknuto, "0" = puštěno
;          bit 5 = příznak změny pravého tlačítka
;          bit 4 = příznak změny levého tlačítka
;          "1" = změna stavu, "0" = beze změny
;          bit 0 = příznak změny souřadnic
;          "1" = změna souřadnic, "0" = beze změny
;      ZF = příznak jakékoli změny (tlačítka, souřadnice)
;          Z = beze změny
;          NZ = je změna

test::
    push    de            ;uschovat obsah DE
    push    bc            ; a BC
    call    odskoky       ;příprava nového a starého
                        ;statusu do B a C
    push    bc            ;uschovat statusy
    ld      a,(mousey)    ;stará souřadnice Y
    call    minl          ;změna souřadnice Y
```

```

ld      h,a          ;nové Y do H

ld      a,b          ;nový status
rrca                    ;bity XA a XB posunout do YA a
                        YB pro vyhodnocení

rrca
ld      b,a          ; do B
ld      a,c          ;starý status
rrca                    ;stejné posunutí bitů
rrca
ld      c,a          ; do C
ld      a,(mousex)  ;stará souřadnice X
call   minl          ;změna souřadnice X
pop    hl            ;v H je nové Y
ld      l,a          ;nové X do L
pop    bc            ;nový a starý status

call   tlacitka     ;vyhodnocení tlačítek

```

; Ještě zbývá otestovat změnu souřadnic a nastavit příznak.

```

ex      de,hl        ;nové X a Y do DE
ld      hl,(mousex) ;staré souřadnice do HL
ex      de,hl        ;přehodit nové se starými
ld      (mousex),hl ;uložit nové souřadnice

ld      a,d          ;staré Y
cp      h            ;porovnat staré Y s novým Y
jp      nz,zmena     ;změna

ld      a,e          ;staré X
cp      l            ;porovnat staré X s novým X
jp      z,nezmena    ;beze změny

```

```

zmena:  inc      c          ;nastavit příznak změny:

```

```

; "1" do bitu 0 v C
nezmena: ld      a,c          ;nový status (úplný)
         and     00110001b   ;test na jakoukoliv změnu
         ld      a,c          ;nový status se vrací v A
         pop     bc          ;obnovit původní BC
         pop     de          ; a DE
         ret                ;návrat z podprogramu

mousex:: db      80h         ;souřadnice X
mousey:: db      80h         ;souřadnice Y

```

3.2.6 Program pro kreslení

Zde si ukážeme použití výše popsaného podprogramu TEST pro komunikaci s myší na příkladu jednoduchého programu pro kreslení bodů a čar. Jeho funkce je taková, že při krátkém stisknutí tlačítka SELECT nakreslí bod, při jeho podržení kreslí souvislou čáru. Stisknutí tlačítka MENU smaže obrazovku. Jako kurzoru je použito invertování aktuálního bodu. Návěští se znaky XX jsou tzv. externí, těmito znaky je označeno volání samostatně překládaných podprogramů. Zde je takto označeno volání grafických podprogramů s touto funkcí:

```

GCLEAR   Smaže celou obrazovku.
GXOR     Invertuje bod na obrazovce na zadané souřadnici
         (L=X, H=Y).
GSET     Rozsvítí bod na obrazovce na zadané souřadnici
         (L=X, H=Y).

```

```

; Vstupní a výstupní parametry:
;      žádné, neboť se nejedná o podprogram, ale o hlavní
      program.

```

```
kresleni::
```

```

        ld      sp,stack      ;nastavení zásobníku
        call   minit         ;inicializace komunikace s
                               myší

;-----
;                               Hlavní smyčka programu
;-----
main::
        ld      hl,main      ;připravit na zásobník
                               návratovou adresu do hlavní
                               smyčky
        push   hl

; Nastavit kurzor - invertovat aktuální bod.

        ld      hl,(mousex) ;souřadnice aktuálního bodu
        call   gxorXX       ;invertovat aktuální bod
        push   hl           ;uschovat souřadnice kurzoru

; Nyní je třeba počkat na stisknutí tlačítka na myši
; nebo na její pohyb.

cykl:   call   test         ;test stavu myši
        jp     z,cykl       ;beze změny = čekat v cyklu

        ld      c,a         ;změna, status do C

; Nejdříve smazat kurzor.

        ex     (sp),hl      ;souřadnice kurzoru do HL,
                               nové aktuální souřadnice
                               uložit na zásobník
        call   gxorXX       ;invertovat aktuální bod

        pop    hl           ;nová souřadnice

```

; Otestovat stisknutí tlačítka MENU = smazat obrazovku.

```
ld      a,c          ;status
and     10100000b   ;ponechat bity tlačítka MENU
cp      10100000b   ;stisknuto tl. MENU?
jp      z,gclearXX  ;ANO = smazat obrazovku
```

; Otestovat držení tlačítka SELECT = rozsvítit bod.

```
ld      a,c          ;status
and     01000000b   ;ponechat bit stavu tl. SELECT
jp      nz,gsetXX   ;tl. SELECT je stisknuto:
                        ; = rozsvítit bod
ret                                           ;návrat do hlavní smyčky

ds      80h          ;místo pro zásobník
stack::                               ;vrchol zásobníku

end     kresleni    ;konec programu, definice
                        startovací adresy
```

3.3 Test myši při přerušení

Pokud jste si vyzkoušeli předchozí kreslicí program, zjistili jste, že myš nelze pohybovat příliš rychle, jinak se nekreslí to, co by mělo. Je to způsobeno stejnou příčinou jako tomu bylo u testovacího programu v BASICu - stav myši je třeba testovat velmi často, jinak některé změny signálů nestihneme zaznamenat. U tohoto kreslicího programu to nebylo až tak problematické, neboť vykreslení kurzoru jen z jednoho bodu je velice rychlé, ale při kreslení kurzorové šipky či při dynamickém kreslení čar by již bylo nutné myš jezdit neúnosně pomalu. Tento problém je nutné bezesbytku

vyřešit, neboť možnost rychlého pohybu myši je základním požadavkem příjemné a rychlé práce uživatele. Vysvětlili jsme si, v čem je problém, teď si povíme o řešení.

Jediným rozumným řešením popsaného problému je použití přerušení. Lze využít cyklického časového přerušení anebo přerušení při změně kteréhokoliv signálu z myši. Druhé řešení je lepší, ale vyžaduje podporu hardware. Řešení s časovým přerušením je o něco méně dokonalé, ale vyžaduje jednodušší, případně žádný (jako u PMD-85), hardware, proto bylo použito při tvorbě GREDITORu.

Princip spočívá v použití časovače, který s frekvencí 1000 - 5000 krát za vteřinu způsobí přerušení běhu mikroprocesoru. Při každém tomto přerušení se otestují signály z myši, jak to bylo dříve popsáno, a výsledek se uloží do tabulky v paměti. Tato činnost probíhá stále, nezávisle na tom, co zrovna dělá hlavní program. Pokud má hlavní program něco důležitějšího na práci, informace o pohybu myši se plní do tabulky. Když hlavní program vykoná vše potřebné, začne si z této tabulky informace vybírat a zpracovávat je.

3.3.1 Uložení informací z myši do tabulky

Podprogram TEST nám vrací tyto informace:

- aktuální souřadnice kurzoru
- stav tlačítek
- příznak změny stavu

Všechny tyto informace je třeba uschovat do paměti. K tomuto účelu slouží tabulka s přístupem typu FIFO (first in, first out), což znamená, že data z ní se vybírají v tom pořadí, v jakém do ní byly ukládány. Tabulka má určitou

velikost (třeba 100 položek) a při jejím zaplnění se do ní píše opět od začátku (cyklický buffer).

Cyklické časové přerušeni s testem signálů z myši a jejich uloženi do tabulky FIFO realizuje následující podprogram. Předpokládá se, že při cyklickém časovém přerušeni se volá PRERUSENI.

```
; Vstupní a výstupní parametry:
```

```
;          žádné - jedná se o obsluhu přerušeni
```

```
preruseni::
```

```
; Podprogram obsluhující přerušeni musí zachovat beze změny  
; všechny registry.
```

```
push    hl          ;uschovat HL
```

```
push    de          ;uschovat DE
```

```
push    bc          ;uschovat BC
```

```
push    af          ;uschovat AF
```

```
call    test        ;test signálů z myši
```

```
call    nz,uloz     ;změna = uložit informace
```

```
; Před návratem z přerušeni je třeba obnovit registry.
```

```
pop     af          ;obnovit AF
```

```
pop     bc          ;obnovit BC
```

```
pop     de          ;obnovit DE
```

```
pop     hl          ;obnovit HL
```

```
ei      ;povolit přerušeni
```

```
ret     ;návrat z podprogramu obsluhy  
        přerušeni
```

```
max     equ     100      ;velikost tabulky
```

```

uloz:                ;uložení informací do tabulky
                    push    hl        ;souřadnice (L=X, H=Y)
                    ld      c,a       ;status
                    ld      hl,nnn    ;adresa čítače položek do HL
                    ld      a,(hl)    ;počet položek v tabulce
                    ld      (lstput),a ;uložit
                    inc     (hl)      ;zvýšit čítač položek o 1
                    ld      a,(hl)    ;vybrat nový počet
                    cp      max       ;překročena kapacita?
                    jp      c,putl    ;NE = pokračovat

```

; Tabulka je plná, informace se zahazují.

```

                    dec     (hl)      ;původní počet položek
                    pop     hl        ;souřadnice
                    ret          ;návrat bez uložení informací

```

```

putl:               ld      hl,putpoi ;adresa ukazatele pro zápis do
                    ;tabulky
                    call   zvys      ;posunout ukazatel na další
                    ;položku, v HL se vrátí adresa
                    ;na aktuální položku
                    pop     de       ;souřadnice

```

; Nyní se uloží informace do tabulky.

```

                    ld      (hl),e    ;souřadnice X
                    inc     hl
                    ld      (hl),d    ;souřadnice Y
                    inc     hl
                    ld      (hl),c    ;status
                    ret          ;návrat z podprogramu

```

```

zvys:               inc     (hl)      ;zvýšení ukazatele o 1
                    ld      a,(hl)   ;vybrat nový stav ukazatele

```

```

cp      max      ;překročen konec tabulky?
jp      c,adr    ;NE = pokračovat

```

; Při zaplnění poslední položky tabulky se začíná opět od
; první položky tabulky.

```

xor     a        ;první položka
ld      (hl),a   ;uložit číslo položky

```

; Adresace do tabulky: každá položka má 3 bajty, index se
; proto vynásobí 3x a přičte se k začátku tabulky.

```

adr:    ld      e,a      ;číslo položky
        ld      d,0      ; do DE jako 16 bitů
        ld      hl,buf   ;adresa začátku tabulky
        add     hl,de     ;přičíst index 3x
        add     hl,de
        add     hl,de
        ret              ;návrat, v HL je adresa
                          položky

```

; Pomocné ukazatele a čítače:

```

lstput: db      0        ;číslo poslední ukládané
                          položky
nnn:    db      0        ;počet položek v tabulce
buf:    ds      3*max    ;místo pro tabulku

```

3.3.2 Výběr informací o myši z tabulky

Tabulka BUF se při přerušení plní informacemi o pohybu myši. Tyto informace je třeba z tabulky vybírat a zpracovávat. Přitom je třeba zajistit správnou synchronizaci s paralelním procesem vyvolávaným přerušením. To realizují

následující dva podprogramy. Liší se tím, že BMIN bere z tabulky všechna data, zatímco MIN pouze položky s informacemi o změně stavu tlačítek.

```
; Vstupní parametry:  
;      žádné
```

```
; Výstupní parametry: .  
;      L = souřadnice X  
;      H = souřadnice Y  
;      A = status (význam bitů byl popsán dříve)  
;      ZF = příznak změny  
;      Z = beze změny  
;      NZ = změna tlačítek nebo souřadnic
```

bmin::

```
push    de            ;uschovat DE  
push    bc            ; a BC  
  
ld      b,00110001b ;vracet i položky se změnou  
                        souřadnic  
jp      min0
```

min::

```
push    de            ;uschovat DE  
push    bc            ; a BC  
  
ld      b,00110000b ;vracet pouze položky se  
                        změnou tlačítek
```

```
min0:   ld      c,0            ;nulový status  
min1:   ld      hl,nnn        ;adresa čítače položek v  
                        tabulce  
        ld      a,(hl)        ;vybrat počet položek  
        or      a            ;je v tabulce nějaká položka?
```

```
    jp      nz,mn2      ;ANO = zpracovat
```

; V tabulce nejsou žádné informace, vrátí se prázdný status.

```
mnret:  ex      de,hl      ;souřadnice do HL
        ld      a,c        ;status
        and     00110001b  ;test příznaku změny
        ld      a,c        ;vrátit status v A

        pop     bc         ;obnovit BC
        pop     de         ;obnovit DE
        ret                      ;návrát z podprogramu
```

```
mn2:    dec     (hl)       ;snížit čítač položek o 1
        ld      hl,getpoi  ;ukazatel pro výběr položek z
                           tabulky
        call    zvys       ;na další položku, adresu do
                           HL
```

; Zbývá vybrat informace z tabulky.

```
        ld      e,(hl)    ;souřadnice X
        inc     hl
        ld      d,(hl)    ;souřadnice Y
        inc     hl
        ld      c,(hl)    ;status
```

```
        ld      a,c        ;status
        and     b          ;bereme tento typ položky?
        jp      z,mn1      ;NE = vybrat další
        jp      mnret     ;ANO = návrat z podprogramu
```

```
getpoi: db      0          ;ukazatel pro výběr z tabulky
```

3.3.3 Čekání na informace z myši

Pokud má hlavní program vše zpracováno, musí čekat na příchod nových informací z myši, aby mohl vyvinout další činnost. Toto čekání realizují triviální podprogramy, které se opět liší typem položek, které vracejí jako výstupní parametr (MINW ignoruje položky se změnou pouze souřadnic, BMINW bere i tyto položky).

```
; Vstupní parametry:
```

```
;      žádné
```

```
; Výstupní parametry:
```

```
;      L = souřadnice X
```

```
;      H = souřadnice Y
```

```
;      A = status se změnou
```

```
minw::   ei                ;povolit přerušeni (pro  
                jistotu)
```

```
minwl:   call    min        ;vybrat položku z tabulky  
                jp      z,minwl ;žádná není = čekat v cyklu  
                ret                ;návrat s informacemi
```

```
bminw::  ei                ;povolit přerušeni
```

```
bminwl:  call    bmin       ;vybrat položku z tabulky  
                jp      z,bminwl ;žádná není = čekat v cyklu  
                ret                ;návrat s informacemi
```

3.3.4 Inicializace cyklického časového přerušeni

Inicializace přerušeni v rozhraní pro připojení myši k ZX-Spectru je závislé na typu tohoto rozhraní. Většinou není třeba žádné - při použití astabilního klopného obvodu (AKO).

Zde proto popíšu pouze programové zpracování přerušení při použití tohoto typu rozhraní.

Přerušení není možné zpracovávat v módu IM 1, neboť obslužný podprogram začíná na adrese 38h, kde je v paměti ROM napevno naprogramován podprogram pro ošetření klávesnice. Zbývá použití vektorového modu přerušení: IM 2.

AKO nepředává procesoru žádný vektor přerušení (to dělají pouze periferní obvody od firmy Zilog), mikroprocesor načte ze své datové sběrnice bajt OFFH. Tento bajt tvoří nižší část vektoru přerušení - adresy buňky v paměti, kde je uložena adresa obslužného podprogramu. Vyšší část vektoru tvoří obsah procesorového registru I, který lze programově nastavit dle potřeby. Inicializaci vektoru přerušení a podprogramu TEST realizujeme takto:

```
; Vstupní a výstupní parametry:
```

```
;          žádné
```

```
minit::
```

```
        di          ;zakázat přerušení
```

```
; Nejprve inicializovat proměnné podprogramu TEST.
```

```
ld      hl,6080h    ;souřadnice středu obrazovky
ld      (mousex),hl ; uložit do paměti
xor     a           ;bajt 0
ld      (tlcitac),a ;nulovat čítač pro potlačení
                        odskoků
ld      (nnn),a    ;nulovat čítač bajtů v tabulce
                        TAB
ld      (getpoi),a ;nulovat ukazatel pro čtení z
                        tabulky TAB
ld      (putpoi),a ;nulovat ukazatel pro zápis do
                        tabulky TAB
```

; Nyní zinicilizujeme vektor přerušení.

```
ld      a,0feh      ;adresa stránky 0fe00h
ld      i,a         ; do registru I
ld      hl,preruseni;adresa podprogramu pro
zpracování přerušení
ld      (0feffh),hl ; do paměti
im      2           ;nastavit vektorový mód
                    přerušení
ei      ;povolit přerušení
ret     ;návrat z podprogramu
```

Na PMD-85 je pro generaci přerušení použit časovač 8253, který je jeho standardní součástí. Tento časovač je třeba naprogramovat - zapnout do správného módu a nastavit časové konstanty. Obvod 8253 obsahuje tři nezávislé časovače, my pro naše účely použijeme pouze dva: CT0 a CT1.

CT1 je použit v módu 3 jako symetrická dělička 10. Na jeho vstupu je frekvence 2 MHz, na výstupu 200 kHz.

CT0 je použit v módu 2 jako asymetrická dělička 200. Na jeho vstupu je frekvence 200 kHz z CT1 a na výstupu pulsy do log. "0" o délce 5 usec. s frekvencí 1 kHz. Tyto pulsy jsou zavedeny do vstupu mikroprocesoru 8080A pro přerušení.

Při přijetí přerušení provede mikroprocesor instrukci RST 38H - zavolá podprogram na adrese 38H.

Inicializaci časovačů, proměnných a vektoru přerušení provede tento podprogram:

```
; Vstupní a výstupní parametry:
;      žádné
```

minit::

```
di      ;zakázat přerušení
```


; Nejprve inicializovat proměnné podprogramu TEST.

```
ld      hl,8080h      ;souřadnice středu obrazovky
ld      (mousex),hl  ; uložit do paměti
xor     a              ;bajt 0
ld      (tlcitac),a  ;nulovat čítač pro potlačení
                        odskoků
ld      (nnn),a      ;nulovat čítač bajtů v tabulce
                        TAB
ld      (getpoi),a   ;nulovat ukazatel pro čtení z
                        tabulky TAB
ld      (putpoi),a   ;nulovat ukazatel pro zápis do
                        tabulky TAB
```

; Nyní zinicilizujeme vektor přerušeni tak, že na adresu
; 38h uložíme instrukci JP PRERUSENI.

```
ld      a,0c3h       ;kód instrukce JP adr.
ld      (38h),a      ; uložit do vektoru
ld      hl,preruseni;adresa podprogramu pro
obsluhu přerušeni
ld      (39h),hl     ; uložit do vektoru
```

; Zbývá inicializovat časovač 8253.

```
ld      a,01110110b ;CT1, mód 3
out     (5fh),a      ; do řídicího registru
ld      a,10         ;děleno 10
out     (5dh),a      ; do registru CT1
xor     a              ;vyšší bajt je 0
out     (5dh),a

ld      a,00110100b ;CT0, mód 2
out     (5fh),a      ; do řídicího registru
ld      a,200+1      ;děleno 200
```

```

out      (5ch),a      ; do registru CTO
xor      a            ; vyšší bajt je 0
out      (5ch),a

ei                          ; povolit přerušeni
ret                          ; návrat z podprogramu

```

3.4 Princip dynamického kreslení

Při práci s grafickým editorem je nejčastější operací umístění grafického objektu do obrázku. Pro příklad si vezmeme kreslení úsečky. Její umístění v obrázku je dáno počátečním a koncovým bodem. Bylo by možné jednoduše kurzorem ukázat na počáteční, pak na koncový bod a teprve potom by se vykreslila úsečka. Při tomto principu si ale jen těžko představujeme kudy úsečka povede, co protne, čeho se dotkne apod. Shrnutí - špatně se bude hledat její přesné umístění.

Z hlediska uživatele je mnohem výhodnější, když se úsečka vykresluje průběžně, takže v každém okamžiku vidí její tvar a může ji optimálně umístit. Tento způsob volby umístění vyžaduje složitější program, o jehož principu si nyní povíme.

Počáteční bod je určen polohou kurzoru při stisknutí tlačítka SELECT, s tím při programování nebudou žádné problémy. Nyní uživatel jede kurzorem do koncového bodu, přitom je třeba neustále kreslit úsečku - spojnici počátečního a koncového bodu. Tato spojnice se vykreslí, při posunu kurzoru se smaže a nakreslí se spojnice do nového aktuálního bodu. Tento proces se neustále opakuje a navenek se projevuje jako spojitý pohyb úsečky uchycené v počátečním bodu. Cyklus končí puštěním tlačítka SELECT, čímž je zadán koncový bod. Jsou dány počáteční a koncový bod, zbývá

vykreslit úsečku mezi těmito body a to v požadovaném grafickém módu a barvě.

Při dynamickém kreslení úsečky je nutné mít možnost ji zase smazat a přitom zachovat pozadí beze změny. Toho dosáhneme použitím grafického módu XOR. Při prvním vykreslení se body tvořící spojnici invertují - objeví se úsečka, při opakovaném vykreslení stejné úsečky se body opět invertují - vrátí se do původního stavu, čímž úsečka zmizí a pozadí je v původním stavu. Aby nedošlo ke změně barevných atributů, úsečku je třeba kreslit v módu "nebarvit", kdy je změna barevných atributů potlačena.

Popsaný princip dynamického kreslení lze schematicky zapsat takto:

1. uložit barevný mód do BARMOD
2. uložit grafický mód do GRMOD
3. nastavit barevný mód "nebarvit"
4. nastavit grafický mód "XOR"

5. vykreslit úsečku
6. čekat na posun myši
7. vykreslit úsečku (smazání)
8. puštěno tlačítko SELECT?
9. NE = cykl od kroku 5.

10. nastavit barevný mód BARMOD
11. nastavit grafický mód GRMOD
12. vykreslit úsečku
13. ukončit podprogram

Stejný princip, jako byl popisován při kreslení úsečky, se používá i pro dynamické kreslení dalších grafických objektů - rámečku, obdélníku, kružnice, textu, úsečky 45° atd. Je proto vhodné výše popsaný podprogram napsat univerzálně, aby šel použit pro kreslení různých grafických

objektů. Mění se pouze vykreslovaný objekt, příslušný podprogram, který je třeba pro jeho vykreslení volat, se může předávat jako vstupní parametr.

Nyní si ukážeme hotový podprogram, který realizuje výše uvedený princip dynamického kreslení:

```
; Vstupní parametry:
;       HL = souřadnice počátečního bodu (L=X, H=Y)
;       DE = adresa podprogramu pro vykreslení objektu

; Výstupní parametry:
;       HL = souřadnice koncového bodu
;       DE = souřadnice počátečního bodu

dox::

; Připravit adresu podprogramu kreslicího grafický objekt.

        ex     de,hl           ;adresa podprogramu do HL
        ld     (govec+1),hl; připravit adresu

        ld     h,d             ;souřadnice poč. bodu do HL
        ld     l,e

; Nastavit grafický mód "XOR" a barevný mód "nebarvit".

        call   modexXX         ;nastavit grafický mód "XOR"
        call   clr0XX         ;nastavit barevný mód
                                "nebarvit"

; Následuje hlavní smyčka dynamického kreslení.

drl:    ld     hl,(mouseXX) ;aktuální souřadnice kurzoru
        call   crsonXX        ;zobrazit kurzor
        call   gofce          ;vykreslit grafický objekt
```

```

push    hl          ;uložit akt. souřadnice
call    minwXX      ;čekat na změnu stavu myši
ld      c,a         ;status do C
ex      (sp),hl     ;obnovit akt. souřadnice
call    gofce       ;smazat grafický objekt
pop     hl          ;nové souřadnice

```

; Test puštění tlačítka SELECT.

```

ld      a,c         ;status
and     01010000b   ;nechat pouze bity tl. SELECT
cp      00010000b   ;puštěn SELECT?
jp      nz,drl      ;NE = pokračovat v cyklu

```

; Bylo puštěno tlačítko SELECT. Obnovit původní grafický
; a barevný mód a ukončit podprogram.

```

call    crsoffXX    ;smazat kurzor
call    mode0XX     ;nastavit původní grafický mód
call    color0XX    ;nastavit původní barevný mód
ret     ;návrat z podprogramu DOX

```

```

gofce:  push    hl          ;uschovat všechny potřebné
                                reg.
                                push    de
                                push    bc
govec:  call    0           ;sem se zapisuje adresa
                                podprogramu kreslicího
                                grafický objekt
                                pop     bc          ;obnovit všechny registry
                                pop     de
                                pop     hl
                                ret     ;návrat do hlavní smyčky

```

Na závěr povídání o dynamickém kreslení si ještě ukážeme využití podprogramu DOX v programu pro kreslení úseček. Stisknutím tlačítka SELECT se zadá počáteční bod úsečky, puštěním se definuje koncový bod. Pro volbu koncového bodu je využito dynamického kreslení - podprogramu DOX.

```

; Vstupní a výstupní parametry:
;      žádné - jedná se o hlavní program

usecky::                                ;vstupní bod programu
    ld      sp,stack                    ;nastavit zásobník
    call    minitXX                     ;inicializace komunikace s
                                         myší

main:                                    ;hlavní smyčka programu
    ld      hl,main                     ;připravit na zásobník adresu
                                         pro návrat do hlavní smyčky
    push   hl

    ld      hl,(mouseXX)                ;aktuální souřadnice kurzoru
    call    crsonXX                     ;vykreslit kurzor
    call    minwXX                       ;vzít nové souřadnice
    ld      c,a                          ;status do C
    call    crsoffXX                    ;smazat kurzor

; Při stisknutí tlačítka MENU smazat celou obrazovku.

    ld      a,c                          ;status
    and    10100000b                    ;ponechat pouze bity tl. MENU
    cp     10100000b                    ;stisknuto MENU?
    jp     z,gclearXX                   ;ANO = smazat obrazovku

; Při stisknutí tl. SELECT spustit dynamické kreslení.

```

```

ld      a,c          ;status
and     01010000b   ;bity tl. SELECT
cp      01010000b   ;stisknuto SELECT?
ret     nz           ;NE = návrat do hlavní smyčky

ld      de,drawXX   ;adresa podprogramu kreslení
                        úsečky do DE
call    doxXX       ;podprogram pro dynamické
                        kreslení
jp      drawXX      ;vykreslit úsečku

stack:  ds          80h      ;místo pro zásobník
                        ;vrchol zásobníku

end     usecky      ;konec programu a definice
                        vstupního bodu

```

3.5 Kurzor - tvary, algoritmy

Slovem **kurzor** označujeme grafickou značku na obrazovce, která slouží ke zviditelnění aktuálního bodu - místa, se kterým pracujeme. Kurzorem pohybujeme po obrazovce pomocí myši a označujeme jím místa pro vykreslení grafických objektů, vybíráme z nabídky, označujeme části textu apod.

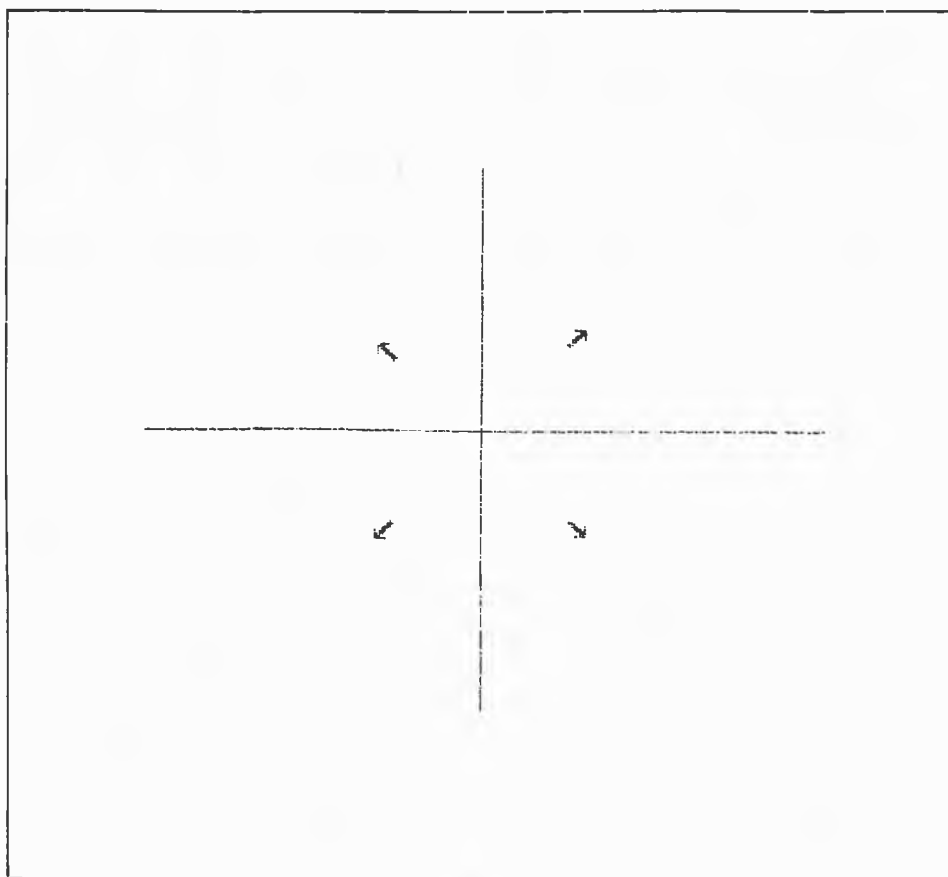
Tvar kurzoru bývá nejčastěji šipka, ale používá se i nitkový kříž (průsečík svislé a vodorovné čáry označuje aktuální bod), křížek, kolečko, pěst s nataženým ukazovákem atd. Často se využívá i průběžné změny tvaru kurzoru, to podle zvoleného pracovního módu. Třeba tužka pro kreslení, nůžky pro vystřižení oblasti, guma pro mazání, přesýpací hodiny - když je třeba chvíli počkat, sprej pro barvení, váleček pro vyplnění plochy... Tvar kurzoru se může měnit podle volby funkce z nabídky, nebo také podle polohy na displeji. Hlavní účel změn tvaru kurzoru je usnadnit práci

uživatelé, připomenout mu, jakou funkci nebo mód má právě zvolenu.

3.5.1 Šipka - kurzor GREDITORu

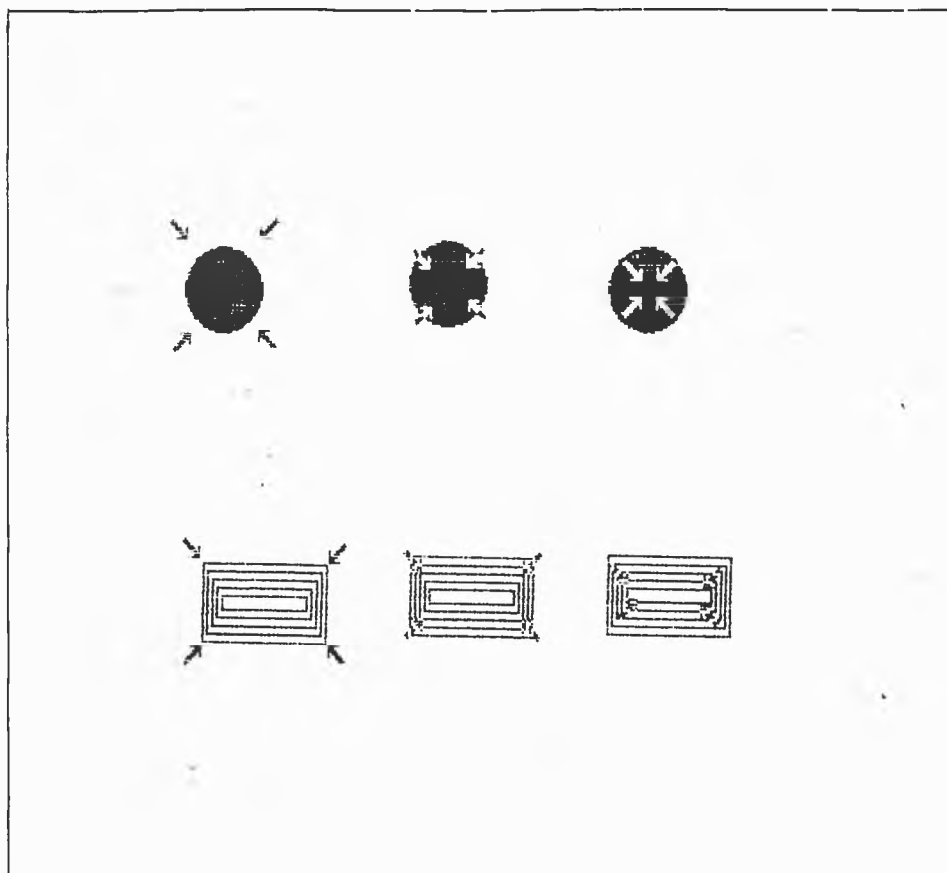
V programu GREDITOR je pro jednoduchost použit pouze jeden tvar kurzoru: šipka.

Aby bylo možné ukázat šipkou na každý bod displeje a to včetně bodů na okraji a přitom byla celá šipka na displeji zobrazena, mění se její směr tak, aby vždy ukazovala směrem k nejbližšímu okraji displeje. Změna směru se provádí v polovině displeje.



3.5.2 Princip kreslení kurzoru

Při kreslení kurzoru musíme řešit obdobný problém jako při dynamickém kreslení grafických objektů, tj. nejdříve kurzor nakreslit a pak ho smazat tak, aby obrázek, přes který se kurzor kreslí, zůstal nezměněn. Pro řešení tohoto problému byla opět použita logická funkce XOR (výhradní nebo). Všechny body, které leží na místě, kam se má vykreslit kurzor, se invertují. Po prvním provedení této operace se objeví šipka - na černém podkladu bílá, na bílém černá. Po druhém provedení operace se všechny body vrátí do původního stavu - šipka zmizí a pozadí zůstane beze změny.



Pro vykreslení i smazání šipky lze použít stejný podprogram, který vypadá takto:

```
; Vstupní parametry:
;      HL = adresa do displeje pro vykreslení kurzoru
;      DE = adresa tabulky tvaru kurzoru
;      BC = směr šipky (nahoru/dolu)

; Výstupní parametry:
;      žádné

;=====
Vykreslení kurzorové šipky do displeje PMD-85.
;=====

sipdej::
    ld     a,8           ;svislý rozměr šipky
sipcykl:
    push  af           ;uschovat čítač Y
    ld    a,(de)       ;vezmi bajt z tabulky tvaru
    inc   de           ;připravit adresu dalšího
                        bajtu
    xor   (hl)         ;logická operace XOR s obsahem
                        displeje
    ld    (hl),a       ;zapsat do displeje výsledek
                        operace
    inc   hl           ;připravit adresu dalšího
                        bajtu

    ld    a,(de)       ;druhý bajt tvaru šipky
    inc   de
    xor   (hl)         ;logická operace
    ld    (hl),a       ;zapsat výsledek do displeje
```

```

add    hl,bc      ;připravit adresu další linky
                        displej (+/- 3fh)
pop    af         ;obnovit čítač Y
dec    a         ;snížit čítač, je 0?
jp     nz,sipcykl ;NE = zapsat další linku
ret

```

```

=====
;Vykreslení kurzorové šipky do displeje ZX-Spectrum
;=====

```

sipdej::

```

ld     b,8        ;svislý rozměr šipky

```

sipcykl:

```

ld     a,(de)     ;vezmi bajt z tabulky tvaru
inc    de         ;připravit adresu dalšího
                        bajtu

```

```

xor    (hl)       ;logická operace XOR s obsahem
                        displeje

```

```

ld     (hl),a     ;zapsat do displeje výsledek
                        operace

```

```

inc    hl         ;připravit adresu dalšího
                        bajtu

```

```

ld     a,(de)     ;druhý bajt tvaru šipky

```

```

inc    de

```

```

xor    (hl)       ;logická operace

```

```

ld     (hl),a     ;zapsat výsledek do displeje

```

```

inc    c         ;otestovat příznak směru

```

```

dec    c

```

```

call   z,incy     ;šipka míří nahoru

```

```

inc    c

```

```

dec    c

```

```

call    nz,decy    ;šipka míří dolu

dec     b          ;snížit čítač Y, je 0?
jp      nz,sipcykl ;NE = zapsat další linku
ret     ;návrát z podprogramu

```

; Podprogramy INCY a DECY jsou popsány v popisu adresace
; displeje.

3.5.3 Výpočet polohy kurzoru

V předchozím odstavci jsme si ukázali podprogram pro vykreslení a smazání šipky. Ten ale sám o sobě nestačí, ještě je nutné z logických souřadnic kurzoru (souřadnice aktuálního bcdu) vypočítat fyzickou adresu do bitmapy displeje, směr šipky a její polohu v bajtu - tedy adresu na příslušný řádek tabulky tvaru šipky. Tato tabulka je závislá na hardware počítače. Pro PMD-85 má 6 řádek po 16 bajtech, pro ZX-Spectrum 8 řádek po 16 bajtech.

Kromě výpočtu fyzické adresy kurzoru je ještě třeba řídit mazání kurzoru na předešlé souřadnici a vykreslení na nové. Obě tyto funkce realizuje následující podprogram, který tvoří samostatný modul použitelný k libovolnému programu.

Podprogram je závislý na typu počítače, ale protože verze pro PMD-85 a ZX-Spectrum se liší jen v některých detailech, jsou tyto verze odlišeny pomocí podmíněného překladu.

```

; Vstupní parametry:
; CRSON:
;          HL = souřadnice aktuálního bodu (L=X, H=Y)
; CRSOFF:
;          žádné

```

```

;
; Výstupní parametry:
;      žádné

; Nejprve definice verze pro zvolený počítač.
; Pro ZX-Spectrum do příštího řádku zapsat 0.

pmd      equ      1          ;verze: PMD-85 / ZX-Spectrum

; Podprogram CRSOFF smaže kurzor z displeje, pokud byl
; vykreslen.

crsoff::
        push     af          ;uschovat původní obsah AF
        ld      ' a,(cursor) ;vybrat příznak vykreslení
                           kurzoru
        or      a           ;je kurzor vykreslen?
        jp      z,return    ;NE = návrat z podprogramu
        xor     a           ;0
        jp      curl

; Podprogram CRSON smaže kurzor z minulé pozice, pokud byl
; vykreslen, a vykreslí ho na nové pozici.

crson::
        push     af          ;uschovat původní obsah AF
        ld      a,(cursor)  ;vybrat příznak vykreslení
                           kurzoru
        or      a           ;je kurzor vykreslen?
        jp      z,crs2     ;NE = vykreslit nový

; Kurzor je vykreslen. Srovnat starou polohu s novou, při
; shodě souřadnic nic nemazat ani nekreslit.

```

```

push    de            ;uschovat původní obsah DE
ex      de,hl         ;nové souřadnice do DE
ld      hl,(lastcur) ;staré souřadnice do HL
ld      a,h           ;srovnat staré a nové Y
cp      d
jp      nz,crs1      ;liší se
ld      a,l           ;srovnat staré a nové X
cp      e
crs1:   ex      de,hl         ;nová souřadnice do HL
pop     de            ;obnovit původní obsah DE
jp      z,return     ;beze změny = návrat

call    crsoff       ;smazat kurzor na staré pozici

crs2:   ld      a,-1        ;příznak vykreslení kurzoru
ld      (lastcur),hl;uložit novou polohu kurzoru
curl:   ld      (cursor),a  ;uložit příznak vykreslení
                                kurzoru
push    hl            ;uschovat registry HL, DE, BC
push    de
push    bc

ld      hl,(lastcur);souřadnice aktuálního bodu
if      pmd           ;verze pro PMD-85
ld      bc,40h-1     ;směr nahoru
else
                                ;verze pro ZX-Spectrum
ld      c,0          ;směr nahoru
endc

ld      a,h           ;souřadnice Y
or      a             ;horní/dolní polovina?
jp      p,sipl       ;horní
if      pmd
ld      bc,-40h-1    ;směr dolu
else
ld      c,l          ;směr dolu

```

```

        endc
        dec     h           ;šipka ukazuje shora
        dec     h
sip1:   inc     l           ;šipka ukazuje zprava
        call   addrXX      ;adresace do displeje

; Podle polohy v bajtu zvolit příslušnou tabulku.

        ld     d,-1        ;čítač
sip2:   inc     d           ;zvýšit čítač
        if     pmd
        rrca                    ;hledáme bit "1"
        else
        rlca
        endc
        jp     nc,sip2     ;hledat dál

        ld     a,d         ;tabulka 0..
        add    a,a         ;vynásobit 16x pro indexaci
        add    a,a
        add    a,a
        add    a,a
        push   hl         ;uschovat adresu do displeje
        ld     hl,tabsipka ;adresa tabulky tvarů šipky
        ld     e,a         ;připravit index jako 16 bitů
        ld     d,0
        add    hl,de       ;indexace
        ex     de,hl      ;adresu tvaru do DE
        pop    hl         ;adresa do displeje

; Nyní jsou připraveny parametry pro vykreslení šipky:
;     HL = adresa do displeje
;     DE = adresa tabulky tvaru šipky
;     BC = příznak směru (nahoru/dolu)

```

```

call    sipdej    ;vykreslení/smazání šipky

pop     bc        ;obnovit původní obsah
                    registrů
pop     de
pop     hl

return: pop     af        ;původní stav
ret     ;návrat z podprogramu

cursor: db       0        ;příznak vykreslení kurzoru
lastcrs: dw      0        ;souřadnice kurzoru

```

K podprogramu ještě patří tabulka tvarů šipky. Zde z ní uvedu jen první část (jedna poloha šipky v bajtu), další jsou podobné, pouze bity jsou o jednu pozici pootočený.

```

;=====
;      Tabulka tvaru šipky pro PMD-85
;=====

```

```

.radix  2          ;nastavit binární soustavu

```

```

tabsip:

```

```

db      011111,000000
db      000111,000000
db      001111,000000
db      011101,000000
db      111001,000000
db      110000,000001
db      100000,000011
db      000000,000001

```

```

;=====
;      Tabulka tvaru šipky pro ZX-Spectrum

```



```
;=====
```

```
        .radix 2  
tabsip:  
        db      11111110,00000000  
        db      11100000,00000000  
        db      11110000,00000000  
        db      10111000,00000000  
        db      10011100,00000000  
        db      10001110,00000000  
        db      10000110,00000000  
        db      00000011,00000000
```

Nyní si ukážeme jednoduchý program pro pohyb kurzoru po obrazovce:

```
; Vstupní a výstupní parametry:  
;          žádné - hlavní program  
  
jezdit::  
        ld      sp,stack      ;nastavit zásobník  
        call   minitXX       ;inicializovat myš  
cykl:   call   minwXX        ;převzít data z myši  
        call   crsonXX       ;vykreslit kurzor  
        jp     cykl          ;zůstat v nekonečné smyčce  
  
        ds     80h           ;místo pro zásobník  
stack:  
  
        end     jezdit       ;konec programu
```

3.5.4 Jiný princip kreslení kurzoru

Dříve popsaný způsob kreslení kurzoru pomocí logické funkce XOR má výhody - jednoduchost a rychlost - a jednu nevýhodu - na členitém pozadí (směs černých a bílých bodů) je kurzorová šipka špatně zřetelná. Tento nedostatek lze odstranit použitím jiné metody kreslení, ovšem za cenu větší složitosti a o něco menší rychlosti podprogramu. V dalším textu si vysvětlíme princip této lepší metody kreslení kurzoru a pak si ukážeme příslušný podprogram.

Novou metodu kreslení kurzoru můžeme nazvat **sprajtovou**. Sprajt (někdy se toto slovo překládá jako **skřítek**, anglický originál je **sprite**) je myšlený obdélníček, ve kterém je nakreslen nějaký obrázek. Tento obdélníček se na displeji kreslí tak, jako by byl **před** pozadím, tzn. zakrývá kresbu v pozadí a sám je vždy vidět celý. Sprajt většinou bývá realizován pomocí hardware - videoprocesoru. Stejný princip je možné realizovat i programově. Postup je takový, že si nejprve uschováme část pozadí, přes kterou se má sprajt (kurzorová šipka) vykreslit. Poté tuto oblast danou maskou vymažeme (log. funkce AND) a nakonec do ní zapíšeme tvar kurzoru (OR). Výsledkem těchto operací je to, že kurzor je vždy vykreslen stejně - bez ohledu na pozadí - a jeho obrys je vyznačen odlišnou barvou. Takto vykreslený kurzor je velmi dobře viditelný na libovolném pozadí. Při mazání kurzoru jednoduše obnovíme přeepsanou část pozadí do původního stavu.

Vykreslení kurzoru sprajtovou metodou a jeho vymazání realizují následující podprogramy:

```
; Vstupní parametry:  
;  
; HL = adresa do displeje 'pro vykreslení kurzoru  
;  
; DE = adresa tabulky masky a tvaru  
;  
; BC = příznak směru nahoru/dolu  
;  
; pro PMD-85 to je offset +/-
```

```

;                               pro ZX-Spectrum to je adresa podprogramu
;                               A = rozměr Y

; Výstupní parametry:
;                               žádné

pmd      equ      1              ;"1"= PMD-85, "0"= ZX-Spectrum
maxy     equ      16            ;maximální rozměr Y

        if      pmd
sizeX    equ      4              ;počet bajtů kurzoru v řádce
        else
sizeX    equ      2
        endc

sipdej::                               ;začátek podprogramu

; Nejdříve uložit parametry pro smazání kurzoru podprogramem
; SIPBER.

        ld      (lastscr),hl;uložit adresu kurzoru v
displeji
        ld      (ry),a          ;uložit počet linek

; Připravit směr kreslení: nahoru/dolu přímo do programu.

        push    hl              ;uschovat HL
        ld      h,b              ;příznak směru do HL
        ld      l,c
        ld      (sipdir+1),hl ;uložit směr
        pop     hl

        ld      bc,rozdil       ;adresa rozdílové tabulky

cykl:

```

```
push    af          ;uschovat počet linek
```

```
; Následující blok realizuje dříve popsané logické operace  
; pro zápis kurzoru sprajtovou metodou. Pro zvýšení  
; rychlosti je animace bajtů v řádce řešena opakováním  
; instrukcí a nikoli cyklem.  
; Pseudoinstrukce REPT opakuje blok instrukcí až k ENDM,  
; počet opakování je zadán jako parametr za REPT.
```

```
rept    sizeX       ;opakovat podle počtu bajtů  
ld      a,(de)      ;vybrat masku z tabulky  
cpl                    ;invertovat masku  
and     (hl)        ;vymazat z pozadí tvar daný  
                        maskou  
ex      de,hl       ;adresu tabulky tvarů do HL  
inc     hl          ;na bajt tvaru  
or      (hl)        ;vložit tvar kurzoru do bajtu  
inc     hl          ;na masku dalšího bajtu  
ex      de,hl       ;do HL opět adresu displeje
```

```
; V A je nový obsah displeje. Do pomocné tabulky si nyní  
; uložíme rozdíl mezi starým a novým obsahem displeje, to  
; pro pozdější obnovení jeho obsahu.
```

```
xor     (hl)        ;rozdíl do A  
ld      (bc),a      ;uložit rozdíl do tabulky  
inc     bc          ;zvýšit ukazatel  
xor     (hl)        ;nový obsah displeje do A  
ld      (hl),a      ;zapsat pozadí s kurzorem do  
                        displeje  
inc     hl          ;posunout adresu do displeje  
                        na další bajt v lince  
endm                    ;konec opakovaného bloku
```

```
; Byla zapsána jedna linka kurzoru.
```

; Přesuneme adresu do displeje na další linku.

```
call zmena ;změna adresy do displeje
```

; Snížit čítač linek a případně opakovat cykl pro další linku.

```
pop af ;do A počet linek
```

```
dec a ;snížit čítač
```

```
jp nz,cykl ;opakovat cykl
```

```
ret ;návrat z podprogramu SIPDEJ
```

```
;=====
; Podprogram ZMENA posouvá HL na další linku displeje
;=====
```

zmena:

```
if pmd
```

```
push bc ;uschovat BC
```

```
sipdir: ld bc,0 ;sem se připraví offset
```

```
add hl,bc ;adresa další linky displeje
```

```
pop bc ;původní BC
```

```
else
```

; ZX-Spectrum

```
rept sizex ;opakovat podle počtu bajtů
```

```
dec hl ;vlevo na začátek kurzoru
```

```
endm
```

```
sipdir: call 0 ;sem se připraví adresa podprogramu pro změnu Y
```

```
endc ;konec podmíněného překladu
```

```
ret ;návrat z podprogramu ZMENA
```

```
;=====
;      Podprogram SIPBER smaže kurzor z displeje.
;=====
```

```
; Vstupní a výstupní parametry:
;      žádné
```

```
sipber::
```

```
    push    hl          ;uschovat všechny registry
    push    de
    push    bc
    push    af
```

```
; Připravit adresy tabulek do registrů.
```

```
    ld      hl,(lastscr);adresa kurzoru v displeji
    ld      de,rozdil   ;adresa rozdílové tabulky
    ld      a,(ry)      ;počet linek kurzoru
    ld      b,a         ; do B
```

```
cykl2:
```

```
    rept    sizex      ;opakovat pro všechny bajty
    ld      a,(de)     ;rozdílový bajt do A
    xor     (hl)       ;výsledkem je původní obsah
    ld      (hl),a     ;zapsat původní obsah
    inc     hl         ;další bajt displeje
    inc     de         ;na další rozdílový bajt
    endm          ;konec opakovaného bloku
```

```
    call   zmeny      ;na další linku displeje
    dec    b           ;snížit čítač linek
    jp     nz,cykl2   ;znovu do cyklu
```

```
    pop    af         ;obnovit původní obsah
                        registrů
```

```

        pop     bc
        pop     de
        pop     hl
        ret                               ;návrat z podprogramu SIPBER

lastscr: dw     0                        ;adresa kurzoru v displeji
ry:      db     0                        ;počet linek kurzoru
rozdil:  ds     sizex*maxy              ;místo pro rozdílovou tabulku

```

```

; Pro úplnost ještě tabulku tvaru šipky a její masky.
; Tato tabulka se nepoužívá přímo pro kreslení kurzoru, ale
; podle ní se vytváří více pracovních tabulek (rotace bitů,
; změna směru).
; Pro lepší znázornění tvaru je při zápisu tabulky tvaru
; použito MACRO definice a místo "0" je napsána ".".

```

```

crsnormal::                               ;tvar kurzoru: šipka
        db     9,9                       ;definice rozměru: 9x9 bodů
        .radix 2                          ;nastavit dvojkovou soustavu

```

```

; tvar šipky

```

```

        mdb     .....
        mdb     .111111.,.....
        mdb     .111111.,.....
        mdb     .1111...,.....
        mdb     .11111..,.....
        mdb     .11.111.,.....
        mdb     .11..111,.....
        mdb     .....1.,.....
        mdb     .....

```

```

; maska šipky

```

mdb 11111111,.....
mdb 11111111,.....
mdb 11111111,.....
mdb 1111111.,.....
mdb 11111111,.....
mdb 11111111,1.....
mdb 11111111,1.....
mdb 111.1111,1.....
mdb111,.....

3.6 MENU - volba z nabídky

Základním principem komunikace s programem pomocí elektronické myši je volba z nabídky - MENU.

Po stisknutí příslušného tlačítka na myši se na displeji objeví tabulka s názvy funkcí, ze kterých můžeme volit. Pomocí myši ukážeme kurzorem na vybraný řádek a zvolená funkce se provede. V dalším textu si popíšeme, které akce je třeba vykonat pro programovou realizaci popsaného principu. Konkrétní podprogram pro volbu z nabídky je příliš dlouhý, povíme si proto pouze o jeho hlavních blocích a ukážeme si některé jeho zajímavé části.

3.6.1 Postup při volbě z nabídky

Postup při volbě z nabídky obsahuje tyto základní kroky:

1. Vstup: souřadnice kurzoru, text nabídky.
2. Volba místa pro výpis nabídky.
3. Uložení pozadí z místa výpisu nabídky.
4. Výpis textu nabídky.
5. Inverze řádky s kurzorem - výběr funkce.
6. Vrácení pozadí - výmaz nabídky.
7. Výstup: číslo zvoleného řádku.

Ad 1.-2.

Pro urychlení práce uživatele byla v GREDITORu zvolena taková koncepce, kdy se text nabídky vždy vypisuje co nejbližší kurzoru. Jako vstupní parametr se proto předává souřadnice kurzoru. Z ní se vypočítá poloha nabídky s ohledem na její velikost tak, aby se celá vešla na displej. Souřadnice X se ještě modifikuje tak, aby jednotlivé znaky textu nabídky ležely ve svém celém bajtu, ne na hranici dvou

sousedních bajtů. To je důležité pro snadný a rychlý výpis textu nabídky.

Ad 3.

Po určení polohy nabídky na displeji je nutné uschovat obsah této oblasti pro její pozdější obnovení. Data z displeje se ukládají do paměti do vyhrazeného místa (BFMENU) sloužícího pouze tomuto účelu. Musí se dbát na to, aby přenos byl co nejrychlejší a nezapomenout na odlišnou adresaci displeje a paměti. Ve verzi pro ZX-Spectrum je třeba do paměti uložit i barevné atributy.

Ad 4.

Do určené oblasti se vypíše text nabídky, který je připraven v paměti v tabulce a adresa této tabulky se předává jako vstupní parametr. Kromě textu jsou v tabulce ještě obsaženy informace o počtu řádek a sloupců nabídky. Při programování se nesmí zapomenout na doplnění kratších textů mezerami až do konce řádky nabídky.

Ad 5.

Z polohy nabídky a aktuálních souřadnic kurzoru se určuje řádek, na který kurzor ukazuje. Tento řádek se vypíše inverzně. Při přesunu kurzoru na další řádek se předchozí obnoví (zruší se inverze) a invertuje se nový. Při poloze kurzoru mimo nabídku není invertován žádný řádek - žádná funkce není zvolena.

Ad 6.

Při ukončení výběru z nabídky se obnoví původní obsah displeje z paměti, kam byl uložen - bod 3. Podprogram je ten samý, pouze smysl přenosu dat je opačný - z paměti do displeje.

Ad 7.

Při návratu z podprogramu pro volbu z nabídky se jako výstupní parametr předává číslo zvolené řádky (1..max.). Pokud nebyl zvolen žádný řádek (kurzor mimo nabídku), vrátí se 0.

3.6.2 Hlavní smyčka podprogramu pro volbu z nabídky

; Bylo vybráno místo po výpis nabídky, uloženo pozadí
 ; z tohoto místa a byla vypsána nabídka a zarámována.
 ; Následující programový blok realizuje hlavní smyčku
 ; výběru z nabídky - pohyb kurzoru, inverzi řádku
 ; s kurzorem a návrat z podprogramu volby z nabídky.

```

      ld      hl,(posx)   ;souřadnice kurzoru
      ld      b,-1       ;žádné pole není invertováno

mmain::                                ;hlavní smyčka
      call   crsonXX     ;vykreslit kurzor
      call   minwXX     ;vzít informace z myši
      call   crsoffXX   ;smazat kurzor

      ld      c,a       ;status myši
      and    01010000b  ;bity tlačítka SELECT
      cp     01010000b  ;stisknuto tlačítka SELECT?
      jp     z,zrusmenu ;AND = zruší MENU bez
                        výběru

      ld      a,c       ;status myši
      and    10100000b  ;bity tlačítka MENU
      jp     z,mmain   ;beze změny = čekat ve smyčce
      cp     00100000b  ;puštěno tlačítka MENU?
      jp     z,volfunkci ;AND = určit zvolenou funkci

```

; Tlačítka MENU je stisknuto, invertovat políčko s kurzorem.

```

call   pole?      ;určit políčko s kurzorem
cp     b          ;stejně jako minule?
jp     z,mmain    ;ANO = opět do cyklu

push   af         ;číslo nového řádku s kurzorem
call   normal     ;zapnout normální mód tisku
ld     a,b        ;číslo invertovaného řádku
call   prline     ;vypsat: zrušit inverzi

pop    af         ;nový řádek pro inverzi
ld     b,a        ;uschovat číslo právě
                    vybarveného řádku
call   inverze    ;zapnout inverzní mód tisku
call   prline     ;vypsat inverzně řádek

```

; Při přechodu kurzoru na nový řádek se volá podprogram,
; jehož adresa se předává jako vstupní parametr volby
; z nabídky. To je využito např. pro okamžité nastavení
; barvy pozadí nebo palety (viz. popis volby barev).

```

ld     a,b        ;číslo řádky s kurzorem
push   hl         ;uschovat všechny registry
push   de
push   bc
inmenu: call 0     ;sem se zapisuje adresa
                    volaného podprogramu
pop    bc         ;obnovit všechny registry
pop    de
pop    hl
jp     mmain      ;znovu do hlavní smyčky

```

; Při stisknutí tlačítka SELECT se provede ukončení
; volby z nabídky. Volba se neprovede, vrátí se 0.

zrusmenu:

```
    xor    a            ;0 jako výstupní parametr
    jp    volf         ;obnovení pozadí, návrat
```

; Po puštění tlačítka MENU se provede výběr řádku,
; obnoví se pozadí a ukončí se podprogram volby z nabídky.
; Jako výstupní parametr se předává číslo zvolené funkce
; v registru A.

```
volf:    or     a            ;nastavit příznak Z/NZ
        push  af          ;uschovat číslo funkce
        call  crsoffXX    ;smazat kurzor
        call  vrat        ;vrátit pozadí - zrušit
                          nabídku
        pop   af          ;výstupní parametr je v A
        ret             ;návrat z podprogramu volby z
                          nabídky
```

3.6.3 Úschova a obnova pozadí pod textem nabídky

; Vstupní parametry:
; TABXY = souřadnice levého horního rohu nabídky
; RY = počet linek nabídky (rozměr Y v bodech)
; XCHAR = počet znaků nabídky (rozměr X ve znacích)

; Výstupní parametry:
; žádné

; Podprogram SCHOVEJ uloží pozadí do paměti do BFMENU,
; podprogram VRAT obnoví pozadí z paměti z BFMENU.

vrat: ;podprogram obnovy pozadí

```
; Kód instrukce LD A,(DE) je 1ah,  
; kód instrukce LD (HL),A je 77h.
```

```
ld hl,771ah ;přenos (DE) = (HL)  
jp sch0
```

```
schovej: ;podprogram úschovy pozadí
```

```
; Kód instrukce LD A,(HL) je 7eh,  
; kód instrukce LD (DE),A je 12h.
```

```
ld hl,127eh ;přenos (HL) = (DE)  
sch0:  
ld (schmd),hl ; příprava přenosových  
instrukcí  
  
ld hl,(tabxy) ;souřadnice levého horního  
rohu  
ld a,(ry) ;rozměr Y  
add a,4 ;+ rámeček  
ld b,a ; do B  
  
ld a,(xchar) ;počet znaků = počet bajtů  
add a,2 ;+ rámeček  
ld c,a ; do C  
  
call addrXX ;adresace do displeje = HL  
ld de,bfmenu ;místo v paměti pro uložení  
pozadí
```

```
; V registrech jsou připraveny parametry pro přenos:  
; HL = adresa do displeje  
; DE = adresa místa v paměti  
; C = počet bajtů v jedné lince
```

```

;      B = počet linek

uschl:                ;hlavní smyčka přenosu
    push    bc        ;uschovat BC a HL
    push    hl

schcmd:  nop          ;sem se zapisují kódy
                    ;instrukcí
    nop            ; -"-

    inc     hl        ;na další bajt v řádce
    inc     de        ;zvýšit ukazatel do paměti
    dec     c         ;snížit čítač bajtů v řádce
    jp     nz,schcmd ;v cyklu přenést jeden řádek

```

; Nyní je nutné adresovat v displeji další linku.
; Zde se liší verze pro PMD-85 a ZX-Spectrum.

```

    pop     hl        ;adresa začátku řádky
    if     pmd
    ld     bc,40h     ;offset mezi řádky
    add    hl,bc      ;adresa další linky displeje
    else

    call   incyXX     ;posun HL na další linku
    endc            ;konec rozdílné části

    pop     bc        ;obnovit čítače
    dec     b         ;snížit čítač linek
    jp     nz,uschl  ;pokračovat v přenosu další
                    ;linky
    ret          ;návrat z podprogramu

```

3.6.4 Určení řádky nabídky s kurzorem

```
; Vstupní parametry:
;     HL = souřadnice kurzoru
;     TXTXY = souřadnice levého horního rohu nabídky
;     RX, RY = rozměr tabulky v bodech

; Výstupní parametry:
;     A = číslo řádky s kurzorem,
;         0 = kurzor je mimo nabídku,
;         1..max = číslo řádku

pole?:
    push    de            ;uschovat DE a BC
    push    bc
    ex      de,hl        ;souřadnice kurzoru do DE

    ld      hl,(rx)      ;rozměr tabulky nabídky
    ld      bc,-(ramy*256+ramx) ;odečíst rámeček
    add     hl,bc
    ld      b,h          ; do BC
    ld      c,l
    ld      hl,(txtxy)   ;souřadnice začátku tabulky

; Jsou připraveny potřebné parametry v registrech:
;     HL = souřadnice levého horního rohu tabulky MENU
;     DE = souřadnice kurzoru
;     BC = velikost tabulky v bodech

    call    pole2        ;určení řádky s kurzorem do A

    ex      de,hl        ;souřadnice kurzoru do HL
    pop     bc           ;obnovit původní obsah BC a DE
    pop     de
    ret     nc           ;kurzor je na tabulce
```



```

xor    a            ;mimo tabulku = vrací se 0
ret    ;návrat z podprogramu POLE?

```

pole2:

```

ld     a,e          ;Xkurzor
sub    l            ;Xkurzor - Xtab
ret    c            ;vlevo od tabulky
cp     c            ;-RX
ccf    ;invertovat vlajku Cy
ret    c            ;vpravo od tabulky

```

```

ld     a,d          ;Ykurzor
sub    h            ;Ykurzor - Ytab
ret    c            ;nad tabulkou
cp     b            ;-ry
ccf    ;invertovat Cy
ret    c            ;pod tabulkou

```

; Souřadnice kurzoru je v mezích tabulky, budeme vybírat
; řádek.

; V registru A je Ykurzoru - Ttabulky, tedy offset od
; horního konce tabulky. Řádek určíme dělením offsetu
; počtem linek na jeden řádek nabídky.

```

ld     c,msizey    ;počet linek na jeden řádek
                        textu
call   divubXX     ;dělení A/C = A,
                        ;tedy offset/výška = řádek
inc    a           ;počítat od 1
or     a           ;NC, shodit vlajku Cy
ret    ;návrat, v A je číslo řádku s
                        kurzorem

```

3.7 Základní grafické algoritmy

Základem každého grafického editoru, tedy i GREDITORu, jsou podprogramy kreslící různé grafické objekty. Protože v programech používajících elektronickou myš se často kreslí různé grafické obrázky, seznámím Vás se základními grafickými algoritmy a podprogramy.

3.7.1 Adresace displeje PMD-85

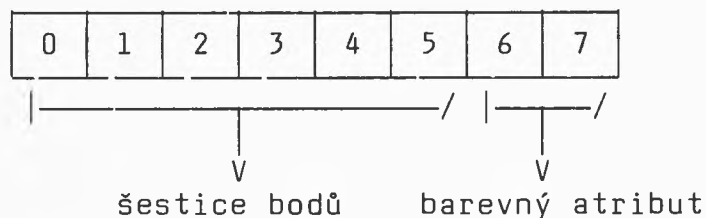
Před programováním jakékoli operace s obrazovou pamětí je potřebné znát přesně její adresaci, která se silně liší u každého typu mikropočítače.

Mikropočítač PMD-85 je řešen tak, že obrazová paměť je v adresním prostoru v oblasti C000H - FFFFH. Bajty jsou řazeny sekvenčně za sebou, vždy 48 se jich zobrazuje (288 bodů v jedné lince) a dalších 16 se nezobrazuje. Bajt zobrazený jako první v levém horním rohu displeje je na adrese C000H, bajt vpravo od něj na adrese C001H atd. Poslední bajt zobrazený v pravém rohu první linky je na adrese C02FH. Bajty C030H až C03FH se nezobrazují. Druhá linka začíná na adrese C040H, třetí na C080H atd. První bajt poslední linky (zobrazuje se 256 linek) je na adrese FFC0H. Poslední zobrazovaný bajt v pravém dolním rohu displeje je na adrese FFEFH. Popsanou situaci přehledně ukazuje následující tabulka:

```
C000H, C001H, C002H, C003H ... C02FH
C040H, C041H, C042H, C043H ... C06FH
C080H, C081H, C082H, C083H ... C0AFH
C0C0H, C0C1H, C0C2H, C0C3H ... C0EFH
... ..
FF80H, FF81H, FF82H, FF83H ... FFAFH
```

FFC0H, FFC1H, FFC2H, FFC3H ... FFEFH

Z každého bajtu se 6 nižších bitů (bity 0 až 5) zobrazuje. Bit s log. "1" svítí, bit s log. "0" nesvítí. Bit 0 je ze šestice nejvíce vlevo, bit 5 nejvíce vpravo. Bity 6 a 7 tvoří barevný atribut pro těchto 6 bodů. Každá šestice bodů obsažených v jedné bajtu musí mít stejnou barvu - jednu ze čtyř, nebo barvu pozadí. Barva pozadí je stejná pro celý displej a dá se zvolit 1 z 8 (rozumí se u PMD-85 s barevnou úpravou).



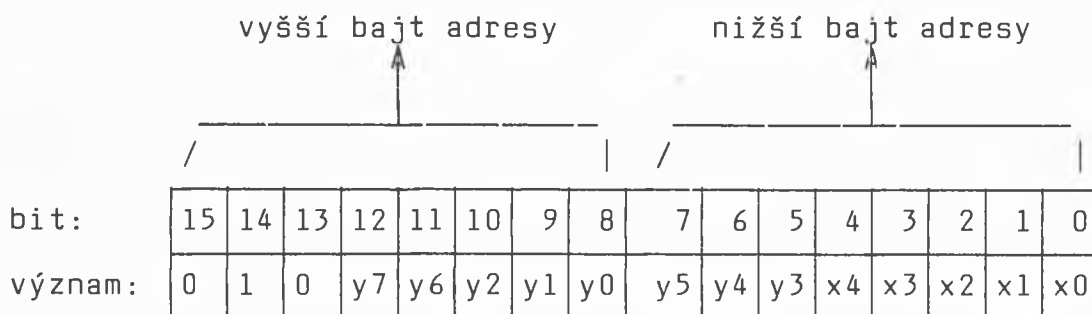
U PMD-85 s barevnou úpravou se barvy volí na výstupních branách takto:

pozadí: bity 7,6,5 na adrese 0F4H
paleta: bit 4 na adrese 0F4H
 bity 5,4 na adrese 0F6H
inicializace brány 8255: bajt 82H na adresu 0F7H

Slovem "adresa" míním adresu prostoru I/O, nikoli adresu paměti!

3.7.2 Adresace displeje ZX-Spectrum

Displej ZX-Spectrum má dvě části - bitmapu a atributy. Bitmapa je v adresním prostoru v oblasti 4000H až 57FFH. V rámci jedné zobrazované linky je adresace lineární. První zobrazený bajt na první lince je na adrese 4000H, bajt vpravo od něj na 4001H atd. až poslední bajt první linky je na 401FH. Na jedné lince se zobrazuje 32 bajtů, tj. 256 bodů. Zobrazuje se 192 linek. Adresace ve svislé ose je poněkud divoká a náročnější na pochopení. Je to dáno přehozením některých adresních bitů. Nejnázornější bude tabulka přiřazení adresních bitů.



Bajty v prvním sloupci displeje (první bajt každé linky) mají tyto adresy:

4000H, 4100H, 4200H, 4300H, 4400H, 4500H, 4600H, 4700H,
 4020H, 4120H, 4220H, 4320H, 4420H, 4520H, 4620H, 4720H,
 4040H, 4140H ...

Jeden bajt se zobrazuje jako osmice bodů - každý bit představuje jeden bod. Bit 7 je zobrazen nejvíce vlevo z osmice, bit 0 nejvíce vpravo. Bit s log. "1" má barvu popředí, bit s log. "0" má barvu pozadí.

Barevné atributy jsou v adresním prostoru v oblasti 5800H až 5AFFH. Adresace je lineární a vždy jeden atribut (bajt) je barevným atributem pro 8 nad sebou ležících bajtů


```

;           0..191 pro ZX-Spectrum
;           L = souřadnice X (0..255)

; Výstupní parametry:
;           HL = adresa do displeje na adresovaný bajt
;           A  = bitová maska
;           - v adresovaném bitu je "1", v ostatních "0"

;=====
;           Adresace displeje PMD-85.
;=====

addr::

    push    bc           ;uschovat BC a DE
    push    de

    push    hl           ;uschovat souřadnice
    ld      a,l           ;souřadnice X
    and     1             ;lichá/sudá
    push    af
    xor     1             ;X s vymaskovaným bitem 0
    rrca                    ;podělit 2
    ld      l,a           ; do L
    ld      h,0           ; do HL jako 16 bitů
    ld      de,tab6       ;tabulka pro adresaci v ose X
    add     hl,de         ;indexovat do tabulky
    ld      a,(hl)        ;obsah tabulky
    and     3             ;číslo bitu: 0,1,2
    add     a,a           ;vynásobeno dvěma: 0,2,4
    ld      e,a           ; do E
    pop     af            ;příznak sudá/lichá
    add     a,e           ;číslo bitu: 0..5
    ld      e,a           do E

    ld      a,(hl)        ;obsah tabulky

```

```

ld      d,0          ;číslo bitu jako 16 bitů v DE
ld      hl,tbit      ;tabulka bitových masek
add     hl,de        ;indexace do tabulky
ld      c,(hl)       ;bitová maska: 1,2,4,8,10h,20h

rrca                    ;obsah tabulky/4
rrca
and     3fh          ;fyzická adresa X
ld      l,a          ; do L
pop     af           ;v A je souřadnice Y
rrca                    ;zarotovat o 2 bity
rrca
ld      e,a          ; do E
and     00111111b    ;vyšší část fyzické adresy Y
ld      h,a          ; do H
xor     e            ;nižší část fyzické adresy Y
or      l            ;připojit fyzickou adresu X
ld      l,a          ;do L
ex     de,hl         ;fyzickou adresu XY do DE
ld      hl,0C000h    ;adresa začátku obrazové
                    ;paměti
add     hl,de        ;indexace do displeje

ld      a,c          ;bitová maska do A
pop     de           ;obnovit původní obsah DE a BC
pop     bc
ret                    ;návrat z adresačního
                    ;podprogramu

```

Tabulka bitových masek:

```
tbit:    db      1,2,4,8,10h,20h
```

; Tabulka pro výpočet fyzické souřadnice X se vytvoří
; pomocí cyklu:

```

11      defl    0          ;vynulovat čítač
tab6:
      rept    128        ;tabulka má 128 položek
      db      (11/3)*4 + (11 mod 3)
11      defl    11+1      ;zvýšit čítač o 1
      endm      ;konec opakovaného bloku

```

```

;=====
;      Adresace displeje ZX-Spectrum.
;=====

```

addr::

```

      push    bc          ;uschovat BC a DE
      push    de          ;
      ld      a,l         ;souřadnice X
      and     7           ;číslo bitu
      ld      c,a         ; do C
      xor     1           ;vymaskovat číslo bitu
      rrca                    ;podělit 8
      rrca                    ;
      rrca                    ;
      ld      a           ;fyzické X do L

      ld      a,h         ;souřadnice Y
      rlca                    ;pootočit o dva bity
      rlca                    ;
      and     11100000b    ;y5, y4, y3
      or      1           ;přidat fyzické X
      ld      l,a         ; do L

      ld      a,h         ;souřadnice Y
      and     7           ;y2, y1, y0
      ld      b,a         ; do B

```



```

ld      a,h          ;souřadnice Y
and     11000000b    ;y7, y6
rrca                    ;posunout o tři bity
scf                    ;"1" do bitu 14
rra                    ;
rrca                    ;
or      b            ;přidat y2, y1, y0
ld      h,a          ; do H

ex      de,hl        ;celou fyzickou adresu do DE
ld      b,0          ;číslo bitu jako 16 bitů v BC
ld      hl,tabbit    ;adresa tabulky bitových masek
add     hl,bc        ;indexace do tabulky
ld      a,(hl)       ;vybrat bitovou masku
ex      de,hl        ;adresu do displeje do HL

pop     de           ;obnovit původní obsah DE a BC
pop     bc           ;
ret                    ;návrat z adresačního
                    ;podprogramu

```

; Tabulka bitových masek:

```
tabbit:  db      80h,40h,20h,10h,8,4,2,1
```

3.7.4 Posun adresy displeje

Dalšími důležitými podprogramy pro grafiku jsou adresace sousedních bodů, tedy změna fyzické adresy do displeje a bitové masky na sousední bod a to v libovolném ze čtyř směrů. Tyto funkce realizují čtyři jednoduché podprogramy:

```

INCX      posun o bod doprava
DECX      posun o bod doleva

```

```
INCY      posun o bod dolu
DECY      posun o bod nahoru
```

```
; Vstupní parametry:
;      HL = fyzická adresa do displeje
;      C  = bitová maska
```

```
; Výstupní parametry:
;      HL = upravená adresa displeje
;      C  = upravená bitová maska
```

```
;=====
; Posun adresy bodu v displeji PMD-85 o bod doprava
;=====
```

```
incx::      ;posun o bod doprava
            ld      a,c      ;bitová maska
            rlca      ;posun o bod doprava
            ld      c,a      ;uložit novou masku
            and     00111111b ;překročen rámeč 6 bodů?
            ret     nz      ;NE = hotovo
```

```
; Byl překročen rámeč jednoho bajtu, je třeba adresovat
; další.
```

```
inc      hl      ;adresace dalšího bajtu
ld      c,l      ;nová bitová maska
ret      ;hotovo = návrat
```

```
;=====
; Posun adresy bodu v displeji PMD-85 o bod doleva
;=====
```

```

decx::                                ;posun o bod doleva
    ld      a,c                        ;bitová maska
    rrca                               ;posun o bod doleva
    ld      c,a                        ;uložit novou masku
    ret    nc                          ;Cy? NE = hotovo

```

```

; Byl překročen rámeček jednoho bajtu, je třeba adresovat
; předchozí.

```

```

    dec    hl                          ;adresace předchozího bajtu
    ld     c,20h                       ;nová bitová maska
    ret                                       ;hotovo = návrat

```

```

;=====
; Posun adresy bodu v displeji PMD-85 o bod dolu
;=====

```

```

incy::                                ;posun o bod dolu
    push   de                          ;uschovat DE
    ld     de,40h                      ;offset adres sousedních linek
    add    hl,de                       ;na další linku
    pop    de                          ;obnovit DE
    ret                                       ;hotovo, návrat

```

```

;=====
; Posun adresy bodu v displeji PMD-85 o bod doprava
;=====

```

```

decy::                                ;posun o bod nahoru
    push   de                          ;uschovat DE
    ld     de,-40h                    ;offset adres sousedních linek
    add    hl,de                       ;na předchozí linku
    pop    de                          ;obnovit DE

```

```
ret ;hotovo, návrat
```

```
;=====
; Posun adresy bodu v displeji ZX-Spectrum o bod doprava
;=====
```

```
incx:: ;posun o bod doprava
      rrc c ;rotace bitové masky
      ret nc ;Cy? NE = hotovo
```

```
; Byl překročen rámeček jednoho bajtu, je třeba adresovat
; další.
```

```
inc hl ;adresace dalšího bajtu
ret ;hotovo = návrat
```

```
;=====
; Posun adresy bodu v displeji ZX-Spectrum o bod doleva
;=====
```

```
decx:: ;posun o bod doleva
      rrc c ;rotace bitové masky
      ret nc ;Cy? NE = hotovo
```

```
; Byl překročen rámeček jednoho bajtu, je třeba adresovat
; předchozí.
```

```
dec hl ;adresace předchozího bajtu
ret ;hotovo = návrat
```

```
;=====
```

; Posun adresy bodu v displeji ZX-Spectrum o bod dolu

=====

```
incy::                                ;posun o bod dolu
    inc    h                            ;zvýšit o 1 y0..y2
    ld     a,h                          ;nová adresa
    and    00000111b                    ;překročen rozsah y0..y2?
    ret    nz                            ;NE = hotovo

    ld     a,l                            ;nižší bajt adresy
    add    a,20h                         ;zvýšit o 1 y3..y5
    ld     a,l                            ; do L
    ret    c                              ;rozsah y3..y5 překročen

    ld     a,h                            ;nová adresa
    sub    8                             ;zachovat y6..y7
    ld     h,a                            ; do H
    ret
```

=====

; Posun adresy bodu v displeji ZX-Spectrum o bod nahoru

=====

```
decy::                                ;posun o bod nahoru
    dec    l                              ;snížit y0..y2
    ld     a,l                            ;vyšší bajt adresy
    and    00000111b                    ;y0..y2
    cp     7                              ;překročen rozsah y0..y2?
    ret    nz                            ;NE = hotovo

    ld     a,l                            ;nižší bajt adresy
    sub    20h                            ;snížit y3..y5
    ld     l,a                            ; do L
    ret    c                              ;překročen rozsah y3..y5
```

```

ld      a,h          ;nová adresa
add     a,8          ;ponechat původní y6..y7
ld      h,a          ; do H
ret

```

3.7.5 Operace s bodem

Již jsme si ukázali jak adresovat jednotlivé body v displeji, ale ještě s nimi neumíme provést žádnou operaci. To je již velice jednoduché a ukážeme si to na podprogramech pro rozsvícení, zhasnutí, inverzi a test bodu v displeji.

```

; Vstupní parametry:
;      L = souřadnice X
;      H = souřadnice Y

; Výstupní parametry:
;      pouze u testu bodu GPOINT:
;      Z, NC = nesvítí
;      NZ, c = svítí

;=====
;      Rozsvícení bodu
;=====
gset::
    push    hl          ;uschovat HL
    call    addrXX      ;adresace displeje
    or      (hl)        ;nastavit adresovaný bit do
                        ;"1"
    ld      (hl),a      ;zapsat výsledek do displeje
    pop     hl          ;obnovit HL
    ret              ;návrat z podprogramu

```

```
;=====
;          Zhasnutí bodu
;=====
```

gres::

```
    push    hl          ;uschovat HL
    call    addrXX     ;adresace displeje
    cpl          ;"0" do adresovaného bitu,
                  ;"1" do všech ostatních
    and     (hl)        ;nastavit adresovaný bit do
                  ;"0"
    ld      (hl),a      ;zapsat výsledek do displeje
    pop     hl          ;obnovit HL
    ret                ;návrat z podprogramu
```

```
;=====
;          Inverze bodu
;=====
```

gxor::

```
    push    hl          ;uschovat HL
    call    addrXX     ;adresace displeje
    xor     (hl)        ;invertovat adresovaný bit
    ld      (hl),a      ;zapsat výsledek do displeje
    pop     hl          ;obnovit HL
    ret                ;návrat z podprogramu
```

```
;=====
;          Test bodu - svítí / nesvítí
;=====
```

gpoint::

```
    push    hl          ;uschovat HL
    call    addrXX     ;adresace displeje
    and     (hl)        ;otestovat adresovaný bit
```

```

pop     hl             ;obnovit HL
ret     z             ;nesvítí: Z, NC

scf                    ;svítí: nastavit Cy
ret                    ;návrat z podprogramu

```

3.8 Vrácení o krok

Uživatel GREDITORu má možnost kdykoliv se vrátit o jeden krok zpět. O tom, k čemu je tato funkce dobrá, již řeč byla, nyní si povíme o její programové realizaci.

Základní princip je v tom, že před provedením libovolné funkce si do paměti uložíme aktuální stav obrazové paměti. Teprve poté se vykoná zvolená funkce, která změní obrázek. Pokud se nyní chceme vrátit o krok zpět, přeneseme z paměti dříve uložený stav do obrazové paměti, čímž získáme obrázek ve stavu před provedením funkce.

Přenesení obrazové paměti do operační realizuje podprogram GETSCR. Pro pochopení jeho funkce je důležité si uvědomit rozdílnou adresaci operační a obrazové paměti.

```

; Vstupní a výstupní parametry:
;      žádné

```

```

;=====
;      Uložení obrazové paměti PMD-85
;=====

```

```

getscr::
    push    hl             ;uschovat všechny registry
    push    de
    push    bc
    push    af

```



```

ld      hl,0c000h   ;začátek obrazové paměti
ld      de,scrbufXX ;adresa místa pro uložení dat
ld      b,0         ;256 linek = čítač Y

gets1:                                     ;hlavní smyčka ukládání dat
push   bc          ;uschovat čítač Y

ld      c,44       ;počet bajtů na lince = čítač
                        X
dirl:   ld      a,(hl)   ;přenesení jednoho bajtu
        ld      (de),a
        inc    hl       ;zvýšit ukazatel do displeje
        inc    de       ;zvýšit ukazatel do paměti
        dec    c        ;snížit čítač X
        jp     nz,dirl  ;cykl přenesení jedné linky

```

; Přenesli jsme jednu linku, adresovat další

```

ld      bc,40h-44   ;offset linek
add     hl,bc       ;adresa začátku další linky

pop     bc          ;čítač Y
dec     b           ;snížit čítač Y
jp     nz,gets1    ;cykl - další linku

pop     af          ;obnovit všechny registry
pop     bc
pop     de
pop     hl
ret                                     ;návrat z podprogramu GETSCR

```

;=====
; Uložení obrazové paměti ZX-Spectrum

;=====

getscr::

```
    push    hl            ;uschovat všechny registry
    push    de
    push    bc
    push    af
```

```
dir1:  ld     hl,4000h     ;začátek obrazové paměti
        ld     de,scrbufXX ;adresa místa pro uložení dat
        ld     bc,1b00h   ;délka obrazové paměti
        ldir                    ;blokový přenos dat

        pop    af        ;obnovit všechny registry
        pop    bc
        pop    de
        pop    hl
        ret              ;návrat z podprogramu GETSCR
```

Pro zpětný přenos dat z operační paměti do obrazové stačí zaměnit dvě instrukce u návěští DIR1 takto:

; PMD-85

```
dir1:  ld     a,(de)      ;vzít bajt z operační paměti
        ld     (hl),a     ;zapsat ho do displeje
```

; ZX-Spectrum

```
dir1:  ld     hl,scrbufXX ;adresa dat v paměti
        ld     de,4000h   ;adresa displeje
```

3.9 Hlavní smyčka GREDITORu

Na závěr programátorské kapitoly ještě pro ilustraci a poučení uvádím výpis hlavní smyčky GREDITORu, tak jak je použita v programu, který máte na kazetě. Je shodná pro PMD-85 i ZX-Spectrum.

```
;=====
;       Hlavní smyčka grafického editoru GREDITOR
;=====

main::                                ;začátek hlavní smyčky
    ld      sp,spbaze                 ;nastavení zásobníku
    ei                                     ;povolení přerušení

    ld      hl,main                   ;adresa hlavní smyčky
    push    hl                        ;uložit ji na zásobník

    call    curlbl                    ;vykreslit textový kurzor,
                                        ;pokud je nastaven mód "text"
    ld      hl,(mousex)               ;souřadnice kurzoru
    call    crsonXX                   ;vykreslit kurzor
    call    mode0                     ;nastavit zvolený grafický mód
    call    color0                    ;nastavit zvolený barevný mód

wtkey:
    ld      a,(funkce)                ;zvolená funkce
    cp      fcebof                    ;funkce "bod"?
    jp      nz,wtk1                   ;NE

; Funkce "bod" se zpracovává trochu jinak - čeká se na data
; z myši.

    call    bminXX                    ;vezmi informace z myši
    jp      wtk3
```

```
wtk1:    call    minXX      ;vezmi informace z myši, bez
                                čekání
```

```
wtk3:    jp      nz,wtk2    ;informace jsou = zpracovat
```

```
ld      a,(funkce) ;zvolená funkce
```

```
cp      fcelbl      ;funkce "text"?
```

```
jp      nz,wtkkey   ;NE = znovu do smyčky
```

; V módu "text" se testuje i klávesnice - vypisují se znaky.

```
call    $inkeyXX    ;test klávesnice bez čekání
```

```
jp      z,wtkkey    ;žádná klávesa = do smyčky
```

; Stisknuta klávesa v módu "text", vypsát znak.

```
call    crsoffXX    ;smazat kurzor
```

```
call    curlbl      ;smazat textový kurzor
```

```
call    mode0       ;nastavit zvolený grafický mód
```

```
call    color0      ;nastavit zvolený barevný mód
```

```
jp      pischar     ;vykreslit znak
```

; Zpracovat změnu stavu myši.

```
wtk2:    ld      b,a      ;status myši
```

```
call    crsoffXX    ;smazat kurzor
```

```
call    curlbl      ;smazat textový kurzor
```

```
ld      a,b          ;status myši
```

```
and    10100000b    ;bity tlačítka MENU
```

```
cp      10100000b    ;stisknuto MENU?
```

```
jp      z,gomenu    ;ANO = vypsát MENU atd.
```

```
ld      a,b          ;status myši
```

```
and    01000000b    ;bity tlačítka SELECT
```

```

    jp      nz,fce      ;drženo tlačítko SELECT =
                        funkce
    ret                                ;návrat do hlavní smyčky

;=====
;      Vyvolání zvolené funkce
;=====

;      HL = souřadnice kurzoru
;      B  = status myši

fce:
    ld      de,frtn     ;adresa pro návrat z funkce
    push   de           ;připravit ji na zásobník

    push   hl           ;uschovat souřadnice
    ld     a,(funkce)   ;nastavená funkce
    add    a,a           ;vynásobit dvěma pro indexaci
                        do tabulky
    ld     l,a           ;index jako 16 bitů do HL
    ld     h,0
    ld     de,tabfci    ;adresa tabulky funkcí
    add    hl,de         ;indexace do tabulky
    ld     a,(hl)       ;vybrat z tabulky adresu
                        funkce

    inc    hl
    ld     h,(hl)
    ld     l,a           ;v HL je adresa funkce

    ex     (sp),hl      ;do HL souřadnice
    ld     a,b           ;status myši do A
    and    00010000b    ;změna stavu tlačítka SELECT?
    ret    z             ;NE = skok na funkci
    jp     getscrXX     ;ANO = uložit obrazovku

```

```

frtn:                ;návrat z funkcí
                    ret      ;do hlavní smyčky

```

```

;=====
;      Vyvolání volby z nabídky
;=====

```

```

;      HL = souřadnice kurzoru

```

gomenu:

```

ld      hl,tabmenu   ;adresa tabulky hlavní nabídky
call    menuXX      ;volba z nabídky
ld      c,a          ;zvolená funkce do C
add     a,a          ;vynásobit 2x pro indexaci
ld      l,a          ;do HL jako 16 bitů
ld      h,0
ld      de,mnfce    ;tabulka podprogramů pro
                    funkce
add     hl,de        ;indexace do tabulky
ld      e,(hl)      ;vybrat adresu z tabulky
inc     hl
ld      d,(hl)      ;adresa je v DE
push    de          ;adresu na zásobník
ld      a,c          ;číslo zvolené funkce do A
ld      hl,(mousex) ;souřadnice kurzoru
ret     \           ;skok na podprogram

```

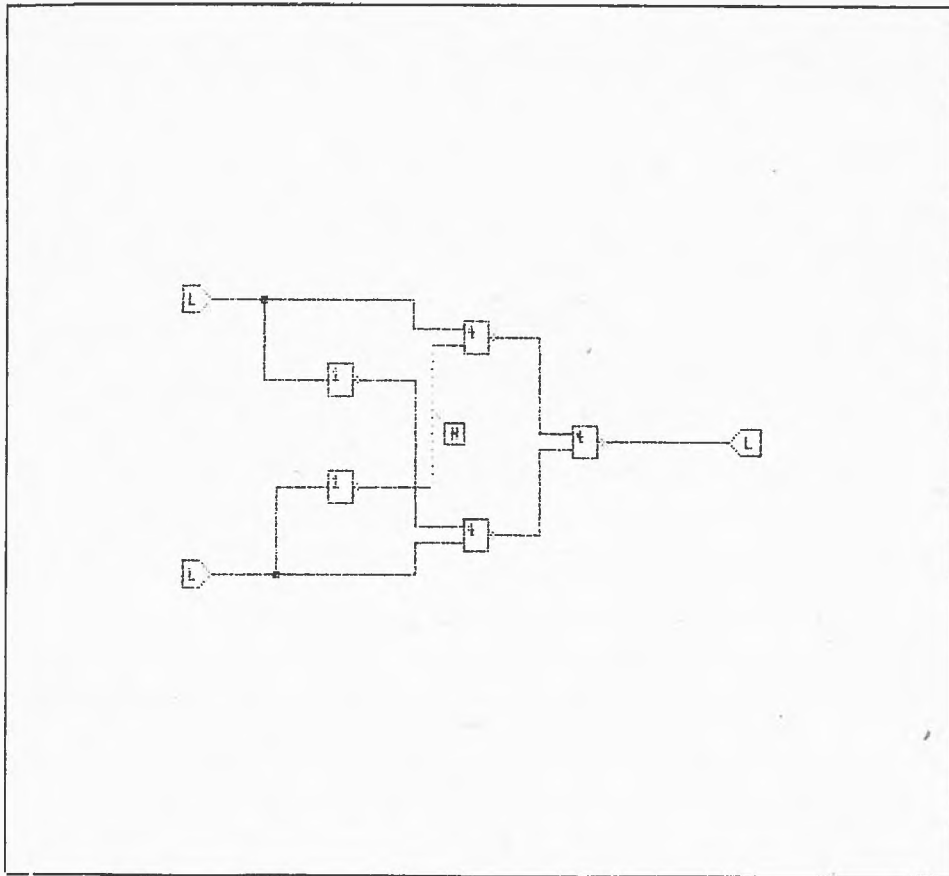
3.10 Příklad další aplikace elektronické myši

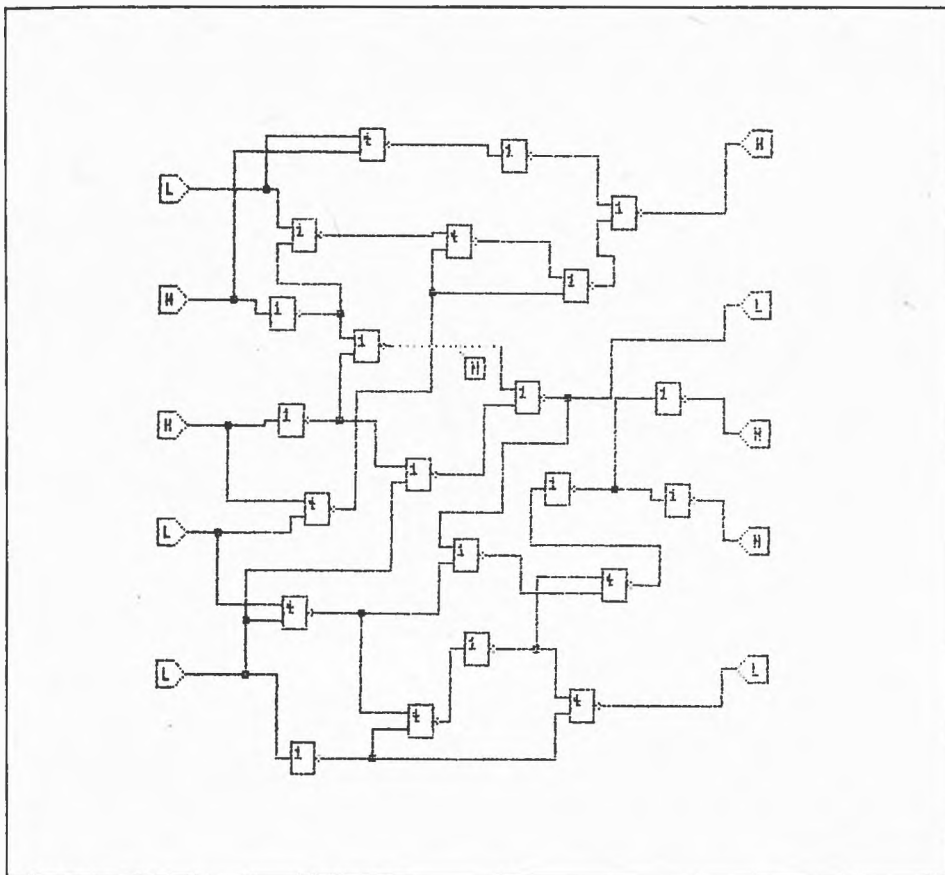
Na závěr našeho povídání Vás seznámím s programem **SLOU**, který má zcela jiný účel než GREDITOR, ale přesto s ním má mnoho společného. Nejdříve stručný popis funkce programu **SLOU**.

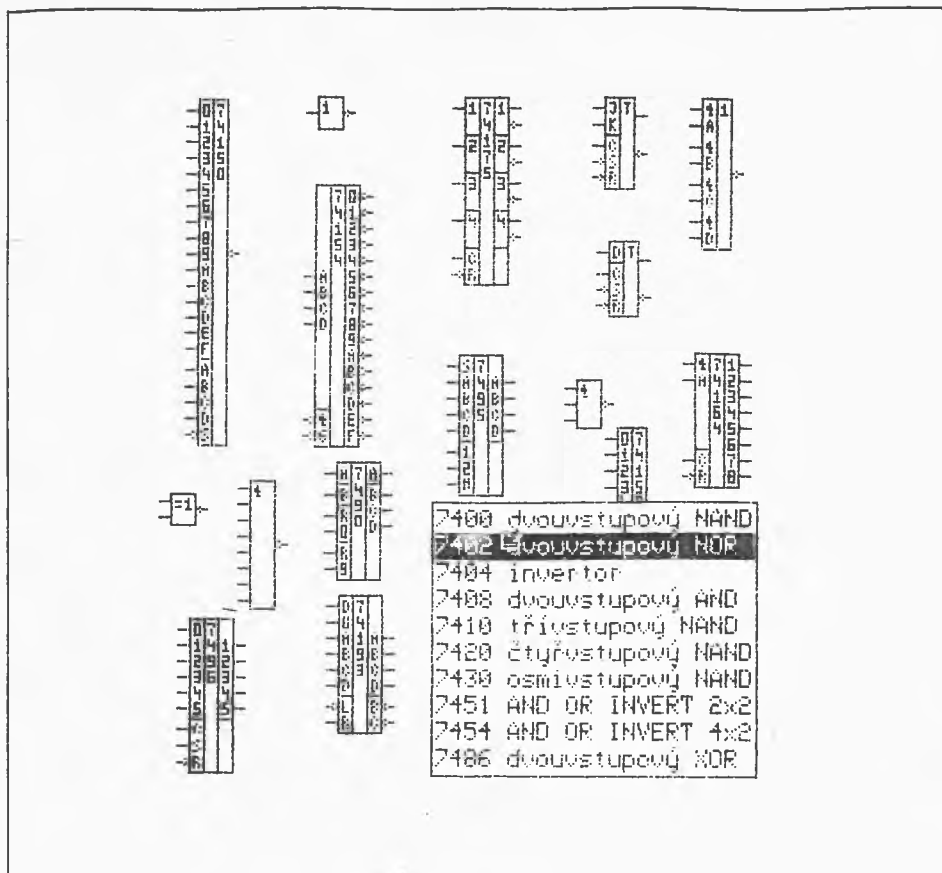
Program **SLOU** je Simulátor logických obvodů. Uživatel má k dispozici knihovnu základních digitálních integrovaných obvodů typu TTL - asi 50 typů základní řady 74.. Z knihovny si vybere potřebné typy a jejich schematické značky rozmístí na displeji. Jejich vstupy a výstupy propojí drátovými spoji. Po zapojení celého navrženého obvodu se provede simulace funkce tohoto obvodu. Přitom se pro jednoduchost předpokládá shodné zpoždění průchodu signálu všemi typy obvodů. Logickou hodnotu důležitých bodů obvodu si může zobrazit pomocí výstupních prvků, pomocí logické sondy si může zobrazit logickou hodnotu v libovolném místě obvodu. Při nalezení chyby může rušit nebo doplňovat spoje i prvky až do vytvoření obvodu se správnou funkcí. Hotový obvod lze vytisknout na tiskárně, vykreslit na souřadnicovém zapisovači nebo uložit na magnetofon k dalšímu použití.

Program **SLOU** je počítačovou simulací stovebnice DS 200 a má sloužit k výuce.

Pro ilustraci ještě několik ukázek z práce s programem **SLOU**:







Nevíte, co má program SLOU společného s grafickým editorem? Celý princip ovládání je opět orientován na MYŠ! Všechny funkce se volí pomocí myši z nabídky, rozmísťování či rušení prvků a spojů se opět jednoduše a rychle děje pomocí myši. Opět je zde hojně využito grafického zobrazení i dynamického kreslení, je zde již použito několika tvarů kurzoru a kreslení kurzoru sprajtovou metodou.

V obou programech jsou použity některé shodné podprogramy:

- komunikace s myší
- kreslení kurzoru
- volba z nabídky
- základní grafické funkce

- dynamické kreslení
- operace s magnetofonem

Proč zde popisují program SLOU? Ze srovnání GREDITORu a SLOU je vidět, že využití myši je opravdu univerzální a výhodné a není omezeno jen na oblast kreslení obrázků grafickým editorem. Další výhodou při použití myši k ovládní programu je univerzálnost základních podprogramů. Stačí je vytvořit pouze jednou a pak už je lze takřka beze změny (většinou úplně beze změny) využívat v různých programech, což podstatně zvyšuje efektivitu programování.

ZÁVĚR

Závěr

Účelem knihy, kterou jste právě dočetli, bylo seznámit Vás s problematikou využívání elektronické myši a grafického editoru z hlediska uživatele a posléze i programátora. Cíle Vaší práce s touto knihou můžeme definovat těmito body:

- připojit sestavenou myš k počítači a oživit ji
- naučit se používat myš k ovládní grafického editoru
- kreslit pomocí GREDITORu obrázky
- na základě předchozích bodů si uvědomit obecnější souvislosti - možnosti elektronické myši
- pokud jste programátor, seznámit se se základními problémy při komunikaci s myší a způsoby jejich řešení a tím si umožnit efektivní tvorbu vlastních programů, které budou využívat elektronickou myš.

**Na závěr mi dovolte popřát Vám mnoho
hezkých chvil s myší v ruce!**

Děkuji za pozornost.

Ing. Vít Libovický

Grafický editor

Vydal ústřední výbor Svazarmu a ZO Svazarmu 4006/602 jako součást dokumentace a materiálu dálkového interaktivního kursu elektronické myši - samostatně neprodejné. Vydání první. Adresa redakce a organizačního sekretariátu kursu: 602. ZO Svazarmu, Wintrova 8, 160 41 Praha 6.

