****
** **
** Vše o MZ - 800 **
** **
****

BASIC MZ-800

Tento BASIC je ve dvou verzích, pro Q-disk a pro CMT. Zásoba povelů obou verzí BASICu je stejná. Ve verzi pro CMT jsou chyby, které byly ve verzi pro Q-disk opraveny. Také kapacita paměti u obou verzí je stejná.

Jestliže máte možnost se dostat k verzi pro Q-disk, potom vám radím používat tutu verzi přetočenou na kazetu.

Proto již 100% nesouhlasí jednotlivé adresy programů. Jedině prvních 8 KB je stejných. My se zde budeme zabývat hlavně Q-disk verzí

Další povely BASICu

V BASICu MZ-800 jsou k dispozici další povely, které nejsou nikde uvedeny. Pro přehlednost jsou zde vyjmenovány a krátce vysvětleny.

BEEP - písknutí (440 Hz)
EDIT - vypíše naposledy zpracovávané číslo řádky. Je také možno psát EDIT n .
FRAC - vypíše desetinou část desetinného čísla
A=FRAC (12.345) => PRINT A => .345
HEX$ (x) - převod dekadického čísla (x) na hexadecimální
MOD - modulo dělení. Vypíše celočíselný zbytek při dělení.
B=11 MOD 3 => PRINT B => 2
SPÁCE$ - vytvoří prázdný řetězec znaků.
LOAD ALL - tento příkaz provede nahrátí všech programů na Q-disku do RAM-disku. Tam se mohou snadno provést příkazy RENAME nebo DELETE.
SAVE ALL - teto příkaz provádí nahrátí všech prograů z RAM na Q-disk.

Úprava klávesnice - psací stroj

MZ-800 nemá na klávesnici žádné zvláštní klávesy pro přehlásky a německé znaky. to bude nepříjemné především pro ty uživatele, kteří často používají mikropočítač pro zpracování textů a dat.

Zvláštní znaky a přehlásky jsou uloženy ve znakovém generátoru a nechají se nadefinovat místo funkčních kláves F1 - F10.

```
10 DEF KEY (1) = CHR$ (187): REM ü
20 DEF KEY (2) = CHR$ (186): REM o
20 DEF KEY (3) = CHR$ (173): REM u
```

```
30 DEF KEY (4) = CHR$ (174): REM
40 DEF KEY (6) = CHR$ (185): REM A
50 DEF KEY (7) = CHR$ (168): REM O
60 DEF KEY (8) = CHR$ (178): REM U
```

Tímto krátkým programem se nadefinují přehlásky na funkční klávesy.

Ukládání programu v BASICu do paměti (TOKEN)

Program v BASICu se neukládá do paměti tak, jak je zobrazen na obrazovce, nebo jak se vkládá, ale interpret BASICu má pro každý příkaz číslo, kterému se říká kód. Podle tohoto kódu může potom interpretr s pomocí tabulky zjistit, o jaký příkaz se jedná. Výhoda je jasná. Místo aby si např. pro příkaz -RESTORE- pamatovat 7 naků, stačí aby si pamatovat pouze dva znaky (kod), čímž je malá spotřeba paměti.

Příkazy BASICu MZ - 800

Vtéto části si zopakujeme všechny příkazy BASICu, jejich zápis v kódu, přípustné skratky a začátek programů v paměti. Koho by toto zajímalo, může si pomocí DISASSEMBLERu tyto programy (tabulky) prohlédnout, nebo i pozměnit.

Kódy se rozdělují do tří skupin. V jedné skupině jsou příkazy, které nepotřebují žádný před-byt. Ve zbývajících skupinách jsou příkazy, které tento byt obsahují. Známe dva druhy 254 (FEh) a 255 (FFh). Na druh příkazu to nemá vliv.

Mezi příkazy se vyskytují kódy, které nejsou obsazeny. Tyto jsou v naší tabulce označeny jako "frei" (volné) a spůsobují přímo skok do podprogramu, který vydává chybové hlášení "Syntax Error. Tento program začíná u obou verzí BASICu na adrese 6364h.

příkaz	zkratka	TOKEN	QD-BASIC	CMT-BASIC
GOTO	G.	128	69AF	69AF
GOSUB	GOS.	129	692A	692A
volný		130	6364	6364
RUN	R.	131	606E	606E
RETURN	RE.	132	68F9	68F9
RESTORE	RES.	133	6E9E	6E9E
RESUME	RESU.	134	64B6	64B6
LIST	L.	135	6A9D	6A9D
volný		136	6364	6364
DELETE	D.	137	620C	620C
RENUM	REN.	138	62AB	62AB
AUTO	A.	139	60CA	60CA
EDIT	E.	140	6125	6125
FOR	F.	141	6602	6602
NEXT	N.	142	6609	6609
PRINT	P.	143	6C9F	6C9F
volný		144	6364	6364
INPUT	I.	145	6DD4	6DD4
volný		146	6364	6364
IF		147	69C2	69C2
DATA	DA.	148	67FE	67FE
READ	REA.	149	6DCD	6DCD

DIM	DI.	150	8DCC	8D55
REM		151	67FE	67FE
END	EN.	152	60B9	60B9
STOP	S.	153	6468	6468
CONT	C.	154	6499	6499
CLS	CL.	155	7972	78FB
volný		156	6364	6364
ON.	O.	157	6952	6952
LET	LE.	158	650D	650D
NEW		159	618E	618E
POKE	PO.	160	68A1	68A1
OFF	OF.	161	6364	6364
PMODE	PM.	162	A015	9F9E
PSKIP	PS.	163	A07A	A003
PLOT	PL.	164	A428	A3B1
PLINE	PLI.	165	A0BC	A045
RLINE	RL.	166	A0BE	A048
PMOVE	PMOV	167	A0C2	A04B
RMOVE	RM.	168	A0C5	A04E
TRON	T.	169	61CC	61CC
TROFF	TROF.	170	61DF	61DF
INP		171	681F	681F
DEFAULT	DEF.	172	6F41	6F41
GET	GE.	173	686C	686C
PCOLOR	PC.	174	A12A	A0B3
PHOME	PH.	175	A15F	A0E8
HSET	H.	176	A162	A0EB
GPRINT	GP.	177	A16D	A0F6
KEY	K.	178	6B84	6B84
AXIS	AX.	179	A1D5	A15E
LOAD	LO.	180	6FFD	6FFD
SAVE	SA.	181	73C0	733F
MERGE	M.	182	7068	7068
CHAIN	CH.	183	7053	7053
CONSOLE	CONS.	184	6A61	6A61
SEARCH	SE.	185	6A9B	6A9B
OUT	OU.	186	6B0A	6B0A
PCIRCLE	PCI.	187	A215	A19E
PTEST	PT.	188	A3F3	A36C
PAGE	PA.	189	A3F1	A37A
WAIT	W.	190	67EE	67EE
SWAP	SW.	191	74A6	7425
volný		192	6364	6364
EROR	ER.	193	63F8	63F8
ELSE	EL.	194	69E4	69E4
USR	U.	195	67A4	67A4
BYE	B.	196	6A59	6A59
volný		197	6364	6364
volný		198	6364	6364
DEF		199	6C31	6C31
volný		200	6364	6364
volný		201	6364	6364
LABEL	LA.	202	67FE	67FE
volný		203	6364	6364
volný		204	6364	6364
volný		205	6364	6364
WOPEN	WO.	206	6F59	6F59
CLOSE	CLO.	207	6F8A	6F8A
ROPEN	RO.	208	6F56	6F56
XOPEN	X.	209	6F5C	6F5C
volný		210	6364	6364

volný			211	6364	6364
volný			212	6364	6364
DIR			213	742A	73A4
volný			214	6364	6364
volný			215	6364	6364
RENAME	REN.		216	744B	73CA
KILL	KI.		217	6F8B	6F8B
LOCK	LOC.		218	7418	7397
UNLOCK	UN.		219	7416	7395
INIT	INI.		220	6F47	6F47
volný			221	6364	6364
volný			222	6364	6364
volný			223	6363	6364
volný		254	128	6364	6364
CSET	CS.	254	129	6364	6364
CRESET	CR.	254	130	6364	6363
CCOLOR	CC.	254	131	6364	6364
volný		254	132	6364	6364
volný		254	133	6364	6364
volný		254	134	6364	6364
volný		254	135	6364	6364
volný		254	136	6364	6364
volný		254	137	6364	6364
SOUND	SO.	254	138	9BAC	9B35
volný		254	139	6364	6364
NOISE	NO.	254	140	9BE3	9B6C
BEEP	BE.	254	141	6A56	6A56
volný		254	142	6364	6364
volný		254	143	6364	6364
COLOR	COL.	254	144	7675	760B
volný		254	145	6364	6364
SET		254	146	7592	752B
RESET	RESE.	254	147	7593	7529
LINE	LIN.	254	148	75A1	7537
BLINE	BL.	254	149	75A2	7538
PAL		254	150	7619	75AF
CIRCLE	CI.	254	151	774C	76DE
BOX	BO.	254	152	764B	75DE
PAINT	PAI.	254	153	76FD	768F
POSITION	POS.	254	154	75F5	758B
PATTERN	PAT.	254	155	75C4	755A
HCOPY	HC.	254	156	7968	78F1
volný		254	157	6364	6364
volný		254	158	6364	6364
volný		254	159	6364	6364
SYMBOL	SY.	254	160	790F	78A1
volný		254	161	6364	6364
MUSIC	MU.	254	162	9BE6	9B6F
TEMPO	TE.	254	163	9BD9	9B62
CURSOR	CU.	254	164	683B	683B
VERIFY	V.	254	165	73A9	7328
CLR		254	166	619B	619B
LIMIT	LIM.	254	167	68B5	68B5
volný		254	168	6364	6364
volný		254	169	6364	6364
volný		254	170	6364	6364
volný		254	171	6364	6364
volný		254	172	6364	6364
volný		254	173	6364	6364
BOOT	BOO.	254	174	6A95	6A95
INT		255	128	95C2	954B

ABS	AB.	255	129	95BD	9546
SIN	SI.	255	130	9709	9692
COS		255	131	96F9	9682
TAN		255	132	97E8	9771
LN		255	133	9A4F	99D8
EXP	EX.	255	134	992B	98B4
SQR	SQ.	255	135	961F	95A8
RND	RN.	255	136	98E6	986F
PEEK	PE.	255	137	98DA	9863
ATN	AT.	255	138	963D	95C6
SGN	SG.	255	139	98A0	9829
LOG		255	140	9A43	99CC
FRAC	FR.	255	141	7BC0	7B49
PAI		255	142	98B8	9841
RAD	RA.	255	143	98B3	983C
volný		255	144	6364	6364
volný		255	145	6364	6364
volný		255	146	6364	6364
volný		255	147	6364	6364
volný		255	148	6364	6364
volný		255	149	6364	6364
volný		255	150	6364	6364
volný		255	151	6364	6364
volný		255	152	6364	6364
volný		255	153	6364	6364
volný		255	154	6364	6364
volný		255	155	6364	6364
STICK	STI.	255	156	9060	8FE9
STRIG	STR.	255	157	907F	9008
volný		255	158	6364	6364
volný		255	159	6364	6364
CHR$	CHR.	255	160	889F	88BA
STR$		255	161	8931	8879
HEX$	HE.	255	162	88F0	88F0
volný		255	163	6364	6364
volný		255	164	6364	6364
volný		255	165	6364	6364
volný		255	166	6364	6364
volný		255	167	6364	6364
SPACE$	SPA.	255	168	8913	889C
volný		255	169	6364	6364
volný		255	170	6364	6364
ASC	AS.	255	171	8992	891B
LEN		255	172	899C	8925
VAL	VA.	255	173	89A4	892D
volný		255	174	6364	6364
volný		255	175	6364	6364
volný		255	176	6364	6364
volný		255	177	6364	6364
volný		255	178	6364	6364
ERN		255	179	887A	8803
ERL		255	180	8888	8811
SIZE		255	181	8852	87DB
CSRH	CSR	255	182	8864	87ED
CSRV		255	183	8869	87F2
POSH		255	184	886E	87F7
POSV		255	185	8874	87FD
LEFT$	LEF.	255	186	89B6	893F
RIGHT$	RI.	255	187	89D3	895C
MID$	MI.	255	188	89F0	8979
volný		255	189	6364	6364

volný		255	190	6364	6364
volný		255	191	6364	6364
volný		255	192	6364	6364
volný		255	193	6364	6364
volný		255	194	6364	6364
volný		255	195	6364	6364
TI$	TI.	255	196	8A46	89CF
POINT	POI.	255	197	896C	88F5
EOF	EO.	255	198	893F	88C8
FN		255	199	8FBF	8F48
volný		255	200	6364	6364
volný		255	201	6364	6364
volný		255	202	6364	6364
volný		255	203	6364	6364
volný		255	204	6364	6364
volný		255	205	6364	6364
volný		255	206	6364	6364

Celkový přehled BASICu MZ - 800

Na následujicích stránkách si popíšeme obě verze BASICu.

0000 - RAM - monitorová tabulka pro skok na 005A.

005B - Tabulka s počátečními sdresami CTRL programy (řídící programy) až po 009A. Následuje příslušný program.

00DA - Zde začíná BASIC.

017C - Tabulka s počátečnímy adresami různých programů až do 022A.

0247 - Hexadecimální hodnota v registru HL se ukládá v decimálním ASCII formátu do registru DE.

02F8 - Zde se zkoumá, zda byla stisknuta prázdná klávesa (mezerník). Jestli-že ano, čeká se až bude stisknuta jiná klávesa.

0305 - Bude-li stisknuta klávesa BREAK, program bude přerušen.

0314 - Zde jsou krátké programy pro vedení ukazatelů pro proměnné a pro oblast řetězců.

0340 - Chybový text se zde převádí na srozumitelný text podle čísla chyby v akumulátoru.

03BE - Určuje způsob provozu obrazovky a zkoumá se připojení přídavné paměti VRAM.

0419 - Zde se nacházejí různé programy, které inicialisují paletové registry a určují oblast přetáčení obrazu.

04EF - Až po 05AC jsou programy,které přestavují obrazovku na 40 nebo 80 znaků na řádek a mění ostatní programy odpovídajícím způsobem.

05AD - Tento program vypíše znak v akumulátoru na obrazovku.

0652 - Podprogram smaže celou obrazovku (CLS).

06CC - Podprogram maže znak na pozici kursoru.

0848 - Posouvá oblast přetáčení o jeden řádek nahoru.

0992 - Tento podprogram přemístní kursor do levého horního rohu obrazovky (HOME)

099B - Podprogram způsobý posun řádky, jestliže kursor nestojí na začátku řádky.

09AB - Podprogram způsobý v každém případě posun o řádek.

0A19 - Nastavení klávesnice - ALPHA mód.

0A1B - Nastavení klavesnice - psaní malých písmen.

0A1E - Nastavení klávesnice - GRAPH módu.

0A24 - Posun kursoru na pozici dalšího tabulátoru.

0AB4 - Program vyzvedne řádku z klávesnice a uloží ji od adresy DE.

0B21 - Program vyzvedne znak z klávesnce. Kursor přitom bliká. Znak se při skoku zpět nachází v akumulátoru.

0BF6 - Testuje se klávesnice. Není-li stisknuta žádná klávesa, v akumulátoru je nula (00), v opačném případě hodnota příslušné klávesy v ASCII kódu.

0C7C - Testuje se stisknutí funkčních kláves. V HL je uloženo která klávesa byla stisknuta.

0D22 - Testuje se klávesnice. Při stisknutí SHIFT+BREAK se nastaví přepíneč na 0. Byl-li stisknut jen BREAK, čeká se tak dlouho dokud se opět nepustí.

0D47 - Zde začínají tabulky k přepočtu kódu kláves.

0D94 - Program vypisuje na obrazovku text, ktery je od adresy DE. Konečná značka textu musí být 0DH

0DC3 - Tabulka malých písmen ve video kódu

0E17 - Od této adresy jsou tiskové programy pro výpis na obrazovku nebo tiskárnu. Tyto programy sahají až po 0ED8.

0ED9 - V této tabulce se převádějí řídící znaky pro tiskárnu.

0EF9 - K této adrese přijdou skoky z ROM monitoru zpět do BASICu. Program, který je k dispozici se nesmaže.

0F4E - Program testuje, zda VRAM paměť obsahuje 32 Kb. Jestli-že ano, pak se na adrese 1D99 zapíše 01H

12B2 - Od téty adresy leží text funkčních kláves.

13D2 - Tabulka, která přepočitává SHARP kód na ASCII a zpět.

15E0 - Tabulka udává povely monituru a jejich počáteční adresy.

D	DUMP	16C6
M	MEMORY CORECTION	1738
P	PRINTER	15FE
G	GOTO	16C1
F	FIND	175C
R	RETURN (BASIC)	1644
S	SAVE	1607
L	LOAD	161E
V	VERIFY	163B
T	TRANSFER	17AC

2000 - Od této adresy až do 27D0 je celý obsah obrazovky. Při 40 znacích je potřeba jen polovina této paměti (23E7). Může být ale pouze čtena, například pro výpis na tiskárnu. Zápis do této oblasti nemá žádný vliv na obraz obrazovky.

Po těchto programech začíná vlastní BASIC, tj. všechny povely atd. Protože BASIC je velmi rozsáhlý a podporuje různé podprogramy, nebudeme se jím zde zabývat, protože k tomuto účelu je podrobnější literatura.

Rozšíření BASICu

V následující části vám chceme ukázat, jak můžete program ménit. K tomuto účelu jsme napsali několik malých programů ve strojovém jazyce, kteřé BASIC modifikují. Protože existují dvě verze BASICu, (kazetová a Q-disk verze), napsali jsme je pro obě dvě.

Rozšíření nepotřebuje žádné přídavné místo v paměti, protože se používají oblasti, které jsou volné v samotném BASICu.

1. BORDER - barva okraje (BOR.)

Pomocí tohoto povelu si můžeme nadefinovat barvu okraje obrazovky. Za povelem BOR. musí být konstanta, proměnná nebo funkce. Konstanta barev nabývá hodnotu 0 - 15 , což odpovídá tabulce v SHARP manuálu.

pro kazet. verzi BASICu (dále jen, pro CMT)

```
POKE 23017, 66,79,210
POKE 21936, 205,218,132,123,1,207,6,237,121,201
POKE 23709, 176,85
```

pro Q-disk verzi BASICu (dále jen, pro Q-disk)

```
POKE 23017, 66,79,210
POKE 21913, 205,81,133,123,1,207,6,237,121,201
POKE 23709, 153,85
```

2. SEARCH, LIST - s prázdnou řádkou

Tento povel přidá za každou řádku jednu řádku prázdnou. To usnadňuje především u delších programů lepší přehlednost.

pro CMT

```
     POKE 21946, 58,131,107,205,6,0,201
   POKE 27424, 205,146,85
```

pro Q-disk

```
   POKE 21906, 58,131,107,205,6,0,201
   POKE 27424, 205,146,85
```

prázdná řádka se zapíná - POKE 21909, 205
vypíná - POKE 21909, 201

3. Pípnutí při READY

Podprogram vytvoří při každém zobrazení READY ještě krátké pípnutí. Tím máte ještě akustickou kontrolu. To má smysl hlavně při dlouých programových výpočtech, nebo při nahrávání.

pro CMT

```
   POKE 21953, 205,62,0,17,240,99,201
   POKE 22650, 205,193,85
```

pro Q-disk

```
   POKE 21899, 205,62,0,17,240,99,201
   POKE 22650, 205,139,85
```

4. Zakázané zastavení pomocí BREAK

Tento POKE zakáže použití příkazu BREAK. Program se nedá zastavit.

pro CMT + Q-disk

```
   POKE 3368, 201          zastavení je zakázané
   POKE 3368, 200          zastavení je povoleno
```

5. Zakázané přerušení pomocí SHIFT / BREAK

Tento POKE zakáže přerušení programu pomocí SHIFT / BREAK.

pro CMT + Q-disk

```
     POKE 22754, 62          přerušení je zakázané
   POKE 22754, 202         přerušení je povoleno
```

6. Změna kursoru v ALPHA módu

Pomocí tohoto POKE se může změnit zobrazení kursoru v ALPHA

módu. Může být též zabudován do vlastních programů.

pro CMT a Q-disk

```
POKE 5009, 255,255,255,255,255,255,255,255
```

Toto je normální kursor. Chcete-li ho změnit musíte zadat vždy vžech 8 bytů. Je tedy téměř neomezené množství změn kursorů. Kursor se tvoří stejně jako PCG znak.

```
POKE 5009, 255,129,129,129,129,129,129,255
```

Zde je návrh nového kursoru, vytvoří krabici.
Pozn.: Za touto adresou následuje kursor v SHIFT / ALPHA a GRAPH módu.

7. Možnosti ochrany programů v BASICu

Tím že programy po nahrátí nejsou aoutomaticky spuštěny. není v BASICu žádná moznost programy 100% chránit proti nedovolenému kopírování. Přesto existuje možnost programy po spuštění pomocí RUN chránit. Některé z těchto možností vám zde chceme krátce představit. Pomocí následujících POKE příkazů je BASIC modifikován takovým způsobem, že některé příkazy nemůže provádět.

pro CMT

```
ochrana proti SAVE      POKE 29503, 201
                        POKE 29503, 205    původ. nastav.
      proti LIST        POKE 27293, 201
                        POKE 27293, 62     původ. nastav.
      proti PEEK        POKE 39011, 201
                        POKE 39011, 229    původ. nastav.
      proti BYE         POKE 27228, 201
                        POKE 27228, 223    původ. nastav.
```

pro Q-disk

```
ochrana     proti SAVE  POKE 29632, 201
                        POKE 29632, 205    původ. nastav.
      proti LIST        POKE 27293, 201
                        POKE 27293, 62       původ. nastav.
      proti PEEK        POKE 39130, 201
                        POKE 39130, 229      původ. nastav.
      proti BYE         POKE 27228, 201
                        POKE 27228, 223      původ. nastav.
```

Pozn.: Tato ochrana je nedostačující. Jednodoší je chránit program proti zastavení, blokováaním BREAK při běhu aINPUT, ošetření chyb a úpravou CTRL+RESET

8. Odstranění otazníku při INPUT

Především při obchodním použití bývá často otazník, který se zobrazuje při příkazu INPUT X považován za rušivý.

pro CMT a Q-disk

```
POKE 28213, 0,0
POKE 28213, 63,32       vymazání otazníku
POKE 28213, 32,58       bude psát dvojtečku
```

Pozn.: Provedete-li INPUT "";X otazník tištěn není.
Otazník se vytiskne pouze při neodeslání čísel.

9. CONSOLE bez mazání obrazovky

Jistě jste si všimli, že při provádění příkazu CONSOLE se nejprve smaže obrazovka a pak se teprve provede CONSOLE

Často by bylo vhodné, aby zbytek obrazovky zůstal zachován. Je možné provádě CONSOLE i bez předchozího smazání obrazovky.

Ale pozor, musí se opatrně aby nebyl obraz na obrazovce zničen. (Teto postup není příliš korektní, může dělat samovolné přesuny na obrazovce.)

pro CMT a Q-disk

```
POKE 1257, 0,0,0        bez smazání obrazovky
POKE 1257, 205,82,6     obrazovka se smaže
```

Prakticky není výhodné v případě, že byl obraz ji SCROLLován.

10. Zrychlené SAVE a LOAD

Pomocí povelů POKE máte možnost zvýšit rychlost nahrávání a přehrávání o výce než dvojnásobnou rychlost. Naše pokusy prokázaly, že se zvýšenou rychlostí nezhoršila spolehlivost záznamu Také příkazy WOPEN a ROPEN jsou zrychleny.

Samozřejmě je také možnost staré, pomalé nahrávky převzít do nového formátu. K tomu potřebujete POKE aktivovat až po nahrátí. Nové rychlé nahrávky nemohou být náhrány bez rozšířeného BASICu a naopak.

pro CMT a Q-disk

Rychlé SAVE, LOAD	Normální SAVE, LOAD
POKE 15249, 35	POKE 15249, 76
POKE 15255, 11	POKE 15255, 24
POKE 15259, 49	POKE 15259, 105

11. Zlepšený příkaz TRON

Jistě jste si jiš všimli, že povel tron je v BASICu 800 naprogramovan poněkud nedostatečně. A tak je například těžké hledat chybu v programu, když je obrazovka stále popsána čísly

My jsme povel TRON modifikovali tak, že je nyní možné zobrazovat číslo řádky stále v levém horním rohu. Vlastní stavba obrazovky tím zůstává zachována a je jednoduší program sledovat. Stisknutímy BREAK můžeme být průběh programu ještě zpomalen.

por CMT

```
POKE 22400, 42,134,16,229,42,130,16,229,33
POKE 22409, 0,0,34,130,16,33,0,128,34,134
POKE 22421, 218,157,205,33,
POKE 22425, 121,223,11,6,5,62,32,205,18,0,16,249,225,34
POKE 22439, 130,16,225,34,134,16,195,6,98
```

Tento program je nový TRON program.

spustí se — POKE 25078, 195,128,87,
starý TRON prog.(pro tisk) — POKE 25078, 62,91,223

pro Q-disk

```
POKE 22421, 81,158,205,152
```

MZ - 700 BASIC 1Z-013B v1.0A

Tato část knihy se zabývá interpretem BASICu MZ-700. Speciálně půjde o kazetovou verzi BASICu, která se od Q-disk verze BASICu liší hlavně tím, že adresy přesně neodpovídají.

K mikropočítači MZ-800 patří dvě různé verze BASICu. jedna z nich je MZ-700 BASIC, který sice ponechává více volného místa v paměti, ale nepodporuje zváštní možnosti grafiky a generování tónů. Proto se zvlášť hodí pro psaní programů, které tyto zvláštnosti nepotřebují jako nabříklad práce s daty.

MZ-700 BASIC se vždy hlásí bílou barvou na modrém pozadí a tím se dá sndno rozeznat od MZ-800.

BASIC se nahrává z kazety nebo disku a potom se sám spustí. Nyní můžete vkládat vlastní programy, nebo je nahrát z kazety, nebo disku.

Některé příkazy v manuálu nejsou uvedeny, ale BASIC jim přesto rozumí.

TRON Tento příkaz vypisuje čísla řádků, které jsou interpretem právě zpracováván. Tento příkaz je užitečný zejména při hledáníchyb nebo tvoření programu.

TROFF Vypíná příkaz TRON

POKE O tomto příkazu je sice v manuálu zmínka, ale je zamlčeno, že může být použit také s několika parametry např.

```
POKE 53248, 65,66,67    je stejné jako
     POKE 53248, 65
     POKE 53248, 66
     POKE 53248, 67
```

JOY Tento příkaz kontroluje Joystick. V MZ-800 jsou ale definovány nové programy pro Joystick takže tento příkaz by mohl být bez účinku. (Blíže neznámá činost, parametry 0 - 7 vrací log hodnotu)

CLS Příkaz maže obrazovku a přesouvá kursor do levého horního rohu.

**HEX$** Převádí dekadické číslo na hexadecimální, např.: PRINT HEX$ (255) dává FF.

Organizace řádků BASICu v paměti

Programový řádek se do paměti neukladá tak, jak se objevuje na obrazovce nebo jak se zadává, ale v kódech. Každému příkazu se přiřadí hexadecimální kód (číslo), který je dlouhý jeden nebo dva byty. To má tu výhodu, že každý příkaz potřebuje maximálně jen dva byty, většina povelů se dokonce obejde s jedním bytem.
Např.: Příkaz RESTORE má kód 85h místo 7 znaků.
Program v MZ-700 BASICu začíná vždy od adresy 6BCFh.

Struktura programového řádku je násedující:

1. První dva znaky určují počet znaků v program. řádku. Může mýt maximálně 255 bytů. To vychází z podmínek, že druhý byt (High byte) musí být vždy 00H.

2. Třetí a čtvrtý byt tvoří číslo programové řádky BASICu.

3. Náslsedující byty až po koncový byt 00H jsou kódy, ASCII kódy, proměnné nebo řetězce.

4. Koncový byt 00H. Ten ukazuje konec programové řádky.

Je-li počet znaků 00, číslo řádky 00, je to konec celého programu v BASICu.

Příklad uložení programu v paměti:

Abychom ješťe jednou zdůraznili velý postup ukládání řádků do paměti, vylistujeme si krátký program.

```
10 REM BBG-DEMO
20 FORA=1TO255
25 PRINTCHR$(A);:NEXT
30 PRINT"Info BASIC"
40 GOTO10
60 END
```

Vúpis paměti:

adresa	obsah	význam
6BCF	0F	počet Bytů v řádce (low Byte)
6CD0	00	(high Byte)
6CD1	0A	10 - číslo řádky (low Byte)
6CD2	00	(high Byte)
6CD3	97	REM
6CD4	20	SPACE (mezera)
6CD5	42	B
6CD6	42	B
6CD7	47	G
6CD8	2D	-
6CD9	44	D
6CDA	45	E
6CDB	4D	M
6CDC	4F	O
6CDD	00	konec řádky

```
6CDE  15          počet Bytů v řádce (low Byte)
6CDF  00                          (high Byte)
6CE0  14          20 - číslo řádky (liw Byte)
6CE1  00                          (high Byte)
6CE2  8D          FOR
6CE3  41          A
6CE4  F4          =
6CE5  15          tento Byt znamená, že následuje
                        velká konstanta
6CE6  81          konstanta (ve vnitřním kódu)
6CE7  00                skok na adresu 1581h
6CE8  00                -/-
6CE9  00                -/-
6CEA  00                -/-
6CEB  E0          TO
6CEC  15          následuje velká konstanta
6CED  88          konstanta (ve vnitřním kódu)
6CEE  7F                skok na adresu 1500h
6CEF  00                -/-
6CF0  00                -/-
6CF1  00                -/-
6CF2  00          konec řádky
6CF3  0E          počet Bytů v řádce (low Byte)
6CF4  00                          (high Byte)
6CF5  19          25 - číslo řádky (low Byte)
6CF6  00                       (high Byte)
6CF7  8F          PRINT
6CF8  FF          1. Token CHR$
6CF9  A0          2. Token CHR$
6CFA  28          (
6CFB  41          A
6CFC  29          )
6CFD  3B          ;
6CFE  3A          :
6CFF  8E          NEXT
6D00  00          konec řádky
6D01  12          počet Bytů v řádce (low Byte)
6D02  00                          (high Byte)
6D03  1E          30 - číslo řádky (low Byte)
6D04  00                          (high Byte)
6D05  8F          PRINT
6D06  22          "
6D07  49          I
6D08  B0          n
6D09  AA          f
6D0A  B7          o
6D0B  20          SPACE (mezera)
6D0C  42          B
6D0D  A1          a
6D0E  A4          s
6D0F  A6          i
6D10  9F          c
6D11  22          "
6D12  00          konec řádky
6D13  09          počet Bytů v řádce (low Byte)
6D14  00                          (high Byte)
6D15  28          40 - číslo řádky (low Byte)
6D16  00                       (high Byte)
6D17  80          GOTO
6D18  0C
6D19  CF          1. Byt adresa skoku
```

```
6D1A  6B          2. Byt adresa skoku (6BCF)
6D1B  00          konec řádky
6D1C  06          počet Bytů v řádce (low Byte)
6D1D  00                        (high Byte)
6D1E  3C          60 - číslo řádky (low Byte)
6D1F  00                     (high Byte)
6D20  98          END
6D21  00          konec řádků v BASICu
6D22  00          konec číslování řádků
6D23  00          konec číslování
```

Příkazy BASICu, adresy a kódy (TOKEN)

Zde je ještě jednou vypsáno postavení všech povelů BASICu, jejich kódů a odpovídajících adres programů. Některé příkazy jsou sice obsaženy v tabulce kódů, ale vedeou k hlášení Syntax Error. Příkazy jsou v seznamu odpovídajícím způsobem označeny. Tyto příkazy jsou převážně pro disketu, a proto z důvodu šetřením místa nejsou definovány pro kazetovou verzi BASICu, která je zde probírána.

příkaz	TOKEN	adresa	poznámka
GOTO	80	3807	
GOSUB	81	36CD	
RUN	83	1C6E	
RETURN	84	3694	
RESTORE	85	25A8	
RESUME	86	36FB	
LIST	87	4102	
DELETE	89	3456	
RENUM	8A	3471	
AUTO	8B	21D5	
FOR	8D	1CBF	
NEXT	8E	1D6F	
PRINT	8F	1E6A	
INPUT	91	22CA	
IF	93	383F	
DATA	94	3323	viz REM
READ	95	25E3	
DIM	96	5AEA	
REM	97	3323	viz DATA
END	98	21B7	
STOP	99	2071	
CONT	9A	209F	
CLS	9B	38CE	
ON	9D	3792	
LET	9E	1959	
NEW	9F	2248	
POKE	A0	33D7	
OFF	A1	20FE	ERROR
MODE	A2	4DD0	
SKIP	A3	4E2C	
PLOT	A4	3A31	
LINE	A5	4E8C	
RLINE	A6	4EFE	
MOVE	A7	4F05	
RMOVE	A8	4F14	
TRON	A9	227E	

TROFF	AA	2282	
INP	AB	3350	
GET	AD	3389	
PCOLOR	AE	4F1B	
PHOME	AF	4F50	
HSET	B0	4F64	
GPRINT	B1	4F76	
KEY	B2	4321	
AXIS	B3	4FFC	
LOAD	B4	41D1	
SAVE	B5	42A4	
MERGE	B6	41AB	
CONSOLE	B8	39CB	
OUT	BA	3339	
CIRCLE	BB	5050	
TEST	BC	5227	
PAGE	BD	523B	
ERASE	CD	20FE	ERROR
ERROR	C1	2105	
USR	C3	3305	
BYE	C4	13C2	
DEF	C7	251B	
WOPEN	CE	46A3	
CLOSE	CF	4545	
ROPEN	D0	46DF	
KILL	D9	20FE	ERROR
TO	E0		FOR program
STEP	E1		-/-
THEN	E2		IF program
USING	E3		PRINT program
TAB	E6		PRINT -/-
SPC	E7		PRINT -/-
OR	E8	20FE	ERROR
AND	EC	20FE	ERROR
> <	EE		MATEMAT. program
< >	EF		-/-
= <	F0		-/-
< =	F1		-/-
= >	F2		-/-
> =	F3		-/-
=	F4		-/-
>	F5		-/-
<	F6		-/-
+	F7		-/-
-	F8		-/-
/	F8		-/-
*	FC		-/-
EXP(^)	FD		-/-
SET	FE 81	3902	
RESET	FE 82	3935	
COLOR	FE 83	4473	
MUSIC	FE A2	443C	
TEMPO	FE A3	4463	
CURSOR	FE A4	336C	
VERIFY	FE	A5	42D1
CLR	FE	A6	224E
LIMIT	FE	A7	340B
BOOT	FE	AE	3A6A skok na 0000H
INT	FE 80	6277	
ABS	FF 81	6272	
SIN	FF 82	63C6	

COS	FF 83	63B0	
TAN	FF 84	648D	
LN	FF 85	6736	
EXP	FF 86	6615	
SQR	FF 87	62D0	
RND	FF 88	65D0	
PEEK	FF 89	65B5	
ATN	FF 8A	62EE	
SGN	FF 8B	657A	
LOG	FF 8C	672A	
PIA	FF 8E	65A2	
RAD	FF	8F	659D
EOF	FF 95	20FE	
JOY	FF 9E	39A3	
STR$	FF A1	5677	
HEX$	FF A2	55EC	
ASC	FF AB	56A0	
LEN	FF AC	56AC	
VAL	FF AD	56B4	
ERN	FF B3	5562	
ERL	FF B4	556A	
SIZE	FF B5	5548	
LEFT$	FF BA	56C7	
RIGHT$	FF BB	56E4	
MID$	FF BC	5702	
TI$	FF C4	573F	
FN	FF C7	5CF7	

Manipulace s PEEK a POKE

Tato část knihy se zabývá BASIC monitorem MZ-700, který leží v oblasti 0000h až 0FFFh. Jsou zde udány specielní adresy, jejichž změna může sloužit k vytváření efektů.

Tato paměťová místa mohou být měněna pomocí POKE nebo USR. Adresy v paměti jsou udány jednak v dekadickém, jednak v hexadecimální tvar

hexa	dekad	význam
0015	21	Zobrazení řetězce na obrazovce. USR(21,A$) A$ - vypíše od pozice kursoru
0030	48	Zahraje hudební řetězec
003E	62	Vyvolá se pomocí USR. Vytvoří krátký tón. USR (62), USR ($003E)
0044	68	USR (68) zapíná frekvenci tónového generátoru, řiditelná přes 2717 a 2718
0047	71	USR (71) vypíná tónový generátor
004D	77	Plotter zap/vyp. Když je oblast této paměťové buňeky 1 zapíná, 0 vypíná
004E	78	CONSOLE zap/vyp. Když je obsah 1, jsou parametry zadány, když je 0 nejsou
0054	84	Tato adresa obsahuje momentálí sloupec,

ve kterém je kursor.

0055 85 Tato adresa obsahuje momentální řádek, na které se nachází kursor.

0056 86 Zde je první argument příkazu Console.

0057 87 Zde je poslední řádek oblasti přetáčení obrazu pro příkaz Console.

0059 89 Pomocí této adresy může být vytvořen autorepeat. Klávesa potom může zůstat stisknuta a nemusí se opětovně mačkat. To se dosáhne pomocí POKE 89,240. Původní nastavení POKE 89,83

005B 91 Tato adresa obsahuje počáteční sloupec oblasti přetáčení obrazu.

005C 92 Tato adresa obsahuje poslední sloupec oblasti přetáčení obrazu.

005D 93 Zde jsou uschovány informace o barvách. POKE 93,39 by změnilo barvu popředí na černou a barvu pozadí na bílou. Jestliže se k 39 přičte ještě 128, pak následuje výpis ve 2. sadě znaků.
např. POKE 93,39+128:PRINT"BBG BUCH"

005F 95 Tato buňka obsahuje ASCII hodnotu naposledy stisknuté klávesy.

0060 96 Tato adresa obsahuje znak kursoru v zobrazovacím kódu. Základní nastavení je 239 POKE 96,73 by změnilo kursor na ? .

0064 100 AM/PM (dopol./odpol.) přepínání vnitřních hodin. Jestliže je TI$ méně než 12.00 hod. dopol., je tato buňka 0, jinak 1. POKE 100,1 by nastavilo TI$ z 164312 na 044312.

0124 292 Při POKE 292,1 se při každém stisknuti klávesy CR vytvoří krátký tón. POKE 292,4 písknutí vypíná.

0288 648 Vtéto buňce může být měněna rychlost, kterou se pohybuje kursor při trvalém testování klávesnice. Základní nastavení je 10. Větší než 10 je rychlejší, menší než 10 je pomalejší.

04B3 1203 Jestiže se během provádění programu stiskne klávesa BREAK, provádění programu se na krátký čas přeruší. Tomu se může zamezit POKE 1203,201. Základní nastavení je POKE 1203,216.

0655 1621 Přetáčení obrazu nahoru. USR(1621) má stejný efekt, jako stisknutí klávesy SHIFT+cursor nahoru.

067D 1661 Přetáčení obrazu dolů. Stejně jako nahoru.

07F9 2041 Tím se může měnit vzhled Alfa-módového
            kursoru. Změna se ale provede až po
            návratu z grafického módu.

0A32 2610 Obsahuje hodnotu tempa pro muziku. Když
            je tempo 1, je zde 7, když je tempo 6,
            je zde 2 atd.

0A39 2617 Byty 2617 a 2618 řídí tónový generátor.
            Generátor se zap. USR(68) a vyp. USR(71).

0FFC 4092 Zde je kód (File code) souboru,
            naposledy nahraného programu.

0FFD 4093 Zde začíná jméno naposledy nahraného
            programu. Délka může mýt maximálně
            16 znaků.
1010 4112 Na této a následující adrese je zapsaná
            délka naposledy nahraného programu.

RAM monitor

Ještě jednou si zopakujem nejdůležitější standartní programy, které jsou k dispozici v BASIC-monitoru a následně i v RAM monitoru. Tento monitor je značně kompatibilní s MZ-700 ROM monitorem. Tabulka skoků na začátku až po 0066h je s ROM-monitorem identické.

adresa	obsah	mnemonic	význam
0000	C3DA00	JP 00DA	Začátek a do 1819h je 0
0003	C32001	JP 0120	Vstup řádky - nepoužit
0006	C3FE04	JP 04FE	Nová řádka 2
0009	C3FA04	JP 04FA	Nová řád., když kursor není na nové řádce
000C	C33105	JP 0531	Print Space
000F	C30205	JP 0502	Print Tab 0,10,20,30,....
0012	C33305	JP 0533	Print Chr. (ASCII)
0015	C3DA04	JP 04DA	Print zprávy není použit
0018	C3DA04	JP 04DA	Print zpráva siehe 0051h
0018	C3F002	JP 02F0	Vstup klávesy není použit
001E	C3A004	JP 04A0	BREAK
0021	C33B0A	JP 0A3B	Zápis záhlaví
0024	C33F0A	JP 0A3F	Zápis programu
0027	C37A0A	JP 0A7A	Čtení záhlaví
002A	C37E0A	JP 0A7E	Čtení programu
002D	C3940A	JP 0A94	Kontrola
0030	C3E408	JP 08E4	Melody od 'DE' konec 0Dh
0033	C35A0D	JP 0D5A	Nastavení času
0036	0000	NOP NOP	žádný příkaz
0038	C3E40D	JP 0DE4	program přerušení
003B	C3E40D	JP 0DA7	čtení času, Akumulátor obsahuje 0=AM, 1=PM DE obsahuje sekundy.
003E	C3130A	JP 0A13	Beep. Krátký tón

```
0041  C3040A      JP 0A04          Tempo. Regostr A (1-7)
0044  C3B709      JP 09B7          Začátek melodie
0047  C3D209      JP 09D2          Konec melodie
004A  C32001      JP 0120          Vstu řádky (0003)
004D  00                Plot Flag 1=zap, 2=vyp
004E  00                Console Flag 1=zap, 2=vyp
004F  FF                Basicflag FF=Basic,
                        64=Monitor
0050  00                Použit při nahrávání CMT
0051  C3DA04      JP 04DA          Tisk zprávy
0054  00                Kursor sloupec
0055  00                Kursor řádek
0056  00                Console start. řádek
0057  18                Console - počet řádek
0058  C35303      JP 0253          Vstup klávesy, bez opak.
005B  00                Console Start Spalte
005C  27                Console ANzahl der Spalten
005D  71                Barva obrazovky, 71=bílá-
                        modrá
005E  00
005F  00                ASCII kód naposledy
                        stisknuté klávesy
0060  00                Zobrazovací kód kursoru
0061  C36A0E      JP 0E6A          Program Joystick
0064  00                AM/PM Flag. 0=AM, 1=PM
0065  00                Pro Joystick
0066  00                     -/-
```

Dále sou uvedeny, adresy kde začínají jednotlivé standartní programy, které jsou k dispozici v RAM-monitoru MZ-700. Jsou udány vstupní adresy v hexadecimálním tvaru

```
1443  SAVE
1469  LOAD
14A9  VERIFY
14B0  RETURN
15BD  MEMORY SET
1545  GOTO
15F4  FIND
154A  DUMP
165A  TRANSPER
```

Užitecné adresy v S-BASICu.

Následující seznam S-BASIC monitoru a BASICu podává přehled o užitečných adresách a podpogramech a o struktuře a organizaci BASICu. Všechny údaje o adresách jsou v hexadecimálním kódu.

0067 až po 00A6 jsou adresy CTRL (řídící). Ty určují adresy skoků po ?CHR$ (příkazu nebo CTR klávesy společně s jinou klávesou.

00EA Tento standertní program je vyvolán příkazem PEEK pro přístup na VIDEORAM

00F2 Tento standartní příkaz je zapotřebí na VIDEORAM pomocí příkazu POKE.

04A0 Příkaz testuje, zda byla stisknuta klávesa BREAK, nebo SHIFT+BREAK.

04CD Zde se převádí ASCII kód do zobrazovacího kódu (Display kód)

0EFC Tabulka pro přepočítávání z ASCII kódu do zobrazovacího kódu.

0FFC Na této adrese je uložen kód souboru naposledy nahraného programu.

0FFD Od této adresy je jméno naposledy nalezeného programu.

100E Zde je délka naposledy nahraného programu. Na prvním místě je nižší slabika (low Byte) a potom vyšší (high Byte).

1010 Na tomto místě je adresa autostartu. V BASICu je ale bez významu

110F Vyrovnávací paměť (Buffer). Zde je poslední řádka, která byla vložena klávesnicí.

1132 Odtud jsou povely, kterými jsou obsazeny klávesy funkcí. Tabulka sahá až po 13C1H.

1832 Sem skáče BASIC po nahrátí. Tento program také maže paměť až do konce. Na adrese 182EH je vyšší byt. Jestliže se sem vloží např. A0H, paměť se vymaže jen po A000H. Zde je také text hlášení po startu.

1848 Sem skáčou všechny skoky z ROM-monitoru do BASICu.

1869 Z této adresy vychází READY.

1872 Od téty adresy leží standartní program pro vkládání měnění řádků programu BASICu.

18FE Zde začíná prováděcí program RUN.

19B9 Odtud začíná startovací program, který převádí programy MZ-80 K na MZ-700.

20C7 Zde jsou ERROR tabulky a příslušná čísla.

21A7 Tato adresa obsahuje právě aktuální čísla řádky programu v BASICu.

2287 Od této adresy je standartní program TRON.

242F Až po adresu 245BH jsou nesmyslné věci.

2AF5 Odtud až po 2CA8H je první tabulka klíčových slov.

2CA9 Odtud až po 30A8H je nevyužita paměť. Sem se mohou zavést vlastní programy.

30A9 Odtud je druhá tabulka klíčových slov.

3147 Zde jsou adresy jednotlivých klíčových slov. Jsou vyhledávány analogicky k tabulkám.

38D5 Stack Pointer

38D9 RUN přepínač 1=Stop 0=RUN

38DD TRON přepínač 1=TRON 0=TROFF

38E6 Tato a následující adresa obsahuje aktuální číslo řádky.

3D86 Odtud je standartní program přepočtu z binárních do dekadických hodnot v paměti.

4795 Odtud je standartní program, který převádí řádky v BASICu na kódy. V tomto způsobu zápisu (kódech), je program též uložen v paměti.

49CD Působí právě opačně než standartní program na adrese 4795H.

4B72 Zde jsou chybová hlášení ve srozumitelné řeči. Tabulka sahá až po 4D4FH.

528B Zde se zkoumá správnost matematických programů.

6AB3 Konec programu v BASICu.

6AB5 Ukazatel proměnných

6AB7 Ukazatel zásobníku pro BASIC.

6ABD Konec paměti, která je k dispozici pro BASIC.

6ABF Začátek programu v BASICu.

Paměť barev a druhá sada znaků.

Při zobrazování znaků a barev mějte vždy na paměti následující! Paměť znaků leží v oblasti D000H - D3E7H. To odpovídá přesně 1000 znakům a odpovídá to 25 řádkům po 40 znacích. Vtéto oblasti jsou uloženy pouze znaky.
Příslušné informace o barvě leží přesně o 2048 Bytů výše. Leží tedy v oblasti D800H - DBE7H.

Pozn.: BASIC MZ-700 používá celkem 50 řádků ve kterých lze editovat (SHIFT+nahoru,dolů nebo USR).

Příklad: Jak může být vytvořen znak s barvou. Znak se objeví v levém horním rohu.

```
10 POKE 53248, 83:REM Vytvoření znaku
20 POKE 53248+2048, 64:REM Přebarvení ze zelené na černou
```

Druhá sada znaků je aktivována vždy, když je v paměťi barev

více než 128 (80H). Proto musí být k barvě v našem příkladu přičteno 128.

Příklad: Přepnutí celého obsahu obrazovky na druhou sadu znaků a nastavení barvy na zelenou.

```
FOR X=55296 TO 55295:POKE X,64+128:NEXT
```

Ale také příkazy PRINT mohou být zobrazeny přímo ve 2. sadě znaků. K tomu slouží místo v paměti 93 (5DH).

Příklad:

```
POKE 93,PEEK(93)+128:?"700 Příklad"
```

Převod programů z MZ-80 K K tomu slouží místo v paměti 93 (5DH).

Příklad:

```
POKE 93,PEEK(93)+128:?"700 Příklad"
```

Převod programů z MZ-80 K

S-BASIC má standartní program, který převádí programy z MZ-80 K do MZ-700. Tento program umí překládat pouze kódy.
Především příkazy POKE, PEEK a USR by měli být před zpoštěním překontrolovány ještě jednou. Všechny POKE mezi 53248 až 54217 sahají do oblasti Videa a nemusí se měnit.

Ostatní POKE většinou přímo ovlivňují BASIC a proto by měly být používány opatrně. Následuje výpis POKE, které by měli být změněny v každém případě. Adresy 4465 a 4466 řídí v MZ-80 K kursor(4465 je sloupec, 4466 je řádka). To může být změněno tím, že se oba argumenty napíší za příkaz CURSOR. 17828 je u MZ-80 K adresa pro trvalé testování kláves. Nová adresa je nyní 89.

Adresy 4513 a 4514 jsou u MZ-80 K určeny pro řízení tónového generátoru. Tyto adresy jsou u MZ-800 (v módu MZ-700) 2617 a 2618. POKE 4464, 1 přeřazuje u MZ-80 K klávesnici na malá písmena. Též může být zadáno PRINT CHR$(5). Přeřazení zpět na velká písmena bylo u MZ-80 K pomocí POKE 4464, 0. Tento příkaz může být proveden pomocí PRINT CHR$(6).

To jsou v podstatě všechny důležité POKE, které se musí změnit. U všech ostatních POKE se musí postupovat s nejvyšší opatrností, protože by mohlo dojít ke smazání BASICu i celého programu. Pro jistotu by měly být vschny neznámé POKE nejprve opatřeny REM, dokud Vám nebude jasný jejich význam.

Adresy jako 10167,10680,10682 a 7125 můžete v každém případě bez náhrady vymazat.

Logické operace AND / OR

V mnoha časopisech a výpisech programů se vyskytují příkazy AND a OR. S-BASIC těmito příkazy ale nedisponuje. To by však nemělo být problémem, protože tyto příkazy se dávají snadno

nahradit. (Pouze v podmíce, ne v logických operacích)

znak " * " se použije pro AND
Znak " + " se použije pro OR

Příklad:

1) místo IF A=1 AND B=2 THEN
lze psát IF (A=1) * (B=2) THEN

2) místo IF A= 1 OR B=2 THEN
lze psát IF (A=1) + (B=2) TFEN

Je důležité sledovat, aby všechny takto použuté argumenty byly v závorkách

Vyloučení činosti klávesy BREAK a SHIFT+BREAK

Zde bude uvedeno, jak se dá program zajistit proti přerušení nebo zastavení klávesou SHIFT+BREAK.
Stisknutí samotné klávesy BREAK nepřeruší běžící program, ale na krátkou dobu ho zastavit.

```
POKE 1203, 201          zastavení zakázané
POKE 1203, 216          zastavení povoleno
```

Při zabránění přerušení pomocí SHIFT+BREAK se musí změnit více paměťových míst.

```
POKE 6452, 54,25  přerušení zakázané
POKE 6452, 113,32 přerušení povoleno
```

Nyní může být program přerušen příkazem INPUT. Tomu zabráníme pomocí těchto POKE

```
POKE 8987, 29,35  přerušení zakázané
POKE 9056, 98,35
POKE 8987, 105,32 přerušení dovoleno
POKE 9056, 105,32
```

Zména barev bez vymázání obrazovky

```
COLOR,,6,1:USR($072D)
```

Tímto příkazem se provádí změna barvy. Je ale třeba dát pozor, protože se kursor přesune do levého horního rohu

Odvozené funkce

MZ-700 BASIC má množství matematických funkcí, ale mnoho funkcí nemá v sadě příkazů obsaženo. Tyto funkce mohou být velmi užitečné především v matematických programech.

Tyto funkce mohou být odvozeny z funkcí, které jsou k dispozici a dají se též psát v MZ-800 BASICu. Proto zde uvádíme sestavu dalších důležitých a uzitečných funkcí.

```
Secans              SEC(X)=1/COS(X)
Cosecans       CSC(X)=1/SIN(X)
Cotangens      COT(X)=1/TAN(X)
Arcussinus     ARCSIN(X)=ATN(X/SQR(-X*X+1))
Arcuscosinus        ARCCOS(X)=ATN(X/SQR(-X*X+1))*PI/2
Arcussecans    ARCSEK(X)=ATN(SQR(X*X-1))+(SGN(X)-1)*PI/2
Arcuscosecans       ARCCSC(X)=ATN(1/SQR(X*X-1))+(SGN(X)-1)*PI/2
Arcuscotangens      ARCCOT(X)=-ATN(X)*PI/2
Hyperbelsinus       SINH(X)=(EXP(X)-EXP(-X))/2
Hyperbelcosinus     COSH(X)=(EXP(X)+EXP(-X))/2
Hyperbeltangens     TANH(X)=-EXP(-X)/(EXP(-X))*2+1
Hyperbelsecasns     SECH(X)=2/(EXP(X)+EXP(-X))
Hyperbelcosecans   CSCH(X)=2/(EXP(X)-EXP(-X))
Hyperbelcotangens  COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
Areasinus      ARSINH(X)=LN(X)+SQR(X*X+1))
Areacosinus    ARCOSH(H)=LN(X+SQR(X*X-1))
Areatangens    ARTANH(X)=LN((1+X)/(1-X))/2
Areasecans     ARSECH(X)=LN((SQR(-X*X+1)+1)/X)
Areacosecans        ARCSCH(X)=LN((SGN(X)*SQR(X*X+1)+1)/X)
Areacotangans       ARCOTH(X)=LN((X+1)/(X-1))/2
```

Pořízení bespečnostní kopie S-BASICu

Doporučuje se pořídit si bespečnostní kopii S-BASICu a tuto také potom používat k denní potřebě, protože originální kazeta by mohla být náhodou vymazána nebo přehrána. Dejte ale pozor na to, že je dovoleno udělat pouze jednu kopii a tu jen pro osobní použití

Pro vytvoření této kopie postupujte takto:

- zapnout počítač
- stisknout klávesu M, tím se dostanete do monitoru
- vložit J00AD a stisknout CR
- MC000 CR
- CD CR
- 27 CR
- 00 CR
- CD CR
- 2A CR
- 00 CR
- C3 CR
- 08 CR
- 11 CR
- stisknout SHIFT+BREAK
- vložit originál kazetu a přetočit na začátek
- vložit JC000 a stisknout CR
- stisknout PLAY na magnetofonu
- program je nahraný po objevení "HIT ANY KEY"
- vyndát originál kazetu
- vložit prázdnou kazetu

- stisnout libovolnou klávesu
- stiknout klávesu RECORD PLAY na magnetofonu
- čekat asi 7 minut
- vyndat nahranou kazetu s kopii BASICu

Rozšíření sady příkazů S-BASICu

Zde budou ukázána některá rozšíření pro kazetovou verzi S-BASICu. Všechny následující rozšíření nepotřebují žádné další místo v paměti, protože používají prázdnou paměť uvnitř BASICu
Pro vložení standartních programů a rozšíření nejprve nahrajte S-BASIC z kazety do počítače a potom vložte všechny řádky, před kterými je uvedeno číslo řadky. Texty mezi řádky programu jsou pouze pro vysvětlení a slouží jako komentář.

1) READY s tónem

Následující standartní program vytvoří při každém READY krátký tón. Toto umožňuje, aby po LOAD nebo jiných dlouhých procedurách byl jejich konec vnímán i akusticky.

```
100 REM READY TON
110 POKE 6256,129,48:POKE 12417,175,205,62,0,205,6,0,201
```

2) Hudba při ERROR

Při každé chybě se pomocí tohoto programu vytvoří krátký hluboký tón. Tím se chyba neobjeví jen na obrazovce, ale je provázen hlubokým tónem.

```
120 REM Hudba při ERROR
130 POKE 8561,137,48
140 POKE 12425,205,81,0,229,17,149,48,205,48,0,225,201,
    45,65,13
```

3) Zastavení LIST pomocí SPACE

Tento program umožňuje při listování programu listování zastavit klávesou SPACE, aby jste mohli např. dělat poznámky Při každém dalším stlačením SPACE se listuje řádeka po řádkce. Při stisknutí libovolné jiné klávesy probíhá listování dále.

```
150 REM Zastavení LIST
160 POKE 16786,77,36,204,83,2,0:POKE 9293,205,27,0,
    254,32,201
```

4) Zlepšený příkaz TRON

Při tomto zlepšení příkazu TRON se bude číslo právě zpracovávané řádky zobrazeno v levém horním rohu obrazovky, místo na aktuálním pozici kursoru. Toto zlepšuje přehlednost. Příkaz TROFF ruší příkaz TRON.

```
170 REM Nový TRON
```

```
180 POKE 8841,42,230,56,205,167,33,33,0,208,6,5,26,
    205,205,4
190 POKE 8856,205,242,0,35,19,16,245,225,241,201
```

5) Zobrazení ROPEN a WOPEN

U přdcházejících modelů MZ-700 bylo při ROPEN nebo WOPEN definována výzva pro PLAY nebo RECORD PLAY. V S-BASICu pracuje tato výzva pouze v přímém způsobu a na v programu.

Dá se to však snadno realizovat i v programu. K tomu slouží následující program.

```
200 REM PLAY/RECORD PLAY při ROPEN/WOPEN
210 POKE 18442,205:POKE 18252,175
```

S tímto rozšířením ale stále ještě není zobrazeno, zda byla data nalezena a nahrána. Následující program vypíše jméno souboru dat před jejich nahráním na obrazovku.

```
220 REM ROPEN se zobrazením jména
230 POKE 18167,205,153,48
240 POKE 12441,17,11,67,33,252,15,205,200,22,58,252,15,201
```

6) Speciální LIST

Tato pozměnění standartního programu LIST způsobí, že po každé řádce programu v BASICu je vynachaná řádka. To usnadňuje přehled především u delších programů. Toto rozšíření platí pro obrazovku i pro tiskárnu.

```
250 REM Speciální LIST
260 POKE 12410,205,249,23,205,249,23,201
270 REM Zapnuto:POKE 16781,122,48
280 REM Vypnuto:POKE 16781,249,23
```

7) Ochrana programů

Cránění programů proti LIST nebo SAVE může být provedeno, jestliže byl program spuštěn pomocí RUN. Proto není možné program 100% chránit.

Přesto vám zde ukážeme,jak může být program chráněn proti různým povelům.

ochrana	proti	původní stav
POKE 16642,201	LIST	POKE 16642,175
POKE 17060,201	SAVE	POKE 17060,205
POKE 26037,201	PEEK	POKE 26037,254
POKE 5058,201	BYE	POKE 5058,229

Výše uvedené řádky vložíme do počítače. Nyní máme basický program, který by po odstartování způsobyl změny. Uživatel by si mě jenom zapamatovat adresy příkazů POKE, které jsou na řádcích 270 a 280, aby mohl zapínat a vypínat speciální LIST. Toto rozšíření by se mě-

lo po vložení do počítače nahrát. Podle potřeby potom může být vždy přehrána. Jestliže byl program jednou odsratován, může být smazán příkazem NEW. Změny zůstanou zachovány až do vypnutí počítače.

KURZ STROJOVÉHO JAZYKA.

Pro uživatele každého počítače je důležité a užitečné, když se naučí systémový vnitřní jazyk. Ale není to tak jednoduché, aby si mohl člověk jednoduše sednout a začít programovat.

Nejprve srovnejme znaky BASICu a strojového jazyka. V BASICu je obsluha velmi pohodlná, protože např. programy pro počítání jako *,+,-,/ jsou již definovány. U komfortnějších BASICů, jako je např. MZ-800 je přirozeně definováno mnohem více funkcí jako např. SIN,COS,TAN,EXP atd. Naproti tomu si ve strojovém jazyce musíme i obyčejné násobení dobře rozmyslet; opravdu těžké jsou např. programy pro počítání v reálném čase.

Hledisko rychlosti je jeden z rozhoduj1icích argumentů pro programování v assembleru. A tak je někdy zcela nemožné napsat některé programy v BASICu, protože rychlost je úplně nedostačující. Totéž platí pro potřebu místa v paměti, která je u BASICu velmi velká. Proti tomu se ve strojovém jazyce stejný problém vyřešit na jedné desetině místa,zvláště, když se BASIC nemusí nahrávat. Toto uspoří při pozdějším provádění programu mnoho času.

Nyní ale začneme odhrnovat roušku tajemství strojového jazyka.

1. Programovací medium.

Jako se v BASICu nedá programovat bez BASICu, nedá se programovat ve strojovém jazyce bez strojového jazyka. Zde je několik možností:

A. programování v assembleru

B. programování ve strojovém jazyce

C. programování v Monitoru

Chceme zde jen ukázat rozdíly mezi assemblerem a strojovým jazykem (zkráceně ML = machine language), ale dále se budeme zabývat programováním v ML (viz. též khiha: "ASSEMBLERPROGRAMMIERUBG AUF DEM MZ-700/800").

Proto vám doporučujeme, abyste si pro tento ML-kurs obstarali dobrý strojový jazyk, který by měl v každém případě ovládat následující povely:

a. Disassembler

b. zapsání hex. čísla do paměti

c. listování pamětí

d. ukládání ASCII - řetězců do paměti (textwrite)

e. ASCII - listing

Toto vám umožní např. strojový jazyk 8002
od firmy BBG Software.

2. Dvojkový systém.

Pro znalce strojového jazyka mohou být následující kapitoly příliš lehké a může je proto klidně přeskočit. Jestliže někomu není dvojkový neboli binární systém známý, měl by zde dávat dobrý pozor, protože toto je klíč k porozumění ML.

Jak již jméno napovídá, jedná se zde o systém čísel, který má něco společného s dvojkou (bi = řecky dvě). Jak možná víte, v elektronice a tím i u počítačů existují jen dva stavy, a to 0 a 1 (vypnuto a zapnuto). Tento dvojkový sytém používá jen tyto dva stavy. Princip je podobný naší desítkové soustavě. Pro ty, kteří náš desítkový systém ještě dostatečně neznají, máme jeden příklad.

Vezměme si číslo 15326. Musíme zde jednotlivé číslice rozdělit podle jednotek, desítek, stovek, tisíců atd. To znamená:

```
15326  =  1 deset tisíc ..... 1x10

                          3
           5 tisíc ............ 5x10

                          2
           3 sta .............. 3x10

                          1
           2 desítky .......... 2x10

                          0
           6 jednotek ......... 6x10  (pro všechny nematema-
                                       tiky: deset na nultou
                                       je jedna!)

           4        3        2        1        0
15326 = 1x10  +  5x10  +  3x10  +  2x10  +  6x10

           n        n-1        n-2
číslo = Xx10  +  Yx10    +  Zx10    +  Ax10 + Bx1
```

Právě tak se chová dvojkový systém. Desítkový systém má za základ 10, dvojkový systém 2. U desítkového systému je největší číslice 9, u dvojkového systému tomu odpovídající číslo 1. Jsou zde tedy pouze číslice 0 a 1.

```
          n        n-1        n-2
dv.č. = Xx2   +  Yx2     +  Zx2     +  Ax2x1 + Bx1
```

Jako příklad přepočítáme decimální číslo 39 na binární. Na to musíme znát mocniny dvojky. Tak jako 10,100,1000,10000 atd. jsou mocniny deseti, tak jsou 2,4,8,16,32,64,128 atd. mocniny dvojky.

```
            x
číslo = 2 = mocnina 2

             x
číslo = 10 = mocnina 10
```

Číslo 39 se skládá z mocnin 2, následujícím způsobem:

$$39 = 32 + 4 + 2 + 1$$

$$39 = 2^5 + 2^2 + 2^1 + 2^0$$

To se pokusíme zvládnout nejlépe pomocí následující tabulky:

128	64	32	16	8	4	2	1	
0	0	1	0	0	1	1	1	= 39

Decimální číslo 39 se v binární soustavě rovná číslu 00100111. Jako jiný příklad decimální číslo 111. Sestavíme opět tabulku tak, že zapíšeme pod příslušnou mocninu 2 jedničku nebo nulu, aby vzniklo správné číslo.

128	64	32	16	8	4	2	1	
0	1	1	0	1	1	1	1	= 111

Decimální číslo 111 je tedy v binární soustavě 01101111. Předkládáme vám ale ještě jednu dobrou metodu, jak určit, kdy se musí pod mocninu 2 napsat 0 nebo 1.

Ukážeme si to opět na čísle 111. Nejdříve hledáme mocninu 2, která je nejbližší nižší k desítkovému číslu. 128 je větší než 111, ale zároveň je číslo 111 větší než číslo 64. Napíšeme tedy pod 64 číslo 1, a potom 64 odečteme od 111 (111 - 64 = 47). Nyní zacházíme s číslem 47 právě tak, jako s číslem 111, takže 32 by bylo nejbližší nižší mocnina 2. Zapíšeme tedy pod 32 číslo 1 a odčítáme (47 - 32 = 15). Příští nejbližší nižší mocnina 2 je 8, takže pod ní zapíšeme 1, odečtením dostaneme 7, což se dá z hlavy zpočítat jako dvojkové číslo 111 (1 + 2 + 4 = 7). Tam, kam nebyla zapsána žádná 1, je logicky zapsána 0. Tím dostaneme dvojkové číslo 01101111. Čím větší je decimální číslo, tím větší musí být mocniny 2.

Jestliže máme dvojkové číslo a chceme ho přepočítat na desítkové, sečteme jednoduše mocniny 2. To může sloužit jako kontrola např. 01101111 = 64 + 32 + 8 + 4 + 2 + 1 = 111.

3. Hexadecimální systém.

Hexadecimální systém se může z počátku zdát pro začátečníka zmatený, ale přesto je to velice logický systém a základ pro naučení strojového jazyka. Ptáte se proč se za základ pro počítač nevzal desítkový nebo dvojkový systém. Odpověď je zřejmá. Desítkový systém není pro počítač zpracovatelný, a přepočítávaní desítkového systému do dvojkového spotřebuje mnoho času. Přepočítávání hexadecimálního systému do dvojkového je velice rychlé, protože 16 je přece mocina 2. Tím se ještě budeme zabývat. Jak již jméno napovídá, je číslo 16 základ tohoto číselného systému. Obecná přepočtová rovnice tedy tentokrát zní:

```
                n        n-1        n-2
číslo = Zx16  +  Yx16    +  Zx16    +  Ax16
```

Mocniny 16 jsou:

```
                        0   1   2   3
 1, 16, 256, 4096 = 16 ,16 ,16 ,16
```

Potrebujeme jen tyto mocniny, protože v paměti můžeme adresovat jen po 63 535, takže vyšší mocninu nepotřebujeme.

Pravděpodobně vás již napadlo, jak jak může být hexadecimální systém vybaven číslicemi, když od 10 do 15 žádné číslice nejsou. Zde bylo rozhodnuto vzít za základ písmena A - F, takže A=10, B=11, C=12, D=13, E=15. Výpočet hexadecimálního čísla se provádí podle stejného schématu jako při výpočtu dvojkového čísla. Jenom se zde samozřejmě hledá nejbližší nižší mocnina 16. Podívejme se na to na příkladu. Decimální číslo 83 má být přepočítáno na hexadecimální.

Nejdříve tedy napíšeme tabulku:

```
-----------------------------------------
    4096      256      16      1

      0        0        5      3  =  83
-----------------------------------------
```

Postupujme analogicky k dvojkovému systému. Nejprve hledáme nejbližší nižší mocninu 16, čili číslo 16. Toto je v 83 obsaženo 5 krát, napíšeme tedy pod 16 číslo 5, 5x16=80, dostaneme tedy zbytek 3. Dohromady to tedy dává hexadecimální číslo 83. Zkusme to na větším decimálním čísle, třeba 15311. Nejlépe bude, vemete-li si na pomoc kapesní kalkulačku. První mocnina je 4096. Toto číslo je v 15311 obsaženo 3x, 3x4096=12288. Nyní dostaneme jeho zbytek (15311-12288) 3023, který musíme porovnat s další menší mocninou 16. Do 3023 se 256 vejde 11x. Nyní musíme opět spočítat zbytek (3023-(11x256)=207). 207 děleno 16 je 12 a zbytek je 15.

```
-------------------------------------------------
15311  =   3x4096  +  11x256  +  12x16  +  15

             3          2          1         0
15311  =   3x16    +  11x16   +  12x16  +  15x16
-------------------------------------------------
```

Nyní musíme čísla přeformovat na hexadecimální čísla. Jak jsme se již zmínili, představují čísla přes 9 samostatné hexadecimální cifry(A=10,B=11 atd.). Nyní musíme vypočtené hodnoty čili 3,11,12,15 přeformovat na hexadecimální číslo, což dává 3BCF. Nejvyšší hexadecimální hodnota by byla FFFF. Od teď budou hexadecimální čísla na konci oznacena písmenem H. To nám také ulehčí identifikaci, zda se jedná o čísla mající decimální nebo hexadecimální hodnotu. FFFFH je tedy 15x4096 + 15x256 + 15x16 + 15=65535. Volné místo v paměti je 65535 bytů, což je 64 Kbytů.

3.1. Přepočítání z dvojkové do hexadecimální soustavy.

Při programování ve strojovém jazyce je později důležité převádět dvojkové hodnoty na hexadecimální a naopak. K tomu ně-

kolik dobrých rad: budeme se zde zabývat jen 8-bitovými hodnotami, např. dvojkovou hodnotou 01111111. Potom se 8-bitové číslo rozdělí na dvě poloviny, a to přesně uprostřed a vznikne 0111 a 1111. To jsou decimální čísla 7, resp. 15, což jsou hexadecimální hodnoty 7 a F. Hexadecimální číslo tedy je 7FH.

4. Bit a byte.

S těmito pojmy jsme se již setkali občas v předcházejících kapitolách. Kdo ještě neví oč se jedná, tomu budiž řečeno následující. Bit je nejmenší jednotka, kterou počítač zná. Byte se skládá z 8 bitů. Bity si mohou pamatovat jen stavy 0 a 1, jeden byte má proti tomu 2 na osmou, tj. 256 možností zapamatování. Byte je normálně udáván jako hexadecimální číslo. Celá paměť MZ-800 má podletoho 8x65535 = 524280 bitů.

5.Procesor Z-80.

Mikroprocesor Z-80 je srdcem MZ-800. Každý počítač má mikroprocesor, ATARI, APPLE, COMMODORE mají především procesory 6502. Jiné známé procesory jsou 6809 a nové 32-bitové procesory 68000, 80186 a mnoho dalších. Z-80 procesor má asi 400 oficiálních povelů a ještě mnohem více neznámých povelů, kterými se budeme zabývat později. Povely jsou 1,2,3,4 bytové.

5.1. Registry.

Registry jsou zvláštností procesoru. S registry se dá počítat, vyjímat hodnoty z paměti a přenášet hodnoty do paměti. Mnoho logických operací, sčítání, odčítání a další se provádí pomocí regustrů. Přitom existují jednobytové a dvoubytové registry. Dvoubytové neboli také 16 bitové registry jsou zvláště praktické pro matematické operace. To je výhoda, kterou má procesor Z-80 proti procesoru 6502.

Sada registrů:

8-bitové registry	16-bitové registry
A (akumulátor)	PC (programmcounter-čítač programů)
B	BC (skládá se dohromady
C	z B a C)
D	DE (skládá se
E	z D a E)
H	HL (skládá se
L	z H a L)
F (příznak přepínače)	SP (ukazatel zásobníku)
LX	IX (skládá se z
HX	registrů LX a HX)
LY	IY (skládá se z
HY	registrů LY a HY)

Dvojitý registr z A,B,C,D,E,F,H,L
R - obnovovací registr

I - přerušovací registr

5.1.1. Akumulátor (A-registr).

Jak již bylo naznačeno, má akumulátor mezi registry zvláštní postavení. Akumulátor je 8-bitový registr, ve kterém se provádí logické operace jako AND,OR, ale též sčítání a odčítání. Jako jediný 8-bitový registr je schopen vyvolat hodnoty z paměti a do paměti je ukládat. Logické operace může provádět s pomocí ostatních registrů nebo s pamětí. To dělá z akumulátoru důležitou součástí procesoru.

5.1.2. Další 8-bitové registry.

Kromě akumulátoru má Z-80 ještě další registry, jako B,C,D,E, H,L. Obecně působí tyto registry jako akumulátor, ale jejich zásoba povelů není tak velká. A tak mohou pracovat jen ve spojení s akumulátorem. Také není možné adresování paměti z těchto registrů.

5.1.3. Příznaky přepínače (flags).

Příznaky přepínače jsou představovány F-registrem, takže je dohromady 8 příznaků přepínače, z kterých je používáno pouze 6.

Zde je grafický přehled:

7	6	5	4	3	2	1	0
S	Z	-	H	-	P	N	C

Jména flagů:

S - znaménkový flag Z - nulový flag
H - poloviční indikátor přenosu
P - paritní flag
N - odčítací flag
C - indikátor přenosu

Flagy se používají pro označení výsledku matematicko-aritmetické operace. Udává se, zda operace dává výsledek jako 0, nebo jestli existuje přeplnění atd.

5.1.3.1. Indikátor přenosu.

Indikátor přenosu nám ukazuje přeplnění bitu 7 (u 16-bitových přeplnění bitu 15). Indikátor přenosu si můžeme představit jako aktivní devátý bit. Chceme provést operaci, při které vznikne přeplnění 8-bitů, takže při ní překročíme výpočetní oblast. Např. A obsahuje 83H a B obsahuje 90H. Součet dává potom 113H, což by bylo ale 9-bitové číslo. V akumulátoru je potom 13H a v indiká-

toru přenosu 1.

5.1.3.2. Odečítací flag.

Tento příznak přepínče není programátorem požíván, používá se při speciálních desítkově-aritmetických povelech jako vnitřní znaménkový bit. Pro programování je relativně nedůležitý.

5.1.3.3. Poloviční indikátor přenosu.

Poloviční indikátor přenosu funguje podobně jako indikátor přenosu, ale ukazuje se zde přenosu ze 3 bitů na 4. Pro programátora zpracovávajícího program ve strojovém jazyce zde není žádná možnost tento indikátor kontrolovat, a proto je praktický význam minimální.

5.1.3.4. Paritní flag.

Paritní flag má 2 hlavní funkce.

a) Určení rovnosti nějakého jevu. Rovnost znamená totéž jako křížový součet. To znamená, že se spočítají jednotky uvnitř jednoho bytu. Jestliže se suma jedotek nerovná, potom je v P-flagu jednotka (1), jinak je obnoven počáteční stav.

b) Druhá hlavní funkce je funkce přeplnění. Udává se, zda při sčítání nebo odčítání nebyl omylem změněn znaménkový bit tím, že výsledek omylem přeběhl do znaménkového bitu.

5.1.3.5. Zero-flag.

Jak jsme se již zmínili ve výše uvedeném příkladu, slouží nulový flag k zobrazení toho, zda některá operace má výsledek 0 nebo ne. Při výsledku 0 má Z-flag nastaveno 1, jinak 0.

5.1.3.6. Znaménkový flag.

Tento flag slouží k zobrazení znaménka, přičemž se osmice bitů rozdělí na 7 datových bitů a jeden bit znaménkový. To má za následek to, že výpočetní oblast je od -127 do +128.

5.1.4. R - registr.

Obnovovací registr slouží interně k obnovení paměti. Přitom počítadlo odpočítává z 7FH na 0, a potom je paměti vydán obnovovací impuls. Z tohoto registru se dá prakticky jen číst, což se však báječně hodí pro generátor náhodných čísel. Tím, že se zamezí impulsu obnovení (do akumulátoru se dá xx a do R se dá A) se dá vymazat celá paměť a zůstane pouze destruktivní program.

5.1.5. I - registr.

Přerušovací registr ukazuje stav uvnitř úrovně operačního systému, takže pro programátora nemá velký praktický význam.

5.1.6. PC - registr.

PC-registr, nebo též čítač programů, ukazuje, na kterém místě právě pracuje základní jednotka Z-80. K čítači programů není možný žádný přímý přístup.

5.1.7. SP - registr.

SP-registr, nebo též ukazatel zásobníku, je registr, ve kterém je 16-bitová adresa místa v paměti. Na tomto místě se nachází zásobník, což je pomocí ukazatele zásobníku definovaná pracovní oblast, do které se ukládají, a ze které se vybírají data. Pro práci s tímto zásobníkem jsou speciální povely, kterými se budeme zabývat později.

5.1.8. HL - registr.

HL-registr je 16-bitový registr, který se skládá ze dvou 8--bitových registrů H a L. Přitom nižší byt (Low Byte) tvoří L, zatím co vyšší byt (High Byte) tvoří H.

Pro osvěžení:
Low znamenalo nižší, High vyšší hodnotu. Zde začínáme mluvit o něčem, na co se v 16-bitovém zápisu musí dávat pozor. V paměti je nejprve uložen Low, a potom High registr. Nechť tedy obsahuje H-registr 06H, L-registr F4H. Potom bude registr HL obsahovat jako clek O6F4H. Jestliže nyní uložíme obsah registru HL do paměti, bude v paměti stát F406, čili přesně obráceně. HL-registr je tedy 16-bitový registr pro logické operace, a je to vedle A nejpoužívanější registr v sadě registrů.

5.1.9. BC a DE - registry.

Tyto dva registry jsou velmi podobné registru HL. Oba se skládají z 8-bitových registrů B a C, resp. D a E, ale povelový komfort je ve srovnání s HL velmi omezený. Musí se dát ale velký pozor na to, že když se při programování použije registr B, změní se i hodnota registru BC, takže se nejprve musí zachránit hodnota registru BC. To je chyba, která se při programování ve strojovém jazyce často stává. To samozřejmě platí pro všechny 16-bitové registry, takže i pro HL,DF,IX,IY.

5.1.10. IX a IY registry.

Registry IX a IY jsou velcí bratři registru HL. Zde se musí dát pozor, protože zde bylo trochu nešťastně zvoleno pojmenování registrů. Za prvé nemají registry IY a IX nic společného s registrem I. Část I z IX registru je HX registr, část X je LX re-

gistr. Právě tak je to u IY registru, kde I je HY, část Y je LY registr. Pozor ale na to, že LY,HY,HX,LX jsou 8-bitové registry, a ne jak se může na první pohled zdát 16-bitové registry. Programování těchto registrů není popsáno v žádné knize o Z-80, jsou to neznámé povely, kterými se budeme zabývat později. IX a IY se též nazývají indexové registry. Sada povelů indexových registrů je velice podobná povelům pro HL registr, ale pro každý povel pro indexový registr je potřeba minimálně tři, většinou dokonce čtyři byty, což značně zvětšuje délku programu. O způsobu adresování budeme mluvit v jiné kapitole, protože je to opravdu těžká část programování ve strojovém jazyce.

5.1.11. Zdvojené registry.

Zvláštností v sadě registrů jsou tzv. zdvojené registry, A', B',C',D',E',H',L',F'. Jsou to duplikáty registrů A,B,C,D,E,H,L,F, které byly vyměněny s normálními registry pomocí povelů pro vyměňování. Procesor dále počítá s hodnotami zdvojených registrů. Jestliže se povely vyměňování požijí dvakrát, logicky se vytvoří původní stav.

5.2. Mnemonic a Opcode.

Další důležitou kapitolou je sovislost mezi symbolem a významem povelů ve strojovém jzyce. Právě tak jako v BASICu má i zde každý povel konkrétní význam. Tak např. povel DIM v BASICu má význam zřízení pole. Ve strojovém jazyce má také každý povel svůj hexa. kód, čili hexa. číslo označuje povel. Tak povel NOP (No OPeration) je označen číslem 00H. Takže když je zpracovávána hodnota 00H, je interpretována jako NOP, a potom zpracovává ihned další povel. Jestliže přijde jako povel 09H, budou sečteny registry HL a BC. Přehled všech těchto povelů Z-80 najdete v dodatku této knihy. Mnemoniccode 09H znamená ADD HL,BC, čímž se dostáváme k mnemotechnickému zůsobu zápisu. Mnemoniccode povelu je pro programátora srozumitelná interpretace hexa. povelu. Tak je pro hexa. povel 09H definován mnemoniccode ADD HL,BL. Pro programátora není možné rozpoznat smysl programu jen na základě čísel, naproti tomu mnemoniccode je pro něj snadno srozumitelný. V mnemoniccode se dá programovat, ovšem pouze v assembleru. V assembleru programujete např. INC A a assembler to převede na 3CH. Disassembler dělá pravý opak, takže převede 3CH na INC A. Pro nás je disassembler plně dostačující, ale každý dobrý stroj. jazyk by měl mít disassembler. Program se vloží v hexa. číslech a potom se může vylistovat pomocí disassembleru. Programování v assembleru je rozsáhlejší, ale má také více možností.

5.3. Struktura paměti

Napsaný program je v paměti na určité adrese. Paměť začíná na adrese 0000H a končí na FFFFH. Normální oblast programování MZ-800 začíná na 1200H. Odtud budeme též psát všechny naše programy. Ke každé adrese v paměti patří také obsah, což může být datový nebo programový byte. Abychom se mohli tímto zabývat, potřebujeme disassembler, který přeloží ze strojového jazyka do jazyka symbolických instrukcí oblast od 1200H. Když se zapne počítač,

nejprve se většinou vymaže paměť. Je tomu tak i u MZ-800. Nyní tedy nalézáme v paměti pouze 00H a FFH. Když uvedený obsah přeložíme, objeví se následující:

```
    ADRESA          OPCODE          MNEMONICCODE

     1200        00                   NOP
     1201            00               NOP
     1202            FF                  RST 38
     1203            FF                  RST 38
```

5.4. Druhy adresování.

V možnostech asdresování není Z-80 vybaven tak bohatě jako jeho protějšek - 6502. Přesto zde musíme říci několik věcí ne právě zanedbatelného významu. U Z-80 má mnemoiccode tři díly:

a) druh povelu
b) cílová adresa / registr
c) strojová adresa / registr

To znázorníme na příkladu:

Příklad: LD A,B

Toto je ukládací povel, při kterém bude hodnota, která je v registru B uložena do registru A. V BASICu by tento povel zněl: A = B. Na tomto se dá vysvětlit struktura mnemonického kódu. Druh povelu je ukládací povel (LD). Cílový registr je zde A-registr, zdrojový registr je B-registr. To je registrové adresování. Zabývejme se nyní povelem LD A,(HL), což je opět ukládací povel. Akumulátor zde opět představuje cílový registr. Jen zdrojový registr zde potřebuje další vysvětlení. Vidíme, že zdrojový výraz je zde (HL). Závorka nám ukazuje, že zde pracujeme s pamětí. Výraz v závorce je HL-registr. HL je 16-bitový registr, který zde ukazuje na místo v paměti. Jestliže by HL měl hodnotu 0000H, pak by byla zpracovávána hodnota, která je v paměti na místě 0000H (u MZ-800 je na adrese 0000H C3H). Tato hodnota je potom uložena do akumulátoru. A ještě další příklad:

```
LD (1000H),A
```

Jedná se opět o ukládací příkaz. Tentokrát nám závorka ukazuje, že cílová adresa je v paměti, zde tedy adresa 1000H. Hodnota, která má být uložena za adresu 1000H, je tedy v akumulátoru (zdrojovém registru).

Rozdílné jsou 16-bitové ukládací povely. Vezměme příklad uvedený výše, ale tentokrát s HL-registrem.

```
LD (1000H),HL
```

Jak si můžete domyslet, bude hodnota v HL-registru uložena na adrese 1000H. Nyní je ale tato hodnota 16-bitová, takže obsadí kromě adresy 1000H ještě adresu 1001H.

Jak jsme se již zmínily, při 16-bitových znacích se v paměti ukládá nejprve nižší a potom vyšší byte. Nechť HL obsahuje před provedením příkazu LD (1000H),HL hodnotu 12F5H. Po provedení je na adrese 1000H hodnota F5H, na adrese 1001H hodnota 12.

Jako poslední bychom se chtěli zabývat dříve zmíněným indexovým adresováním regisrů IX a IY. Jak již bylo řečeno, můžeme tyto registry považovat za velké bratry registru HL. Pro IX platí hexadecimální označení DDH, pro IY platí označení FDH. K hexa. kódu, který platí pro HL, se jednoduše přidá DDH nebo FDH. Jako příklad uvedeme povel INC HL:

```
INC HL = 23
INC IX = DD 23
INC IY = FD 23
```

To platí jen pro ty povely, u kterých nejsou žádné cílové a zdrojové pohyby, protože tam musí být ještě hodnota indexu. Po Opcode se přidá ještě jeden byte.

Zabývejme se např. povelem:

```
LD A,(IX+05)
```

Tento povel má velkou vnější podobu s již zmíněným příkladem LD A,(HL), jen je tentokrát jiná hodnota výrazu v závorce; místo HL je zde IX+05. Jestliže tedy IX obsahuje hodnotu 2000H, přičte se k ní 5 a dostaneme hodnotu 2005H. Potom bude vybrána hodnota z adresy 2005Ha bude uložena do akumulátoru.

5.4.1. Výpočet indexového adresování.

Rozhodující je výpočet indexové hodnoty za Opcode, protože např. existuje povel LD A,(IX-03). Můžete se domyslet, že při indexovém adresování přicházejí v úvahu hodnoty v rozsahu od -128 do +127. Jako další příklad uvedeme povel:

```
LD A,(IX+F0H)
```

Při hexa. hodnotě F0 je nastaven sedmý bit. Tento sedmý bit je znaménkový bit - to znamená, že to je záporné číslo. V tomto případě se jednoduše hodnota neguje (vytvoří se doplněk); doplněk k FOH je 10H (F0H + 10H = 00H). Takže kdyby IX bylo 2000H, od 2000H by se odečetlo 10H, což dává 1FF0H. Z této adresy by byl obsah uložen do akumulátoru.

Příklady:

IX hodnota	Hodnota indexu	Vytvořená adresa
3000H	10H	3010H
3000H	00H	3000H
3000H	C0H	2FC0H

Ještě nezkušený programátor by měl nejprve znát všechny hodnoty HL-registru, než začne s indexovým programováním.

5.6. Dvojková aritmetika.

Dvojková logika je ve strojovém jazyce důležitá věc. Většinou jde o logické operace s akumulátorem, se kterým se dají lépe rozpoznat nebo maskovat určité stavu bitu.

Povely logických operací jsou:

a) AND
b) OR
c) XOR

Nejprve něco o základu logických operací. Provádí se vždy logické operace se dvěma bity podle logického schématu. Cílový bit bude obsahovat novou hodnotu, přičemž se postupuje podle tabulek.

Například tabulka příkazu OR

bit A	bit B	výsledný bit
0	0	0
1	0	1
0	1	1
1	1	1

Na vysvětlenou: Když má bit A hodnotu 1 a bit B hodnotu 0, bude mít výsledný bit logické operace hodnotu 1. Když bude mít bit A, nebo B hodnotu 1, bude výsledný také 1. Ve strojovém jazyce se provádějí logické operace vždy v akumulátoru s jedním s registrů. Například logická operace OR A,B, nebo lépe uspořádáno OR(AaB). Výsledek log. operace je také v akumulátoru. Proveďme operaci OR v akumulátoru a C registru. A obsahuje 45H a C obsahuje 09H. Nyní napište tyto hodnoty ve dvojkovém kódu.

registr A	01000101	45H	
C	00001001	09H	
	---		
výsledek	01001101	4DH	

Porovnávali jsme pod sebou napsané bity podle tabulky OR, a jeko výsledek jsme dostali 4DH.

Nyní tabulka pro příkaz AND

bit A	bit B	výsledný bit
0	0	0
0	1	0
1	0	0
1	1	1

Vezměte nahoře uvedená data a porovnejte podle tabulky AND.

registr A	01000101	45H	C	00001001	09H
	---				
výsledek	00000001	01H			

Po provedení AND A,B je tedy v akumulátoru hodnota 01H. Při log. operaci AND bude výsledek 1, pouze tehdy, když oba původní bity obsahují 1.

Nakonec ještě tabulka pro příkaz XOR.

bit A	bit B	výsledný bit

0	0	0
0	1	1
1	0	1
1	1	0

Znovu porovnáváme uvedená data pomocí tabulky XOR

registr A	01000101	45H
C	00001001	09H
	---	
výsedek	01001100	4CH

Poprovedení příkazu XOR A,B obsahuje akumulátor hodnotu 4CH. Příkaz XOR dá interpretovat také takto. Výsledný bit obsahuje 1 pouze tehdy, když bity A,B jsou rozdílné.

5.7. Logické a početní operace

V sadě příkazů je několik příkazů, které programátorovi při těchto operacích pomáhají

5.7.1. Příkaz ADD

Příkaz ADD je jednoduchý sčítací příkaz, který sčítá dva registry. Například ADD HL,BC. Když HL obsahuje 3245H a BC A753H, pak po provedení příkazu obsahuje HL registr hodnotu D998H.

5.7.2. Příkaz ADC

Příkaz ADC pracuje stejně jako ADD, ale při sčítání je přičten i indikátor přenosu. Kdyby byl indikátor nastaven a provedli buchom příkaz ADC HL,BC s hodnotami dříve uvedenymi, v registru HL by byl výsledek D999H.

5.7.3. Příkaz SUB

Příkaz SUB vzájemně odečítá dva 8-bitové registry. Například příkaz SUB A,B odečítá registr B od akumulátoru.

5.7.4. Příkaz SBC

Příkaz SBC pracuje stejně jako příkaz SUB, ale při nastavení indikátoru přenosu je odečten i indikátor přenosu. SBC příkaz platí na rozdíl od příkazu SUB i pro 16-bitové registry.

5.7.5. Příkaz INC

Příkaz INC jednoduše načítá registr, to znamená, že je jeho hodnota o jedenotku zvýšena. Když HL obsahuje 3323H, po prove-

dení příkazu INC HL bude obsahovat hodnotu 3324H. Podobně jako v BASICu X=X+1.

5.7.6. Příkaz DEC

Příkaz DEC jednoduše odčítává registr, to znamená, že jeho hodnota je o jednotku snížena. Když HL obsahuje 3323H, po provedení příkazu DEC HL bude obsahovat hodnotu 3322H. Podobně jako v BASICu X=X-1.

5.7.7. Příkaz CP

Příkaz CP umožňuje srovnávat registr nebo obsah paměti s akumulátorem. Sronává se, zda hodnota v akumulátoru je větší, menší nebo stejná. Například registry A i B obsahují hodnotu 99H. Po provedení příkazu CP A,B se v registrech hodnoty nemění, ale nastaven Zero-flag, protože hodnoty se sobě rovnají.

5.8. Práce s jednotlivými bity

V sadě příkazů jsou příkzy, které umožňují zjišťování hodnot, nastavení nebo obnovení počátečních hodnot jednotlivých bitů.

5.8.1. Příkaz SET

Příkaz umožňuje nastavení jednotlivých bitů v registrech. Například SET 7,A nastavuje nejvyšší bit v akumulátoru na 1. Může se manipulovat i s hodnotou v paměti pomocí indexového adresování.

5.8.2. Příkaz RES

Příkaz RES umožňuje nastavení počáteční hodnoty jednitlivých bitů v registrech. Například RES 5,B nastavuje pátý bit regisru B na 0. Jestliže B obsahoval předtím 23H (00100011), obsahuje po provedení příkazu hodnotu 03H (00000011)

5.8.3. Příkaz BIT

Příkaz BIT umožňuje zjišťování hodnot jednotlivých bitů v registrech. Například BIT 0,A zde se zjišťuje obsah nultého bitu akumulátoru. Tato částst hodnoty je potom v Z-flagu. Jestliže nultý bit akumulátoru obsahoval 1, pak v Z-flagu je 0.

5.8.4. Příkazy rotací

Příkazy rotací jsou příkazy, pomocí kterých můžeme bity v registrech rolovat, to znamená, že při levotočivé rotaci se 6-tý bit přesune na místo 7-mého bitu, 5-tý bit na místo 6-tého bitu atd. až se přesune 7-mý bit na místo 0-tého bitu. Při pravotočivé rotaci se budou bity posouvat obráceně. Příkazy rotace se liší jen nepatrně. Co bude pro programátory Z-80 jistě nové, je příkaz SLS (CB30 až CB37 jako Opcode). Většinou se obejdeme s příkazy RRCA a RLCA. Zde se ještě zmíníme o použití při násobení. Vezměme si např. číslo 52, napišme ho v binarním kódu 00110100. Když provrdeme RRCA, bude po rotaci v akumulátoru 01101000, což je číslo 104. Provedlo se násobení dvěma. Jestliže byl nastaven 7 bit, výsledek není spravný! Přeplnění je v flagu CY. Pávě tak otáčení doprava znamená dělení. Proto jsou tyto příkazy zvláště při matematických programech nepostradatelné.

5.8.5. Blokové příkazy

Mikroprocesor Z-80 má tu zvláštnost, že může použít příkazy pro operace s bloky. Tyto příkazy se dělí do dvou skupin.

a) příkazy pro hledání bloků
b) příkazy pro přesuny bloků

5.8.5.1. Příkazy pro hledání bloků

Hledání bloků zde bude vysvětleno na příkazu CPIR. Přitom se znak, který je v akumulátoru, hledá v určité oblasti. Adresa začátku je v HL, délka bloku, ve kterém se má hledat je v BC a znak, který se bude hledat je v akumulátoru. Například budeme hledat znak 66H v oblasti od 3000H do 4000H. Jestliže je znak nalezen, bude práce na tomto místě v paměti přerušena a Z-flag je nastaven. Jestliže znak není nalezen, je Z-flag nastaven na původní hodnotu.

```
1200  3E 66       LD   A,66H
1202  21 00 30    LD   HL,3000H
1205  01 00 10    LD   BC,1000H
1208  ED B1       CPIR
120A  CA XX XX    JP   Z,XXXXH
120D  pokračování
```

5.8.5.2. Příkazy přesunu bloků

Funkce přesunu bloků bude vysvětlena na příkazu LDIR. Přitom bude blok v paměti přesunut do jiné části paměti. Budeme přesouvat oblast od 2000H do 3000H za C000H. Pro příkaz LDIR jsou treba tři parametry:

a) počáteční adresa původního bloku HL - registr
b) počáteční adresa přesunu DE - registr
c) délka bloku BC - registr

Jsou-li nastaveny všechny parametry, můžeme použít příkaz LDIR. Pro zhora uvedená data by se musel napsat následující program:

```
1200  21 00 20    LD   HL,2000H     odkud
```

```
1203  11 00 C0    LD   DE,C000H     kam
1206  01 00 10    LD   BC,1000H     délka
1209  ED B0       LDIR
120B  C3 AD 00    JP   00ADH  skok do monitoru
```

5.8.6. Příkazy skoků

Zásadně jsou dva druhy skoků, podmíněné a nepodmíněné.

Nepodmíněné skoky "skáčí" vždy na definovanou adresu, podmíněné skoky "skáčí" na adresu pouze tehdy, je-li splněna podmínka, např. když je Z-flag 1.

5.8.6.1. Absolutní příkazy skoků

Absolutní příkazy skoků jsou jednoduché příkazy, například JP 3000H skočí vždy na adresu 3000H v paměti, nebo například JP NZ,3000H skáčí na adresu 3000H, když výsledek operace není nolový,bude v Z-flagu 0. N v podmínce znamená vždy NE !! Například: NC = not Carry

5.8.6.2. Relativní příkazy skoků

Relativní příkazy skoků jsou o něco komplikovanější než absolutní. Jsou také podmíněné a nepodmíněné, ale způsob adresování je jiný. Podobá se indexovému adresování, popsanému v kapitole 5.4.1.. Jak si vspomínáme, rozsah hodnot byl od -128 do 127, čili 7-mi bitové zobrazení. Právě tak je to u relativního skoku. Nejprve následuje normální byt pro druh příkazu, potom hodnota indexu, například

```
1200  38 06       JR C,PC+2+06      ;PC - obsah čítače
                                     instrukcí
```

V tomto případě bude pokračovat zpracování programu o 6 bytů dále, je-li nastaven indikátor přenosu. Výhoda relativních skoků je možnost přesunutí do libovolné oblasti paměti. Naproti tomu je nevýhodou malý rozsah adresování pouze 256 bytů, což hovoří pro apsolutní adresování. Zvláštní formou je skok závislý na registru například JP (HL).

To znamená, že ve zpracování programu se bude pokračovat na tom místě v paměti, které odpovídá hodnotě registru HL. Jestliže HL obsahuje například hodnotu 2000H, stačí na adresu 2000H. To platí i pro registry IX a IY.

5.8.7. Příkazy CALL

Příkaz CALL je obdobou příkazu GOSUB v BASICu. Je jím vyvolán podprogram, který je zakončen příkazem RETURN. Přitom existují právě tak jako u skoků podmíněných a nepodmíněných příkazy CALL, jako například CALL Z,3000H. Zde je vyvolán podprogram na adrese 3000H. když je nastaven Z-flag.

K příkazu CALL patří také příkaz RET, který může být též podmíněný nebo nepodmíněný. Například RET C skáče zpět, když je nastaven indikátor přenosu.

5.8.8. Příkaz DJNZ

Speciální příkaz skoku je příkaz DJNZ, který provádí funkci příkazu cyklu. Přitom registr B používá jako počítač smyček. Příkaz DJNZ je spojen s relativní adresou skoku a s odpočítáváním B registru. To může znít velmi složité, ale složité to není.

```
1200  06 06       LD   B,06H
1202  CD 3E 00    CALL 003EH  nechá zaznít tón
1205  10 FB       DJNZ 1202H
1207  C3 AD 00    JP   00ADH  skok do monitoru
```

Když se tento program odstartuje z monitoru pomocí J 1200, zazní 6x tón, protože jsem na začátku uložili do registru B hodnotu 6H, potom jsme standartním programem 003EH zavolali standartní tónový program. Nyní čítač programu narazil na příkaz DJNZ, přičemž od B registru odečte jednotku. B registr obsahuje nyní 5H. Protože tento registr neobsahuje 0, řízení programu skočí na 1202 a znovu se vyvolá tón. Po šesti kolech je B registr 0, takžese již neskáče na adresu 1202, ale pokračuje dále v programu. Pomocí tohoto příkazu se vytvoří smyčka. Program v BASICu by zněl takto: FOR T=1 TO 6:USR ($003E):NEXT

5.8.9. Příkazy IN a OUT

Tyto příkazy vysílají a přijímají data přes porty. Tím je umožněno provozování externích zařízení jako floppy, tiskárny apod. Vhodným programováním se dají například řídit signály modelů vlaků a mnoho jiných. U MZ-800 mají příkazy OUT a IN často za úlohu provádět řízení přídavné paměti, čili aktivovat nebo deaktivovat oblasti paměti.

6. Kapitola programování

6.1. Doporučení

Následující příkaz programů jsou napsány tak, že mohou být spuštěny pomocí monitoru MZ-800. Ale pro porozumění je potřeba Disassembler, protože pouze z hexadecimálních čísel se dá těžko vyčíst smysl. Napsané programy se dají spustit z monitoru příkazem J 1200 a vkládat příkazem M. Na to se musí zadat i hexadecimální kód. Po provedení programu se řízení předá zpět do monitoru. Pomocí Disassembleru se dá snadno rozluštit obsah programu.

6.2. Jak tisknout ve strojovém jazyce

6.2.1. Vytisknutí textu na displej

Při tisknutí ve strojovém jazyce je nejpříjemnější, dá-li se použít standartní program monitoru. To se musí text, který chceme vytisknout, napsat na určitou adresu v paměti. Pomocí příkazu M napíšeme na adresu 2000H náš text, například BBG

```
M 2000 (CR)
42 (CR)          B     v ASCII kódu
42 (CR)          B
47 (CR)          G
0D (CR)                konec textu
```

Jak vydíte, musí být text, který chceme vytisknout, okončen ODH. Nyní ještě program, který tento text vytiskne:

```
1200  11 00 20    LD   DE,2000H     adresa, textu
1203  CD 15 00    CALL 0015H  vyvolání tisku
1206  C3 AD 00    JP   00ADH  skok do monitoru
```

Takže se nejprve do DE registru nahraje adresa 2000H, na které je text, který chceme vytisknout. Když zavoláme podprogram 0015H, objeví se text na obrazovce. Po provedení J1200 se tedy objevy text BBG na obrazovce.

6.2.2. Vytištění jednoho znaku

V popsaném příkladě jsme tiskli na obrazovku libovolně dlouhý text. Nyní chceme na obrazovku vytisknout je jeden znak. Přirozeně pro to můžeme použít výše uvedený program, ale je jednodušší možnost, použit standartní program monitoru 0012H (viz. též standartní programy monitoru). Znak, který má být vytištěn musí být v akumulátoru, než je tiskový příkaz zavolán. Například: chceme-li vytisknou 0. Na to musíme znát hodnotu v ASCII kódu a tu uložit do akumulátoru (ASCII kód 0 je 30H).

```
1200  3E 30       LD   A,30H  vložení 0 (30H)
1202  CD 12 00    CALL 0012H  tisk
1205  C3 AD 00    JP   00ADH  skok do monitoru
```

6.3. Čtení klávesnice

6.3.1. GET ve strojovém jazyce

Rovněž velmi důležitou součástí programu je testování klávesnice. V monitoru je již testování klávesnice definováno (program monitoru 001BH). Tento program je volán stisknutím klávesy, její hodnota je uložena v akumulátoru. Nebyla-li stisknuta žádná klávesa, je v akumulátoru 00H. Tento program zároveň vytiskne znak stisknuté klávesy na obrazovku.

```
1200  CD 1B 00    CALL 001BH  GET program
1203  28 FB       JR   Z,1200H      není stisknuta
1205  CD 12 00    CALL 0012H  tisk stisk.klávesy
1208  C3 00 12    JP   1200H  testování kláves.
```

Jestliže nyní stiskneme libovolnou klávesu, zobrazí se znak klávesy obrazovce, Tento program se musí přerušit pomocí RESET.

6.3.2 INPUT ve strojovém jazyce

Zde jde jako v BASICu o to, zpracovávat celý řetězec znaků. Tento program je uložen v monitoru 0003H. Program uloží celý textový řetězec v ASCII kódu na adrese definovanou registrem DE. Konec řetězce je automaticky ukončen 0DH.

```
1200  11 00 20    LD   DE,2000H     oblast textu
1203  CD 03 00    CALL 0003H  progr. INPUT
1206  CD 06 00    CALL 0006H  nový řádek
1209  CD 15 00    CALL 0015H  tisk textu  120C  C3 AD 00    JP   00ADH
skok do monitoru
```

To je program, který přečte text, uloží ho od adresy 2000H a potom text vytiskne ještě jednou na obrazovku (J 1200). Text si můžeme prohlédnout příkazem monitoru D.

6.3.3. Kursorový vstup ve strojovém jazyce

Další možnost vstupu znaku z klávesnice je pomocí programu monitoru 09B3H. Při vyvolání tohoto programu bliká korsor na obrazovce a čeká na vklad znaku z klávesnice. Jestliže je stisknuta klávesa, je hodnota této klávesy ve video kódu v akumulátoru.

```
1200  CD B3 09    CALL 09B3H  vyzvedne znak
1203  32 00 D0    LD   (D000H),A    umístní znak
1206  C3 AD 00    JP   00ADH  skok do monitoru
```

Tento program vyzvedne znak z klávesnice a uloží ho na adresu D000H. Tato adresa patří video-paměti a je to první adresa v této paměti. Zna se tedy objeví na obrazovce vlevo nahoře.

6.4. Dimensování ve strojovém jazyce

Nejpozději při ukládání dat do paměti potřebuje každý programátor pole. V BASICu se pro to používá příkaz DIM, ve strojovém jazyce si takový program musíme sami napsat. Nyní provedeme jednoduché dimensování pole ve strojovém jazyce a porovnáme ho s BASICem. Nejprve příklad v BASICu:

```
10 DIM A(4)
20 A(0)=10000:A(1)=10001:A(2)=10003:A(3)=10005:A(4)=0
30 X=1:PRINT A(X)
```

Tento program by nám vytiskl na obrazovku číslo 10001. Naše data musíme v paměti ukládat od definované adresy a to ne v dekadickém systému, ale jako hexadecimální data. Uložíme našich pět dat za adresu 3000H. Dá se říci, že začátek tabulky téměř odpovýdá jménu proměnné. Jestliže bychom zvolili jiný začátek, dosáhli bychom samozřejmě jiných hodnot.

Za adresou 3000H v paměti musí být následující hodnoty (překontrolujte pomocí příkazu D monitoru):

```
D3000
3000 10 27 11 27 13 27 15 27
3008 00 00
```

To jsou hexadecimální hodnoty A(0) - A(4). Nyní botřebujeme dva parametry

a) naši proměnnou (zde adresa 3000H)
b) naše X (příklad v Basicu)

Jestiže první parametr máme 3000H a druhý parametr 1H, měla by nám být vydána hodnota 2711H (10001 dek.), jako příklad v BASICu.

```
1200  3E 01       LD   A,01H  v Basicu X=1
1202  21 00 30    LD   HL,3000H     začátek dat
1205  87          ADD  A            když 16 bit  A*2
1206  06 00       LD   B,06H  uložení hodoty z
1208  4F          LD   C,A    akum. do BC reg.
1209  09          ADD  HL,BC
120A  5E          LD   E,(HL)
120B  23          INC  HL
120C  56          LD   D,(HL)
120D  EB          EX   DE,HL
120E  CD BA 03    CALL 03BAH  tiskne hodnotu HL
1211  C3 AD 00    JP   00ADH  skok da monitoru
```

Jestliže nyní vložíme J1200, tak se na obrazovce objeví číslo 2711H. Nyní změníme adresu na 1201H, která odpovídá našemu X. Jestliže napíšeme 0, dostaneme 2710H (10000 dek.).

Uvnitř tohoto programu jsou části, které můžeme často použít jako užitečné programy. Podívejme se na program ještě jednou.

Neprve máme vstupy obou parametrů. potom. Potom provedeme ADD,A proč ? Zcela jednoduše, jestliže máme 1 jako první parametr a 3000H jako druhý parametr, pak adresa, na které je naše hodnota, musí být 3002H, protože máme dvou bytové hodnoty. Čili musíme A zdvojnásobit, pomocí ADD A. Potom musíme tuto novou hodnotu akumulátoru přičíst k našemu začátku tabulky (začátek tabulky je 3000H), což musí dát 3002H. Potřebujeme tedy jen program, který sčítá HL a akumulátor (ADD HL,A).

```
2000  C5          PUSH BC           zabraň BC, nutné
2001  06 00       LD   B,00H  vlož do BC
2003  4F          LD   C,A          akumulátor
2004  09          ADD  HL,BC
2005  C1          POP  BC           vrať BC
2006  C9          RET
```

Nyní chybí program, který z této adresy, která je v HL, uloží 16-ti bitovou hodnotu do registru HL, čili ulož do HL obsah adresy z HL (LD HL,(HL)).

```
2007  D5          PUSH DE           zabraň DE, nutné
2008  5E          LD   E,(HL)
2009  23          INC  HL
200A  56          LD   D,(HL)
200B  EB          EX   DE,HL
200C  D1          POP  DE
200D  C9          RET
```

Nyní můžeme program napsat ještě jednou s použitím podprogramu.

```
1200  3E 01       LD   A,01H
1202  21 00 30    LD   HL,3000H
1205  87          ADD  A
1206  CD 00 20    CALL 2000H  ADD HL,A
1209  CD 07 20    CALL 2007H  LD  HL,(HL)
120C  CD BA 03    CALL 03BAH  tisk na obrazovku
1210  C3 AD 00    JP   00ADH  skok do monitoru
```

Program můžeme odstartovat opět pomocí J1200, měnit adresu 1201H.

6.5. Smyčky

Smyčky jsou při vytváření programu velice důležitá věc, ale při smyčkových proměnných, které jsou větší než 255, není možno použít příkaz DJNZ. Zde se musí zhotovit nová smyčka. Napíšeme si program, který nechá 1000 x zaznít tón (prosím raději nezkoušet, chudák reproduktor).

```
1200  01 E8 03    LD   BC,03E8H     uloží do BC 1000
1203  C5          POSH BC           zajistí BC
1204  CD 3E 00    CALL 003EH  zazní tón
1207  C1          POP  BC
1208  0B          DEC  BC
1209  78          LD   A,B
120A  B1          OR   C
120B  C2 03 12    JP   NZ,1203H
120E  C3 AD 00    JP   00ADH  skok do monitoru
```

Jak vidíte, registr BC je použit jako čítač smyček. Proto musí být tato hodnota zajištěna pomocí PUSH BC, pro případ, že by registr BC byl změněn uvnitř smyčky. Po provedení standartního programu, ležícího ve smyčce (zde 003EH),musí být hodnota BC opět zajištěna. Potom musí být počet smyček sníženo jednu a musí být zjištěno, zda již není nolový, protože v tom případě by byl program přerušen. Zjištění, zda registr BC je nulový se provádí kombinací LD A,B a OR C. Vzpomeňme si, že pokud jsou obě části 0, je výsledek 0. Takže pouze pokud jsou nulové oba registry B i C, je nastaven Z-flag. Potom je smyčka přerušena, jinak se pokračuje v jejím provádění.

6.6. Dvojkový výraz v akumulátoru

Zde je krátký příklad programu, který vytiskne obsah akumulátoru na obrazovku jako 8-místné dvojkové číslo. Dá se použít jako podprogram.

```
2000  C5          PUSH BC
2001  F5          PUSH AF
2002  06 08       LD   B,08H
2004  07          RLCA
2005  F5          PUSH AF
2006  3E 30       LD   A,30H
2008  30 01       JR   NC,200B
200A  3C          INC  A
```

```
200B  CD 12 00    CALL 0012H
200E  F1          POP  AF
200F  10 F3       DJNZ 2004
2011  F1          POP  AF
2012  C1          POP  BC
2013  C9          RET
```

Základní myšlenka programu je v tom, že obsah akumulátoru je otá-
čen pomocí indikátoru přenosu. PO každé rotaci je indikátor
přenosu nastaven nebo ne, podle toho co obsahuje 7-mí bit s
akumulátoru. Jestliže je idikátor přenosu nastaven, vytiskne se
1, je-li nulový, vytiskne se 0.

6.7. Převod hexadecimálního čísla na dekadické

Pomocí tohoto programu se obsah registru HL ohjeví na
obrazovce dekadické číslo.

```
1200  A7          AND  A
1201  ED 52       SBC  HL,DE  dělení mocninou 10
1203  38 04       JR   C,1209H
1205  3C          INC  A
1206  10 F8       DJNZ 1200H
1208  C9          RET
1209  19          ADD  HL,DE   při přenosu korigovat
120A  C9          RET
120B  C5          PUSH BC           DE = BC
120C  D1          POP  DE
120D  AF          PUSH AF
120E  C9          RET
120F  00          NOP
1210  CD 0B 12    CALL 120BH
1213  06 0A       LD   B,0AH
1215  CD 00 12    CALL 1200H  program dělení
1218  C6 30       ADD  30H    převod do ASCII
121A  C3 12 00    JP   0012H  vytisknutí
121D  01 10 27    LD   BC,2710H     10000 dekadicky
1220  CD 10 12    CALL 1210H
1223  01 E8 03    LD   BC,03E8H     1000 dekadicky
1226  CD 10 12    CALL 1210H
1229  01 64 00    LD   BC,0064H     100 dekadicky
122C  CD 10 12    CALL 1210H
122F  01 0A 00    LD   BC,000AH     10 dekadicky
1232  CD 10 12    CALL 1210H
1235  01 01 00    LD   BC,0001      1 dekadicky
1238  C3 10 12    JP   1210H
```

Když chceme tento program otestovat, např. chceme znát dekadickou
hodnotu hexadecimálního čísla C000H, musíme napsat

```
2000  21 00 C0    LD   HL,C000H     2003  CD 1D 12    CALL 121DH
2006  C3 AD 00    JP   00ADH
```

Provedeme-li příkaz J 2000 se na obrazovce objeví číslo 49152.
Nyní musíme jen uložit hexadecimální číslo na adresu 2001H,
spustit program a dostaneme příslušné decimální číslo.

Ještě něco k porozumění programu. Dělíme naše číslo přísluš-
nými mocninami deseti tak dlouho, dokud nevznikne přenos, čili
nejprve 10000. Ve 43243 je obsaženo 4x, vytiskne se 4, zbytek

dělíme 1000 atd.

6.8. Převod dekadického čísla na hexadecimální

Napíšeme krátký program, který vyzvedne pětimístné dekadické číslo s klávesnice a převede ho na hexadecimální

```
1200  CD B3 09    CALL 09B3H  vyzvedne číslo
1203  CD CE 0B    CALL 0BCEH  Video kód na ASCII
1206  F5          PUSH AF           zachraň vlož.číslo
1207  CD 12 00    CALL 0012H  tisk čísla
120A  F1          POP  AF           vyzvedne číslo
120B  D6 30       SUB  30H    převádí na hexa.
120D  47          LD   B,A    nastav. smyček
120E  B7          OR   A            je číslo nula (0)
120F  C8          RET  Z            když ano, pak zpět
1210  19          ADD  HL,DE  přečte mocninu
1211  10 FD       DJNZ 1210H
1213  C9          RET
1214  CD 06 00    CALL 0006H  posun řádky
1217  21 00 00    LD   HL,0000H     inicializace HL
121A  11 10 27    LD   DE,2710H
121D  CD 00 12    CALL 1200H  vyvolání výpočtu
1220  11 E8 03    LD   DE,03E8H     1000 dekadicky
1223  CD 00 12    CALL 1200H
1226  11 64 00    LD   DE,0064H     100 dekadicky
1229  CD 00 12    CALL 1200H
122C  11 0A 00    LD   DE,000A      10 dekadicky
122F  CD 00 12    CALL 1200H
1232  11 01 00    LD   DE,0001H     1 dekadicky
1235  CD 00 12    CALL 1200H  1238  CD 06 00    CALL 0006H  posun řádeky
123B  CD BA 03    CALL 03BAH  tisk obsahu HL
123E  C3 AD 00    JP   00AD   skok do monitoru
```

Tento program ze dá volat z monitoru příkazem J 1214. Zadané dekadické číslo se převedede na hexadecimální. Ještě něco k porozumění. Při pětimístném čílse, např. 45225 se mohou jednotlivé mocniny deseti nasčítat, takže principiálně se sčítání provádí podle následujícího schematu.

```
45225 = 4*10000 + 5*1000 + 2*100 + 2*10 + 5*1
```

6.9 Přídavné příkazy Z-80

Přídavné příkazy Z-80 je něco, o čem není zmínka v žádné literatuře o Z-80.

Rozsah příkazů procesoru zahrnuje totiž asi 1000 příkazů, místo známých 400. Z největší části se přitom jedná o příkazy, pro řízení registrů HX, HY, LX a LY. Jak jsme se již zmínili, jedná se přitom o 8-mi bitové registry a to vždy o vyšší a nižší byt registru IX a IY. Přitom je opcode přikloněn ke kódům registrů H a L, právě tak jako IX a IY, právě tak jako IX, IY jsou přikloněny k opcode registru HL.

```
2000  26 77       LD    H,77H
2002  DD 26 77    LD    HX,77H
```

Takže se jednoduše napíše DD nebo ED a už je možnost výběru z obrovské palety příkazů. Musíme si dát pozor, protože téměř žádný Disassembler není schopen tyto příkazy přečíst, protože memí schopen tyto příkazy přečíst, protože o nich není nikde zmínka. V naší nabídce naležnete Disassembler, který je schopen tyto příkazy přeložit.

MONITOR 1Z-013B

Monitor 1Z-013B je 4 KB dlouhý provozní systém, který je stále uložen v paměti ROM. V tomto monitoru jsou uloženy všechny potřebné podprogramy, které jsou nutné k programování MZ-800. Monitor je převzat od MZ-700, proto je s ním kompaktibilní. To znamená, že všechny programy z MZ-700 mohou být provozovány na MZ-800.

Monitor 1Z-013B je uložen v paměti EPROM od adresy 0000H až do 1000H. Normálně jsou volány pouze podprogramy tohoto monitoru. Vlastní monitor se nikdy sám neaktivuje. Abychom se dostali k tomuto monitoru musíme nejdříve provést následující úkony:

1) Zapnout počítač, nebo stiskmout klávesu RESET
2) Stlačit klávesu "M"
3) Po objevení Cursoru "*" můžete pracovat

Nyní jste v monitoru 1Z-013B. Tento monitor obsahuje téměř tytéž povely (tj.příkazy a instrukce) jako monitor popsaný v příručce SHARP. Zmíníme se však o dvou rozdílech:

1) Povel GO (klávesa "G") je nyní klávesa "J" jako Jump
2) Pomocí povelu Bank Switching tj. stránkování paměti (klávesa "#") se dá ROM monitor vypnout. Současně způsobí tento povel "teplý" start Basic-interpreteru. Takže když jste nedopatřením skočili z Basic-interpreteru do ROM monitoru, můžete se tímto způsobem znovu dostat do Basicu, aniž byste ztratili program. (Jako stlačení klávesy "CTRL + RESET"současně.)

V monitoru 1Z-013B jsou definovány nejen základní povely,ale také jsou k disposici všechny důležité podprogramy, jako například testování klávesnice, tisk na obrazovku nebo řízení magnetofonu. Tyto standartní programy (rutiny) a způsob jejich funkce budou posány dále v textu.

Pro porozumění dalšího popisu je nutná základní znalost programování ve strojovém jazyce, pro nováčky je vhodné si nejprve přečíst kurs programování ve strojovém jazyce.

První díl monitoru tvoří tabulka skoků, která splňuje následující hlavní funkce:

a) Umožňuje co největší kompatibilitu s předchozími modely MZ-80K a MZ-80A.
b) Důležité programy monitoru se snáze pamatují. Na programy monitoru se skáče pomocí příkazu ve strojovém jazce CALL ****H Po provedení programu v monitoru se automaticky skočí zpět k adrese, ze které byl program v monitoru zavolán (jako příkaz GOSUB v Basicu).

Podprogramy monitoru

0000H Start provozního systému v oblasti E
Tento program způsobí skok na adresu E800H, to znamená na počáteční (startovací) adresu oblasti E monitoru.

0003H GET LINE umožňuje vkládání řádků, čili více znaků z klávesnice. Vložený řetězec znaků je uložen do paměti na místo určené registrovým párem DE. Program končí stisknutím klávesy "CR". Po skončení podprogramu se za řetězec znaků přidá koncová značka (0DH). Tato značka je nutná k označení konce vloženého řetězce znaků. Kromě vložení řetězce znaků se tento řetězec současně objeví na obrazovce. Maximálně může být vloženo 80 znaků.
Při současném stisku kláves SHIFT a BREAK se na adresu, která je v registru DE, uloží BREAK-Code místo řetězce znaků a kód návratu je uložen na následující adrese.

0006H NEW LINE kursor se nastaví na začátek další řádky.

0009H NEW LINE kursor se nastaví na začátek další řádky, pokud není na začátku řádky.

000CH PRINT SPACE vytiskne prázdný znak na momentální pozici kursoru

000FH PRINT 10 SPACES vytiskne od momentální posice kursoru maximálně 10 prázdných znaků až po další TAB-Posici(0,10, 20,30)

0012H PRINT CHARAKTER vytiskne ASII znak uložený v akumulátoru. Řídící znaky 11H - 16H nevytvoří žádný znak, ale jejich funkce se provedou. Takže např. jestliže uložíme do akumulátoru 16H a zavoláme tento program, obrazovka se smaže (CLEAR-HOME)

0015H PRINT MESSAGE vytiskne řetězec znaků počínaje posicí kursoru na obrazovce. Počáteční adresa řetězce znaků , čili příslušná adresa v paměti musí být předem zadaná v registru DE. Řetězec znaků musí být ukončen 0DH (tj.znak klávesy "CR"}. Řídící znaky 11H-16H budou provedeny.

0018H PRINT MESSAGE vytiskne řetězec znaků počínaje posicí na obrazovce. Počáteční adresa v paměti se předá v registru DE. Řetězec musí být ukončen znakem 0DH. Řídící znaky 11H-16H se neprovádí, ale jsou obrácně vytištěny.

0018H GET KEY z klávesnice se přečte znak v ASCII kodu a tato hodnota je uložena v akumulátoru. Jestliže není stisknuta žádná klávesa, je v akumulátoru při návratu hodnota ODH. Zvláštní klávesy dávají odpovídající ASCII - hodnotu.

Tabulka zvláštních kláves:

klávesa	Hex - code
DEL	60H
INST	61H
ALPHA	62H
GRAPH	63H
SHIFT&BREAK	64H

CR (Return)	66H
Malá volná	90H

Malá volná klávesa (tj.neoznačená klávesa nad klávesou CR) se u japonských modelů používá pro přepínání sad znaků mezi Hiragana a Katagana. Klávesy funkcí nemohou být tímto programem testovány.

001EH SHIFT, BREAK, CTRL — když je stisknuta klávesa BREAK, je příznak přenosu C (CARRY-Flag) nastaven na "1". Když je stisknuta klávesa SHIFT, je 6. bit v akumulátoru nastaven na "1". Při stisknutí klávesy CTRL je nastaven 5. bit v akumulátoru na "1" a při současném stisknutí kláves SHIFT a CTR je nastaven 4. bit v akumulátoru na "1".

0021H WRITE HEADER — návěští začátku programu (header) je zapsáno na kazetu. Návěští začátku programu obsahuje následující data, která specifikují program:
jméno programu, začátek programu, délka programu, kód souboru a startovací adresu. Vyjmenovaná data jsou uložena na adresách 11F0H-1167H. Blíže viz obsazení pomocných buněk monitoru.

0024H WRITE DATA — program (blok dat) určený počáteční adresou a délkou se nahraje na kazetu. Program nebo blok dat se pro jistotu nahrává dvakrát. Přenos je 1200 bitů za vteřinu.

0027H READ HEADER — návěští začátku program je zavedeno do paměti. Návěští začátku programu obsahuje jméno programu, začátek programu, délku programu, kód souboru a startovací adresu.

002AH READ DATA — nahrává do paměti program, určený počáteční adresou a délkou

002DH VERIFY — srovnává se program nahraný na kazetě s programem v paměti. Při chybě ve srovnání, tzn. CHECK SUM ERROR, je vydáno chybové hlášení.

0030H MELODY — hraje hudbu odpovídající řetězci znaků, jehož počáteční adresa je v registru DE. Řetězec musí být ukončen znakem 0DH nebo C8H. Navíc je v tomto standartním programu obsaženo testování stisknuté klávesy BREAK. Při stisknutí této klávesy je nastaven příznak přenosu C.

0033H Nastavení a spuštění zabudovaných hodin. Do akumulátoru je vložena 0 nebo 1 (AM nebo PM) a registr DE je inicialisován počtem vteřin (naximálně 43200, což odpovídá 12 hodinám).

0038H INTERRUPT — při povelu RESTART provede procesor Z-80 v souvislosti s hardwarem větvení skok na adresu 0038H, protože tato adresa je v oblasti monitoru a nemůže být změněna, skáče se automaticky na adresu 1038H, na kterou se může uložit libovolný INTERRUPT vektor.

003BH READ TIME — přečte se momentální čas. V registru DE se po vyvolání toho standartního programu nachází počet vteřin (maximalně 43200, což odpovídá 12 hodin) a v akumulátoru je uložena 0 nebo 1.

003EH BELL vytvoří krátký tón o frekvenci 880 Hz.

0041H TEMPO SET určuje rychlost přehrání hudby v závislosti na bsahu akumulátoru. Jsou povoleny hodnoty od 01 do 07. Přitom 01H odpovídá pomalé rychlosti, 04H střední a 07H je rychle.

0044H SOUD START vytvoří stálý tón udané frekvence. Frekvence se vypočte následující rovnicí: F= 895 kHz/NM , přičemž NM odpovídá 2 bytovému číslu v paměti na adrese 11A1H. Proto nutno dát pozor na to, že N musí být na adrese 11A2H a M na adrese 11A1H.

0047H SOUD STOP ukončí tón, vytvořený pomocí SOUD START

0041H Start Monitoru 1Z-013B, ale po něm ihned následuje skok do E-oblasti za E800H.

00ADH Teplý start Monitoru 1Z-013B. Po skoku na tuto adresu se nacházíte v monitoru 1Z-013B. V tomto monitoru jsou vám k disposici následující povely:

J	Jump
L	Load (z kazety)
F	Flopy Boot
B	Key BELL (tón při každém stisknutí klávesy)
#	Teplý start Basicu (pro MZ-700 i MZ-800)
P	Printer Test (Test tiskárny)
M	Memory Correction (oprava paměti)
S	Save (na kazetu)
D	Hexdump
*	*

00F3H JUMP (příkaz) na tento standartní program se skočí při stlačení klávesy "J".

00F7H BEEP (příkaz on/of) na tento standartní program se skočí při stlačení klávesy "B".Při každém stisknutí klávesy zapnut respetive vypnut.

00FFH QUICK DISK jesliže je připojena disková jednotka, skočí se na F000H, tzn. že je inicialisován Quid disk

0111H LOAD (příkaz) na tento standartní program se skočí při stlačení klávesy "L".

0155H Ploter Printer Test Befehl povel testu kreslícího zařízení ev. tiskárny. Na tuto standartní adresu se skočí při stlačení klávesy "P".

018FH PLOT/PRINT CHARACTER ASCII hodnota v akumulátoru se předá na plotter event. tiskárnu a vytiskne se. Některé hodnoty jsou reservovány pro řízení plotteru. Podrobný popis tohoto podprogramu je v kapitole řízení ploteru.

01A5H PLOT/PRINT MESSAGE ASCII hodnoty v řetězci se jedna po druhé předávají na plotter ev. tiskárnu a vytisknou se. Posice řetězce znaků se předává v registru DE. Řetezec musí být ukončen znakem 0DH. Podrobný popis tohoto podprogramu je v kapitole řízení plotteru.

01C7H MELODY jako 0030H.

02A6H INCREMENT DE registr je čtyřikrát inkrementován (inkrementovat znamená zvýšit obsah o +1).

02ABH SOUND START jako 0044H

02BEH SOUND STOP jako 0047H

02E5H SET TEMPO jako 0041H

O2F3H CRT MANAGMET tento základní program se používá pro interní řízení obrazovky. Po naskočení programu jsou do registru HL vloženy souřadnice kursoru. Souřadnice X je v L, souřadnice Y je v H. Současně je v registru DE uložena značka, která udává, zda momentální řádka je dvojnásobná (tj.delší než 40 znaků). Značka leží v oblasti od 1173H do 118BH. Kromě toho je do akumulátoru vložena 0 nebo 1 podle toho, zda je nebo není použita dvojnásobná délka.

0308H SET TIME jako OO33H

0358H READ TIME jako 003BH

O38DH TIME INTERRUPT když byl nastaven indikátor příznaku AM, smaže se a nastaví se indikátor PM, eventuelně obráceně. Současně se do časovače uloží 43200 vteřin.

03B1H DISPLAY SPACE & CHARACTER vytiskne prázdný znak a ASCII znak, který je určen obsahem akumulátoru. Výtisk na obrazovku se provede na Video RAM adresu určenou registrem HL.

03B1H ASCII PRINT FOR HL Obsah registu HL se vytiskne na obrazovku od momentální posice kursoru

03C3H ASCII PRINT FOR ACC obsah akumulátoru se vytiskne na obrazovku od momentální posice kursoru.

03D5H Data k přepnutí plotru na 80 znaků na tomto místě v paměti jsou uložena data (01H, 09H, 09H, 09H, 0DH), která jsou nutná k přepnutí plotru na 80 znaků.

03DAH HEX TO ASCII hexadecimální číslo ve 4 nižších bitech akumulátoru se převede na odpovídající ASCII kód a uloží se v akumulátoru.

03E5H ASCII TO HEX převádí hexadecimální číslo, které je jako ASCII znak uloženo v akumulátoru, na dvojkové číslo a ukládá výsledek do 4 nižších bitů akumulátoru. Při chybě je nastaven příznak C (tj. CARRY -Flag).

03F9H ASCII TO HEX jako 03E5H.

0410H FOUR CHARACTER ASCII CONVERSION převádí řetězec, který představuje 4 místné hexadecimální číslo v ASCII formátu na hexadecimální číslo a ukládá ho do registru HL. Registr DE musí obsahovat počáteční adresu řetězce znaků, který obsaje čtyřmístné hexadecimální číslo v ASCII formátu. Tento standartní program tedy převádí čtyřmístný řetězec znaků např. "3410" na 16-ti bitové hexadecimální číslo. Při chybě, která může být způsobena neplatným řetězcem

znaků, se nastaví indikátor přenosu C na "1".
(Flag C je "1") .

041FH TWO CHARACTER ASCII CONVERSION převádí řetězec, který představuje dvoumístné hexadecimální číslo v ASCII formátu,na hexdecimální číslo a ukládá ho do akumulátoru. Registr DE musí obsahovat počáteční adresu řetězce znaků který představuje dvoumístné hexadeciální číslo v ASCII formátu. Při chybě, která musí být způsobena neplatným řetězcem znaků, je indikátor přenosu CARRY nastaven na "1"

0436H WRITE HEADER jako 0021H

0470H Data k přepnutí plotru na 40 znaků. Na tomto místě v paměti jsou uložena data (01H, 09H, 09H, 0BH, 0DH), která jsou nutná k přepnutí plotru na 40 znaků.

0475H WRITE DATA jako 0024H

04D8H READ HEADER jako 0027H

04F9H READ DATA jako 002AH
O577H BELL jako 003EH

057EH FLASHING AND KEYIN z klávesnice se přečte znak ve Video-códu a uloží se do akumulátoru. Současně bliká kursor. Není-li stisknuta žádá klávesa, je do akumulátoru uloženo F0H a je nastaven příznak Zero
(bit Z v registru F je nastaven na "1")

0588H VERIFY jako 002EH

05FAH NEW LINE AND ASCII PRINT FOR HL na obrazovku se vytiskne na začátek nové řádky obsah registru HL.

069FH CASSETTE MOTOR ON zapíná motor magnetofonu

0700H CASSETTE MOTOR OFF zapíná motor magnetofonu

0759H 107 MICROSECOND DELAY časové zpoždění 107 mikrosekund

07A8H M-BEFEL povel opravy paměti

07E6H GET LINE jako 0003H

087CH Tento standartní progaram provádí návrat vozíku a nastavuje kursor na začátek na další řádky.

0893H PRINT MESSAGE jako 0015H

08A1H PRINT MESSAGE jako 0018H

08BDH GET KEY jako 001BH

090EH NEW LINE jako 0006H

0918H NEW LINE jako 0009H

0920H PRINT SPACES jako 000CH

0924H PRINT 10 SPACES jako 000FH

0935H PRINT CHARACTER jako 0012H

0996 7 MILLISECOND DELAY časové zpoždění 7 milisekund.
 09B3H INPUT KEY tento standartní program čeká s blikajícím kursorem tak dlouho, dokud není vložen znak z klávesnice Ovšem nemusí být stisknuto "CR" (Return). Odpovídající hodnota stisknté klávesy ve Video-codu je uložena v akumulátoru.

0A32H SHIFT, BREAK, CONTROL jako 001EH

0A50H Testování sítě klávesnice. pomocí tohoto standarního programu můžete přímo testovat klávesnici. Můžete testovat i klávesy funkcí.
Jestliže se skočí do tohoto programu, zatím co je stisknuta nějaká klávesa budou registry B a C obsahovat následující data:

B -registr:

7 bit je 0 ,jestliže nebyla stisknuta žádná klávesa
7 bit je 1 ,jestliže byla stisknuta nějaká klávesa
6 bit je 0 ,jestliže nebyla stisknuta klávesa SHIFT
6 bit je 1 ,jestliže byla stisknuta klávesa SHIFT
5 bit je 0 ,jestliže nebyla stisknuta klávesa CTRL
5 bit je 1 ,jestliže byla stisknuta klávesa CTRL
4 bit je 0 ,jestliže nebyly současně stisknuty klávesy SHIFT a CTRL
4 bit je 1 ,jestliže byly současně stisknuty klávesy SHIFT a CTRL

C -registr:

bity 0, 1, 2 ,dávají číslo řádku sítě klávesnice
bity 3, 4, 5 ,dávají číslo sloupce sítě klávesnice

MATICE KLÁVESNICE

Tastaturabtastung {$E000H} Tastendaten {$E001H}

```
            F0  - F1 - F2 - F3 - F4 - F5 - F6 - F7  -  F8
 řada/sloup. 1   - 2  - 3  - 4  - 5  - 6  - 7  - 8   -  9  - 10
-------------------------------------------------------------
D7/11     --- neoz  - Y  - Q  - I  - A  - 1  - \ -INST - BREAK- F1
     !        !    !    !    !    !    !    !     !      !
D&/12 --- GRA   - Z  - R  - J  - B  - 2  - ~ - DEL -  CTRL- F2
     !      !    !    !    !    !    !    !     !      !
D5/13 --- libra - @  - S  - K  - C  - 3  - - - nahor      - F3
     !      !    !    !    !    !    !    !     !      !
D4/14 --- ALPH  - [  - T  - L  - D  - 4  - S - dolů       - F4
     !      !    !    !    !    !    !    !     !      !
D3/15 ---       - ]  - U  - M  - E  - 5  - 0 - vlevo      - F5
     !      !    !    !    !    !    !    !     !      !
D2/16 ---   :   -    - V  - N  - F  - 6  - 9 - vprav      -
     !      !    !    !    !    !    !    !     !      !
D3/17 ---   ;   -    - W  - O  - G  - 7  - , -  ?  -      -
     !      !    !    !    !    !    !    !     !      !
D0/18 ---  CR   -    - X  - P  - H  - 8  - . -  /  - SHFT -
```

0A92H ASCII TO DISPLAY CODE TABLE od této adresy jsou data pro převádění z ASCII kódu do Video-kódu.

0BB9H ASCII TO DISPLAY převádí ASCII znak v akumulátoru do Video-kódu (Display-code, zobrazovací cód) a ukládá tento kód do akumulátoru.

0BCEH DISPLAY TO ASCII převádí Video-kód v akumulátoru na ASCII znak a ukládá ho do akumulátoru.

0BEAH MATRIX TO DISPLAY CODE TABLE od této adresy v paměti jsou Video-bity, odpovídající síti klávesnice, Když např. po vyvolání 0A50H, je v registru C uloženo 07H, potřebujeme si v této tabulce najít pouze 7 byte (počínaje od 0 a ne od 1 !) 7 byte je CDH a tím odpovídá klávese CR.

0BEAH KEY MATRIX TO DISPLAY CODE TABLE Normaler normální

0C2AH KEY MATRIX TO DISPLAY CODE TABLE při stisknutí SHIFT

OC6AH KEY MATRIX TO DISPLAY CODE TABLE při stisknutí GRAPH
OCAAH KEY MATRIX TO DISPLAY CODE TABLE při stisknutí CTRL

0CE9H KEY MATRIX TO DISPLAY CODE TABLE při přepnutí na Katagana (tj. neoznačené tlačítko). Tato tabulka není u evropských modelů použita.

OD29H D - povel k zobrazení zvolené části paměti.

ODA6H VERTICAL BLANKING CHECK zjišťuje zda probíhá období vertikálního nulování (zatemnění). Řízení je vráceno vyvolávajícímu programu, je-li vstupováno během tohoto období.

ODB5H DISPLAY CODE CHARACTER PRINT na momentální pozici kurzoru na obrazovce se vytiskne znak, jehož Video-kode je uložen v akumulátoru.

ODDCH DISPLAY CONTROL pomocí tohoto standartního programu se dá měnit zobrazení na obrazovce. Přitom akumulátor obsahuje odpovídající řídící znak. Řídící znaky a jejich působení jsou následující:

akumulátor	způsobí
COH	přetočení obrazovky o řádku dolů
C1H	kurzor se posune o řádku dolů
C2H	kurzor se posune o řádku nahoru
C3H	kurzor se posune o znak doprava
C4H	kurzor se posune o znak doleva
C5H	kurzor se nastaví na pozici HOME(D000H)
C6H	Video - RAM a barvy v RAM se vymažou, kurzor je nastaven na pozici HOME.
C7H	písmeno za kurzorem se vymaže a kurzor se nastaví na tuto pozici. Stejná funkce jako klávesa "DEL".
C8H	Na momentální pozici kurzoru se vsune písmeno a všechna ostatní písmena za

tímto se posunou doprava. Stejná funkce
jako klávesa "INST".

C9H přepíná klávesnici z grafického módu
do alfa módu

CDH provede se návrat vozíku (CR) a kurzor
se nastaví na začátek dalšího řádku na
obrazovce.

0F5EH S - povel SAVE uložení.

0FB1H momentální pozice kurzoru se převede na Video RAM adresu
a uloží se do registru HL.

0FCBH V - povel VERIFY ověřovací příkaz.

0FD8H vymaže paměť od adresy v regustru HL. Délku mazané paměti
udává registr B. Vymazání znamená uložit do oblasti
paměti data 00H.

Monitor 1Z-013B potřebuje oblast RAM od 1000H - 11FFH jako pracovní oblast, tzn., že v této oblasti paměti se ukládají všechna potřebná data.

Obsazení pomocných buněk monitoru (pracovní oblast)

adresa	význam
1000H - 1037H	volný
1039H	nižší byt vektoru přerušení
103AH	vyšší byt vektoru přerušení Při přerušení se skočí na zde uvedený vektor přerušení (adresu).
103BH - 10EFH	oblast zásobníku Normálně se v této oblasti nachází u každého programu zásobník.
10EFH	TOP DE STACK konec zásobníku
10F0H - 1107H	data návěští začátku programu
10F0H	kód souboru FILECODE (druh programu) označují program v Basicu, Assembleru atd. 01 = program ve strojovém jazyku 02 = program v MZ-80K Basiku 03 = data v MZ-80K Basicu 05 = programy v Basicu MZ-700 a MZ-800
10F1H - 1101H	jméno programu Jméno programu může být dlouhé maximálně 16 znaků a musí být ukončeno znakem 0DH (CR). Jestliže na adrese 10F1H je 0DH, nemá program žádné jméno.
1102H	nižší byte délky programu
1103H	vyšší byte délky programu Zde uložená délka programu je rozhodující pro

počet bytů, které se při programu SAVE uloží
na kazetu

1104H nižší byte začátku programu

1105H vyšší byte začátku programu
Od zde uložené počáteční adresy se začne
program nahrávat na kazetu při skoku do stan-
dartního programu SAVE (0024H)

1106H nižší byte startovací adresy

1107H vyšší byte startovací adresy
Na této adrese závisí, zda bude nahraný program
automaticky proveden. Jestliže má být program
automaticky odstartován, musí na těchto dvou
adresách být startovací adresa programu, který
má být proveden. Startovací adresa musí být
větší nebo rovna 1200H. Při 0000H se program
neodstartuje.

1108H - 116FH vstupní buffer (vyrovnávací paměť)
Tyto buňky paměti se používají jako vstupní
vyrovnávací paměť.

1170H značka pro malá nebo velká písmena
Obsah 0 odpovídá velkým písmenům.
Obsah 1 odpovídá malým písmenům.

1171H Y - pozice kurzoru (0-18H)
Označení řádky pozice kurzoru.

1172H X - pozice kurzoru (0-27H)
Označení sloupce pozice kurzoru.

1173H - 118BH pro každou řádku obrazovky jeden byt, který
udává, zda je řádka druhá část dvojité řádky.
V tom případě je obsah 1, jinak 0.
118EH značka pro uložení znaku, na kterém je momentálně
kurzor

118FH značka pro pozici kurzoru - nižší byt
1190H značka pro pozici kurzoru - vyšší byt
Použito jen u modelu předcházejícího MZ-80K.

1191H značka, zda je zobrazen kurzor nebo znak.
0 = znak, 1 = kurzor. Použito jen u modelu
předcházejícího MZ-80K.

1192H zobrazovací kód znaku kurzoru

1193H STRING-flag
Použito jen u modelu předcházejícího MZ-80K.

1194H značka pro počet znaků ve vkládané řádce

1195H délka předpětí kazety nižší byt

1196H délka předpětí kazety vyšší byt
Označení délky předpětí pri nahrávání, resp.
čtení kazety.

1197H kontrolní součet při nahrávání do počítače
nižší byt

1198H kontrolní součet při nahrávání do počítače
vyšší byt
V těchto buňkách je uložena kontrolní suma
při nahrávání z kazety. Kontrolní suma je
počet jednotkových bitů v nahraných bytech.
Tento součet slouží ke zjištění chyb nahrávání.

1199H kontrolní součet při nahrávání na kazetu
nižší byt

119AH kontrolní součet při nahrávání na kazetu
vyšší byt
Označení kontrolní sumy při nahrávání na kazetu.
Kontrolní suma jako u 1197H 1198H.

119BH označení pro AM/PM (hodiny)
AM = 0, PM = 1

119CH označení běhu hodin
Při obsahu FOH běží zabudované hodiny

119DH označení KEY-SOUND
Označení pro vytvoření tónu při stisku klávesy.
Tón = 0, bez tónu = 1.

119EH paměť pro tempo hudby

119FH paměť pro délku tónu

11A0H paměť pro číslo oktávy

11A1H paměť pro frekvenci nižší byt

11A2H paměť pro frekvenci vyšší byt

11A3H - 11F3H vstupní vyrovnávací paměť klávesnice

11F4H - 11FFH volné

Monitor E - oblasti

Monitor E - oblasti je druhý, přes 4 KB dlouhý provozní systém, který je též stále uložen v oblasti ROM. Monitor E - oblasti leží, jak již jméno napovídá, v horní oblasti paměti od E010H po F36CH.

V této oblasti leží všechny standartní programy pro Quick disk a další monitor 9Z-504M.

Při spuštění počítače a při každém stisknutí RESET se automaticky skočí na adresu 0000H. Odtud se ale ihned skočí na E800H a objeví se ihned počáteční obraz. Jestliže stisknete klávesu "M" jako Monitor, nalézáte se v Monitoru 9Z-504M.

Podrobnější popisy standartních programů pro Quick disk viz též v kapitole Quick-disk.

E010H QD-IOS hlavní inicializační program operačního systému Quick-disku. Podle toho, jaký je obsah buňky paměti QDPA (1130H), vyvolají se následující standartní programy:

1 = Quickdisk READY-Check
Zkoumá, zda je Quickdisk připojen a připraven k provozu

2 = FORMAT
Formátuje Quick-disk

3 = READ
Čte z Quick-disku

4 = WRITE
Píše na Quick-disk

E08AH standartní program kontroly připravenosti Quick-disku (READY-Check)

E090H standartní program FORMAT

E0DAH standartní program READ

E14EH standartní program WRITE

E29BH QD MOTOR ON zapíná motor Quick-disku

E2E8H QD MOTOR OFF vypíná motor Quick-disku

E44AH program řízení floppy disku řídí připojenou disketovou jednotku

E4DCH program kontroly připravenosti floppy disku
Kontroluje, zda je připojena disketová jednotka a zda je připravena k provozu.

E517H FD MOTOR ON zapíná motor disketové jednotky

E530H FD MOTOR OFF vypíná motor disketové jednotky

E800H tabulka skoků na toto místo se skáče okamžitě po zapnutí nebo po resetu

E801H skok do monitoru MZ-800 (počáteční obraz)

E804H skok do monitoru 9Z-504M

E807H nahrání z kazety za 1200H

E80AH nahrání na kazetu od 1200H

E80DH verifikace kazety od 1200H

E810H skok na QD-IOS, tedy na E010H

E813H začátek monitoru MZ-800 (počáteční obraz)
V této části programu se inicializují všechny důležité součásti a adresy, jako např. část port 8255, časovač 8253, Z-80 PIO, SIO

E876H inicializace barev obrazovky a tempa hudby

E8A9H do paměti programovatelného generátoru znaků se nahraje obsah generátoru znaků ROM, tzn., že znaky pevně uložené v CG-ROM (CG = znakový generátor) se přenese do PCG - RAM.

E8E1H inicializuje se paleta barev a barvy okraje

E945H IPL nahrávač z kazety nahrání programu z kazety a odstartování

EA5EH testování povelů monitoru 9Z-504M
Monitor 9Z-504M obsahuje následující povely:

J = JUMP
L = nahrání z kazety
F = inicializace floppy disku
B = BELL on/off (vypnutí nebo zapnutí tónu při stisku klávesy)
M = oprava paměti
S = nahrání na kazetu
V = verifikování kazety
D = HEXdump
Q = Quickdisk (s dalšími parametry, jako QL, QS, QD atd)
E = Exit Ramboard
G = Goto$

EAA9H J - povel

EAB5H Q - povel
Jsou možné následující povely pro Quickdisk:

QL = nahrání z Quickdisku
QS = nahrání na Quickdisk
QC = kopírování Quickdisku
QF = formátování Quickdisku
QX = přenos z kazety na Quickdisk
QD = adresář Quickdisku

EB34H GETLINE s testováním BREAK
Tento standartní program umožňuje vložení řádky, čili více znaků, z klávesnice. Vložený řetězec znaků se uloží do vstupní vyrovnávací paměti, takže od místa v paměti 11A3H. Standardní program končí po stisknutí klávesy CR. Po skončení tohoto podprogramu se na konec řetězce znaků automaticky přidává 0DH.
EB4CH L - povel

EBAEH S - povel

EC00H V - povel

EC1EH B - povel

EC29H M - povel

EEA7H QUICK DISK LOAD povel nahrání z quick disku. Od této adresy je uložen kompletní povel (Quickpodprogram) s dotazem na parametry.

EF2EH QUICK DISK SAVE povel nahrání na quick disk
Od této adresy je uložen kompletní povel nahrání na

quick disk s dotazem na parametry.

EFEFH QUICK DISK DIRECTORY povel adresáře quick disku.

F0B5H QUICK DISK FORMAT povel formátování quick disku.
Od této adresy je uložen kompletní standartní program formátování quick disku s dotazem "OK (Y/N)".

F12CH QUICK DISK COPY povel kopírování quick disku.
Od této adresy je uložen kompletní program kopírování quick disku s dotzem na jméno programu.

F1A2H CASSETTE TO QUICKDISK COPY povel kopírování z kazety na quuick disk. Od této adresy je kompletní program kopírování z kazety na quick disk s dotazem na jméno programu.

F202H Error Tabelle tabulka chyb. Hodnotám chyb se zde přiřazují odpovídající chybová hlášení.

Obsazení pomocných buněk monitoru E-oblast

(pracovní oblast)

Monitor E-oblasti potřebuje právě tak jako monitor 1Z-013B pracovní oblast. Většina paměťových buněk, důležitých pro uživatele je používána pro řízení quick disku.

Právě tak, jako v monitoru 1Z-013B se pro ukládání na disk musí zřídit návěští začátku programu, jak je již známo z povelů pro řízení magnetofonu.

Toto je ale podrobně probráno s komentovaným příkladem v Basicu v kapitole o quick disku.

GRAFIKA S VYSOKOU ROZLIŠITELNOSTÍ

U MZ-800 jsou dvě rozlišitelnosti obrazovky.

a) 320 x 200 bodů = 64000 bodů/bitů = 8 kBytů
b) 640 x 200 bodů = 128000 bodů/bitů = 16 kBytů

Již v základním modulu je možné použít obě rozlišitelnosti, ale opravdu efektivní je teprve přídavná grafická paměť, která umožňuje určení 128000 bodů ve 4 ze 16-ti barev. Ale již se základní verzí MZ-800 máte možnost si vyzkoušet práci s barvami.

Máte následující možnosti:

a) Základní verze MZ-800

V-RAM velikost	barev	rozlišitelnost	část DMD
2x8 kByte	4	320 x 200	00H
16 kByte	2	640 x 200	04H

b) MZ-800 s grafickým rozšířením (+16 kByte)

2x8 kByte	4	320 x 200	00H
		použita stará V-RAM (Frame A)	
2x8 kByte	4	320 x 200	01H

		použita přídavná V-RAM (Frame B)	
4x8 kByte	16	320 x 200	02H
			obě V-RAM
1x16 kByte	2	640 x 200	04H
		použita stará V-RAM (Frame A)	
1x16 kByte	2	640 x 200	05H
		použita přídavná V-RAM (Frame B)	
2x16 kByte	4	640 x 200	06H
			obě V-RAM
MZ-700 systém	8	40 x 25	08H
			1/2 staré V-RAM

Jistě jste se již zajímali o to, co znamená DMD. DMD znamená Displey mode register a slouží pro výběr druhu grafického zobrazení. Přes port CEH se vydává odpovídající hodnota v akumulátoru.
Příklad:

```
6000 3E 00        LD A,00H
6002 D3 CE        OUT (CEH),A
```

Příslušná grafická paměť leží od adresy 8000H až po maximálně BFFFH, když je grafická paměť plně otevřena. Při tom leží všechny paměti paralelně s vyjímkou módu 640 x 200 ve 2 barvách. To znamená, že při způsobu zobrazení DMD 00H leží 2 paměti paralelně u sebe od adresy 8000H po 9FFFH. Vždy jsou 2 barvy paměti. Jestliže tedy leži paralelně 4 paměti, můžeme použít současně 2^4 => 16 barev. Proč současně? Nyní přecházíme ke zvláštnosti v druhu a způsobu, jakým musí být barvy použity. Barvy musí být kódovány přes palety. Zásadně se každé paletě přiřadí jedna ze 16-ti barev. Zde bychom se chtěli zabývat možnostmi, které poskytuje základní verze MZ-800.

Jak již bylo jasné z předcházejícího přehledu, mohou se použít 4 barvy. Nyní tedy nejprve přiřadíme každé paletě jednu barvu. K dispozici jsou následující barvy:

```
0H ... černá
1H ... modrá
2H ... červená
3H ... fialová
4H ... zelená
5H ... zelenomodrá
6H ... žlutá
7H ... bílá
8H ... šedá
9H ... světle modrá
AH ... sv. červená
BH ... sv. fialová
CH ... sv. zelená
DH ... sv. zelenomodrá
EH ... sv. žlutá
FH ... sv. bílá
```

V základním vybavení máme nyní k dispozici 4 barevné palety. Můžeme tedy říci: paleta 0 = sv. červená, paleta 1 = zelená atd. Toto přiřazení se provádí pomocí povelu OUT. Hodnota, která se bude přes port F0H vydávat, se rozdělí na nižší a vyšší 4 bitové slovo. Do vyššího 4 bitového slova se potom napíše číslo palety, do nižšího slova barva. Například chceme paletě 2 přiřadit šedou barvu (šedá = 08H). Na to musí být napsán následující program:

```
1200 3E 28  LD A,28H       2=paleta;8=barva
```

1202 D3 F0 OUT (F0H),A   inicializace palety 2
šedou barvou

Pozn.: Platí pouze pro 2 a 4 barevný režim (ne pro 16 bar. režim)

Velice důležité je ještě aktivování grafiky všeobecně. To se provádí povelem IN A,(E0H). Grafika se znovu deaktivuje povelem IN A,(E1H). Tento způsob zakódování palet má výhody, ale i nevýhody. Nevýhoda je zakódování jako takové; nejdříve se musí inicializovat, což může stát nervy. Naproti tomu se ale musí říci, že se pomocí nového přiřazení dají okamžitě změnit např. všechny zelené body na červené. Důležité je ještě, že při použití grafického rozšíření máte možnost kreslit dva obrazy přes sebe a tím míchat barevné tóny. Pomocí části DMD se potom může vybrat, který obraz má být zobrazen.

Nyní se dostaváme k další kapitole, čtení dat. To se provádí s pomocí RF(READ - format register). Nižší čtyřbitové slovo obsahuje specifikaci detekovaného grafického plann a nebo definici kódu palety, který je potřebný pro READ (čtecí) 0perace. Existují dva způsoby čtení:

1. Single

```
RF=|0|x|x|1/0|1/0|1/0|1/0|
           { p l a n y }
```

Čte z planů specifikovaných jako 1. Je-li nalezeno více planů provádí operaci AND. Není-li specifikován žadný plan vrací hodnotu FFH.(Při čtení z VRAM).

2. Search

```
RF=|1|x|x|A/B|1/0|1/0|1/0|1/0|
           |   {číslo palety}
           |
           |
           FRAME A=0;B=1
```

Vrací bit 1 tam, kde se vyskytuje specifikovaná paleta. (Jinak 0).

Výsledná kontrola pro RF registr se vydává přes port CDH.

Podobně jako pro čtení platí též pro psaní. Zde se používá tomu odpovídající Writhe Format Register. Jako u RF registru odpovídají nižší čtyři bity paletě,kterou se má psát. Bit 4 se používá opět jen tehdy, když chceme současně zpracovávat dva obrazy. (Má vliv pouze na REPLACE a PSET). Bit 5-7 ale představují něco jako mód způsobu provozu. Je 6 způsobů, jak se dá použít.

a) Single Write (0H)
   - - - - - - - - -
   Při tomto způsobu se zeptá na specifikované plany; jiné plany se nemění.

b) Exor (1H)
   - - - - -
   Zde se programem ukládaná data spojují pomocí logické operace exklusivní nebo (XOR), a to v nižším 4 bitovém slově stojící hodnoty; ostatní se nemění.

c) Or (2H)
   - - - -

Funguje jako EXOR, ale s logickou operací nebo.

d) Reset (3H)
- - - - - -
Ve specifikovaných grafických planech se obnoví počá-
teční stav těch bodů, které byly nastaveny daty.

e) Replace (4H)
- - - - - - -
Ukládá data do specifikovaných grafických planů(desek);
ostatní grafické plany obdrží hodnotu 0. REPLACE umož-
nuje psaní v jedné určené barvě palety.

f) Pset (6H)
- - - - -
PSET je pro nás nejzajímavější povel, protože se s ním
mohou barevně nastavit jednotlivé body. Mód palety daný
nižším čtyřbitovým slovem se dosadí na defiovanou adre-
su v grafické paměti.

Je vidět, že není jednoduché se s grafikou s velkou rozliši-
telností spřátelit, a proto napíšeme krátký příklad, který nas
taví bod na obrazovce na libovolnou barvu. Ještě se musí dát po-
zor na to, že programy pro grafiku s vysokou rozlišitelností se
nikdy nesmí psát od oblasti paměti 8000H a výše, protože při
stránkování paměti by se program zhroutil, neboť by najednou pracoval ve špatné
paměti. Jako příklad chceme nastavit bod ve
středu obrazovky-červený. Přitom pozadí má být černé.

```
6000 DB E0    IN  A,(E0H)    aktivace grafiky
6002 3E 00    LD  A,00H      aktivace graf. provozu 320x200
                             pixel
6004 D3 CE    OUT (CEH),A    aktivace
6006 3E 00    LD  A,00H      paleta 0=černá
6008 D3 F0    OUT (F0H),A
600A 3E 03    LD  A,00H      mód single write s paletou 0
600C D3 CC    OUT (CCH),A    paleta 0
600E 21 00 80 LD  HL,8000H   zač. RAM s vys. rozlišitelností
6011 01 00 20 LD  BC,2000H   délka RAM ----------"----------
6014 AF       XOR A          akumulátor=0
6015 77       LD  (HL),A     vymaž paměť pro grafiku s vy-
                             sokou rozlištelností
6016 23       INC HL
6017 0B       DEC BC
6018 78       LD  A,B
6019 B1       OR  C
601A C2 14 60 JP  NZ,6014H   maž dál, když BC není 0
601D 3E 32    LD  A,32H      přiřazuje paletě 3 červenou
601F D3 F0    OUT (F0H),A
6021 3E C3    LD  A,C3H      zpracovávat módem PSET s pale-
6023 D3 F0    OUT (F0H),A    tou 3
6025 3E 01    LD  A,01H      1 pixel přidat do středu
6027 32 00 90 LD  (9000H),A  obrazovky
6029 DB E1    IN  A,(E1H)    obnovení počátečního stavu ob-
                             razovky
602B 76       HALT
```

Tento program se může odstartovat z monitoru pomocí J 6000,
ale potom se musí udělat RESET, protože byl použit povel HALT
(STOP). V tomto módu není možné používat jednoduše písmo i gra-
fiku. Na to se musí vytvořit balík podprogramů. Definice módu

WRITE se provádí pomocí vyššího čtyřbitového slova hodnoty, která se vydává přes port CCH.

PCG grafika.

V této kapitole se budeme zabývat PCG grafikou. PCG znamená Programmable Charackter Generator (programovatelný generátor znaků). Je schopen sám si naprogramovat každý znak sady znaků a tím vytvářet například sadu ozdobných písmen pro psaní a mnohem více jiných věcí. PCG nedělá v principu nic jiného, než povel Basicu PATTERN, ale PCG může být mnohem účiněji, protože znaky mohou být uloženy do paměti a používány místo původních znaků.

Abychom toho dosáhli, nejprve si vysvětlíme hardwerovou funkci PCG. MZ-800 má vnitřní paměť, která leží u 1000H paralelně k hlavní paměti. V této leží sada znaků (tedy 512 znaků), která je uložena v 8 bytovém řádu. MZ-800 má, jak jsme se již zmímili v kapitole MZ-700 Basic, 2 sady po 256 znacích. Tyto znaky tabulky zobrazovacích kódů jsou také všechny ještě jednou vytištěny v dodatku této knihy. Pro PCG grafiku je rozhodující tabulka zobrazovacích kódů, ne ASCII tabulka, která je i v SHARP příručce. Každý znak potřebuje tedy 8 bytů, protože se skládá z 8x8 jednotlivých bodů, takže z 64 jednotlivých bodů. S 8 byty máte k dispozici přesně 64 bitů, takže každý ze 64 bitů odpovídá jednomu bodu.

Písmeno "A" se podle toho skládá z následujících bitů, které přímo vyplývají z nastavených:

Hexadecimální	Dvojkový kód	zobrazení znaku
18	00011000	11
24	00100100	1 1
42	01000010	1 1
7E	01111110	111111
42	01000010	1 1
42	01000010	1 1
42	01000010	1 1
00	00000000	1 1

1 znamená, že příslušný světelný bod je nastaven
0 znamená, že příslušný světelný bod není nastaven

Uložení v paměti je tedy takové, že každý znak obsazuje v PCG paměti 8 bytů. Pro celou sadu znaků je tedy třeba 8x512 bytů a přitom se musí rozlišovat mezi pevnou a dynamickou PCG pamětí. Při inicializaci se sada znaků přenese z pevné (ROM) PCG paměti do dynamické (RAM) pomocí blokového přenosu. Tento postup je bezpodmínečně nutný, protože základní znaky někde definovány být musí, aby počítač po zapnutí vůbec mohl zobrazit písmena a znaky.

Dynamická PCG paměť leží u C000H. Tato skutečnost vás ale vůbec nemusí znepokojovat, protože PCG paměť neovlivňuje celkové rozdělení paměti MZ-800 a vy jako uživatel si toho vůbec nevšimnete. Tohoto bychom chtěli využít a vytvořit program, kterým můžeme sadu sadu znaků měnit. Přitom si musíme být vědomi,ze přímo nemůžeme psát do pevné, ani do dynamické PCG paměti, ale jen do naší normální paměti. Do ostatních oblastí se musíme dostávat pomocí IN(přepínat).Takže tedy program.

```
Program pro nahrání sady znaků na adresu A800H.
- - - - - - - - - - - - - - - - - - - - - - - - -
Adresa    Opkód    Mnemon. kód  Poznámka

A000      D3 E4    OUT (E4H),A
A002      DB E0    IN  A,(E0H)  ;zapnutí PCG módu
A004      21 00 C0 LD HL,C000H  ;adresa PCG dynamické paměti
A007      01 00 10 LD BC,1000h  ;délka dat
A00A      11 00 A8 LD DE,A800H  ;adresa naší paměti
A00D      ED B0    LDIR         ; přenos dat
A00F      DB E1    IN A,(E1H)   ;obnovení počátečního stavu PCG
                                  módu
A011      C3 AD 00 JP 00ADH     ;skok do monitoru

Program pro inicializaci PCG sady znaků do dynamické PCG paměti.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
A014      D3 E4    OUT (E4H),A  ;
A016      DB E0    IN A,(E0H)   ;zapnutí PCG módu
A018      21 00 A8 LD HL,A800H  ;adresa naší paměti
A01B      01 00 10 LD BC,1000H  ;délka dat(4096 bytů)
A01E      11 00 C0 LD DE,C000H  ;adresa PCG paměti
A021      ED B0    LDIR         ;přenos dat
A023      DB E1    IN A,(E1H)   ;obnovení počáteč. stavu PCG mód
A025      C3 AD 00 JP 00ADH     ;skok do monitoru
```

Oba tyto programy můžete vložit z monitoru (jednoduše - opcode s pomocí povelu M, a s JA000 resp. JA014 odstartovat). Od adresy A800H leží tedy data sady znaků v hexadecimálním formátu. Když vložíme J A014, přeneseme data na A800H. Nyní můžeme sadu znaků měnit. Příklad pro změnu znaku SPACE:

Protože každý znak zabírá 8 bytů, je znak 0 (SPACE) nyní uložen v paměti od A800H do A807H. Znak 1 (A) je uložen v pamětových buňkách A808H až A80FH.

a) JA000
b) MA800 00 FF (CR)
c) JA014

Ihned po vložení se změní vzhled obrazovky. Objeví se černé pruhy, které téměř představují FFH. Chtěli bychom ještě provést takovou změnu pro každý znak.

Následující vzorec výpočet adresy v paměti, na které je libovolný znak, respektive kde jsou uloženy jeho hodnoty.

Adresa=A800H + 8 x číslo měněného znaku

Pro znak DFH (223 decimálně) tedy platí

Adresa=A800H + 8 x DFH = AEF8H

Nyní se dostáváme ke zvlástnosti v uložení PCG hodnot. Tyto jsou totiž v paměti napsány zrcadlově obráceně. Hned to prakticky vyzkoušíme tím,že znak DFH což původně byla zmije, nyní změníme na třešni. Musíme tedy nejdříve definovat třešni.

12345678	bitový vzor(zrcadlově obráceně)	hexadecimálně
*** 1	00000111	07 H
**** 2	00001111	0F H
* * 3	00010010	12 H
* * 4	00100010	22 H
* * ** 5	11010010	D2 H
** **** 6	11110110	F6 H

```
* ** **  7        01101101                    6D H
 **      8        00000110                    06 H
```

Nyní vložíme z monitoru:

```
MAEF8
AEF8  00 07 (CR)
AEF9  03 0F (CR)
AEFA  13 12 (CR)
AEFB  2A 22 (CR)
AEFC  2A D2 (CR)
AEFD  AA F6 (CR)
AEFE  44 6D (CR)
AEFF  00 06 (CR)
SHIFT&BREAK

JA014
```

Když necháme znak DF ojevit na obrazovce vidíme naši třešni.

```
MD000
D000  00 DF (CR)   (Objevení se třešně na obrazovce)
SHIFT & BREAK
```

Vypočítat, kde se od A800H nachází správná adresa, je ale relativně těžké. Nechme to proto prostě dělat počítat. Napíšeme program, který nahrazuje naši rovnici. Adresa = A800H+znak*8.

```
A100  3E xx       LD  A,xx          ;xx=hodnota znaku
A102  6F          LD  L,A           ;uloží se do HL
A103  26 00       LD  H,00          ;
A114  29          ADD HL,HL         ;
A115  29          ADD HL,HL         ;x8
A116  29          ADD HL,HL         ;
A117  01 00 A8    LD  BC,A800H      ;naše PCG paměť
A11A  09          ADD HL,BC         ;
A11B  CD BA 03    CALL 03BAH        ;zobrazení adresy
A11E  C3 AD 00    JP  00ADH         ;skok do monitoru
```

Napíšeme tedy na adresu A101H náš znak, který chceme změnit a odstartujeme program pomocí J A100.

Potom se na obrazovce objeví adresa, kde je v paměti uložen znak s číslem xx (v zobrazovacím kódu). Při vložení by se potom objevilo A800H.

Ještě něco všeobecného. PCG znaky nejsou jednotlivě, tedy bod po bodu, barevně nastavitelné. Je tedy možné nastavit jen barvu a barvu pozadí pro celý znak. Toho dosáhneme pomocí grafiky s vysokou rozlišitelností. Zde PCG grafika imponuje svojí rychlostí a relativně jednoduchou použitelností.

Programovatelný zvukový generátor (PSG)

MZ-800 má oproti mnoha jiným počítačům velkou výhodu ve velmi velké a komplexní zásobě povelů v Basicu. Při tom bylo pamatováno i na vnitřní zvukový čip, který muže být v určitých mezích programován bez problémů v Basicu.

Na tomto místě budiž řečeno, že v příručce OWNER'S MANUAL je chyba v syntaxi. Na straně 6-83 je pod povelem NOISE napsáno, že je možno použít dvou šumových tónů najednou, jestliže se

jednotlivé parametry šumových tónů oddělí čárkou. Vnitřní PSG ale může vytvářet pouze jedno šumění a tak není divu, že se podle návodu k povelu NOISE nedá dosáhnout požadovaného efektu. Je tedy současně možné pouze jedno šumění. Je to tzv. bílý šum (podle SHARP).

Později se budeme blíže zabývat ještě jedním druhem šumu, ale ten nemůže být vyvolán pomocí povelu NOISE.

Vycházíme z toho, že pro uživatele je programování PSG pomocí povelů v Basicu běžné. Nyní se budeme zabývat tím, jak se dá PSG řídit přímo pomocí povelů OUT. To se může provádět jednak z Basicu povelem OUT@, jednak z programu ve strojovém jazyce.

```
Např.:    LD A,3FH
          OUT (F2H),A
```

Výstupní port pro PSG je F2H.

Povelem OUT F2H tedy může být zpřístupněn PSG. Aby se ale mohl vytvořit tón, musí se ještě ledacos udělat, protože PSG má přesný předpis, který udává, kde musí být který bit nastaven eventuelně nenastaven, aby mohl hrát určitý tón.

Kromě toho je nezbytné vyslat více bytů po sobě (sériově), aby se PSG mohla sdělit frekvence a hlasitost. Tak obsahují první dva byty data o výběru jednoho ze tří tónových generátorů a o frekvenci. Výběr tónového generátoru a frekvence se tedy dělá dvěma byty, které musí být na PSG vyslány sériově. Na první pohled to vypadá velmi komplikovaně, protože frekvence musí být udána v 10 bitech. Protože ale pro 1 byt může být přeneseno pouze 8 bitů, je nutně těchto 10 bitů rozdělit na dva byty.

Přitom musí být bity rozděleny následovně:

BYTE 1

Sedmý bit prvního bytu by měl obsahovat 1. Tato "1" ukazuje PSG, že se jedná o první byte. Šestý, pátý a čtvrtý bit udává, který třítónový generátor má být osloven. Třetí, druhý, první a nultý bit udává nižší hodnotu frekvence.

BYTE 2

Sedmý a šestý bit se nastaví na 0. 5-tý,4-tý,3-tí,2-hý,1-ní a 0-tý bit obsahují vyšší hodnotu frekvence.

Můžeme si tedy představit 6 bitů z bytu 2 a 4 bity z bytu 1 seřazeny za sebou do řady. Potom by to vypadalo asi takto:

```
BIT:  5 4 3 2 1 0 3 2 1 0
      =========== =======
         Byte 2    Byte 1
```

Nyní se ale nabízí otázka, jak mají být bity vypočítány a nastaveny. I toto je jednodušší než by se mohlo zpočátku zdát.

Musíme dávat pozor na to, že deset bitů, které představují frekvenci, neudává přímo hodnotu frekvence v Hz.

Následující funkce objasňuje desetibitové (F10) číslo, které musí být zvoleno, aby generátor hrál odpovídající frekvenci (FR).

$$F10 = 11094/(FR/10)$$

Nakonec musíme číslo frekvence F10 přepočítat do 10 bitů. Nyní příklad:

Chceme spočítat 10 bitů,které jsou nutné k tomu,aby se vytvořil tón o frekvenci 442 Hz. (442 Hz je přibližně frekvence, kterou uslyíte zvednete-li telefon a uslyšíte oznamovací tón).

Nejprve dosadíme 442 do rovnice jako FR.

Tedy:

F 10=11094/(442/10) dává 250,995

Nyní musíme zaokrouhlit na celá čísla.

F 10=INT(F 10+.5)

Čili dostaneme pro F 10=251

Těchto 251 nyní chceme přepočítat na 4 bitovou a 6 bitovou část. 6-ti bitovou část dostaneme, vydělíme-li číslo F 10 šestnácti (což odpovídá 2^4).

Tedy:

LSB=F 10/16 dává 15,6875

I teď nás zajímá pouze číslo před desetinou tečkou. Proto jej vypočítáme podle následujícího vzoru:

LSB=INT(LSB) dává 15

Šesti bitová část je tedy 15 nebo 2^0+2^1+2^2+2^3=1111
Těchto 1111 je ve dvojkové soustavě.

Jestliže chceme nyní spočítat 4 bitovou část, odřízneme desetinou část a výsledek (v našem případě 0,6875) vynásobíme opět 16.

Tedy:

MSB=FRAC(LSB)x16 dává 11

Čtyřbitová část je tedy 11 nebo 2^0+2^2+2^3=1011
Těchto 1011 je ve dvojkové soustavě.

Nyní jsme spočítali jak 4-bitovou část, tak i 6-ti bitovu část a to:

LSB(4 bit)=11 (decimálně) nebo 1011 (dvojkově)
MSG(6 bitů)=15 (decimálně) nebo 001111 (dvojkově)

Jestliže se vám tento výpočet zdá příliš komplikovaný, můžete použít následující program v Basicu, který si nyní objasníme. Na začátku (po startu pomocí RUN) se program zeptá na frekvenci v Hz, která má být rozložena. Výsledky jsou obsaženy v proměných B4(4 bitová část) a B6(6-ti bitová část).

```
10 CLS
15 INPUT"ZADEJTE FREKVENCI V Hz";FR
20 F0=11094/(FR/10)
30 B6=INT(F0/16)
40 B4=(F0+.5)MOD 16
50 IF FRAC(F0)=>.5F0=INT(F0)+1:ELSE F0=INT(F0)
55 F0=INT(F0+.5)
60 PRINT"6-TI BITOVA CAST";B6
70 PRINT"4 BITOVA CAST";B4
80 PRINT"F0 :";F0
90 PRINT:PRINT:GOTO 15
```

Vypočítejte nyní ručně LSB(4 bitové) a MSG(8 bitové) pro frekvenci f=109 Hz.
Potom si přezkoušejte vypočtený výsledek pomocí výše uvedené-

ho programu. Pro 6-ti bitové číslo (MSB) vychází 63, pro 4 bitové číslo (LSB) vychází 10. F0 je 1018.

Nyní můžete frekvence posílat na PSG, ale ještě mu musíte říci, kterým ze tří generátorů má funkci vytvořit.
Nato přičteme k LSB(4 bitové číslo) v následujícím seznamu uvedené číslo:

Tónový generátor 1: 128
Tónový generátor 2: 160 Tónový generátor 3: 192

Chceme-li tedy spustit tónový generátor 2,přičteme k B4(LSB) číslo 160. B6(MSB) zde zůstává nezměněn. Nyní to bude vypadat takto:

```
B4=B4+160  nebo  B4=11+160    dává 171
```

Nyní musíme tyto dva byty (B4 a B6) poslat na PSG. K tomu nám poslouží OUT, který můžeme použít i v Basicu např. formou:

```
OUT@ 242,B4     242 má stejný význam jako
OUT@ 242,B6     hexadecimální hodnota F2
```

Jestliže tedy chceme vytvořit tónovým generátorem druhý tón o frekvenci 442 Hz, jsou zapotřebí dva povely OUT a to:

```
OUT@ 242,171
OUT@ 242,15
```

Nyní ještě neuslyšíme žádný tón,protože jsme ještě nedefinovali hlasitost. K tomu potřebujeme ještě jeden povel OUT. Povel OUT pro nastavení tlumení si prosím vyberte z následující tabulky. V tabulce jsou uvedeny i tlumící povely pro šumový generátor o kterém budeme mluvit dále. Pro to co nyní popisujeme, nehraje žádnou roli.

Tlumení	Generátor 1	Generátor 2	Generátor 3	Šum
0	144	176	208	240
1	145	177	209	241
2	146	178	210	242
3	147	179	211	243
4	148	180	212	244
5	149	181	213	245
6	150	182	214	246
7	151	183	215	247
8	152	184	216	248
9	153	185	217	249
10	154	186	218	250
11	155	187	219	251
12	156	188	220	252
13	157	189	221	253
14	158	190	222	254
15	159	191	223	255

Když chceme náš tón o frekvenci 442 Hz nechat zaznít v plné hlasitosti, zadáme tlumení 0. To znamená, že má zaznít netlumený tón.

```
OUT@ 242,176     176 odpovídá tlumení 0 pro tónový generátor 2
```

Nyní už musí náš tón zaznít. Samozřejmě můžeme během tónu tlumení měnit.

Uděláme to následujícím programem:

```
10 FOR I=176 TO 191
20 OUT@ 242,I
30 WAIT 500
40 NEXT I
50 FOR I=191 TO 176 STEP-1
60 OUT@ 242,I
70 WAIT 500
90 NEXT I
```

Tón bude pomalu tišší, a potom znovu hlasitější. Provádí se to takto: na řádce 10 necháváme I běžet od 176 do 191. To jsou hodnoty tlumení tónového generátoru 2 pro povel OUT ve stoupajícím pořadí. Na řádce 20 potom toto tlumení jde přímo na PSG. Aby se rozdíl v hladitosti dal lépe vnímat, zpomalujeme program na řádce 30 povelem WAIT 500. Díky tomuto povelu čeká program půl vteřiny, a potom zvyšuje stupeň tlumení skokem na NEXT I na řádce 40.

Jestliže byla dosažena nejvyšší hodnota tlumení (tónový generátor vypnut), začne další smyčka stejným způsobem tlumení ubírat.

Tomuto tlumení jako funkci času se všeobecně říká obalová křivka. Můžeme ji zvýraznit formou diagramu, ve kterém zakreslíme závislost hlasitosti na čase.

V posledním programu by obalová křivka vypadala následovně:

To znamená, že tón bude nejprve tišší a potom hlasitější. Nyní tedy můžeme obalové křivky tónů libovolně měnit.

To můžeme sice dělat i v BASICu, ale se syntaxí tohoto jazyka můžeme volit pouze 9 různých obalových křivek. Tyto možné obalové křivky jsou také znázorněné v uživatelské příručce (OWNER'S MANUAL) na straně 6-80.

Pomocí povelů OUT můžeme ale vytvořit úplně libovolnou obalovou křivku. Získali jsme tedy značnou výhodu oproti programování se sadou povelů BASICu.

Nyní můžeme vytvářet tóny na třech tónových generátorech o navzájem nezávislé frekvenci a hlasitosti. Dále se budeme zabývat tím, jak může být detekován generátor šumu.

Nejlepší možnost detekovat šumový generátor je detekovat třetí tónový generátor tou frekvencí, kterou má kmitat šumový generátor. K tomu musíme ale nejprve vypnout hlasitost tónového generátoru 3. To uděláme tak, že zvolíme maximální tlumení (povelem OUT@ 242,223).

Nakonec můžeme již popsaným způsobem spočítat odpovídající bity frekvence. Předpokládejme, že jsme B4(LSB) a B6(MSG), tedy 4 a 6 bitová čísla, spočítali. Nyní přičteme 192 (tónový generátor 3) k B4 (4 bit. číslo), a potom vydáme frekvenci v pořadí LSB,MSB. To všechno seprovádí podle již popsaného postupu.

Nyní musíme PSG sdělit, že má frekvenci inicializovanou v ge-

nerátoru 3 považovat za šumovou frekvenci. Toho docílíme povelem OUT@ 242,227. Pomocí tohoto povelu se vytvoří synchronní šumění.

Dále můžeme pomocí PSG vytvořit i bílý šum, a to tím, že místo OUT@ 242,227 nyní vložíme OUT@ 242,231. Potom může být hlasitost měněna, jak je uvedeno v tabulce hlasitosti pod šuměním.

Také zde může být samozřejmě vytvořeno mnoho libovolných obalových křivek. Protože zde můžeme vytvořit synchroní šumění, máme velkou výhodu oproti programování v BASICu povelem NOISE, který může vytvořit pouze bílý šum. Vyplatí se tedy řídit zvukový generátor povely OUT, protože získáváme možnosti, které se pomocí normálních povelů v BASICu nedají realizovat.

[POZN. Tyto možnosti se dají realizovat pomocí příkazu, který v obecném manuálu není uveden. Je to příkaz SOUND=(X,Y). Je tedy nesmysl řídit zvuk příkazy OUT, protože to jde lépe a rychleji pomocí tohoto specifického SOUNDu a není-li to nutné, použijeme příkazů MUSIC a NOISE, protože pracují přes přerušení.]

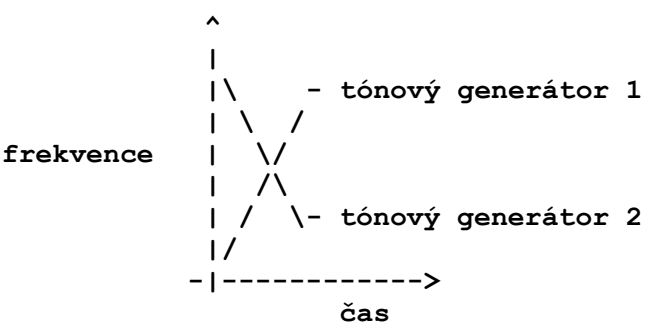
Zvláštnosti:

Mohou být zásadně hrány pouze tři tóny polifonně (současně). Přitom nemůže být hrán více než jeden šumový tón. Můžeme tedy hrát současně maximálně tři normální tóny nebo dva tóny a jeden šumový tón.

[POZN. Výhodné pro BASIC MZ-700,KU-BASIC,FORTRAN,Assembler apod. Ne pro BASIC MZ-800.]

Kruhová modulace

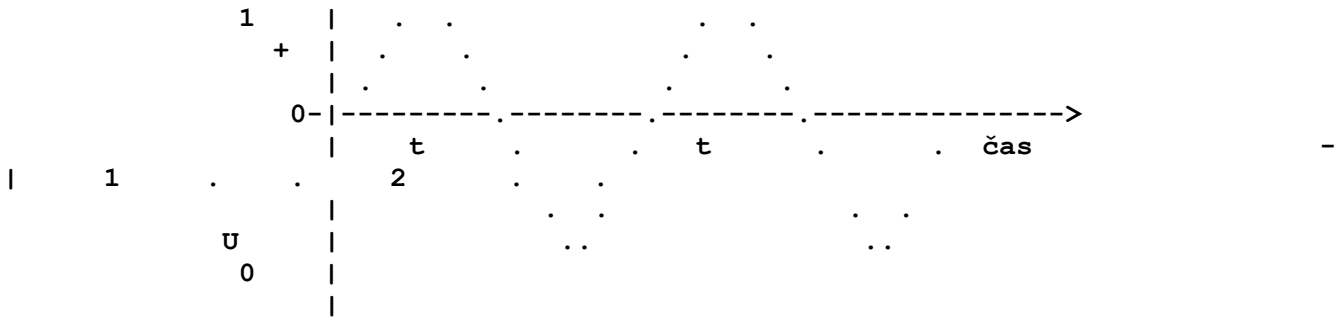
Kruhová modulace se dá vytvořit tím, že se nechá kmitat např. tónový generátor 1 frekvencí od 200 do 1000 Hz a ve stejnou dobu tónový generátor 2 frekvencí od 1000 do 200 Hz. To tedy znamená, že frekvence (výšky tónů) běží protismyslně, jak je zřejmé z následujícího diagramu:



Fázování

Abychom mohli vysvětlit pojem fázování, museli bychom odbočit velice daleko - na samotnou stavbu kmitu. Vysvětlíme si tedy pouze na příkladu, co znamená fázování.

Frekvence, kterou slišíme, není nic jiného, než kmitání. Jestliže kmitání probíhá asi tak, jak je zobrazeno na následujícím diagramu, jedná se o kmitání sinusové.



Na základě této křivky může být ledacos vyvětleno. Rozdíl mezi U1 a U0 odpovídá hlasitosti tónu. Čím vyšší je tedy U1, tím hlasitější je tón.

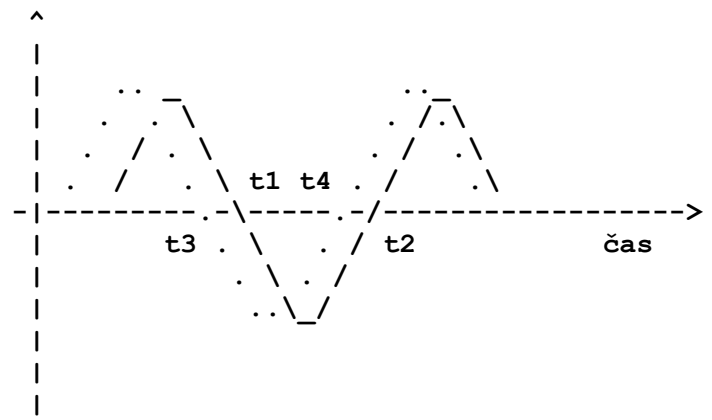
Frekvenci odpovídá čas, který proběhne mezi t1 a t2. Jestliže tyto hodnoty leží dále od sebe, nezmění funkce tak rychle znaménko. Frekvence je potom nižší. Jestliže hodnoty leží blíže u sebe, je frekvence vyšší než v našem příkladě.

Všeobecně se dá říci, že zvýšení frekvence vnímáme jako zvýšení tónu. Označíme-li čas mezi t1 a t2 jako "T" a frekvenci jako "t", vychází následující vzorec:

$$f=1/T \quad \text{nebo} \quad T=1/f$$

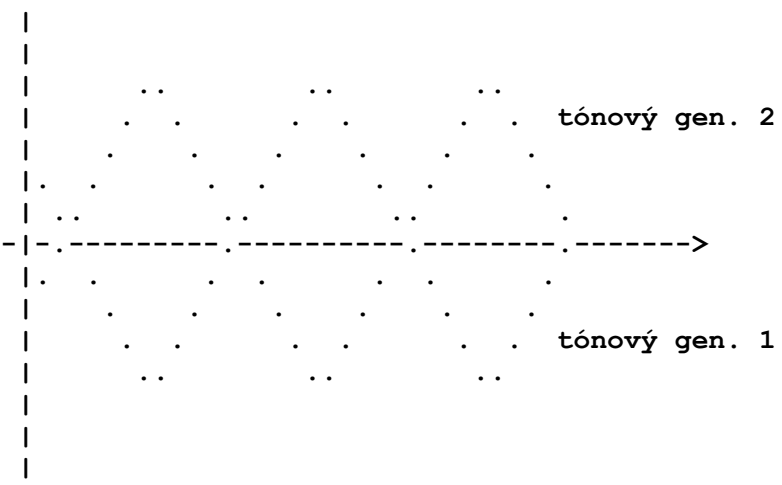
Jestliže tedy vytvoříme tón o frekvenci 400 Hz, potom víme, že odstup mezi t1 a t2 je nyní 1/400 Hz, tedy 0.0025 sekund. Abychom nyní vytvořili fázování, můžeme využít vlastnost, která není popsána v žádné uživatelské příručce. Spustíme dva tónové generátory se stejnou frekvencí. Tzn., že při obou tónech je odstup mezi t1 a t2 stejný.

Tónový generátor nechá křivku začínat v přesně stejnou dobu. A tak dostáváme následující diagram.



Jak vidíte, je doba, která uběhne mezi tím, než křivka proběhne nulou t1 a t2 tónového generátoru 1 přesně rovna době t3 a t4 tónového generátoru 2. Máme tedy stejnou frekvenci. Fáze (průběhy frekvencí) jsou ale časově posunuty, a to o rozdíl mezi t1 a t3. Tomuto efektu se říká fázování. Protože se tím získá velice zajímavý zvuk, vyplatí se to vyzkoušet. Posluchač totiž vnímá v každé chvíli součet kmitů.

Posunutí fází se ale mění také během hraní dvou stejných tónů, takže dochází ke změně součtové křivky a tím také ke změně hlasitosti. Myslitelný by byl také ten případ, že by oba kmity ležely u sebe následovně:



Potom by byl součet obou tónů v každé době roven nule. Neslyšeli bychom tedy žádný tón. To se také skutečně někdy stává.

Abychom mohli všechno vyzkoušet a tónový generátor se mohl optimálně detekovat, je jak vidíme nutné velké množství programátorské práce.

Proto na tomto místě doporučujeme program, který bravurně řeší všechny možnosti v tvoření hudby. Jmenuje se SYNTHY-800 a je k dostání v každé prodejně BBG.

Pomocí programu SYNTHY-800 může uživatel pomocí menu zadávat celé posloupnosti tónů, ve kterých může být kdykoliv volně vložena přes obrazovku jak frekvence, tak hlasitost. Grafická křivka je zobrazena na obrazovce pomocí dialogu s uživatelem.

Může být vytvořen libovolný tón a může být uložen na libovolné paměťové médium (kazetu,RAM disk,quick nebo floppy disk). Tóny nebo řady tónů mohou být potom znovu nahrány do paměti a měněny nebo zlepšeny. Mohou být najednou hrány až tři hudební sekvence. Samozřejmě je detekovaný i šumový generátor. I úplní začátečníc jsou s tímto programem schopni vytvářet dobře znějící hudbu. Člověk tedy nemusí být zvláštně muzikální, aby tento program mohl obsluhovat. Může být vytvořeno přes 20 000 různých tónů. Hotové posloupnosti tónů s jedním nebo více tónovými generátory mohou být vkládány do vlastních programů v BASICu i ve strojovém jazyce.

Tento rogram nabízí v oblasti tvoření hudby opravdu něco nového a otvírá bohaté možnosti. Program byl dokončen krátce před vydáním této knihy a může proto být u nás zakoupen.

Následující dva programy v BASICu vám přináší trochu hudby.

```
1 TEMPO 6
10 FOR JT=1 TO 2
20 FOR I=1 TO 10
30 READ A$,B$,C$
35 MUSIC A$;B$;C$
40 NEXT I
50 RESTORE "ENTRY"
60 NEXT JT
70 END
100 DATA O3V15S0M10
110 DATA O2V13S0M10
120 DATA O2V11S0M10
160 REM
170 DATA R9
```

```
180 DATA D5E3#FA#F1EM4D3E
190 DATA R9
220 REM
230 DATA R9
240 DATA #F#FEEDR6
250 DATA R9
290 LABEL "ENTRY"
299 REM
300 DATA M10#F3AA5M20A7
310 DATA M6#F5M13A7M4+#C0+D2A3
320 DATA M6D5M20#F6M4R5
325 REM
330 DATA M10#F3AA5M20A7
340 DATA M6D5M20#F7M4R5
410 DATA M6#F5M13A7M4+#C0+D2A3
415 REM
420 DATA M10R3BAABBAA
430 DATA R3M10B3A5M4B3RAA
440 DATA V15-D5-D-D-D
470 REM
480 DATA #F3ABGM18A7M10
490 DATA #F3ABGAM10D1E#F5
500 DATA -D-D-D-DV12
530 REM
540 DATA R3#FAA#FAAA
550 DATA M5R3DM15#F5M10R3#F3AA
560 DATA M5R3M15#F5M10R3D#F#F
590 REM
600 DATA B3B1+#C+D3EA7
610 DATA R5M20D5D7M10
620 DATA R5M20-B5-A7M10
640 REM
650 DATA R3+DA#FEBBB
660 DATA M3R2D5M10R0R3M20E7M10
680 DATA M3R1A5M10R4M20B7M10
710 REM
720 DATA +D3A1#FE5M14#F7M10
730 DATA +D3R5#G0A2+D3+B+A+#F1+E
740 DATA A3R3R7
770 REM
780 DATA R9
790 DATA O3M10D5E3#F3M4A3M10#F1ED3M4E3
800 DATA R9
830 REM
840 DATA R8+D5
850 DATA M3#F3#FEEDR3+A5
860 DATA M4+D3+DAA#FR3+#F5


---------


10 TEMPO 5
20 DIM M$(3,10)
30 FOR J=1 TO 3
40 FOR I=1 TO 9
50 READ M$(J,I)
60 NEXT I,J
70 FOR I=1 TO 10
80 MUSIC M$(1,I);"O2"+M$(2,I);M$(3,I)
90 NEXT I
```

```
100 DATA O3V15S0M10
110 DATA +C5B1+C+D+CBA+C3+C1A+C5B1+C
120 DATA A1GEFG6R1FEDEFGAG6
130 DATA A1BAGFEDEDC5C1DM6E3FM10
140 DATA D5M20G9R4M10+D6+C1BAB+C+D+CB4
150 DATA +C1BA+C1BAF4M5F1FF3A+C1ABD
160 DATA F3F1FF3AM10B1GAF
170 DATA D3D1CD6M5D1DD3FM10A1FGED3D1C
180 DATA D6D1CD3E1FM15G7M10R1FEDC9  190 REM
200 DATA V10S0M10
210 DATA E5D1CDCDFE3G1FE5D1C
220 DATA F1ECDE6R1FEGCDEFE6
230 DATA F1DECDGBGBG5C1GM6A3GM10
240 DATA B5M20E9R4M10B6G1GFDEGEG4
250 DATA E1GFEGFA4M5D1DA3FE1FGG
260 DATA A3A1AC3FM10G1GFA
270 DATA G3F1EB6M5D1FG3AM10C1GECB3G1E
280 DATA B6G1ED3G1FM15E7M10R1DGGG9
290 REM
300 DATA O1V1S0M5
310 DATA C3C1C0CC3C1C0CC3CCE1E0EE3E1E0E
320 DATA C3C1C0CC3C1C0CC3CCC1C0CC3C1C0CG1G0GG1G0G
330 DATA C3C1C0CC3CCC1C0CG3G1G0GG1G0G
340 DATA C3C1C0CC3C1C0CG3GGG1C0CC3C1C0CC3C1C0C3C1C0CC3C1
    C0CC3C1C0C
350 DATA G3G1G0GG3G1G0GC3CCC1C0C
360 DATA C3C1C0CC3CCC1C0C
370 DATA G3C1E0EE3E1E0GG3GCG1C0CC3C1C0C
380 DATA G3G1G0GG3G1G0GG3GCG1C0CC3C1C0CC3C1C0CC3C1C0CC1C0C
```

QUICK DISK MZ - 1F11

Vše v následující se vztahuje na Quick disk model MZ - 1F11. Nebudeme se zde blíže zabývat řízením Quick disku z BASICu, protože je dostatečně přesně popsáno v příruče Sharp. Mnohem více Mnohem více bychom chtěli zdůraznit možnosti použití Quick disku v programech napsaných ve strojovém kódu.

Všeobecně:

Pomocí od E000H do FFFFH ležící ROM(jen u MZ-800 a MZ-700, když je připojen Quick disk) můžete nahrát na disk maximálně 34 souborů o max. 64 Kbytů. Systémem je sice na disk připuštěno až 50 souborů, ale vyrovnávací paměť pro adresář stačí jen pro 34 souborů.Ale vy se pravděpodobně nikdy nedostanete do situace,aby jste museli nahrávat více než 34 souborů na jeden Quick disk. Tyto soubory leží na Quick disku jako v dlouhém řetězu bytů, a ne jako na normálních floppy discích, kde jsou rozděleny na sektory a stopy. To má, ale bohužel za následek, že žádné jednotlivé soubory nemohou být smazány nebo přehrány. Máte pouze možnost Quick disk nově formátovat a tím smazat všechny soubory.

Standartní program E010H provádí kompletní řízení Quick disku ale předtím musí být vždy zadáno příslušným kódem jakou operaci má Quick disk provádět.

Příslušná operace se vybírá řídícím kódem, který se předává na paměťové buňce 1130H.

Řídící a kontrolní kódy pro rutinu E010H.

01 READY CHECK - kontroluje,zda je Quick disk připraven k nasazení. Jestliže není, je indikátor přenosu nastaven na 1.
02 FORMAT - formátuje Quick disk
03 READ - čte blok dat (od záhlaví)
04 SAVE - ukládá blok dat na quick disk (od záhlaví)
05 HEADPOINTCLEAR - nastavuje bod záhlaví na počáteční hodnotu (inicializace)
06 Vypíná motor

Tento kód musí být předán na buňku paměti 1130H, než se zavolá startovací program E010H.

Bod záhlaví představuje hexadecimální kód, který se předává na adrese 113DH, a tzv. označuje bod na quick disku, odkud má číst nebo zapisovat. Pro READ(kód=3) a SAVE(04) je tento bod interpretován jako počáteční bod a po čtení/zapsání je nově nastaven.

Standartní program předává v indikátoru přenosu zpět, zda došlo k chybě. Jestliže je Cy=1, je v akumulátoru popsán druh chyby.

Akumulátor	Druh chyby	Adr.zprávy
28H	not found err(soubor nebyl nalezen)	F290H
39H	bad disc err(disk není v pořádku)	F319H
2EH	write protect(disk s ochranou proti psaní)	F2D6H
32H	not ready err(quick disk není připraven)	F2E7H
35H	no file space err(už žádné místo na disku)	F2F4H
36H	unformat err(disk není formátován)	F309H
2AH	already exit err(jméno souboru již existuje)	F2C4H
33H	too many files err(příliš mnoho souborů na disku)	F2A2H
00H	BREAK(přeruší pomocí SHIFT+BREAK)	F329H
Zbytek znamená - hard error		F2B8H

Od příslušné adresy zprávy je v ROM uloženo odpovídající chybové hlášení v ASCII formátu a může být vytištěno na obrazovku následujícím programem:

```
LD    DE, `adresa zprávy`
CALL  0006H
CALL  0015H
```

Stavba záhlaví quick disku.

Záhlaví quick disku je velmi podobné záhlaví u startovacích programů pro kazetu, a leží v oblasti od 10F0H do 112FH.

Význam dat jednotlivě:

10F0H	kód souboru: 01H u programů ve strojovém jazyce	"OBJ"
	02H u programů v BASICu	"BTX"
	03H u souboru dat v BASICu	"BSD"
	04H u rozptýlených souborů BASICu	"BRD"
	05H	"RB"
	07H	"LIB"
	0AH	"SYS"
	0BH	"GR"
	94H u textových souborů z TEXY	"TXT"
	95H u souborů LISPu	"LSP"

Poznámka:
kódy 94H a 95H adresář monitoru nepozná, ale označuje je jako `???` soubor.

10F1H do 1101H	jméno souboru s koncovou značkou(0DH)
1104H	délka následujícího souboru
1106H	počáteční adresa bloku dat
1108H	adresa autostartu programu

Ostatní adresy oblasti záhlaví jsou nevyužity.

Další užitečné podprogramy v ROM.

EF27H	READY check	odpovídá E010H s (1130H)=01
F25FH	Headpoint clear	odpovídá E010H s (1130H)=05
E2E8H	Vypnutí motoru	odpovídá E010H s (1130H)=06

EEF7H — Filesearch

Hledá na disku soubor, jeho jméno je stejné jako v adrese 11A3H. Jestliže jméno v 11A3 začíná 0DH (čili nebylo žádné jméno vloženo), vezme první soubor z disku. Nastaví se bod záhlaví a záhlaví souboru se uloží od 10F0H po 112FH. Při tomto se nebere v úvahu kód souboru(stojí v 10F0H). Jestliže potřebujete, můžete ještě přezkoušet kód souboru. Filesearch musí být inicializován s F25FH (Headpoint clear). Při novém zavolání filesearch bez nové inicializace se hledá dále od místa, kde skončilo poslední hledání. Jestliže filesearch narazí na chybu,

zastaví se na Cy=1.

Pozor: Mezi vyvoláním více filesearch nesmí mez nimi uplynout příliš dlouhá doba, protože by z technických důvodů mohly být některé soubory přeskočeny.Čas nestačí nnapříklad k zobrazení jména souboru na obrazovce. Proto je pro adresář také zapotřebí vyrovnávací paměť.

EFA2H File end search

Hledá konec dat na quick disku,nastavuje bod záhlaví a zkoumá, zda eventuelně již není soubor s tímto jménem (v záhlaví od 10F1H maximálně 16 znaků + CR),nebo zda na disku není příliš mnoho souborů (max 50).Podprogram file end search se provede úspěšně bez chyby pouze tehdy,je-li nastaven Cy=1 a A=28H (not found).

Nyní k nejdůležitějším programům LOAD a SAVE, které se oba vyvolávají přes E010H.

LOAD

Jak jsme se již nahoře zmínili, je LOAD inicializováno kódem 03 v (1130H). Mimo kódu a bodu záhlaví, který byl nastaven pomocí file search, potřebuje LOAD ještě startovací adresu oblasti, do které má být soubor nahrán, delku souboru k nahrání a další kód, který je zapotřebí k rozpoznání datového souboru (k rozlišení od záhlaví).

Data quick disku:

start. adresa na (1132H)
délka na (1134H)
kód 03H na (1130H) 2.kód 01H na (1131H)

SAVE

Save je popsán kódem 04 v 1130H a kódem 04 v 1131H
Kromě toho ještě musí být předána délka záhlaví 0040H, jeho startovací adresa 10F0H, startovací adresa oblasti, která má být nahrána a její délka. Následující tabulka popisuje, kde musí být data předána.

kód	na (1130H)	= 04
2.kód	na (1131H)	= 04
zač.záhlaví	na (1132H)	= 10F0H
délka z.	na (1134H)	= 0040H
začátek DATA	na (1136H)	
délka DATA	na (1138H)	

DATA znamená oblast k nahrání.

Teď ještě příklady k formátování, LOAD a SAVE.

Formátování:

```
      CALL  EF27H       ;test je-li QD připraven
      JP    C,ERROR           ;jestli Cy=1 error(not ready err)
      LD    A,02H       ;kód pro formátování
      LD    (1131H),A   ;kód odložit
      CALL  E010H       ;hlavní program QD
```

```
      JP    C,ERROR           ;je-li Cy=1 tab.chyb.hlášení

LOAD:

      CALL  EF27H       ;test je-li QD připraven
      JP    C,ERROR           ;je-li Cy=1 error(not ready err)
      LD    DE,EE65H    ;text "jméno souboru ? "
      CALL  0006H       ;posun řádky
      CALL  0015H       ;výtisk textu
      CALL  0006H
      LD    DE,11A3H    ;vyrovnávací pamět pro jméno
                        ;souboru
      CALL  0003H       ;vložit řádku
      LD    A,(DE)            ;1 znak ve vyrovnávací paměti
      CP    1BH         ;je-li break, pak error
      CALL  F25FH       ;headpoint clear
      CALL  EEF7H       ;hledání souboru
      LD    HL,(1104H)  ;délka souboru (v záhlaví)
      LD    (1134H),HL
      LD    HL,(1106H)  ;počáteční adresa souboru
      LD    (1132H),HL
      LD    HL,0103H    ;kód 03 a 2. kód 01
      LD    (1130H),HL
      CALL  E010H       ;hlavní program QD
      JP    C,ERROR
      CALL  E2E8H       ;vypnutí motoru
      RET

SAVE

      CALL  EF27H       ;test zda QD připraven
      JP    C,ERROR
      LD    DE,EE65H    ;text "jméno souboru ? "
      CALL  0006H
      CALL  0015H
      CALL  0006H
      LD    DE,11A3H    ;vstupní vyrovnávací paměť
      CALL  0003H       ;vložené řádky
      LD    A,(DE)
      CP    1BH
      JP    Z,ERROR           ;viz nahoře
      EX    DE,HL       ;vyr.paměť 11A3H do HL
      LD    DE,10F1H    ;vyr.paměť pro jméno souboru
                        ;v záhlaví
      LD    BC,0010H    ;max.délka jména souboru
      LDIR              ;blokový přenos
      LD    A,0DH
      LD    (DE),A            ;nastavit konec
      LD    HL,délka    ;do HL vložit délku souboru
      LD    (1104H),HL  ;uložit adresu do záhlaví
      LD    HL,start.adr.     ;do HL vlozit startovací adresu
                        ;souboru
      LD    (1106H),HL
      LD    HL,Execute  ;do HL vložit adresu aut.startu
      LD    (1108H),HL
      LD    A,01H       ;kód pro cílový soubor
      LD    (10F0H),A
      CALL  F25FH       ;headpoint clear
      CALL  EFA2H       ;file end search
      JP    C,ERROR
      CP    28H         ;kód pro "not found"
```

```
      JP    NZ,ERROR
      LD    HL,(1104H)  ;délka
      LD    (1138H),HL
      LD    HL,(1106H)  ;startovací adresa
      LD    (1136H),HL
      LD    HL,0404H    ;kód 04 pro SAVE a 2. kód 04
      LD    (1130H),HL
      LD    HL,0040H    ;délka záhlaví
      LD    (1134H),HL
      LD    HL,10F0H    ;start záhlaví
      LD    (1132H),HL
      CALL  EO10H       ;hlavní program
      JP    C,ERROR
      CALL  E2E8H       ;motor vyp.
      RET
```

Zde uvedené programy mohou běžet jako podprogramy. Musí být ale ještě přidán chybový program, který potom vydává příslušná chybová hlášení.

Řízení plottru ze strojového jazyka

Tato kapitola by vám měla umožnit řízení nestandartně připojeného čtyřbarevného plottru (MZ-1P16) ze strojového jazyka.

Obsluha plotteru z Basicu je podrobně popsána v original, SHARP příručce.Také ve strojovém jazyce si můžete jako v Basicu vybrat mezi dvěma různými druhy provozu plotteru:

1) Grafický mód

V grafickém módu se dá plotterem kreslit nejrůznějším způsobem.

2) Textový mód

V textovém módu se mohou vytisknout pevně dané znaky (písmena, číslice,zvláštní znaky).Tyto znaky se mohou vytisknout v různých velikostech (26,40 a 80 znaků na řádek).

Plotter má znaky uvedené v tabulce znaků plotteru (viz SHARP příručka str.A-26).Jestliže se plotteru dá povel vytisknout znak,který není v této tabulce, automaticky se vytiskne v další barvě hexadecimální kód znaku.

Standartní programy monitoru pro řízení plotteru.

V monitoru 1Z-013B jsou dva standartní programy pro říz. plotteru:

a) CALL 018FH

Tento standartní program vydává obsah akumulátoru na plotter. Obsah akumulátoru přitom může být řídící znak, nebo znak z tabulky znaků.Před zavoláním tohoto standartního programu se musí do akumulátoru uložit příslušný obsah.

b)CALL 01A5H

Tento standartní program vydává na plotter jednotlivé byty řetězce znaků.Řetězec znaků přitom může obsahovat řídící znaky, nebo znaky z tabulky znaků.Řetězec musí být ukončen 0DH, a počáteční adresa řetězce musí být uložena v registru DE.

Oba standartní programy pro plotter mají stejný účinek: vydávají jeden byte na plotter.Program 01A5H by se měl použít jestliže chceme na plotter vyslat hodně bytů (znaků nebo říd. znaků),jinak se používá program 018FH, který vydává pouze jeden byte na plotter.

Řízení plotteru ve znakovém módu

Po zapnutí MZ-800 je plotter automaticky ve znakovém módu.
Barva tisku je černá a tisková hlava je na levém okraji.
Pomocí obou programů pro řízení plotteru (CALL 018FH a CALL 01A5H) se na plotter dají posílat různé řídící znaky a znaky z tabulky znaků.

Tabulka řídících znaků s příslušným účinkem

Řídící znak	Účinek
01H	Přepíná plotter do znakového módu
02H	Přepíná plotter do grafického módu
03H	Posune o jednu řádku zpět
04H	Test tiskárny (4 čtverce v různých barvách)
0AH	Posune o jednu řádku vpřed
0BH	Změna velikosti písma ze 40 na 26 znaků na ř.
0CH	" " " 26 40 "
0DH	Návrat vozíku 'CR'
0EH	Tisková hlava se posune o pozici do leva
0FH	Posun vpřed o stránku
1DH	Změna barvy
09H,09H,09H	Změna velikosti písma ze 40 na 80 znaků na ř.
09H,09H,0BH	" " " 80 40 "

Příklady programů pro obsluhu plotteru ve znakovém módu.

Pomocí výše uvedených řídících znaků, znaků z tabulky znaků a obou jmenovaných programů se nyní dají provádět různé funkce.

Program 1 :

```
LD    A,0BH       ;26 znaků na řádek
CALL  018FH       ;program monitoru pro výdej
                  ;obsahu akumulátoru
LD    A,42H       ;znak 'B'
CALL  018FH
LD    A,42H       ;znak 'B'
CALL  018FH
LD    A,47H       ;znak 'G'
CALL  018FH
```

Tento krátký program přepne plotter na 26 znaků na řádek a potom vytiskne písmena 'BBG'

Stejný program se dá realizovat také jinak, a to pomocí druhého programu monitoru.

Program 2 :

```
LD    DE,A000H    ;počáteční adr. řetězce znaků
CALL  01A5H       ;program monitoru pro výdej
```

```
                        ;řetězce znaků
```

Přitom musí být v paměti od A000H následující :

```
      A000H 0BH         ;26 znaků na řádek
      A001H 42H         ;znak 'B'
      A002H 42H
      A003H 47H         ;znak 'G'
      A004H 0DH         ;značka konce řetězce
```

Řízení plotteru v grafickém módu

Následující program přepíná plotter ze znakového módu do do grafického módu.

Program 3:

```
      LD    A,O2H       ;řídící znak pro graf. mód
      CALL  018FH       ;program monitoru pro výdej
                        ;obsahu akumulátoru
```

Po provedení tohoto programu se nacházíte v grafickém módu. Řízení plotteru se provádí podobně jako ve znakovém módu. Navíc jsou k dispozici specielní rídící povely a grafické povely. pro řízení se též používají standardní programy monitoru 018FH a 01A5H. Při některých grafických povelech se musí na plotter předat navíc k povelu hodnoty parametrů.

Všeobecně k hodnotám parametrů.

Všechny hodnoty parametrů jsou decimální hodnoty a mohou být maximálně trojmístné (stovky, desítky, jednotky). jestliže povel potřebuje více parametrů, jsou jednotlivé hodnoty parametrů odděleny čárkou (ASCII 2CH). Jednotlivé číslice hodnot parametrů se rovněž zadávají v ASCII formátu.

Povely v grafickém módu.

a) povely bez parametrů

Povel	ASCII	Význam
A	41H	Návrat do znakového módu
H	48H	Zvedá hrot pera a posouvá tiskovou hlavu do výchozí pozice
I	49H	Aktuální pozice tiskové hlavy je definována jako výchozí pozice

b) povely s jedním parametrem

Povel	ASCII	Význam
C	43H	Změna barvy tiskové hlavy Tento povel musí následovat parametr o hodnotě 0-3 (ASCII 30H-33H). 0 = černá ASCII 30H 1 = modrá ASCII 31H 2 = zelená ASCII 32H 3 = červená ASCII 33H

Násl. program provádí změnu barvy na červenou.

```
      LD    A,43H       ;C - Povel
      CALL  018FH       ;program monitoru
      LD    A,33H       ;parametr pro červenou barvu
      CALL  018FH       ;program monitoru
```

L 4CH Určuje druh čáry (průběžná nebo tečkovaná)
Tento povel musí následovat hodnota parametru
mezi 0 - 15 (v ASCII formátu)
0 = druh čáry - průběžná
1 - 15 = tečkované druhy čar

Násl. program určuje druh čáry 15.

```
      LD    A,4CH       ;L - povel
      CALL  018FH
      LD    A,31H       ;desítková část hodnoty parametru
                        ;v ASCII formátu
      CALL  018FH
      LD    A,35H       ;jednotková část   "        "
      CALL  018FH
```

Q 51H Určuje směr ve kterém má být vytištěn znak
Po tomto povelu musí následovat hodnota parametru
mezi 0 až 3 (ASCII 30H až 33H).

0 = vzpřímené
1 = doprava ležící
2 = na hlavě stojící
3 = doleva ležící

S 53H Určuje velikost znaků.
Po tomto povelu musí následovat hodnota parametru
mezi 0 a 63 (v ASCII formátu).

Následující program mění velikost znaků na 60.

```
      LD    A,53H       ;S - povel
      CALL  018FH       LD    A,36H       ;desítkové číslo param. 60
      CALL  018FH
      LD    A,30H       ;jednotkové číslo param. 60
      CALL  018FH
```

c) povely se dvěma parametry

Povel ASCII Význam

D 44H Kreslí čáry počínaje aktuální pozicí tiskové hlavy
na udané souřadnicové body až po konečný bod
(Xn, Yn). D X1,Y1,X2,Y2,....,Xn,Yn kreslí čáru
od aktuální pozive kursoru k bodu (X1, Y1),
z tohoto bodu do bodu (X2, Y2) až do koncového
bodu (Xn, Yn).Hodnota prvního parametru
(X - pozice) musí ležet mezi -480 a 480.
Hodnota druhého parametru (Y - pozice) musí
ležet mezi -999 a 999.Hodnoty parametrů musí být
odděleny čárkou (ASCII 2CH).Po poslední hodnotě
parametru musí následovat označení konce 0DH.

Následující program krslí čáru z aktuální pozice

tiskové hlavy do bodu (-20, 20):

```
      LD    A,43H       ;D-povel
      CALL  018FH
      LD    A,20H       ;" - " znak v ASCII
      CALL  018FH
      LD    A,32H       ;desítkové místo X hodnoty v ASCII
      CALL  018FH
      LD    A,30H       ;jednotkové   "       "        "
      CALL  018FH
      LD    A,2CH       ;čárka (oddělení hodnot obou par.)
      CALL  018FH
      LD    A,32H       ;desítkové místo Y hodnoty v ASCII
      CALL  018FH
      LD    A,30H       ;jednotkové   "         "       "
      CALL  018FH
      LD    A,0DH       ;označení konce dat
      CALL  018FH
```

J **4AH** Kreslí čáry, počínaje aktuální pozicí tiskové hlavy do relativními souřadnicemi daného bodu (X1, Y1) až do koncového bodu (Xn, Yn). Povel "J" je jako takový identický s povel "D", ale u povel "D" se udávají absolutní souřadnice a u povel "J" relativní. X hodnoty parametru musí ležet mezi -480 až 480. Y " " " " " -999 až 999. Hodnoty parametrů jsou navzájem odděleny čárkou (ASCII 2CH) a musí být ukončeny 0DH. Jinak viz. povel "D".

M **4DH** Zvedá hrot pera a posouvá tiskovou hlavu do bodu (X, Y).Po povelu musí následovat dva parametry, které jsou odděleny čárkou (ASCII 2CH). X hodnoty parametru musí ležet mezi -480 až 480. Y " " " " " -999 až 999. Oba parametry musí být ukončeny 0DH. Následující program posouvá tiskovou hlavu na pozici (-1, 123).

```
      LD    A,4DH       ;M povel
      CALL  018FH
      LD    A,2DH       ;minus znak v ASCII
      CALL  018FH
      LD    A,31H       ;jednotkové místo X hodnoty
                        ;parametru v ASCII
      CALL  018FH
      LD    A,2CH       ;čárka (oddělení hodnot param.)
      CALL  018FH
      LD    A,31H       ;stovkové místo Y v ASCII
      CALL  018FH
      LD    A,32H       ;desítkové  "       "
      CALL  018FH
      LD    A,33H       ;jednotkové "       "
      CALL  018FH
      LD    A,0DH       ;označení konce dat
      CALL  018FH
```

R **52H** Zvedá hrot pera a posouvá tiskovou hlavu do pozice bodu (X, Y).Povel "R" je jako takový identický s povelem "M", ale při povelu "M"

se zadávají absolutní souřadnice a s povelu "R"
relativní .
X hodnoty parametru musí ležet mezi -480 až 480.
Y " " " " " -999 až 999.
Hodnoty parametru jsou odděleny čárkou
(ASCII 2CH) a musí být ukončeny 0DH.
Jinak viz. povel "M".

d) povely se třemi parametry

Povel ASCII Význam

X 58H Tento povel kreslí kartézský systém souřadnic.
První hodnota parametru musí být buď 0 nebo 1.
Při 0 se kreslí Y-ová osa, při 1 se kreslí X-ová.
Druhý parametr je měřítkový faktor a musí ležet
mezi -999 až 999.
Třetí parametr obsahuje počet značek na osu,
a musí ležet mezi 1 až 255.
Tento povel jinak odpovídá povelu AXIS v Basicu.
Při dalších otázkách viz. též příručka SHARP.
Jednotlivé hodnoty parametrů musí být odděleny
čárkou a ukončeny 0DH.

e) povely s různým počtem parametrů

Povel ASCII Význam

P 50H Tento povel vytiskne znak.Tento povel musí
následovat alespoň jeden parametr.
Vytištěné znaky se udávají v ASCII kódu jako
hodnoty parametrů. Jednotlivé hodnoty parametrů
nemusí být odděleny, ale za posledním znakem musí
následovat 0DH jako označení konce.
Následující program vytiskne písmena BBG.

```
LD    A,50H       ;P povel
CALL  018FH
LD    A,42H       ;"B"
CALL  018FH
LD    A,42H       ;"B"
CALL  018FH
LD    A,47H       ;"G"
CALL  018FH
LD    A,0DH       ;označení konce
CALL  018FH
```

Kombinace více povelů ve znakovém a grafickém módu.

a) znakový mód

Ve znakovém módu mohou být jednotlivé povely spolu spojeny bez oddělení.

Příklad programu:

```
LD    A,01H       ;znakový mód
CALL  018FH
LD    A,0FH       ;posun o stránku
CALL  018FH
LD    A,09H       ;přenutí na 80 znaků/řádek
CALL  018FH
```

```
      LD    A,09H
      CALL  018FH
      LD    A,09H
      CALL  018FH
      LD    A,42H       ;"B"
      CALL  018FH
      LD    A,42H       ;"B"
      CALL  018FH
      LD    A,47H       ;"G"
      CALL  018FH
      LD    A,0DH       ;označení konce
      CALL  018FH
```

b) grafický mód

**V grafickém módu musí být povely, jenž musí být ukončeny pomocí 0DH odděleny čárkou (ASCII 2CH). Přitom 0DH odpadá.
Ale na konci, po posledním povelu, musí být udáno 0DH.**

Příklad programu:

```
      LD    A,02H       ;grafický mód
      CALL  018FH
      LD    A,49H       ;I povel
      CALL  018FH
      LD    A,4CH       ;L povel
      CALL  018FH
      LD    A,30H       ;spojitý druh čáry
      CALL  018FH
      LD    A,43H       ;C povel
      CALL  018FH
      LD    A,32H       ;zelená barva
      CALL  018FH
      LD    A,4AH       ;J povel
      CALL  018FH
      LD    A,31H       ;"1"
      CALL  018FH
      LD    A,30H       ;"0"
      CALL  018FH
      LD    A,30H       ;"0"
      CALL  018FH
      LD    A,2CH       ;čárka
      CALL  018FH
      LD    A,2DH       ;minusové znaménko
      CALL  018FH
      LD    A,32H       ;"2"
      CALL  018FH
      LD    A,30H       ;"0"
      CALL  018FH
      LD    A,31H       ;"1"
      CALL  018FH
      LD    A,2CH       ;čárka
      CALL  018FH
      LD    A,30H       ;"0"
      CALL  018FH
      LD    A,2CH       ;čárka
      CALL  018FH
      LD    A,30H       ;"0"
      CALL  018FH
      LD    A,2CH       ;čárka (oddělení poslední
                        ;hodnoty parametru a povelu "H")
```

```
      CALL  018FH
      LD    A,48H       ;H povel
      CALL  018FH
      LD    A,41H       ;A povel
      CALL  018FH
      LD    A,0DH       ;označení konce
      CALL  018FH
```

Tento program přepíná plotter do grafického módu.
Potom se určí momentální pozice tiskové hlavy jako nový počátek.
Další povel určuje druh čáry. Následující povel udává barvu.
Potom se udělá čára mezi relativním bodem (100, -201) a bodem
(0, 0).Nakonec se tisková hlava posune do původní pozice
a plotter se přepne do znakového módu. Tyto programy se dají
uspořádat mnohem lépe pomocí druhého standartního programu
monitoru 01A5H. Pro přehlednost ale byly napsány se standartním
programem 018FH.

JOYSTICK

Tato kapitola se věnuje obsluze joysticku.
Joystick se připojuje vzadu na rozšiřující liště. Přes povel
IN A,(F0H) se do akumulátoru ukládá informca zda se s joystickem
hnulo nebo ne. Je možné zjistit zda bylo s joystickem pohnuto
do osmi různých směrů. Přitom dává joystick při nestisknuté
klávese pro střelbu následující hodnoty:

nahoře

```
      FAH   FEH   F6H
vlevo FBH   FFH   F7H   vpravo
      F9H   FDH   F5H
```

dole

Nyní hodntoy při stisknuté klávese pro střelbu:

nahoře

```
      EAH   EEH   E6H
vlevo EBH   EFH   E7H   vpravo
      E9H   EDH   E5H
```

dole

Bitový vzor akumulátoru se dá rozluštit,aby se dalo
zjistit, jestli se s joystickem hnulo nebo ne.

```
Bit  0  :obsahuje  0  , když se s joystickem hnulo nahoru
bit  1  :   "      0  ,   "  "       "          "   dolů
bit  2  :   "      0  ,   "  "       "          "   doleva
bit  3  :   "      0  ,   "  "       "          "   doprava
bit  4  :   "      0  ,   "  "  stiskne klávesa pro střelbu
bit  5 - 7  "    vždy  1
```

Když je tedy stisknuta klávesa pro střelbu a joystick
je tlačen doleva, jsou bity 2 a 4 nastaveny na 0, ostatní

jsou v 1. Potom vychází následující vzor bitů : 11101011 = EB
hexadecimálně (viz. tab.). Potom napíšeme ještě program,
který analizuje joystick z Basicu. Musí být vyvolán povelem
USR ($5700).

Na následujících adresách jsou obsaženy hodnoty pro pohyb joysticku:

```
5800H:       hodnota pro joystick nahoru  1=nahoru  0=neobsaženo
5801H:        "    "     "     dolů 1=dolů    0=     "
5802H:        "    "     "     doleva      1=doleva  0=     "
5803H:        "    "     "     doprava   1=doprava 0=     "
5804H:        "    "  klávesu střelby    1=stisk.  0=nestisknuto
```

Nyní tedy příslušný program, který musí být vyvolán jako pod-
program ve strojovém jazyce :

```
5700 DB F0  IN    A,(F0H)           ;hodnota z joy. 1
5702 2F           CPL               ;komplementace hodnoty
5703 21 FF 57   LD      HL,57FFH
5706 06 05  LD    B,06        ;smyčka pro 5 hodnot
5708 23           INC   HL
5709 F5           PUSH  AF          ;zachraň akumulátor
570A E6 01  AND   01          ;maskování posledního
                              ;bitu
570C 77           LD    (HL),A
570D F1           POP   AF
570E 0F           RRCA              ;otáčej akumulátor
570F 10 F7  DJNZ  5708
5711 C9           RET
```

Samozřejmě mohou být volány 2 joysticky. Pricip volání
druhého joysticku je analogický k volání prvního joysticku.
Ale místo port F0H musí být použit port F1H.

RAMDISK MZ-1R18

Ramdisk MZ=1R18 je hardwarový přídavek, který přpojuje
64k bytový disk. Tento ale není přímo přístupný, ale je
adresovatelný pouze přes porty. Z-80 CPU tuto oblast také může
adresovat. Na to se musí napsat podprogram, aby se na adresu
na ramdisku dala napsat ev. přečíst nějaká hodnota. Adresa
by měla být v HL, příslušná hodnota musí být v akumulátoru.
K tomu malý příklad :

-Program pro psaní na ramdisk

```
1200 F5           PUSH  AF
1201 C5           PUSH  BC 1202 6F        LD    A,L
1203 44           LD    B,H
1204 0E EB  LD    C,EBH
1206 ED 79  OUT   (C),A
1208 C1           POP   BC
1209 F1           POP   AF
120A D3 EA  OUT   (EAH),A
120C C9     RET
```

-Program pro čtení z ramdisku

```
1210 F5           PUSH  AF
```

```
1211 C5           PUSH  BC
1212 6F           LD    A,L
1213 44           LD    B,H
1214 0E EB  LD    C,EBH
1216 ED 79  OUT     (C),A
1218 C1           POP   BC
1219 F1           POP   AF
121A DB EA        IN    A,(EAH)
121C C9           RET
```

Důležitý je program, kterým se dá přenést oblast z paměti do libovolné oblasti ramdisku.

K tomu jsou zapotřebí tři hodnoty:

a) počáteční adresa bloku v paměti (HL registr)
b) " " " v ramdisku (DE registr)
c) délka bloku.

Používáme přitom již napsaný program, který píše 1 byte na adresu v ramdisku. Chceme například oblast od 2000H do 3000H napsat na ramdisk od adresy 0000H

Zde program k tomu:

```
2000 21 00 20     LD    HL,2000H    ;počáteční adresa v paměti
2003 11 00 00   LD      DE,0000H    ;   "        " v ramdisku
2006 01 00 10   LD      BC,1000H    ;délka bloku
2009 1A           LD    A,(HL)
200A E5           PUSH  HL          ;zachraň HL
200B D5           PUSH    DE        ;HL = DE
200C E1           POP   HL
200D CD 00 12     CALL  1200H       ;napiš obsah akumulátoru
                              ;na ramdisk
2010 E1           POP   HL
2011 23           INC   HL
2012 0B           DEC     BC
2013 13           INC   DE
2014 78     LD    A,C         ;BC = 0 ?
2015 B1           OR    C
2016 C2 09 20   JP      NZ,2009H
2019 C3 AD 00   JP      00ADH       ;skok do monitoru
```

Nyní chceme tento blok z ramdisku opět načíst do paměti.Potřebujeme znovu ty samé tři parametry, ale tentokrát má být blok přenesen na adresu 7000H v paměti. Také tentokrát použijeme znovu podprogram, který čte 1 byte z ramdisku.

```
2020 11 00 70     LD    DE,7000H    ;poč.adresa v paměti
2023 21 00 00   LD      HL,0000H    ; "     "   v ramdisku
2026 01 00 10     LD    BC,1000H    ;délka bloku
2029 CD 10 12     CALL  1210H       ;čti byte z ramdisku
202C 12           LD    (DE),A
202D 23           INC   HL
202E 13           INC   DE
202F 0B           DEC   BC
2030 78           LD    A,B         ;BC = 0 ?
2031 B1           OR    C
2032 C2 29 20     JP    NZ,2029H    ;když ne, čti dál
2035 C3 AD 00     JP    00ADH       ;skok do monitoru
```

Tyto podprogramy lze bez úprav vložit do vašich programů. Musíte jen v každém programu změnit adresu bloku.

Přepínání paměti (stránkování paměti)

MZ-800 má různé paměti, které leží vzájemně paralelně. Tyto mohou být přes přepínání paměti (stránkování paměti) vypínány ev. zapínány. Protože CPU je adresovatelný pouze na 64k bytů je toto přepínání nezbytností. Normálně se toto provádí pomocí určitých povelů IN a OUT. Důležité je, že program pro přepínání paměti nesmí být v té oblasti kde se přepínání provádí. Výsledek by zpravidla bylo zhroucení programu. Nyní se podívejme co jednotlové povely IN a OUT způsobují :

a) OUT (CEH),A
Jestliže byla předtím do akumulátoru vložena hodnota 8H, přepne se MZ-800 do módu MZ-700. Při uložení jiných hodnot v akumulátoru (možno pouze 0 až 7) přepne se MZ-800 do specifického grafického módu. K tomu blíže v kapitole o grafice.

b) OUT (E0H),A

Při tomto povelu OUT se oblast paměti 0000H-8000H přepne jako RAM. Zbytek paměti se nezmění.

c) OUT (E1H),A

Zde jsou opět dva módy:

1) MZ-700 mód (přes LD A,8H OUT (CEH),A)
Oblast od D000H do FFFFH se zapojí jako RAM.

2) MZ-800 mód
Oblast od E000H do FFFFH se zapojí jako RAM.

d) OUT (E2H),A

Oblast od 0000H do FFFFH se zapojí jako RAM.

e) OUT (E3H),A
1) MZ-700 mód
Oblast od D000H do DFFFH se zapojí jako VRAM.
Aktivuje se horní monitor od E000H do FFFFH.

2) MZ-800 mód
Aktivuje se pouze horní monitor ROM.

f) OUT (E4H),A
1) MZ-700 mód
Oblasti 0000H-1000H a E000H-FFFFH se zapojí jako ROM.
Aktivuje se VRAM od D000H do DFFFH. Tento mód odpovídá módu MZ-80 K.

2) MZ-800 mód
Oblasti 0000H-1000H a E000H-FFFFH se zapojí jako ROM.
Oblast od 1000H do 1FFFH se aktivuje jako CGROM.
Aktivuje se vysoká rozlišitelnost VRAM od 8000H nahoru (podle rozšíření grafické paměti). Zbytek paměti je RAM.

g) OUT (E5H),A
Jako OUT (E1H),A ale příslušná oblast se zcela odpojí.

h) OUT (E6H),A

Jako OUT (E5H),A , ale návrat ke stavu před zabráněním přístupu.

i) IN   A,(E0H)

Aktivuje uvnitř OUT (E4H) módu grafiku.

j) IN   A,(E1H)

Vypíná grafiku uvnitř OUT (E4H) módu.

Nyní bychom k tomuto chtěli uvést příklad.ROM Monitor MZ-800 má být přemístěn z ROM do paralelní RAM. Potom budeme mít software monitor, který můžeme libovolně měnit.

Nejprve program :

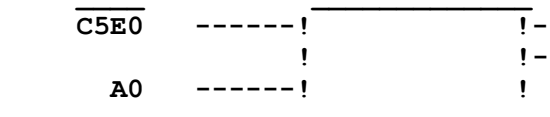
```
1200 21 00 00     LD    HL,0000H    ;počáteční adresa ROM
                                    ;monitoru
1203 11 00 20     LD    DE,2000H    ;volná oblast v paměti
1026 01 00 10   LD      BC,1000H    ;délka ROM monitoru
1209 E5           PUSH  HL          ;zachraň HL
120A C5           PUSH  BC          ;   "    BC
120B D5           PUSH  DE          ;   "    DE
120C ED B0  LDIR                    ;přenos do volné paměti
120E E1           POP   HL          ;HL=2000H
120F C1           POP   BC          ;BC=1000H
1210 D1           POP   DE          ;DE=0000H
1211 D3 E0  OUT   (E0H),A           ;zapoj oblast
                                    ;0000H-1000H jako RAM
1213 ED B0  LDIR                    ;přenes do RAM monitor
1215 C3 AD 00     JP    00ADH       ;horký start RAM monitoru
```

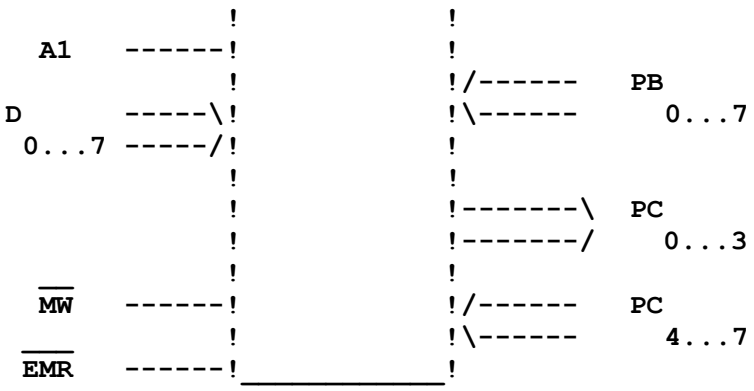
Pro vysvětlení.Nejprve přeneseme ROM monitor, který leží v oblasti od 0000H-1000H do volné oblasti paměti od adresy 2000H. Potom zapojíme oblast od 0000H do 1000H jako RAM a přeneseme sem program ROM monitoru a odstartujeme ho.

Je těžké uvnitř oblasti 0000H-1000H a E000H-FFFFh, když jsou tyto zapojeny jako RAM vyvolat podprogramy jako 0015H, protože tyto podprogramy potřebují mód provozního systému MZ-80 K. Musí se buď upravit software nebo vytvořit specielní podprogramy. Mimochodem můžete, chcete-li, manipulovat pouze s RAM monitorem.
Každou buňku paměti v oblasti 0000H-1000H můžete změnit M-povelem.

HARDWARE

Součást MZ-800 je díl 8255.8255 je zodpovědný za obho- spořadování magnetofonu a klávesnice. Obrázek dole představuje adresování a způsob provozu dílu portů





Druh provozu, používaný v MZ-800 zobrazuje obr.1. Port A je nastaven na výstup, port B na vstup a na portu C jsou čtyři vedení s nižší hodnotou nastaveny na výstup, čtyři s vyšší hodnotou jsou nastaveny na vstup. Toto se provede pomocí CONTROLWORD 8AH (pomocí řídícího slova 8AH). Pomocí tohoto řídícího bytu musí být nastaven mód (druh provozu), než se s tímto dílem pracuje. ROM monitorprogram toto provádí pomocí standartního programu 073EH.

Úplný výčet druhů provozu najdete v následující tabulce.

Tabulka 1 : Řídící slova pro 8255 v módu 0 (adresa E003H)
A = výstup E = vstup

Hex	Řídící slovo (bit) 7 6 5 4 3 2 1 0	port A	port B	port C
		všechny b.	všechny b.	4-7.b.+0-3.b.
80	1 0 0 0 0 0 0 0	A	A	A + A
81	1 0 0 0 0 0 0 1	A	A	A + E
82	1 0 0 0 0 0 1 0	A	E	A + A
83	1 0 0 0 0 0 1 1	A	E	A + E
88	1 0 0 0 1 0 0 0	A	A	E + A
89	1 0 0 0 1 0 0 1	A	A	E + E
8A	1 0 0 0 1 0 1 0	A	E	E + A
8B	1 0 0 0 1 0 1 1	A	E	E + E
90	1 0 0 1 0 0 0 0	E	A	A + A
91	1 0 0 1 0 0 0 1	E	A	A + E
92	1 0 0 1 0 0 1 0	E	E	A + A
93	1 0 0 1 0 0 1 1	E	E	A + E
98	1 0 0 1 1 0 0 0	E	A	E + A
99	1 0 0 1 1 0 0 1	E	A	E + E
9A	1 0 0 1 1 0 1 0	E	E	E + A
9B	1 0 0 1 1 0 1 1	E	E	E + E

Nyní musíme ještě řídící slovo předat do 8255. Na to si musíme ještě blíže prohlédnout adresy dílů.
Adresuje se pomocí C5E0, A0 a A1. Tím je detekovatelný adresami E000H až E003H. Každá z těchto adres má určitý význam (CS = chip select - výběr chipu)

E000H adresa pro port A
E001H " " " B
E002H " " " C
E003H " " " řídící slovo

Tyto adresy slouží společně s MW pro psaní a s EMR pro čtení.
Psaní znamená výstup, čtení vstup.

Testování klávesnice

Porty A a B slouží k testování klávesnice. K tomu se přes výstup portu A posílá na sloupce 1 - 10 matice klávesnice (viz.obrázek klávesnice v dodatku), postupně L potenciál. Při každém nařízení řádku se na vedení řádku 1 - 18 zkoumá, zda L potenciál přes matici prošel nebo ne. Z bodu kde se křižuje vedení řádků a sloupců se dá potom zjistit, která klávesa byla stisknuta. Tento druh testování klávesnice se též nazývá SCANNEN.

Samozřejmě nemusí být vždy testována celá klávesnice, ale je též možne testovat pouze určité klávesy (nebo též netestovat) jako v programy monitoru standartní program 001EH.

Vysvětleme si na základě krátkého programu spolupůsobení hadrware a software. Při tom je nutno mít na zřeteli, že zde uvedená možnost testování klávesnice je realizovatelná, když E oblast v paměti není zapojena jako RAM ale jako KEY TIMER v oblasti od E000H až E070H (druh provozu MZ-80K).

Nejprve se na port A nastaví číslo sloupce, který má být testován (=8) pomocí :

```
LD    A,F8H       3E F8
LD    (KEYPA),A   32 00 E0
```

Pro stabilizaci se přidá NOP. Potom se přes port B vnese zpět informace o řádkách 11 až 18 pro sloupec 8.

```
NOP               00
LD    A,(KEYPB)   3A 01 E0
```

Nastaví se počátečná hodnota Carry bitu (bit přenosu) a dolu na bit 0 (SHIFT)

```
OR    A           B7
RRA               1F
JP    C,?BRK2           DA 89 09
```

Je-li stisknuta klávesa SHIFT, nastaví se indikátor přenosu na 0 a pomocí dvojnásobného posunutí doleva se BREAK bit přenese do indikátoru přenosu a nastane BREAK.

```
RLA               17
RLA               17
JR    NC,?BRK1    30 04       ;SHIFT & BREAK
                              ;CY = 0
```

Jestliže BREAK nebylo stisknuto, uloží se do akumulátoru hodnoty pro klávesu SHIFT (bit 6 = 1) , nastaven indikátor přenosu a skočí se zpět.

```
LD    A,40H       3E 40
SCF               37
RET               C9
```

Jestližé bylo stisknuto BREAK, vymaže se indikátor přenosu, nastaví se Z-flag a skočí se zpět.

```
      XOR   A           AF
      RET               C9
```

Tento program tedy sdělí výsledek testování pomocí stavu flagů a bitů 4 - 6 v akumulátoru. Při tom znamenají :

a) Z=1 bylo stisknuto SHIFT & BREAK
b) D6=1 & Cy=1 " " SHIFT
c) D5=1 & Cy=1 " " CONTROL (CTRL)
d) D4=1 & Cy=1 " " SHIFT & CONTROL

V předcházejícím oddíle bylo provedeno testování zvláštních kláves jako např. SHIFT & BREAK pomocí specielního programu. Výsledek testování byl vyjádřen pomocí flagů. Ve většině případů ale má být testována celá klávesnice a výsledek testování by měl být jako hodnota v registru. Zde SHARP používá specielní, pro klávesy upravený kód, zobrazovací kód. Tato tab. kódů ale bohužel v uživatelské příručce vůbec není otištěna. V dodatku této knihy najdete tab. zobrazovacích kódů. Vlastní testování klávesnice se provádí standartním programem monitoru 0A50H. Výsledek testování je k dispozici v registru BC. Při tom registr B obsahuje stav zvláštních kláves, tj. byla-li stisknuta klávesa SHIFT, BREAK, CTRL atd. To odpovídá zobrazení výsledných hodnot našeho posledního programu, čili když Z = 1, bylo stisknuto SHIFT & BREAK atd. Registr C obsahuje bod křížení čili řádku a sloupec, kde bylo zjištěno stisknutí klávesy. Obsah C se dá interpretovat jako následující :

X X S S S Z Z Z

X = nepoužito
S = číslo sloupce
Z = číslo řádku

Přitom dává např. testování stisknuté klávesy "A" v registru C hodnotu 20H nebo při stisknutí klávesy "B" hodnotu 21H. Program testování klávesnice monitoru nebere v úvahu sloupec 10 = F9H. Tím nemohou být z monitoru testovány klávesy funkcí. Následující program ve strojovém jazyce testuje klávesy funkcí F1 až F5 a potom provádí skok na předem naprogramovanou adresu.

```
FKLAVESA:   LD    A,F9H
            LD    (KEYPA),A
            NOP
            LD    A,(KEYPB)
            CPL
            OR    A
            RET   Z
            EXX
            LD    D,00H
            LD    B,00H

FT1:        INC   B
            RLCA
            JR    NC,FT1
            LD    A,B
            RLCA
            LD    E,A
            LD    HL,FLEI-2   ;od adresy FLEI leží tab.
                              ;skoků pro F1 až F5.
```

```
                        ;Adresu a data skoků mu-
                        ;síte zvolit sami
        ADD   HL,DE
        LD    C,(HL)
        INC   HL
        LD    B,(HL)
        POP   AF
        PUSH  BC
        EXX
        RET
```

Dejte pozor na to, že při provádění tohoto navrženého programu má testování F klávesy funkci GET. Program tedy proběhne pouze jednou, t.zn. že se nečeká na stisknutí klávesy.Jestliže neby-la stisknuta žádná F klávesa, vrátí řízení programu na hlavní program se Z-flag = 0.

Řízení kazetového magnetofonu

Kazetový magnetofon je řízen přes C port 8255. Tím má adresu E002H. Port C se nastavuje řídícím slovem 8AH pro 8255. Bity 0-3 pracují pro směr výstupu, bity 4-7 pro vstup. Než mohou být data vydána ev. přečtena z magnetofonu, musí se spustit motor. To se může udělat pouze ručně stisknutím PLAY nebo RECORD & PLAY. Procesor se ale ptá programem monitoru 'MOTOR' z adresy 069FH přes 4.bit portu C, zda motor běží. Tento prgram monitoru potom zkouší spustit motor řídícím impulsem na 3.bit portu C. Tento pokus má ale úspěch pouze tehdy, je-li PLAY a RECORD & PLAY ještě stisknuto a motor byl zastaven pod-programem 'MSTOP' na 0700H.Tím se dá motor pomocí software pustit a zastavit. To je důležité zvláště tehdy, když programy ev. bloky dat musejí být nejprve zpracovány, než může být pře-čten další blok dat. Vlastní nahrání bitů na magnetofon se provádí vždy přes bit 1 portu C. Nahrání programu je tedy vždy ve dvouch dílech, informačnim díle a v datech. Data jsou vlastní obsah programu,informační díl (označovaný též jako záhlaví) se skládá z:

1. kód souboru — 1 byt pro rozlišení druhu programu
2. jméno programu — 16 bytů+1 byt označení konce
3. délka dat — 2 byty
4. počátečni adresa dat — 2 byty
5. prováděcí adresa (startovací adresa) — 2 byty
6. komentář (bez praktického významu) — 104 bytů

Program pro zapsání z pracovní paměti se podle toho skládá vždy ze dvou dílů.

1. Zapsání informační části (záhlaví). Tuto úlohu provádí standartní program monitoru "WRI" na adrese 0021H
2. Zapsání dat. Tuto úlohu vyřizuje program monitoru "WRD" na adrse 0024H.

Než se odstartuje program pro zápis na kazetu, musí tedy být v oblasti paměti 10F0H až 116FH připravena data pro informační část. Tato data se potom standartního programu monitoru "WRI" zapíší na pásek jako záhlaví. Pro čtení z kazety přes 5.bit portu C se nejprve do této oblasti paměti nahraje informační

část (záhlaví) a z něj se potom převezmou řídící data pro čtení dat.