SHARP

POZNÁMKY

ROM

1 Úvod

Dostala se Vám do rukou jedna ze tří příruček pro programování ve strojovém kódu na mikropočítači **SHARP MZ-821**. Tato řada příruček byla vytvořena zpětným překladem a okomentováním 16KB ROM s operačním systémem.

U čtenáře předpokládáme znalost strojového kódu mikroprocesoru Z80 a ovládání periferních obvodů I8255, Z80-PIO a I8253.

První část je tvořena přehledem jednotlivých skupin podprogramů a je členěna podle jejich zaměření. Jedinou vyjímkou je zde kapitola o grafickém zobrazování v módu MZ-800, které ROM nepoužívá.

Jsme ale přesvědčeni, že sem patří a že informace v ní uvedené Vám pomohou při Vaší práci s tímto počítačem.

Druhou část tvoří okomentovaný zpětný překlad, který je rozdělen na tři relativně samostatné části, podle logické stavby obsahu ROM. Ve třetí části je výpis generátoru znaků, který je v ROM na adresách 1000H-1FFFH a hexadecimální výpis obsahu celé ROM.

Tyto příručky tvoří dostatečnou dokumentaci pro psaní systémových programů pro tento mikropočítač (pracujících v módu MZ-700).

1.1 Použité konvence

Při vytváření symbolů byla dodržována konvence o významu prvního znaku symbolu podle následující tabulky:

@	Podprogram volatelný instrukcí CALL. Obvykle chrání většinu registrů. Nevrací se nikdy jinam, než do programu, kterým byl zavolán. (Výjimku tvoří chybový návrat podpory BASICu)
[	Návěští v dolním monitoru
]	Návěští v horním monitoru. Takto jsou značena jednak pomocná návěští obou monitorů a také podprogramy, které jsou volatelné instrukcí CALL, ale chybový návrat realizují přímo skokem do monitoru.
=	Textová tabulka v "ASCII". Text je ukončen znakem CR (0DH).
!	Libovolná tabulka
<	I/O port vybíraný přes IOREQ I/O port vybíraný přes MREQ
<x>	Symbolická konstanta reprezentující "ASCII" znak
<x>D	Znak v tzv. display kódu
J	Následuje-li adresa, jedná se o pomocné návěští lokálního významu.
M	Následuje-li adresa, jedná se o pomocnou datovou buňku v RAM.

Tato konvence je vytvořena pouze pro přehlednost okomentovaného zpětného překladu. Většina assemblerů nepovoluje tyto znaky jako první v návěští.

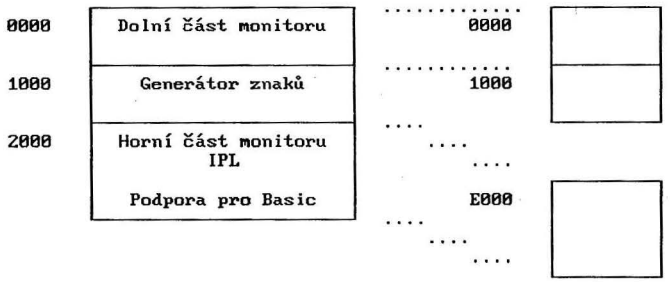
Je vhodné, aby každý při psaní programů dodržoval ve svém zdrojovém textu konvence názvů tak, že při odkazech na ROM bude mít vytvořený symbol (např. jako RHEAD EQU 27H) a ten pak bude používat. (CALL RHEAD).

Přispěje to k zpřehlednění programů i k usnadnění komunikace mezi programátory. K tomuto je vhodné zkráceně ocitovat zásadu číslo 4 z **THE PDP-11 CODEX PROGRAMMATICUS:** "*Každý programátor, který se nepřízpůsobí standardu pojmenování by měl být zastřelen. Jestliže se stane, že není vhodné ho zastřelit, má být zdvořile požádán, aby přepracoval program podle shora uvedených konvencí.*"

Při pojmenování symbolů jsme zachovali názvy použité firmou **SHARP** pro zveřejněné vstupní body do podprogramů a další jsme vytvořili sami tak, aby co nejlépe vyjadřovaly význam návěští.

2 Struktura ROM

V mikropočítači SHARP MZ-821 je použita 16KB EPROM I27128. Mapování fyzických adres v ROM na logické adresy probíhá podle této tabulky:



2-1 Tabulka adres mapování ROM.

Tomu je podřízen i tvar okomentovaného zpětného překladu,který se skládá ze tří samostatných částí:

DOLNÍ MONITOR	00000H - 00FFFH
HORNÍ MONITOR	0E000H - 0F3FFH
PODPORA PRO BASIC	0F400H - 0FFFFH

Generátor znaků je popsán v kapitole **obrazovka**. Detailní informace o generátoru znaků najdete ve třetím dílu těchto materiálů.

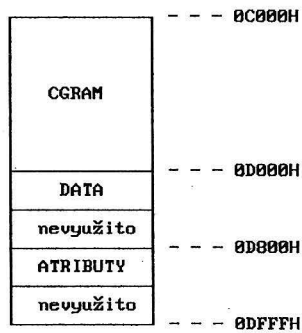
3 Mapování paměti

Pro řízení mapování je využito 7 portů MMC0 - MMC6 na adresách 0E0H - 0E6H. Přepínání pomocí těchto portů není závislé na čtené nebo zapisované hodnotě. Význam většiny portů se mění podle nastaveného módu MZ-700 (DMD = 8) nebo MZ-800 (DMD).
Za základní stav lze považovat ten, kdy je namapováno 64 KB RAM. Následující tabulky ukazují, kterými instrukcemi se mapuje.

1. řádek záhlaví obsahuje instrukce, které namapují tu část paměti, která je označena dvojitou čarou. Jednoduchá čára označu-je, kde se poměry nemění.
2. řádek záhlaví obsahuje instrukce, které namapují do adresového prostoru označeného dvojitou čarou zpět paměť RAM.

Význam použitých zkratek

ROM	obsah EPROM s dolním nebo horním monitorem
CGROM	obsah EPROM s generátorem znaků
CGRAM	obsahuje generátor znaku přenesený do Video RAM
VRAM?	VRAM v módu 640 x 200, RAM v módu 320 x 200
VRAM	Video RAM, v MZ 700 módu s touto organizací:

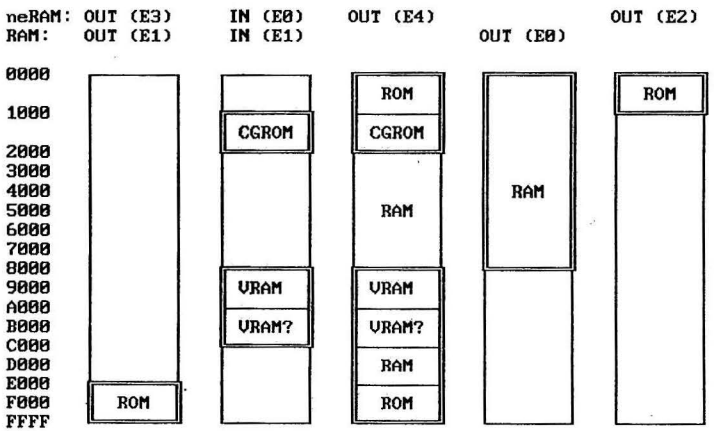


3-1 Struktura VRAM v módu MZ-700

3.1 Mapování paměti v MZ-800 módu

V MZ-800 módu lze samostatně ovládat mapování horního monitoru nebo VRAM současně s CGROM. Velikost mapované VRAM závisí na nastaveném DMD. V módu 320 x 200 zůstává RAM i tam, kde by v 640 x 200 byla VRAM.

Instrukce OUT (0E4H),A namapuje něco jiného než RAM všude tam, kam je to možné. Tohoto stavu se také docílí stiskem RESET.

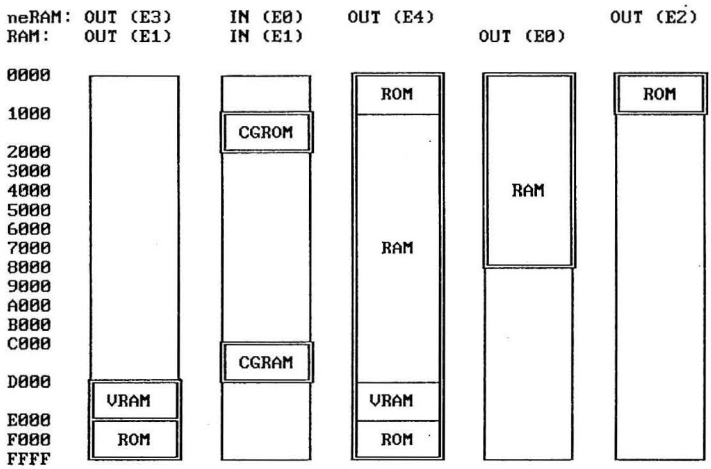


3-2 Mapování paměti v módu MZ-700

3.2 Mapování paměti v MZ-700 módu

V MZ-700 módu se mapuje horní monitor zároveň s VRAM a také s *memory mappingem*, t.j. přístupem k portům pomocí instrukcí pracující s pamětí. Proto po odmapování horního monitoru nelze v módu MZ-700 přistupovat k obvodům I8255 a I8253. (KBD:, CMT:)

Instrukcí OUT (0E4H),A se nastaví základní režim, ve kterém pracují oba ROM monitory.



3-3 Mapování paměti v módu MZ-800

Porty MMC5 a MMC6 slouží pro ovládání signálu INH5, který je sice z videoprocesoru GDG vyveden, ale v MZ-800 není nikam zapojen.

4 Monitor 1Z-013B

Původní vstupní bod spodního monitoru, používaný v MZ-700 byl 004AH. Tento je pro MZ-800 nepoužitelný, protože se zde jako jedna z věcí při inicializaci testuje adresa 0E800H a pokud je nalezena paměť ROM, je na adresu 0E800H předáno řízení. V MZ-700 to předávalo řízení případnému ROM PACKU, v MZ-800 je to spuštění monitoru 9Z-504M (horní monitor).

Pro předání řízení spodnímu monitoru je proto vhodné použít]PRMPT (00ADH), tedy vstup na prompt dolního monitoru.

4.1 Přehled příkazů dolního monitoru

4.1.1 J = Jump

Jxxxx, předá řízení na adresu xxxx

4.1.2 L = Load

Přečte program z pásky a předá mu řízení. Nečte program od standardních zaváděcích adres jako horní monitor, ale od skutečných adres příslušného programu. Proto je někdy vhodnější používat pro nahrávání tohoto monitoru. Program je odstartován, pouze pokud je jeho startovací adresa větší nebo rovna 1200H. V opačném případě je předáno řízení zpět monitoru. Tímto příkazem nelze číst programy pod ROMku.

4.1.3 F = F????

Pokud je obsah adresy 0F000H nulový, tak tam skočí, v opačném případě se vrátí na prompt monitoru. V MZ-800 je tento příkaz nesmyslný. V MZ-700 sloužil pravděpodobně pro předání řízení ROM PACKU pro obsluhu FLOPPY DISKU, eventuálně QUICK DISKU.

4.1.4 B = BEPP ON/OFF

Zapne nebo vypne akustický signál po stisku každé klávesy.

4.1.5 # = Restart

Namapuje všude pamět RAM a skočí na adresu 0000H. Zničí tím oblast paměti od 0FFF0H do 0FFF5H . Efekt tohoto příkazu je totožný s CTRL a RESET současně.

4.1.6 M = Modify

Mxxxx umožní měnit a prohlížet obsah paměti od adresy xxxx dále. Vykonání se zruší klávesou SHIFT + BREAK. Pokus o zapsání nesprávné hodnoty je prostě ignorován a je vyžadována správná hodnota.

4.1.7 P = printer service

Obsluha tiskárny (MZ-1P16 Plot printer). Existují tyto povely:

&L	přepnutí na 80 znaků na řádek
&S	přepnutí na 40 znaků na řádek
&C	výměna pera
&G	přepnutí do grafického režimu
&T	aktivace selftestu tiskárny
text	vytiskne text

Povely následují těsně za příkazem P. Vyskytne-li se povel za textem, je inorován a vytisknut jako součást textu.

4.1.8 S = Save

Uložení souboru na pásku. Typ je vždy 01H, tedy OBJ. Na jméno, délku, zaváděcí a startovací adresu se ptá.

4.1.9 D = Dump

Dxxxxyyyy ... výpis paměti ASCII a v šestnáctkové soustavě. Implicitní hodnota počtu bytů pro výpis yyyy je 00A0H.

4.2 Některé podprogramy monitoru

4.2.1 [HLHEX 013DH

Zprostředkuje volání podprogramu @HLHEX (0410H). Při chybovém návratu přímo předá řízení na prompt monitoru.

4.2.2 [GETL 012FH

Zavolá @GETL (0003H) s nastaveným standardním pracovním prostorem IOBUF (11A3H). Pokud zjistí ukončení vstupu textu klávesou SHIFT + BREAK, tak předá řízení monitoru.

4.2.3 [PCHAR 018FH

Tisk znaku na tiskárně. Na BREAK se vrací přímo do monitoru.

4.2.4 [PTEXT 01A5H

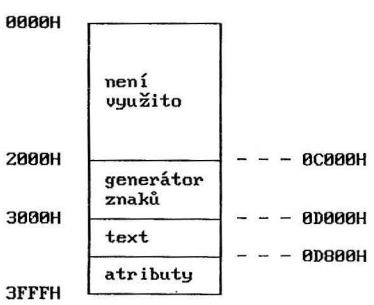
Tisk textu na tiskárně. Text začíná na adrese DE a končí znakem CR (0DH), který se už ovšem netiskne! Chrání všechny registry, ale na BREAK končí v dolním monitoru.

5 Obrazovka

5.1 Připojení CRT:

O řízení zobrazování se v MZ-800 stará zákaznický obvod GDG (100-pin single chip LSI), který zajišťuje také memory management. Programové vybavení v ROM pracuje s obrazovkou tím nejjednodušším způsobem, v MZ-700 módu, kdy je VRAM alfanumerická a obsahuje generátor znaků. Ten je tam při inicializaci zkopírován ze svého vzoru v ROM.

5.2 Uspořádání VRAM v MZ-700 módu:



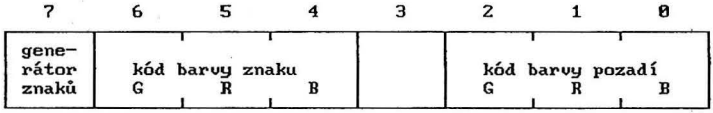
5-1 VRAM v módu MZ-700

Obrazovka v MZ-700 módu má 25 řádků a 40 znaků na řádku. Každému místu přísluší 2 byty ve VRAM, jeden v textové části od D000H a druhý v části atributů od D800H. Obě části zaujímají 1000 bytů, jsou ve VRAM uložené souvisle po řádcích. Zbylých 1048 bytů v textové i v atributové části není využito. Systém je však maže s celou obrazovkou.

5.2.1 Tvar atributu

Každému znaku na obrazovce lze nadefinovat dvě barvy: Barvu popředí a barvu pozadí. V MZ-700 módu je k dispozici výběr z osmi barev s následujícími kódy:

- - -	0	černá
- - B	1	modrá
- R -	2	červená
- R B	3	purpurová (červená + modrá)
G - -	4	zelená
G - B	5	azurová (zelená + modrá)
G R -	6	žlutá (červená + zelená)
G R B	7	bílá



5-2 Struktura atributu

5.3 Generátor znaků

Generátor znaků CGROM zaujímá 4 kB paměti ROM mapující se na adresy 1000H - 1FFFH.

5.3.1 Struktura generátoru

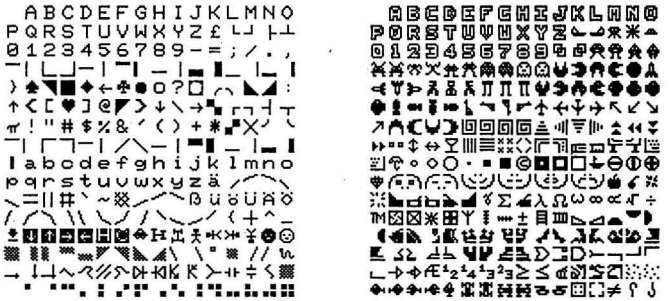
Každý znak je v generátoru reprezentován osmi byty. Tyto byty tvoří matici 8 x 8 bodů, do kterých se rozkresluje každý znak. Byty se ukládají na CRT: od shora dolů s opačným sledem bitů (7. bit je v pravé části znaku). Jednička v generátoru značí bod, který přísluší znaku a nula pozadí. Např: znak 0F1H z 1. generátoru (Pseudografický bod v levém horním rohu) je v generátoru uložen takto: 7,7,7,0,0,0,0,0.

5.3.2 Obsah generátoru

Generátor se skládá ze dvou sad po 256 znacích, celkem je tam tedy 512 znaků. Znaky nejsou uloženy v generátoru podle kódu ASCII, ale podle tzv. display kódu. Přesto však generátor obsahuje všechny znaky ASCII a navíc úplnou pseudografiku 80 x 50 bodů. Podrobnější informace o tvaru jednotlivých znaků najdete ve třetím dílu této příručky.

5.3.3 Generátor znaků CGRAM

Na počítači MZ-700 CGRAM nebyl. Zákaznický obvod pracoval přímo s CGROM. Uživatel si nemohl znaky měnit. Počítač MZ-800 používá CGROM pouze jako vzor. Vlastní generátor je v CGRAM. Mapuje se na adresy 0C000H-0CFFFH těsně pod VRAM. Po inicializaci se přenese obsah CGROM do CGRAM. Uživatelské programy si potom mohou znaky v CGRAM změnit. CGRAM má stejnou strukturu jako CGROM. Software se v MZ-700 módu o rozkreslování nestará, tuto činnost přebírá obvod GDG podle display kódu znaku nalezeného ve VRAM a podle CGRAM. Která sada znaků se použije je určeno 7.bitem atributu. V režimu MZ-800 je CGRAM využíván jako součást VRAM. Uživatel si musí znaky rozkreslovat sám z CGROM nebo užít vlastní generátor znaků.



5-3 Generátory znaků

5.4 Podprogramy pro práci s CRT:

Vstupní vektor MZ-700 obsahuje celou sadu podprogramů pro výstup znaků a textů. Všechny tyto podprogamy chrání registry mimo AF. Předpokládají namapovanou VRAM od 0D000H. Generátor znaků nemění a měnit neumí. I když hardware dovoluje použít až osmi barev, rutiny v ROM používají jen dvě: bílé znaky na modrém podkladu.

Podprogramy využívají tabulku !ATBLN (1173H), která spojuje fyzické řádky na obrazovce do logických. Tabulka má délku odpovídající počtu řádků. Nula u příslušného řádku znamená, že řádek je zahajovací, jednička indikuje pokračování řádku. (má význam pro insert, delete, scroll, a @GETL). Spojeny mohou být maximálně dva řádky. Spojení se dosáhne zápisem znaku do posledního sloupce obrazovky.

5.4.1 @LETNL 0006H

Provede přechod na nový řádek obrazovky. V případě potřeby odroluje.

5.4.2 @IFNL? 0009H

Pokud je kurzor na nulté pozici v řádku, neudělá nic, pokud je kdekoli jinde, provede LETNL.

5.4.3 @PRNTS 000CH

Vytiskne mezeru.

5.4.4 @TAB 000FH

Tabeluje na další tabelační pozici. (po deseti znacích, podle logického ukazatele).

5.4.5 @PRNTC 0012H

Vytiskne znak, který je v A registru v "ASCII" kódu. Pro řídicí znaky zavolá DPCT. Obnovuje obsah registru A.

5.4.6 @MSG 0015H

Vytiskne řetězec znaků, který začíná na adrese DE a končí znakem CR (0DH). Text smí obsahovat řídící znaky, které se provedou.

5.4.7 @RST18 0018H

Jako MSG. Řídící znaky se neprovedou, ale zobrazí.

5.4.8 @?DPCT 0DDCH

Na tyto řídící kódy (vstupují v A registru) provede následující akce, ostatní kódy jsou ignorovány:

0C0H	SCROLL
0C1H	CURSOR DOWN
0C2H	CURSOR UP
0C3H	CURSOR RIGHT
0C4H	CURSOR LEFT
0C5H	HOME
0C6H	CLEAR SCREEN
0C7H	DELETE
0C8H	INSERT
0C9H	SET ALPHANUMERIC MODE
0CAH	SET GRAPHIC MODE
0CBH	není využito
0CCH	není využito
0CDH	CARRIAGE RETURN
0CEH	není využito
0CFH	není využito

5.4.9 @BTHEX 03C3H

Vypíše obsah A registru v šestnáctkové soustavě.

5.4.10 @MHEX 03B1H

Vypíše obsah paměti adresované HL registrem v šestnáctkové soustavě a mezeru.

5.4.11 @HLHEX 03BAH

Vypíše obsah registrového páru HL v šestnáctkové soustavě.

5.4.12 @?NLHL 05FAH

Zavolá IFNL? a HEXHL. Vypíše tedy šestnáctkově obsah HL na nový řádek.

5.4.13 @?POINT 0FB1H

Vrátí v HL adresu kurzoru.

5.4.14 @?ACUR 0FB4H

Očekává v HL registru pozici kurzoru ve tvaru H = řádek, L = sloupec a vrátí v HL tuto adresu přepočtenou na skutečnou adresu ve VRAM.

5.4.15 @CURON 0B92H

Zobrazí kurzor. Znak, který byl pod ním uklidí do systémových proměnných na adresu AKCHAR (118EH).

5.4.16 @CUROF 05F0H

Na místě kurzoru zobrazí správný znak z adresy AKCHAR (11E8H).

5.4.17 @BLIKC 09E3H

Podle stavu PC6 I8255 dosadí na místo kurzoru buď znak kurzoru, nebo znak, který tam má být. (Blikání s kurzorem)

5.4.18 @AVRAM 0DB5H

Zapíše obsah A registru do VRAM a posune kurzor o znak doprava. (Využívá CURSOR RIGHT rutiny DPCT)

5.4.19 @?ADCN 0BB9H

Převede znak v A registru z "ASCII" kódu do display kódu.

5.4.20 @?DACN 0BCEH

Převede znak v A registru z display kódu do "ASCII" kódu.

5.4.21 @PRNTA 096CH

Pomocí AVRAM zobrazí znak a posune CSRH (1194H Logická pozice znaku na řádku)

5.4.22 @ICSRH 096FH

Posune CSRH, při překročení maximální délky logického řádku (80 znaků) ho vynuluje.

6 Klávesnice

6.1 Připojení klávesnice

Klávesnice je připojena přes obvod I8255, z něhož využívá bity PA0-PA3 pro výběr řádku matice kláves a celou bránu B pro vstup dat. Strobe klávesnice je přiveden z PA0-PA3 na vstup obvodu pro výběr 1 z 10 a teprve na výstupu toho obvodu je matice klávesnice. Bity PA4-PA7 se musí při strobe nastavit na jedničku, protože slouží k ovládání joysticků. Z toho důvodu je také vhodné nastavovat PA před každým čtením z PB. Nelze se spoléhat na to, že nebude změněn. Data přečtená z PB odpovídají stavu kláves vybraného sloupce v okamžiku čtení. Logika je na tomto portu negativní, jednička odpovídá nestisknuté klávese, nula stisknuté. PA je na adrese 0D0H resp. 0E000H, PB na adrese 0D1H resp. 0E001H. Podprogram pro zjištění, zda je stisknuta klávesa ve sloupci II a na řádku JJ by mohl vypadat takto:

```
OUT     (0D0H),II.or.11110000B
IN      A,(0D1H)
BIT     JJ,A
JP      Z,klávesa stisknuta
```

	0	1	2	3	4	5	6	7	8	9
7	blank	Y	Q	I	A	1	\	inst	break	F1
6	graph	Z	R	J	B	2	^	del	ctrl	F2
5	libra	@	S	K	C	3	-	up		F3
4	alpha	[	T	L	D	4	spc	down		F4
3	tab	]	U	M	E	5	0	right		F5
2	;		V	N	F	6	9	left		
1	:		W	O	G	7	,	?		
0	cr		X	P	H	8	.	/	shift	

6-1 Rozmístění kláves

6.2 Podprogramy pro práci s klávesnicí

Vstupní vektor MZ-700 zpřístupňuje dva podprogramy pro práci s klávesnicí: GETKY a BRKEY. Další podprogramy jsou použitelné pouze pro případ, že by si uživatel chtěl napsat novou klávesnici, se stejně špatnými vlastnostmi, ale s generováním jiných kódů.

6.2.1 @??KEY 09B3H

Začne blikat s cursorem (není-li to zakázáno nastavením bitu 1 proměnné CONMOD (1170H)) a čeká na stisk klávesy. Přečtenou klávesu vrací v A registru. Znak se neechuje.

6.2.2 @GETKY 001BH

Tento podprogram vrací číslo právě stisknuté klávesy v "ASCII" kódu, nebo nulu pokud není stisknuta žádná klávesa, respektive klávesa, která nemá generovat znak.

6.2.3 @GETKD 08CAH

Vrací číslo klávesy v display kódu nebo 0F0H, pokud není nic stisknuto nebo pokud je stisknuta klávesa, která nemá generovat znak. K převodu mezi kódem klávesnice a display kódem používá 5 tabulek:

!KBD	0BEAH	žádná řídící klávesa
!KBDS	0C2AH	s klávesou SHIFT
!KBDC	0C6AH	s klávesou CTRL
!KBDG	0CAAH	v grafickém režimu
!KBDGS	0CE9H	v grafickém režimu s klávesou SHIFT

6.2.4 @WGKEY 0830H

Počká cca 7 milisekund a zavolá KBDIN. Ničí BC.

6.2.5 @KBDIN 0A50H

Podprogram, který provádí vlastní očtení klávesnice a vrací v registru B kód odvozený z uspořádání kláves.

<kód> = <sloupec> * 8 + (7 - řádek)

Je-li stisknuto více kláves najednou, uvažuje první klávesu na kterou narazí. Tento podprogram neobsahuje čekací smyčku.

6.2.6 @BRKEY 001EH

Otestuje klávesnici na existenci řídících povelů a zachová se podle tabulky:

Stav klávesnice:			Vrací se hodnoty:			Popis situace:
SHIFT	BREAK	CTRL	A	CY	Z	
yes	yes	-	00H	0	1	SHIFT + BREAK
yes	no	-	40H	1	0	SHIFT znaky
no	-	yes	20H	1	0	CTRL znaky
no	yes	no	3FH	0	0	<ESC>
no	no	no	7FH	0	0	běžné znaky

7 Magnetofon

7.1 Připojení CMT:

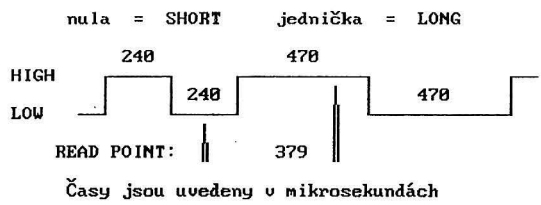
Magnetofon je připojen přes čtyři bity portu C obvodu I8255 (I/O adresa PC je 0D2H resp. 0E002H, CWR je na adrese 0D3H resp. 0E003H)

PC1	výstup	WRITE
PC2	výstup	MOTOR
PC4	vstup	SENSE
PC5	vstup	READ

READ a WRITE slouží pro čtení a zápis dat přes tvarovací obvody. SENSE indikuje stav motoru. Je zde H, pokud je motor zapnutý, resp. pokud se otáči. Náběžná hrana na bit MOTOR přepne momentální stav motoru. Zapíná ho, pokud byl vypnutý a naopak.

7.2 Fyzický formát záznamu.

Je použita frekvenční modulace s průměrnou rychlostí okolo 1350 Baudů. Tvary a časové průběhy signálů znázorňuje tento obrázek:



7-1 Fyzický formát záznamu

7.3 Logický formát záznamu

Je popsán v tabulce. Není-li uvedeno jinak, jsou údaje uloženy v bytech, tzn. jsou spojeny jedním START/STOP bitem typu LONG. Každý soubor se skládá ze dvou částí: hlavičky a dat. Před hlavičkou i daty je odlišný "tape mark" pro rozlišení. Mezi hlavičkou a daty se zastavuje magnetofon. Blok hlavičky i blok dat je uložen vždy dvakrát. Před hlavičkou i blokem je cca 2 sekundy úroveň LOW.

22000 SHORT	Zaváděcí tón 10s
40 LONG	Tape mark 1.část
40 SHORT	Tape mark 2.část
1 LONG	start bit
128 bytů	Hlavička (popsána dále)
2 byty	Kontrolní součet
1 LONG	stop bit
256 SHORT	oddělovač druhého bloku
128 byty	Hlavička (podruhé ta stejná)
2 byty	Kontrolní součet
1 LONG	stop bit

7-2 Formát záznamu hlavičky

11000 SHORT	Zaváděcí tón 5s
20 LONG	Tape mark 1.část
20 SHORT	Tape mark 2.část
1 LONG	start bit
<n> bytů	Data (délka je v hlavičce)
2 byty	Kontrolní součet
1 LONG	stop bit
256 SHORT	oddělovač druhého bloku
<n> bytů	Data (podruhé ta stejná)
2 byty	Kontrolní součet
1 LONG	stop bit

7-3 Formát záznamu bloku dat

Tento formát podstatně snižuje rychlost záznamu a dokonce není ani tak spolehlivý jak by se zdálo. Praktické zkušenosti ukazují, že pokud se v prvním datovém bloku vyskytne chyba, tak se stejně většinou nepodaří nahrát druhý blok. Je to pravděpodobně způsobeno tím, že obslužná rutina přijde na chybu až po přečtení kontrlolního součtu. Jako kontrolní součet ale pravděpodobně přečte buď oddělovací nuly mezi bloky nebo při větší chybě až druhý blok. Potom se na něj ani nemůže zasynchronizovat.

7.4 Hlavička souboru

Ze 128 bytů hlavičky souboru se využívá pouze 24, zbývajících 104 bytů je označeno jako komentář a může být využito nestandardními soubory pro uchování dalších důležitých informací nebo zde může být krátký inicializační program.

Struktura hlavičky souboru:

1 byte	typ souboru
17 bytů	název souboru ukončený CR, tedy maximálně 16 znaků jména
2 byty	délka souboru
2 byty	zaváděcí adresa
2 byty	startovací adresa
104 bytů	není využito

Přehled přípon pro jednotlivé typy souborů:

01 OBJ ..... spustitelný program ve strojovém kódu
02 BTX
03 BSD
04 BRD
05 RB ..... programy v BASICu ve sbalené formě
07 LIB
0A SYS
0B GR

Ostatní kódy jsou považovány za nestandardní a jsou značeny příponou ???.

7.5 Podprogramy pro práci s CMT:

Vstupní vektor MZ-700 monitoru zpřístupňuje pět základních služeb WHEAD, WDATA, RHEAD, RDATA a VERIF. Existují také podprogramy pro řízení CMT: a pro výstup či vstup bytu, bitu. U všech dále uvedených podprogramů platí, že chrání všechny registry kromě AF a pokud došlo k chybě vracejí nastavený CY.

7.5.1 @WHEAD 0021H

Hlavičku souboru, která je v systémové oblasti od adresy HEAD (10F0H) nahraje na pásku.

7.5.2 @WDATA 0024H

Nahraje na pásku blok paměti podle informací obsažených v hlavičce, která je uložena v systémové oblasti.

7.5.3 @RHEAD 0027H

Přečte hlavičku souboru a uloží ji do systémové oblasti.

7.5.4 @RDATA 002AH

Podle informací z hlavičky přečte program z pásky a ukládá ho na místo určení. Je-li uvedena v hlavičce jako délka souboru nula tato rutina se vrátí, aniž by se snažila něco číst. Obdobně i WDATA neprovede žádnou akci, je-li nulová délka bloku.

7.5.5 @VERIF 002DH

Porovná blok z pásky s blokem paměti podle informací uložených v hlavičce, která je v systémové oblasti.

7.5.6 @RBLOK 050EH

Přečte z pásku blok dat o délce BC a uloží od adresy HL. Počítá kontrolní součet a porovná ho s kontrolním součtem přečteným z pásky.

7.5.7 @VBLOK 05ADH

Porovná obsah bloku o délce BC, uloženého od adresy HL a blokem čteným z pásku. Vytváří a porovná i kontrolní součet.

7.5.8 @CHECK 071AD

Vypočítá kontrolní součet bloku o délce BC, který je umístěn od adresy HL a uloží ho na systémové adresy MGCRC (1197H) a MGCRCV (1199H). Kontrolním součtem se rozumí počet bitových jedniček v souboru.

7.5.9 @WBYTE 0767H

Zapíše na CMT byte uvedený bitem typu LONG.

7.5.10 @RBYTE 0624H

Přečte byte z pásky. Aktualizuje kontrolní součet na adrese MGCRC (1197H).

7.5.11 @MG0 0A01H

Zapíše bit typu SHORT.

7.5.12 @MG1 0A1AH

Zapíše bit typu LONG.

7.5.13 @WTMRK 077AH

Zapíše zaváděcí signál a TAPE MARK. V registru E očekává CHEAD (0CCH) jako příznak hlavičky nebo cokoli jiného jako příznak datového bloku.

zaváděcí signál TAPE MARK:

hlavička	22000 SHORT 40 LONG a 40 SHORT
datový blok	11000 SHORT 20 LONG a 20 SHORT

7.5.14 @RINTR 0FE2H

Čeká na souvislou oblast aspoň 100 bitů typu SHORT na pásce. Používá se k nalezení zaváděcího tónu.

7.5.15 @RTMRK 065BH

Počká na zaváděcí signál a přečte TAPE MARK. V registru E očekává buď CHEAD (0CCH) nebo cokoli jiného. (význam jako u WTMRK). Po přečtení TAPE MARK počká na náběžnou hranu dalšího bitu.

7.5.16 @MGON 069FH

Pokusí se desetkrát zapnout CMT: a když zjistí, že mechanika není zapnuta, napíše hlášení a čeká na její zapnutí nebo BREAK. V registru D se očekává CWRITE (0D7H) jako příznak zápisu nebo cokoli jiného jako příznak čtení. (Má význam pouze pro tvar hlášení, které žádá o zapnutí CMT:) Po zapnutí magnetofonu se čeká téměř 2 sekundy na ustálení chodu motoru.

7.5.17 @MGOFF 0700H

Desetkrát se pokusí vypnout magnetofon a když se to nepodaří, tak se vrátí.

7.5.18 W0TO1 0601H

Počká na náběžnou hranu signálu z CMT:. Vrací CY = 1 pokud detekuje BREAK. Očekává v BC a DE připravené adresy. (V BC KBDIN (0E001H) a v DE PORTC (0E002H)).

8 Reálný čas

Reálný čas je udržován obvodem I8253. Tento obvod obsahuje tři na sobě nezávislé dekrementující čítače. Ke každému čítači je přiveden vstup hodinových signálů a signál GATE zabraňující čítání a vyveden výstup čítače.

I8253 je dostupný v MZ-800 režimu na portech 0D4H-0D7H. V režimu MZ-700, ve kterém také pracují dále popisované podprogramy, je I8253 namapován na adresách 0E004H-0E007H. Nejvyšší adresa je adresa řídícího slova, druhá nejvyšší je adresa CTC2, dále CTC1 a CTC0.

CTC0 je užit k řízení akustických výstupů, CTC1 a CTC2 jsou spojeny do kaskády tak, že výstup CTC1 je připojen na vstup CTC2. Na vstup CTC1 je přiveden kmitočet řádkového televizního rozkladu o frekvenci 15611 Hz.

CTC1 pracuje jako dělič kmitočtu. Na jeho výstupu je frekvence 1 Hz. CTC2 počítá počet sekund, které uplynuly od půlnoci nebo od poledne. Jakmile CTC2 dočítá do nuly a je povoleno přerušení bitem PC2 I8255, dojde k přerušení. Přerušovací rutina změní příznak dopoledne/odpoledne a inicializuje CTC2. Kdyby se v přerušovací rutině CTC2 neinicializovalo, došlo by k dalšímu přerušení ihned po návratu z přerušovacího podprogramu.

8.1 Podprogramy pro práci s reálným časem

8.1.1 @TIMST 0308H

Nastavení času, inicializace CTC1 a CTC2 I8253. V **DE** registru očekává čas v sekundách, který uplynul od začátku půldne, v **A** registru příznak 0/1 = dopoledne/odpoledne.

8.1.2 @TIMRD 0358H

Čtení aktuálního času. V **DE** registru vrací počet sekund uplynulých od začátku dne, v **A** registru příznak dopoledne/odpoledne.

8.1.3 @CLOCK 038DH

Standardní rutina obsluhy přerušení. Nastavuje CTC2 na hodnotu 43200, což je délka dvanácti hodin v sekundách. Dále rutina přepne příznak dopoledne/odpoledne. Aktivuje se s periodou 12 hodin.

ROM monitor pracuje v přerušovacím režimu IM 1. Na adrese 38H je skok do RAM na 1038H. Proto na této adrese musí vždy začínat rutina přerušení. Standardně je tam skok na @CLOCK.

9 Akustický výstup

9.1 Řízení akustického výstupu

V zásadě existují tři možnosti řízení akustického výstupu:
1) Přímý výstup na PC0 I8255
2) Nastavení periody tónu do I8253
3) Práce s PSG (SN76482)

Na tomto místě se zabýváme pouze způsoby 1) a 2), způsob 3) je podrobně popsán ve firemním manuálu.

Tón je generován obvodem I8253 a to čítačem CTC0. Na vstup tohoto čítače je přivedena frekvence 554202 Hz (frekvence krystalu/32). CTC0 pracuje jako dělič kmitočtu se stejnou střídou signálu (mód 3 I8253). Frekvence se do I8253 nastaví následujícími instrukcemi:

```
LD    A,36H          mód 3 obvodu I8253
LD    (0E007H),A     nastavit mód a zastavit hudbu

LD    DE,554202/<freqence>
LD    HL,0E004H      adresa CTC0
LD    (HL),E         dolní byte frekvence
LD    (HL),D         horní byte frekvence
```

Na adrese 0E008H je signál GATE0, který zapíná a vypíná čítač.

```
XOR   A
LD    (0E008H),A     povolit čítání

LD    A,1
LD    (0E008H),A     povolit čítání
```

Aby se signál z I8253 dostal na AUDIO IN obvodu PSG, je nutné otevřít cestu nastavením PC0. Toho lze využít i pro softwarové ovládání melodií, kdy se nespustí CTC0, ale softwarově kmitá s PC0.

9.2 Podprogramy pro akustický výstup

Všechny podprogramy nastavují periodu tónu do CTC0 I8253, proto lze zahrát jen jednoduché tóny. S PSG tyto podprogramy nepracují. Není zaručena ochrana všech registrů.

9.2.1 @MELDY 0030H

Zahraje melodii. Adresu vstupního řetězce očekává v registru DE. Řetězec je ukončen znakem CR nebo 0C8H. Obsahuje definici not a jejich délek podle konvencí MZ 700 Basicu.

9.2.2 @MSTA 0044H

Zapne hraní hudby o periodě v proměnné FREQ (11A1H) a nechá ji hrát, dokud nebude vypnuta rutinou MSTP.

9.2.3 @MSTP 0047H

Vypne všechny zvuky generované I8253, t.j. hudbu spuštěnou podprogramem @MELDY nebo @MSTA.

9.2.4 @MELTB 031CH

Zpracuje jednu notu i s její délkou ze vstupního řetězce, adresovaného registrem DE. Registr DE se posune na další notu. Perioda noty je hledána v tabulce HL a uložena do proměnné FREQ (11A1H). Délka noty upravená podle nastaveného tempa se vrací v C registru.

9.2.5 @MELW 028CH

Čeká až dohraje tón, délka čekání je v B registru.

9.2.6 @XTEMP 02E5H

Nastaví tempo melodie 0-7. Hodnotu tempa očekává v A registru. Do proměnné TEMPO (119EH) uloží hodnotu výrazu (8 - obsah A registru.)

9.3 Tabulky melodií

Obsahují označení not a periody, které se ukládají do I8253. Jedna tabulka je pro normální noty, druhá pro noty označené znakem "#".

```
DEFB  "C"
DEFW  perioda noty C
DEFB  "D"
DEFW  perioda noty D
 . .
DEFB  "R"
DEFW  0
```

9.3.1 !MEL 026CH

Tabulka základních not

9.3.2 !MEL# 0284H

Tabulka zvýšených not

9.3.3 !MELEN 029CH

Tabulka délek not

DEFB 1,2,3,4,6,8,12,16,24,32

10 Monitor 9Z-504M

Horní monitor obsahuje kromě příkazů shodných s dolním monitorem také obsluhu RD:, FD: a QD:. Většina podprogramů pro realizaci příkazů je stejná jako v dolním monitoru. Proto nejsou v horní části duplicitní rutiny podrobně komentovány.

Nejsou popsány takřka žádné záležitosti, které se týkají QD:. Máme pro to mnoho různých důvodů, ale pokud by někdo cítil potřebu mít v tomto textu komentáře k této fantastické jednotce s obrovskou kapacitou a fantastickou přístupovou rychlostí (64KB, průměrně 8 sekund, ale klidně až 20), tak nám je může poslat a mi je sem doplníme.

Horní monitor je stavěn s ujasněnou koncepcí ovládání periferních zařízení. Všechny soubory jsou standardně zaváděny od MGBASE (1200H) a dostávají při svém spuštění v BC informaci o zařízení ze kterého byly odstartovány.

BC	periferie
0000H	RD: Sériová paměť
0100H	CMT: Magnetofon
0200H	FD: Floppy disk
0300H	QD: Quick disk

Ke spuštění programu slouží rutina]GOPGM (0ECFCH). Očekává v HL registru adresu tabulky souboru, která musí obsahovat délku, zaváděcí adresu a startovací adresu. V BC' je hodnota, která se má předat do spuštěného programu.

Program musí být uložen od MGBASE (1200H). Má-li se program vykonávat na jiné adrese než MGBASE, je tam před spuštěním přesunut.

10.1 Přehled povelů horního monitoru

10.1.1 J = Jump

Jxxxx, předá řízení na adresu xxxx

10.1.2 G = Gosub

Gxxxx, uloží do zásobníku návratovou adresu do monitoru a předá řízení na adresu xxxx.

10.1.3 L = Load

Přečte program z CMT: a předá mu řízení. Program se ukládá od MGBASE (1200H) a pak volá standardní spouštěcí rutinu, která se postará o jeho umístění v paměti a spuštění.

10.1.4 F = Floppy disk

Pokusí se přečíst z diskety a spustit program, jehož název začíná IPLPRO... a jehož hlavička je na začátku diskety. (Jde vlastně o BOOT příslušného diskového operačního systému).

10.1.5 B = Beep ON/OFF

Zapne nebo vypne akustický signál po stisku každé klávesy.

10.1.6 M = Modify

Mxxxx umožní měnit a prohlížet obsah paměti od adresy xxxx dále. Vykonávání se zruší klávesou BREAK. Pokus o zapsání nesprávné hodnoty je prostě ignorován a je očekáván vstup hodnoty správné.

10.1.7 S = Save

Uložení programu na pásku. Typ je 01 tedy OBJ.

10.1.8 V = Verify

Porovnání obsahu souboru se skutečným stavem paměti.

10.1.9 D = Dump

Dxxxxyyyy, výpis paměti v šestnáctkové soustavě a "ASCII". Implicitní hodnota počtu bytů pro výpis je 000AH.

10.2 Povely pro práci se SRAM

SRAM je vnější *sériová* paměť umožňující dočasné uschování jednoho souboru. (Vlastně se jedná o RAM disk, který dovoluje sériový přístup k datům, která jsou na něm uložena)

10.2.1 ES = SRAM disk save

Přečtení programu z CMT: ve standardním formátu a uložení do sériové paměti. Hlavička se redukuje na 8 bytů: délka, zav.adr., start,adr. a CRC hlavičky. Bezprostředně za ní následuje program ukončený kontrolním součtem. Program je z CMT: čten od adresy MGBASE (1200H) a teprve potom uložen do SRAM:

10.2.2 EB = SRAM disk load

Přečte program ze sériové paměti zavedený příkazem ES a spustí ho.

10.3 Povely pro práci s QUICK-diskem

QUICK-disk je sekvenční datová periferie pracující na principu magnetického záznamu dat na pružné disky. Kapacita jednoho média je 64 KB.

10.3.1 QL = QUICK-disk Load

Zeptá se na jméno souboru a přečte ho z QUICK-disku od adresy 1200H. Soubory jiného typu než OBJ číst neumí.

10.3.2 QS = QUICK-disk Save

Zeptá se na jméno programu, počáteční koncovou a zaváděcí adresu. Pokud soubor uvedeného jména není na QUICK-disku, uloží ho za poslední nalezený soubor.

10.3.3 QF = QUICK-disk Format

Naformátování QUICK-disku. Všechna data z formátovaného média budou ztracena. Přiští příkaz QS ukládá od začátku disku.

10.3.4 QD = QUICK-disk Directory

Výpis adresáře QUICK-disku. Vypisuje jen typ a název souboru.

10.3.5 QC = QUICK-disk Copy

Kopíruje soubory QUICK-disk = = > QUICK-disk. Zeptá se na jméno souboru, nahraje ho od adresy 1200 a vyzve operátora k založení disku, na který se bude soubor kopírovat.

10.3.6 QX = QUICK-disk Xcopy

Kopíruje soubory CMT: = = > QUICK-disk. Přečte soubor z magnetofonu na adresu 1200, vypíše jeho název a zeptá se, jestli ho smí zkopírovat na QUICK-disk. Kopírování opačným směrem neexistuje.

11 IPL loader

IPL loader je umístěn od adresy 0E800H a po zapnutí počítače nebo po RESET je aktivován jako první. Provede příslušné inicializace periferních obvodů a nastaví režim MZ-700.

Pokud zjistí, že je stisknuta klávesa CTRL, tak namapuje RAM přes celou paměť a provede JP 0000.

Jinak nabídne M jako přechod do monitoru a C jako nahrání a spuštění programu z CMT:. Pokud jsou připojeny QD: je nabídnuta i varianta Q pro spuštění programu z QD:

Jsou-li připojeny FD: je automaticky proveden pokus o přečtení programu jehož název začíná IPLPRO.

11.1 Služby pro kopírování programů

11.1.1 @COPYL 0E807H

Přečte soubor z CMT: a uloží ho od standardní zaváděcí adresy MGBASE (1200H).

11.1.2 @COPYS 0E80AH

Zapíše přečtený soubor zpět na CMT:. **POZOR !** není možné aktivovat dvakrát bezprostředně po sobě. COPYS naruší tabulku adres v hlavičce souboru. Každému volání COPYS musí tedy předcházet volání COPYL.

11.1.3 @COPYV

Porovná program uložený od MGBASE s programem na pásce.

12 Floppy disk

12.1 Připojení FD:

Z rozboru softwaru vyplývá, že disk je připojen přes řadič WD 2795 nebo obdobný. Ovládá se přes tyto porty:

0D8H	příkazový registr
0D9H	registr stopy
0DAH	registr sektoru
0DBH	registr dat
0DCH	zapínání a vypínání mechaniky
0DDH	příznak vybrané strany diskety v nultém bitu

12.2 Logický formát

@FDBOOT vyžaduje, aby byla v prvním sektoru nulté stopy standardní hlavička souboru typu 03 = BSD, s názvem začínajícím IPLPRO. Na adrese FDHEAD + 30 musí být číslo bloku, kde začíná program na disketě.

< Blok > = < stopa > * 16 + < sektor > - 1

Další řízení diskových operací musí převzít nahraný program IPLPRO..., který může přizpůsobit logický formát pro jakýkoliv diskový operační systém.

12.3 Podprogramy pro práci s FD:

Všechny podprogramy musí mít na standardní adrese FDARET (0CEFEH) nastavenu adresu podprogramu pro ošetření chyb.

Struktura standardní tabulky pro práci s FD:

1 byte	číslo disku
2 byty	číslo bloku od začátku disku
2 byty	délka čtených dat v bytech
2 byty	závaděcí adresa
1 byte	číslo aktuální stopy
1 byte	číslo aktuálního sektoru
1 byte	počáteční stopa
1 byte	počáteční sektor

Adresa této tabulky se podprogramům předává v IX registru.

12.3.1 @FDBOOT 0E44AH

Provede BOOT standardního programu z FD: a předá mu řízení. Pokud na disketě nenalezne program IPLPRO..., vrátí se s chybovým hlášením do monitoru. Program, kterému bylo předáno řízení smí použít instrukce RET pro návrat do monitoru.

12.3.2 @FDREAD 0E5A7H

Přečte z FD: soubor, podle informací uložených v tabulce, jejíž adresu očekává v IX. Nechrání registry.

12.3.3 @FDON 0E517H

Zapne motor diskové mechaniky a počká cca 960 milisekund. Do proměnné FDON? (0CEF5) uloží jedničku, jako příznak zapnutí disku.

12.3.4 @FDSEL 0E4DCH

Vybere a zapne disk, jehož číslo očekává na adrese IX + 0. Informace o tom, který disk je vybrán jsou uloženy v tabulce !FDRES (0CEF6H). Nechrání registry.

12.3.5 @FDDESL 0E530H

Vypne a odpojí všechny diskové mechaniky. Chrání všechny registry.

12.3.6 @FDTR0 0E548H

Vystaví hlavičku na 0. stopu.

12.3.7 @FDSTOP 0E658H

Ukončí operaci na FD:

12.3.8 @FDTR 0E61BH

Nastavení stopy podle A registru.

12.3.9 @FDSEC 0E62BH

Nastavení sektoru a stopy podle tabulky, jejíž adresu očekává v IX registru.

12.3.10 @FDSEEK 0E528H

Provede příkaz SEEK řadiče, musí být správně nastaven registr stopy a registr dat.

12.3.11 @FDTR? 0E696H

Podle údajů z tabulky, jejíž adresa se předává v registru IX, provede výpočet počátečního čísla stopy (vrací v H) a sektoru (vrací v L) a uloží na příslušná místa do tabulky.

12.3.12 @FDNEXT 0E63CH

Zvětší číslo stopy, pokud detekuje, že je nastaven sektor číslo 17 a vybere sektor číslo 1. V IX je adresa tabulky.

12.3.13 @FDCMD 0E555H

Vyšle z A registru příkaz řadiči FD a počká s time outem na potvrzení o dokončení operace.

12.3.14 @FDSTRT 0E64EH

Vyšle z A registru příkaz řadiči FD a počká s time outem na potvrzení o zahájení operace.

DWT1 0E568HE = Čeká na FD ready.

12.3.15 @FDWT2 0E587H

Čeká na FD not ready.

12.3.16 ???FD 0E8D5H

Otestuje, zda jsou připojeny disky. Vrací Z = 0 pokud ne.

13 SRAM: Sériová paměť

13.1 Připojení SRAM:

SRAM: se může nacházet na I/O adresách od 0F8H nebo na adresách od 0A8H. V dalším textu předpokládám adresy 0F8H.

SRAM: se plní a čte pouze sekvenčně, lze užívat jednoduché I/O instrukce nebo také instrukce INIR a OTIR.

IN A,(0F8H) Rewind neboli nastavení ukazatele současné pozice na začátek SRAM:.

OUT (0FAH),A Zápis bytu z registru A do SRAM: na pozici ukazatele, ukazatel se zvětší o 1.

IN A,(0F9H) Čtení bytu ze SRAM:, z pozice ukazatele, ukazatel zvětší o 1.

13.2 Podprogramy pro práci se SRAM:

V ROM je jen několik podprogramů a to těch, které jsou užity příkazy ES a EB monitoru.

Všechny podprogramy očekávají v C registru I/O adresu SRAM: (0F8H nebo 0A8H).

13.2.1 @???RD 0E7BAH

Zjistí, je-li připojena SRAM. Jestliže není, nastaví CY.

13.2.2 @RDCRC 0E70EH

Spočítá kontrolní součet bloku o adrese DE a délce BC. Vrátí ho v registru HL. Tento podprogram zničí i druhou sadu registrů.

13.2.3 @?HEAD 0E729H

Zjistí, zda je ve SRAM: hlavička se správným kontrolním součtem. Když je, vrací Z = 1, není-li tam, vrací Z = 0.

13.2.4 @RDLOAD 0E6DAH

Přečte program ze SRAM: na adresu 1200H a spustí ho. Jestliže tam nebyl, vrací Z = 0.

14 QUICK-disk

QUICK-disk je řízen obvodem Z80-SIO, který je očekáván na adresách 0F4H-0F7H.

Kapacita 64 KB, rychlost přístupu kolem 8 sekund a sekvenční přístup. Takto by se dal QUICK-disk stručně charakterizovat.

Periferií takových parametrů jako je QUICK-disk se nemá vůbec smysl zabývat. Uvedené informace jsou proto velmi kusé.

14.1 Podprogramy pro práci s QUICK-diskem

14.1.1 @QDISK 0E010H

Podprogram zprostředkující veškerý styk s uživatelem a QUICK-diskem. Jeho funkce je určena příkazem v proměnné QDCMD (1130H):

QDCMD

1 kontrola připravenosti disku
2 formátovaní disku
3 čtení dat z disku
4 zápis dat na disk
5 čtecí hlavu na začátek disku
6 vypnutí QUICK-disku

V proměnné QDPAR (1131H) dostává bližší informace.

14.1.2 @MTON 0E29BH

Zapne motor, prečte číslo souboru a CRC.

14.1.3 @MTOFF 0E2E8H

Vypne motor QUICK-disku

14.1.4 @QDCRC 0E3C3H

Přečte CRC a zkontroluje ho.

14.1.5 @EOMSG 0E3B2H

Uloží koncový záznam souboru a CRC.

14.1.6 @QRBYT 0E3F0H

Přečte byte z QUICK-disku do registru A.

14.1.7 @QWBYT 0E3DBH

Zapíše byte z registru A na QUICK-disk.

15 Podpora BASICu

Interpretr Basicu MZ-800 není zcela samostatný a nezávislý na obsahu paměti ROM. Z dolní části monitoru využívá rutiny ADCN a DACN, dále pak tabulky klávesnice !KBD, !KBDC, !KBDG, !KBDGS. Horní část paměti ROM od adresy 0F400H do konce je určena výhradně pro použití Basicem. Obsahuje tyto rutiny:

- práce s bufferovanou tiskárnou
- přístup k RD:
- nastavení a čtení reálného času
- obsluha joysticku (i simulovaného klávesnicí)
- podprogram pro přímý výstup na tiskárnu

Podprogramy předpokládají, že budou pracovat v režimu MZ-800. Práce se simulovaným joystickem a s reálným časem není v režimu MZ-700 provozuschopná vůbec.

Od adresy 0FDA0H do konce paměti jsou umístěny chybové zprávy Basicu. Jednotlivé textové řetězce navazují těsně za sebe. Začátek řetězce je indikován nastavením osmého bitu prvního znaku. Texty obsahují jen velká písmena, pro přechod od velkých k malým je v řetězci zařazen znak 05H. Každé chybové zprávě je přiřazeno číslo chyby totožné s pořadím zprávy v seznamu (počítáno od 1).

16 Tiskárna

16.1 Připojení LPT:

Tiskárna je připojena přes Z80-PIO. Port A je využit pro řídící signály tiskárny, port B pro přenos dat. Při převzetí znaku tiskárnou lze vyvolat přerušení prostřednictvím obvodu PIO v libovolném přerušovacím režimu procesoru. Toto přerušení obvykle vede k vyslání dalšího znaku na tiskárnu z výstupní fronty.

16.2 Obsloužení chyb

Při práci s tiskárnou může docházet k chybovým stavům. Proto si každý program před voláním dále popisovaných podprogramů musí nastavit adresu podprogramu pro obsluhu chyb na adresu ERRSP (12A8H). Po chybovém návratu má SP hodnotu 12AAH, proto je vhodné nejdříve zásobník obnovit.

Rutina obsluhy chyb dostane na adrese ERRFLG (13D9H) typ chyby:

0FFH to je skutečně chyba, její číslo je v registru A. (Je shodné s číslováním chyb v BASICu)

01H detekován stisk SHIFT + BREAK při @HCOPY, jiný podprogram na break nereaguje.

02H tento návrat nastane z příkazu INIT "LPT:M2", pokud už byl nastaven bufferovaný tisk. Obsluhu tohoto parametru si musí udělat volající program sám.

16.3 Podprogramy pro práci s LPT:

Vstupní vektor na adrese 0F400H obsahuje všechny důležité podprogramy pro práci s bufferovanou tiskárnou. Podprogramy většinou chrání všechny registry kromě AF, jak je patrno z výpisu programu. Dále popisujeme i některé podprogramy, které nejsou obsaženy ve vstupním vektoru, ale mohou být užitečné. Bez RD: nelze bufferovaný tisk použít.

16.3.1 @INLPT 0F418H

Inicializační rutina tiskárny. Na adrese HLPAR (12AEH) očekává adresu zbytku řetězce příkazu INIT "LPT:... Parametry M a S jsou popsány ve firemním manuálu, parametr Q má následující význam:

0.bit = 1 po odeslání znaku na tiskárnu se nečeká, až tiskárna znak převezme.
0.bit = 0 čeká se na převzetí znaku tiskárnou (i v přerušovací rutině)
1.bit logika signálu IRT 0/1 = pozitivní/negativní
2.bit logika signálu RDP 0/1 = pozitivní/negativní
Je-li v příkazu užit parametr Q nebo M0, dojde k inicializaci výstupní fronty.

16.3.2 @INICF 0F41EH

Inicializuje výstupní frontu tiskárny. Adresu počátku fronty najde na adrese 0 a 1 v RAM disku, konec fronty je vždy 0FFFFH.

16.3.3 @PSTR 0F8F1H

Pošle řetězec znaků na tiskárnu. Rutina má speciální volání:

CALL @PSTR
DEFB < počet bytů >
DEFB < 1.byte >, < 2.byte >,.... < n.byte >

16.3.4 PCHR1,PCHR2 0F412H,0F415H

Výstup znaku na tiskárnu s konverzí podle typu tiskárny. Obě volání se téměř neliší. V případě tisku na ASCII tiskárnu se očekává v IY registru adresa rutiny pro převod znaku do ASCII. Návrat z této uživatelské rutiny se provede instrukcí JP (IX). To má tu výhodu, že lze použít rutin v ROM pro jakoukoli tiskárnu.

16.3.5 @CRLPT 0F41BH

Pošle znak CR definovaný rutinou INLPT na tiskárnu

16.3.6 @SLPT 0F634H

Pošle mezeru na LPT:

16.3.7 @BTLPT 0F41BH

Znak z registru A bez jakékoli konverze pošle na LPT: nebo ho uloží do výstupní fronty. Všechny výše popsané rutiny volají pro vlastní výstup znaku tento podprogram.

16.3.8 @BTLP1 0F636H

Chová se stejně jako rutina BTLPT, jenomže neschovává registry do zásobníku.

16.3.9 @BTLP2 0F637H

Pošle znak z B registru na LPT:, chrání jen indexregistry.

16.3.10 @INTP 0F400H

Rutina obsluhy přerušení od LPT:. Je-li ve frontě znak a není-li tisk pozastaven rutinou @SSLPT, pošle ho na LPT: a očekává další přerušení. Nechrání registry.

16.3.11 @INTPX 0F6BEH

Obslužná rutina tiskárny s ochranou registrů.

16.3.12 @LPTSS 0F421H

Pozastaví výstup znaků z bufferu na tiskárnu

16.3.13 @LPTGO 0F424H

Obnoví tisk pozastavený podprogramem @LPTSS

16.3.14 @SSGO 0F427H

Pozastaví nebo obnoví tisk v závislosti na předchozím stavu.

16.3.15 @NOBUF 0F42AH

Bufferovaný tisk přepne na nebufferovaný. Znaky ve frontě vystupují dál na tiskárnu mezi novými znaky. Proto je potřeba po zavolání @NOBUF počkat, až se fronta vyprázdní.

16.3.16 @PLNA? 0F6B0H

Nastaví CY, když není místo v bufferu tiskárny. V opačném případě ho nuluje.

16.3.17 @BTISK 0F6F0H

Pošle znak na tiskárnu přímo z A registru. Předpokládá, že tiskárna je zapnuta a připravena přjmout znak.

16.3.18 @PSTB0 0F705H

Shodí signál "data platná".

16.3.19 @PSTB 0F706H

Nastaví nebo nuluje signál "data platná". Respektuje pozitivní nebo negativní logiku nastavenou parametrem Q v příkazu INIT.

16.3.20 @PSTAT 0F70FH

Zajistí, aby tiskárna byla připravena přijmout znak. Jestliže se tiskárna do tolika sekund, kolik je definováno proměnnou PTO (1094H) neozve, končí chybou.

Tento podprogram je volán vždy při posílaní znaku na tiskárnu.

16.3.21 @HCOPY 0F40FH

Opíše obsah obrazovky v režimu MZ-800 na tiskárnu MZ-80P5/K/. Jinou tiskárnu ignoruje.

16.3.22 @PBYTE 0F436H

Pošle byte z registru A přímo na tiskárnu. Není-li tiskárna schopna pracovat, vrací v registru A hodnotu 0FFH, převezme-li tiskárna znak, vrací nulu.

Tato rutina je zcela samostatná. Nesouvisí z výše popsanými rutinami, nespolupracuje s výstupním bufferem, parametr Q v příkazu INIT nemá na činnost tohoto podprogramu vliv. Logika řídících signálů se musí nastavit přepínači na zadní stěně počítače.

17 RAM disk pro Basic

Rutiny pro práci s bufferovanou tiskárnou používají pro svůj buffer RD:. Kapacita tohoto média je 64 kB, přístupová doba k jednomu bytu je několik mikrosekund. Dolní část RD: se využívá na soubory Basicu, v horní části je buffer tiskárny. Mezní adresa, t.j. adresa prvního bytu v bufferu, je na adrese 0 v RD:. Zde popisované rutiny jsou využity jen pro LPT:, v Basicu jsou napsány identické rutiny pro práci se soubory.

17.1 Připojení RD:

0EBH registr dolní části adresy, horní část adresy se posílá při adresaci portu 0EBH v horní části adresové sběrnice procesoru. Proto se musí RD: adresovat přes BC registr.

0EAH port pro vstup a výstup dat z a na RD:

17.2 Podprogramy pro řízení RAM disku

17.2.1 @RDOA 0F743H

Výstup bytu z registru A na adresu HL.

17.2.2 @RDIA 0F74EH

Vstup bytu z adresy HL do registru A.

17.2.3 @RDODE 0F759H

Výstup slova z registru DE na adresu HL.

17.2.4 @RDIDE 0F759H

Vstup slova z adresy HL do registru DE.

18 Nastavení reálného času

Čas se nastavuje do obvodu I8253 způsobem popsaným v kapitole **Reálný čas**. Rozdíl je v adresování I8253. V následujících rutinách se používá adresování instrukcemi vstupu a výstupu. Proto je nelze použít v režimu MZ-700.

Formát nastavovaného a čteného času: **DE** = čas v sekundách od začátku půldne **A** = příznak dopoledne/odpoledne = 0/1

Nastavovaný čas se uloží do tříbytového prostoru v RAM. I8253 začíná čítat od nuly. Uložený čas se potom použije ke korekci při čtení času. Tento systém má tu výhodu, že lze bez problémů mít k dispozici několik časů.

18.1 Podprogramy pro práci s reálným časem

18.1.1 @TMSTX 0F433H

Nastaví reálný čas. V HL očekává adresu tříbytové položky v RAM pro uložení času.

18.1.2 @TMGTX 0F436H

Čte čas. V HL je totéž co u @TMSTX.

18.1.3 @TMSET 0F436H

Nastaví čas. Adresa úschovy času je 1366H.

18.1.4 @TMGET 0F406H

Získá čas. Na adrese 1366H očekává počáteční čas, uložený podprogramem @TMSET.

19 Obsluha joysticku

19.1 Připojení joysticku

Joysticky jsou k počítači připojeny prostřednictvím portů na adresách 0F1H respektive 0F2H. Po provedení instrukce IN A,(port) dostaneme do registru A obraz stisknutých spínačů. Jednička odpovídá stisknutému spínači.

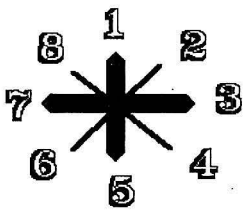
bit 0 dolů
bit 1 nahoru
bit 2 doprava
bit 3 doleva
bit 4 trigger

Aby mohl být joystick otestován, musí být vybrán bitem A5 respektive A6 I8255.

19.2 Podprogramy pro joystick

Podprogramy v ROM jsou přímou vazbou na příkazy Basicu STICK a STRIG. V registru A očekávají číslo joystiku jako v Basicu:

0 = simulovaný na klávesnici
1 = joystick číslo 1 (I/O adresa 0F0H)
2 = joystick číslo 2 (I/O adresa 0F1H)



19.2.1 @STICK 0F409H

V registru A vrací stav joysticku 0 až 8 podle následujícího schématu. Když je ovládací páka v klidu, vrací nulu.

Data přečtená z portů jsou ale v jiném formátu. Pro konverzi se použije šestnáctibytová tabulka !JOY12 (0FA1DH) pro skutečný joystick nebo !JOY0 (0FA2DH) pro joystick simulovaný.

19.2.2 @STRIG 0F49CH

V registru A vrací stav tlačítka "trigger".

0 .. trigger nestisknut

1 .. trigger stisknut

19.2.3 @JOYIN 0F9FFH

Přečte byte z portu 0F0H resp. 0F1H je-li hodnota A registru 1 nebo 2. Simulovaný joystick (číslo 0) neobsluhuje.

19.2.4 @KBLIN 0FA15H

Přečte sloupec matice klávesnice. V registru A očekává číslo tohoto slopce ve spodních čtyřech bitech. Přečtený byte vrací v registru A. (Užíváno pro simulovaný joystick)

20 Užitečné podprogramy

V oblasti nad 0F400H se nachází některé podprogramy, pokusíme se o stručný přehled těch, které mohou být užitečné.

Také je zde soubor chybových hlášení BASICu. Proč je v ROM něco takového to ví jen pracovníci firmy SHARP.

20.1 Obecně užitečné podprogramy.

20.1.1 @BEEPD 0F430H

Akustický signál o frekvenci 545 Hz s délkou v registru D.

< Délka v milisekundách > = D * 2.3

20.1.2 @NUMB 0F76FH

Převede číslo zapsané ASCII v dekadické nebo šestnáctkové soustavě do vnitřního binárního vyjádření. Adresa na řetězec čísel je v HL registru, výsledek se uloží do DE registru. HL pak ukazuje na následující byte za číslem. Veškeré mezery se ignorují.

20.1.3 @NIBLH 0F7BAH

Převede hexa číslici v ASCII v A registru do binárního kódu.

20.1.4 @SKIP 0F7CAH

Přeskočí všechny mezery v řetězci o adrese HL.

20.1.5 @FNDA 0F737H

První nemezerový znak v řetězci HL porovná se zadaným znakem a nastaví flagy. Když se znaky neshodují, ukazuje HL na něj, v opačném případě na znak následující.

Příklad volání:

```
CALL  FNDA
DEFB  <znak>
```

20.1.6 @OUTY 0F940H

Pošle B bytů na porty podle tabulky s adresou HL.
Tabulka má délku 2 * B bytů a následující strukturu:
DEFB <data1>,<port1>
DEFB <data2>,<port2>
. . .
DEFB <dataB>,<portB>
Po skončení podprogramu ukazuje HL za tabulku

20.1.7 @SUBDE 0F9D2H

Je-li HL větší než DE, tak HL od DE odečte.
Vždy přepne nultý bit registru A. (používáno v podprogramech pro reálný čas).

20.1.8 @ADDHL 0F44CH

HL <-- HL + A.

20.1.9 @PUSHA 0F45AH

Uklidí registry včetně AF.
Při návratu přes RET registry opět obnoví.
Ničí registr IX.

20.1.10 @PUSH 0F467H

Uklidí registry kromě AF.
Při návratu přes RET registry opět obnoví.
Ničí registr IX.

20.2 Texty umístěné od 0FDA0H

číslo chyby	adresa textu	text
1	FDA0	Syntax
2	FDA7	Over flow
3	FDB1	Illegal data
4	FDBE	Type mismatch
5	FDCC	String length
6	FDDA	Memory capacity
7	FDEA	Array def.
8	FDF5	Linelength
9	FE00	
10	FE01	GOSUB nesting
11	FE0F	FOR-NEXT
12	FE17	DEF FN nesting
13	FE26	NEXT
14	FE2A	RETURN
15	FE30	Un def. function
16	FE41	Un def. line
17	FE4E	Can't CONT
18	FE5A	Memory protection
19	FE6C	Instruction
20	FE78	Can't RESUME
21	FE86	RESUME
22	FE8C	PAL
23	FE8F	
24	FE90	READ
25	FE94	SWAP level
26	FE9F	
27	FEA0	
28	FEA1	System id
29	FEAB	Framing
30	FEB3	Overrun
31	FEBB	Parity
32	FEC2	
33	FEC3	
34	FEC4	
35	FEC5	
36	FEC6	
37	FEC7	
38	FEC8	
39	FEC9	

40	FECA	File not found
41	FED9	Hardware
42	FEE2	Already exist
43	FEF0	Already open
44	FEFD	Not open
45	FF06	
46	FF07	Write protect
47	FF15	
48	FF16	
49	FF17	
50	FF18	Not ready
51	FF22	Too many files
52	FF31	Disk mismatch
53	FF3F	No file space
54	FF4D	Unformat
55	FF56	Too long file
56	FF64	
57	FF65	
58	FF66	Dev. name
59	FF70	Can't execute
60	FF7E	Illegal filename
61	FF8F	Illegal filemode
62	FFA0	
63	FFA1	Out of file
64	FFAD	Logical number
65	FFBC	LPT:Not ready
66	FFCA	
67	FFCB	
68	FFCC	Dev. mode
69	FFD6	Unprintable
70	FFE2	Check sum
	FFEC	84.10.08 V1.0C

21 Grafický mód zobrazení

21.1 Úvod

Tento díl se jako jediný zabývá problematikou, která není standardně obslužována softwarem v paměti EPROM. Rozhodli jsme se tomu věnovat podrobněji také proto, protože obrazovka je základní periferií, zprostředkující styk mezi počítačem a uživatelem.

Jsou zde všechny doposud známé informace o videoprocesoru, obrazovkové paměti a práci s nimi. Po prostudování tohoto textu by uživatelé měli být schopni efektivně využívat všech vymožeností práce s obrazovkou.

V úvodu do problematiky zobrazování je nutné si položit tři základní otázky:

1) Jaká data musí být v obrazové paměti, aby na obrazovce bylo to co potřebujeme ?

2) Jak softwarově zařídit, aby se data do obrazovkové paměti dostala ?

3) Jak softwarově zjistit, co je ve videopaměti ?

Nejprve se zabýváme problémem 1) v kapitolách o organizaci obrazovky. V kapitolách zápis a čtení se zabýváme strukturou přístupu k obrazovce. Nakonec uvádíme kapitoly, ve kterých je ukázáno praktické využití schopností videoprocesoru, jednak při práci s grafikou a jednak při práci s texty.

21.2 Organizace obrazovky

V MZ-800 módu je obrazovka organizována zcela odlišně než tomu bylo u MZ-700. Obrazovka je plně grafická, lze zobrazit současně až 16 barev. Matice grafické obrazovky má rozměr 320 x 200 bodů, respektive 640 x 200 bodů podle nastaveného režimu. Tomu odpovídá možnost jednoduše zobrazovat 25 řádků textu o 40 respektive 80 znacích. Každý bod může nabývat libovolné barvy nezávisle na ostatních bodech. Počet barev, které můžeme současně vidět na obrazovce se pohybuje v rozmezí 2 až 16. Závisí jednak na velikosti rastru a jednak na velikosti VRAM (video RAM). Standardní MZ-800 totiž nemá nainstalovanou celou VRAM, ale jen její polovinu. Pokud pracuje obrazovka v režimu, ve kterém je možné zobrazit současně 2 nebo 4 barvy, lze je vybrat z 16 možných: základní aditivní: ČERVENÁ, ZELENÁ, MODRÁ, základní subtraktivní: ŽLUTÁ, PURPUROVÁ, AZUROVÁ, neutrální: BÍLÁ, ČERNÁ a tyto barvy mohou být buď tmavé nebo světlé.

21.3 Obrazovková paměť

Obsah obrazovky je uložen ve video paměti RAM, realizované dvěma nebo čtyřmi integrovanými obvody 4416. V dalším textu budeme tuto paměť označovat symbolem "VRAM".

O její obsluhu, včetně oživování se stará podle pokynů procesoru zákaznický obvod GDG.

21.4 Videoprocesor GDG

GDG je 100-pinový zákaznický obvod LSI plnící všechny funkce týkající se mapování paměti, práce se zobrazováním atd. GDG je přístupný instrukcemi IN a OUT na patřičných adresách, popsaných v příslušných kapitolách. Zde se budeme zabývat jen funkcemi, které jsou zajímavé z hlediska řízení obrazovky.

21.5 Seznam I/O adres videoprocesoru

Seznam obsahuje všechny adresy, používané při práci s VRAM. Ty porty, které mají uvedenou vyšší hodnotu adresy je nutné adresovat šestnáctibitově pomocí registru BC.

I/O adresa		funkce
-- CC	OUT	"Write format register" (WF)
-- CD	OUT	"Read format register" (RF)
-- CE	OUT	"Display mode register" (DMD)
--CE	IN	"Status read register"
01 CF	OUT	"Scroll offset register low" (SOF1), 8 bitů
02 CF	OUT	"Scroll offset register high" (SOF2), 2 bity
03 CF	OUT	"Scroll width register", 7 bitů
04 CF	OUT	"Scroll start adress register" (SSA), 7 bitů
05 CF	OUT	"Scroll end adress register (SEA), 7 bitů
06 CF	OUT	"Border color register" (BCOL), 4 bity
07 CF	OUT	"Superimpose bit" (D7) (CKSW),1 bit
-- F0	OUT	"Pallet register"
-- E1	IN/OUT	Odmapování VRAM/mapování horní ROM
-- E0	IN/OUT	Namapování VRAM/odmapování generátoru znaků a dolní ROM

21.6 Organizace VRAM

21.6.1 Kapacita VRAM, standardní a rozšířená VRAM

Kapacita VRAM je maximálně 32 KB, ve standardní verzi MZ-821 je však instalováno jen 16 KB. Paměť je rozdělena do dvou sad A,B zvaných "frame":

sada A ..... standardně nainstalovaná

sada B ..... rozšíření VRAM

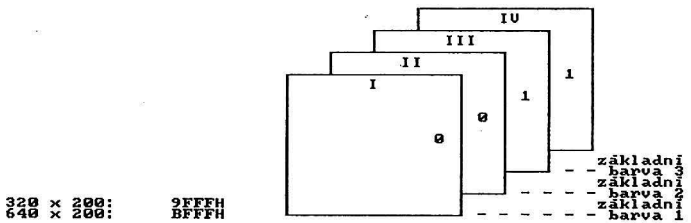
Pojem sada A a sada B má fyzické opodstatnění, slouží k rozlišení standardně nainstalované a rozšířené paměti. Není-li počítač doplněn o dva integrované obvody IO 4416, neexistuje sada B.

21.6.2 Rozdělení VRAM do rovin

Kromě fyzického rozdělení VRAM na sady, lze paměť logicky ještě rozdělit na tzv. "roviny" (plane). Počet, uspořádání a velikost rovin je určující pro počet barev naráz zobrazitelných na obrazovce a počet bodů na řádku. Konfigurace rovin se nastavuje s definicí režimu obrazovky (viz. 4.7). Maximální počet rovin je 4. Označují se římskými číslicemi I, II, III a IV.

Velikost roviny závisí na módu, ve kterém se obrazovka nachází:

- mód velikost roviny
- 320 x 200 8 KB
- 640 x 200 16 KB



21-1 Princip rozložení rovin VRAM

Číslice 1100 uvnitř rovin znázorňují nějaký bod, kód jeho barvy je 12 = 1010B

21.6.3 Adresování VRAM

První adresa VRAM je 8000H, poslední závisí na počtu zobrazovaných bodů na řádku. Všechny roviny začínají i končí na stejných adresách, to znamená, že jedné adrese přísluší jeden byte v každé rovině.

Každá rovina pokrývá celou obrazovku a sama o sobě určuje její obsah pouze dvoubarevně. Barevnosti se dosahuje kombinací dat z více rovin.

Každému bodu obrazovky, přísluší jeden bit každé roviny, to znamená, že každý bod je určen tolika bity, kolik je v daném módu rovin. (Například: 4 roviny = = > 16 barev, protože 2^4 = 16)

Jeden byte z každé roviny je zobrazen jako vodorovná osmice bodů. Nejnižší bit bytu je vlevo a nejvyšší vpravo, což je v rozporu s vžitými konvencemi. Proto se při předělávání programů z jiných počítačů musí byty zapisované do VRAM otočit.

Byty jsou uspořádány souvisle po mikrořádcích až do vyčerpání celé obrazovky. Rozložení adres v každé rovině je znázorněno na následujících tabulkách:

Adresování VRAM v módech 320 x 200

-	0	1	2	3	39
- 0	8000H	8001H	8002H	8003H	8027H
- 1	8028H	8029H	802AH	802BH	8050H
- ..	..	..	..	.. ..	..
- ..	..	..	..	.. ..	..
- 199	9F18H	9F19H	9F1A	9F1BH	9F3FH

Adresování VRAM v módech 640 x 200

-	0	1	2	3	9
- 0	8000H	8001H	8002H	8003H ...	804FH
- 1	8050H	8051H	8052H	8053H ...	809FH
- ...	...	...	...	...	...
- ...	...	...	...	...	...
- 199	BE30H	BE31H	BE32H	BE33H	BE7FH

Na obrazovce se z každé roviny zobrazuje 8000 resp. 16000 bytů. Zbylých 19[illegible] resp. 284 bytů není využito.

21.6.4 Logické a fyzické číslo barvy

Logické číslo barvy

Kdyby jednomu bodu příslušel 1 bit, mohl by bod nabývat pouze dvou barev. Proto je VRAM rozdělena do již zmíněných rovin. Když se k některému bodu vybere z každé roviny po jednom bitu, vznikne uspořádaná čtveřice (nebo jen dvojice) bitů, kterou lze považovat za binární číslo. Takto získané číslo vyjadřuje KÓD PALETY (viz. 4.10), tedy logické číslo barvy.

Fyzické číslo barvy

Fyzické číslo barvy je číslo neoddělitelně spojené s danou barvou. Lze jej obdržet z tabulky v kapitole 4.10. Každému kódu palety (logickému číslu barvy) je zvláštním mechanismem (viz. 4.10) přiřazena skutečná barva, která bude svítit na obrazovce (fyzické číslo barvy).

O této možnosti se budeme bavit až v kapitole 4.10. Bude-li v jiných kapitolách řeč o barvě nebo o čísle barvy, rozumíme tím logické číslo barvy neboli kód palety. Barvou pozadí se myslí vždy barva s kódem palety 0, bez ohledu na to, jak je nastaven PAL.

21.6.5 Definice režimu obrazovky

Již v úvodu bylo řečeno, že lze pro obrazovku definovat několik režimů, lišících se ve velikosti rastru, počtu barev a velikosti používané VRAM. K definici

režimu slouží osmibitový registr DMD dostupný instrukcí OUT (0CEH),A. Čtení z registru nevrátí zapsanou hodnotu, ale status obrazovky (viz. 4.12).

Přehled módů DMD

	rastr	display DMD	roviny	registr
-		4 barvy A	0	I,II
-	320 x 200	4 barvy B	1	III,IV
-		16 barev AB	2	I,II,III,IV
-		2 barvy A	4	I
-	640 x 200	2 barvy B	5	HI
-		4 barvy AB	6	I,III
-	MZ-700	data,atb,CGRAM	8	

Tabulka uvádí všechny známé a možné kódy, ktere lze zapsat do registru DMD. Dále je z ní patrná velikost obrazovky, počet barev a používané roviny.

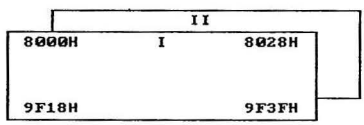
Režim se nastavuje ještě před zahájením práce s obrazovkou, a pak už se až na výjimky nemění. Jediná přípustná změna je střídání módů 0,1 resp. 4,5, tzv. alternující režim práce.

Po nastavení režimu je vhodné okamžitě smazat celou VRAM, protože se změní poměry v adresování.

Pozn:
Protože registr DMD určuje jen mód obrazovky, budeme symbolem DMD označovat i tento mód. Z kontextu je zřejmé, zda se jedná o označení registru nebo módu.
Bude-li v některých případech lhostejný počet barev, ale důležitá velikost obrazovky, budeme říkat "módy 320 x 200" resp. "módy 640 x 200", rozumí se tím jeden z DMD 0,1,2 resp. DMD 4,5,6.

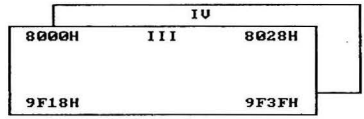
21.6.6 Přehled jednotlivých režimů obrazovky (DMD)

V tomto odstavci jsou stručně popsány jednotlivé módy DMD. Ke každému módu je nakreslen názorný obrázek, ze kterého je vidět uspořádání rovin v paměti i na obrazovce. Adresy uvedené v rozích jedné z rovin jsou společné pro všechny zobrazené roviny.



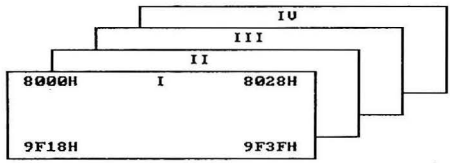
21-2 DMD 0 - 320x200, 4 barvy, sada A

VRAM je organizována do dvou rovin (I a II), obě roviny jsou ze sady A. Sada B se nevyužívá. Barva je z intervalu <0,3>.



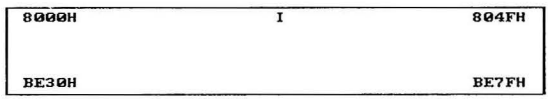
21-3 DMD 1 - 320 x 200, 4 barvy, sada B

Stejný případ jako A, s tím rozdílem, že se zobrazuje sada B (rozšířená paměť, jestliže ji nemáte, budete mít stále bílou obrazovku)



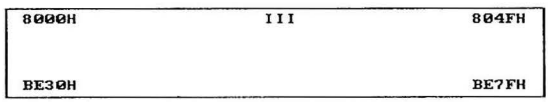
21-4 DMD 2 - 320 x 200, 16 barev, obě sady

VRAM je organizována ve čtyřech rovinách (I, II, III, IV). To dává dohromady šestnáct kombinací a tedy 16 barev. Barva je z intervalu ,15.



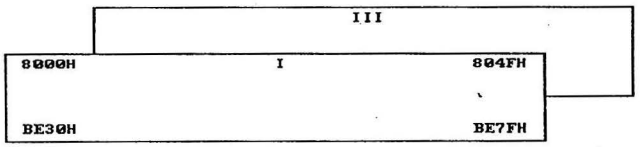
21-5 DMD 4 - 640 x 200, 2 barvy, sada A

VRAM je celá zobrazena jen do jedné roviny v sadě A. Ta má velikost 16 KB, což je dvojnásobek než v módech 0, 1, 2. Sadu B lze obdobně jako v módu 0 využít za jiným účelem.



21-6 DMD 5 - 640 x 200, 2 barvy, sada B

Grafika je stejná jako v minulém případě, zobrazuje se však sada B. Sada A se nevyužívá.



21-7 DMD 6 - 640 x 200, 4 barvy, obě sady

VRAM je rozdělena do dvou rovin: I a III, (ta trojka není překlep, ale skutečnost). Rovina I je v sadě A, rovina III v sadě B. Jejich velikost je stejná jako v režimech 4 a 5 Využívá se celá VRAM pro grafiku.

21.7 Zápis dat do VRAM

Doposud jsme popisovali jakým způsobem se obrazovka chová a co na ní můžeme a nemůžeme vidět, nyní pojednáme stručně o všech způsobech zápisu a kreslení.

21.7.1 Mapování VRAM

VRAM je za normálních okolností schována před zraky programátora. Její zpřístupnění se provede instrukcí:

IN A,(0E0H)

Od tohoto okamžiku je od adresy 8000H po adresu 9FFFH v módech 320 x 200 a po adresu 0BFFFH v módech 640 x 200 videopaměť. Zároveň s ní se na adresy 1000H-1FFFH namapuje ROM generátor znaků. Program, obsluhující obrazovku samozřejmě nesmí být na uvedených adresách a také by zde neměl mít zásobník.

Před návratem z kreslící rutiny se nesmí zapomenout VRAM odmapovat instukcí:

IN A,(0E1H)

Touto instrukcí se obnoví původní obsah paměti na adresách 8000H-9FFFH resp. 0BFFFH a na adresách 1000H-1FFFH. Před další prací s VRAM je nutné ji opět namapovat instrukcí IN A,(0E0H).

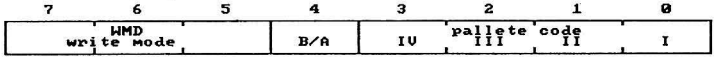
Provedení popsaných instrukcí není vidět na obrazovce.

21.7.2 Přístup do VRAM

Data se do VRAM zapisují po jejím namapování normálními instrukcemi pro práci s operační pamětí. Na rozdíl od běžné paměti se data nezapisují přímo, ale prostřednictvím videoprocesoru GDG, který data nezapisuje a ani nemůže zapisovat tak jak jsou, ale musí provést nějaké konverze. Jaké konverze se mají provádět je určeno jednak módem a jednak speciálním registrem zvaným "Write format register" WF (0CCH).

21.7.3 WF registr

WF registr je 8-bitový registr videoprocesoru. Jeho obsah definuje roviny do kterých se zapisuje a způsob, jakým se zapisuje.



21-8 WF "Write format register"

WMD .... mód zápisu do VRAM (0-7) - určuje druh konverze, kterým budou procházet zapisovaná data. Symbolem "WMD" označujeme jak část WF registru, nesoucí informaci o módu zápisu, tak vlastní mód zápisu. Nepleťte si mód zápisu WMD s módem obrazovky DMD.

B/A .... Tento bit má specifický význam v závislosti na WMD. Určuje sadu, do které se zapisuje a také zapíná transformaci adres pomocí scroll (viz. 21.13).

I až IV Na tyto bity se lze dívat ze dvou pohledů: jako na čísla rovin do kterých se bude zapisovat nebo jako kód barvy, kterou se bude kreslit.

21.7.4 Módy zápisu WMD

Existují dva základní druhy režimů zápisu:

1) Zápis do jednotlivých rovin VRAM

WMD	
0	SINGLE
1	EXOR
2	OR
3	RESET

2) Zápis do všech rovin ve specifické barvě

WMD	
4	REPLACE
5	------"------
6	PSET
7	- --"---

Oba druhy režimů se od sebe liší svým významem i způsobem práce, proto jsou popsány samostatně.

21.7.5 Zápis do jednotlivých rovin (WMD = 0 až 3)

Dělí se íle na 4 různé podrežimy podle obsahu WMD. Co se děje se zapisovaným bytem zřejmé z vásledující tabulky. WD (write data) jsou právě zapisovaná data, VD j u data, která ve VRAM byla.

WMD		funkce
SINGLE	000	*zápis: WD tak jak jsou*
EXOR	001	*zápis: WD .xor. VD*
OR	010	*zápis: WD .or. VD*
RESET	011	*zápis: .not.WD .and. VD*

WD ..... zapisovaná data
VD ..... původní obsah VRAM

Spodní čtyři bity registru WF (I až IV) určují roviny, do kterých se zapisuje. Zapisovat lze do jedné roviny nebo i do všech současně. Booleovská konverze, určená obsahem WMD, se provádí v každé rovině zvlášť, nezávisle na rovinách ostatních.

Pokud je daný z bitů I až IV ve WF registru:
1 ..... data se do příslušné roviny zapisují s konverzí určenou bity 5-7 WF registru.
0 ..... data se do příslušné roviny nezapisují, její obsah se vůbec nemění.

B/A bit v režimech SINGLE, OR, RESET, EXOR

B/A bit nemá ten význam, který je popsán v servisním manuálu, ten podle něhož dostal označení. Specifikuje pouze, zdali se bude či nebude uvažovat scroll, podle tabulky:

DMD	320 x 200			640 x 200		
	0	1	2	4	5	6
B/A						
0	ano	ne	ano	ano	ne	ano
1	ne	ano	ano	ne	ano	ano

V režimech pro zápis do jednotlivých rovin (SINGLE, OR, EXOR, RESET) řídí bit B/A uvažování scrollu bez ohledu na to, je-li daná rovina zobrazovaná, nebo nezobrazovaná.

V těchto čtyřech režimech nezáleží na zvoleném DMD. Rozdíl je jen mezi režimy 320 x 200 a 640 x 200, vyplývající z odlišné velikosti rovin VRAM a odlišného přiřazení adres do fyzické VRAM. Protože v módech 640 x 200 existují pouze roviny I a III, bity II a IV v registru WF jsou nevýznamné.

SINGLE ..... Jednoduchý zápis, data se zapisují tak, jak leží a běží do VRAM. Na předchozím obsahu VRAM nezáleží. Roviny do kterých se nezapisuje se nemění.

EXOR ..... Se zapisovanými daty a daty ve VRAM se provede operace XOR (exkluzivní nebo). V důsledku toho jedničky zapisovaných dat provádějí inverzi ve VRAM, nuly zachovávají původní obsah VRAM.

OR ..... Se zapisovanými daty a daty ve VRAM se provede operace OR (logický součet). Důsledek je ten, že jedničky zapisovaných dat se ukládají jako jedničky do VRAM, nuly zachovávají původní obsah VRAM.

RESET ..... Zapisovaná data se negují a provede se operace AND (logický součin) s daty ve VRAM. Důsledek je ten, že jedničky zapisovaných dat se ukládají jako nuly do VRAM, nuly původní obsah VRAM zachovají.

Režim SINGLE je využitelný pro přímý přístup do VRAM. Je jediný vhodný pro neobrazovkovou manipulaci s VRAM (např. buffery). Režimy XOR, OR a RESET umožňují manipulovat jednotlivými body ve zvolených rovinách, to lze využít např. pro realizaci jednoduché sprajtové grafiky.

21.7.6 Zápis ve specifikované barvě (REPLACE, PSET)

Dělí se na dva podrežimy, použitelné pro zápis textu nebo pro grafiku. Oba jsou, co se týče řízení totožné, proto je popisujeme společně.

Následující tabulka udává význam bitů I - IV v jednotlivých DMD. Platí pro režim REPLACE, PSET a pro SEARCH při čtení.

	DMD	B/A	IV	III	II	I
320 x 200 4 barvy	0,1	0	x	x	0/1	0/1
		1	0/1	0/1	x	x
16 barev	2	x	0/1	0/1	0/1	0/1
640 x 200 2 barvy	4,5	0	x	x	x	0/1
		1	x	0/1	x	x
4 barvy	6	x	x	0/1	x	0/1

DMD ..... nastavený mód obrazovky
0/1 ..... významný bit
x ..... nevýznamný bit
B/A ..... určuje, do které sady se zapisuje

Zapisuje se jen do rovin, které jsou v tabulce označeny symbolem 0/1. Roviny označené "x" se nemění.

Zde je interpretace bitu B/A poněkud odlišná. Určuje skutečně sadu, do které se zapisuje, je možné zapisovat i do viditelné i do neviditelné sady. Je nutné dát pozor na jednu maličkost:

Při zápisu do NEVIDITELNÉ roviny VRAM v módechPSET a REPLACE se neuplatňuje scrolí.

Bity, označené "x" jsou nezajímavé, s rovinami označenými těmito bity se neděje nic. Všimněte si, že neměněné roviny jsou obvykle ty, které se nezobrazují. Vhodnou volbou B/A, který skutečně určuje sadu VRAM se kterou se pracuje, lze zapisovat i do neviditelné VRAM a do viditelné nezapisovat. Nikdy se ale data nazapisují do obou současně. Výjimku samozřejmě tvoří DMD 2 a 6, ve kterých se zapisuje vždy do všech rovin.

Bity označené 0/1 jsou významné. Jejich konkrétní význam je závislý na WMD. V obou režimech je tímto bitem specifikována barva, přesněji kód palety, se kterým se pracuje.

WMD	režim zápisu	Co se píše, když bit I až IV je	
		1	0
4 nebo 5	REPLACE	WD	0 (nula)
6 nebo 7	PSET	WD + VD	.not.WD.and.VD

REPLACE ..... Jedničky ze zapisovaných dat se zapíší jako specifikovaný pallete code, nuly se zapíší jako kód palety 0, t.j. obvykle černá. Výsledek je ten, že se změní celá osmice bodů bez ohledu na její předcházející stav. Tímto způsobem se jednoduše zapisují barevné znaky na černém pozadí, původní znak je překryt. Na pozadí barvy s kódem různým od nuly se v módu REPLACE psát nedá.

PSET ..... Jedničky ze zapisovaných dat se zapíší jako body ve specifikované barvě, nuly ze zapisovaných dat VRAM nemění. Tento režim slouží pro rozsvícení bodu v určité barvě, tedy pro grafiku. Texty se v tomto režimu píší obtížně, protože by se znaky nepřekrývaly, ale prolínaly.

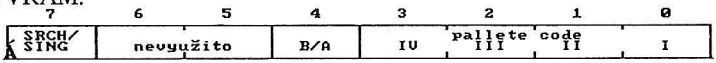
21.8 Čtení z VRAM

Přístup do paměti je stejný jako při zápisu, t.j. pomocí běžných instrukcí pro práci s operační pamětí po namapování VRAM.

Přečtená data nejsou přesným obrazem VRAM (ani by to nešlo, kvůli překrývání rovin), ale data jsou upravena obvodem GDG podle "Read format register" RF (0CDH). Význam jeho bitů je následující:

21.8.1 RF registr

RF registr je 8-bitový registr videoprocesoru, specifikující konverze při čtení z VRAM.



21-9 RF "Read format register"

Existují pouze dva režimy určené bitem SRCH/SING:
0 ... "Single color data read" jednoduché čtení z vybrané roviny nebo vybraných rovin
1 .. "Specified color search" čtení vybrané barvy ze všech definujících rovin.

21.8.2 Jednoduché čtení

Odpovídá jednoduchému zápisu, bit B/A má také stejný význam. Čtení probíhá z roviny, jejíž bit (I až IV) je v RF jedničkový. Aby byla přečtená data jednoznačná, nemělo by se nastavit více bitů I až IV. Když se tak stane, jsou byty přečteny z každé specifikované roviny a provedena s nimi logická operace AND.

Funkce RF v tomto režimu nezávisí na DMD. V režimech 640 x 200 jsou bity II a IV nevýznamné.

21.8.3 Čtení specifikované barvy

Odpovídá zápisu ve specifikované barvě (módy REPLACE a PSET). Následující tabulka udává význam bitů I - IV pro jednotlivé DMD. Platí pro mód SEARCH a také pro módy REPLACE a PSET při zápisu.

	DMD	B/A	IV	III	II	I
320 x 200 4 barvy	0,1	0	x	x	0/1	0/1
		1	0/1	0/1	x	x
16 barev	2	x	0/1	0/1	0/1	0/1
640 x 200 2 barvy	4,5	0	x	x	x	0/1
		1	x	0/1	x	x
4 barvy	6	x	x	0/1	x	0/1

DMD ..... nastavený mód obrazovky
0/1 ..... významný bit
x ..... nevýznamný bit
B/A ..... určuje, ze které sady se čte

Z rovin, kde je v tabulce x se nečte vůbec. Data z rovin označených 0/1 se přečtou, a provede se operace XOR s příslušným bitem v I až IV v RF registru. Výsledný byte vznikne operací NOR z těchto bytů.

V praxi to znamená, že se zjišťuje, kde se vyskytuje barva specifikovaná v RF. Tam, kde tato barva je, se vrací 1, tam, kde je kterákoliv jiná, se vrací 0.

Čtením ve specifikované barvě nezjistíte jaká je barva v daném místě, ale jen jestli zde je nebo není vámi zadaná barva.

Bit B/A má stejný význam jako při zápisu v módech PSET a REPLACE. Určuje sadu ze které se čte, na druhé sadě je výsledek čtení nezávislý. To umožňuje číst ze sady, která právě není zobrazována. V DMD 2 a 6 na hodnotě bitu B/A nezáleží, čte se ze všech rovin. I při čtení je nutné upozornit na důležitou maličkost:

Při čtení z NEVIDITELNÉ roviny VRAM v módu SEARCH se neuplatňuje scroll

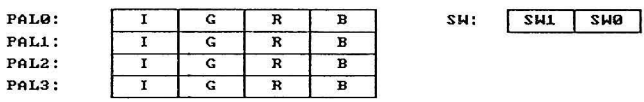
21.9 Pallet

Při zobrazování na display dostane GDG z VRAM pro každý bod kombinaci 1 - 4 bitů určující barvu tohoto bitu. To ale není skutečná barva, jakou bude daný bod svítit. Je to jen kód palety.

Skutečnou barvu je možné každému kódu palety přiřadit libovolně. Dokonce lze dvěma různým kódům palet přiřadit stejné barvy a tím zneviditelnit obsah obrazovky. Skutečné barvy mají svůj kód, určený následující tabulkou:

0	Černá	(black)
1	Modrá	(blue)
2	Červená	(red)
3	Purpurová	(magenta)
4	Zelená	(green)
5	Azurová	(cyan)
6	Žlutá	(yellow)
7	Bílá	(white)
8	Šedá	(gray)
9	Světle modrá	(light blue)
10	Světle červená	(light red)
11	Světle purpurová	(light magenta)
12	Světle zelená	(light green)
13	Světle azurová	(light cyan)
14	Světle žlutá	(light yellow)
15	Světle bílá	(light white)

Přiřazení barev je dáno obsahem registrů palet (značka PAL). Jsou to čtyři čtyřbitové registry PAL0 až PAL3 a jeden dvoubitový SW. (SW se používá jenom v módu DMD 2)



21-10 Registry palet

Když GDG zobrazuje data, tak vybírá z VRAM kód palety, tím adresuje příslušný PAL registr a jeho obsah použije jako kód barvy podle uvedené tabulky.

Protože PAL registry jsou čtyři, může být kód palety maximálně 4. Pokud je použito šestnáct barev, je PAL poněkud pozměněn.

21.9.1 PAL v DMD 2 (16 barev)

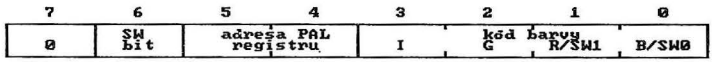
To co bylo dosud o paletách řečeno, neplatí pro DMD 2 (16 barev). V tomto módu jsou kódy palet 0 - 15, registry palet jsou však jen 4. Proto se v tomto módu využívá dvoubitový registr SW. Když GDG zobrazuje data z VRAM, porovnává horní dva bity kódu palety barvy (bity z roviny III a IV) s registrem SW Mohou nastat tyto případy:

1) Hodnoty jsou shodné. Pak se spodní dva bity (z rovin I a II) použijí jako adresa PAL, ze kterého se vybere příslušný kód barvy.
2) Hodnoty se liší. Kód palety z rovin I, II, III a IV se použije bez transformace jako kód barvy.

Z popisu je patrno, že přiřazování kódů barev kódům palety není úplné. Toto přiřazování je výhodné při psaní programů pro rozšířenou i nerozšířenou VRAM. V praxi to vypadá tak, že jedné čtveřici kódu palet: (0-3, 4-7, 8-11 nebo 12-15) je přiřazována barva pomocí registrů palet, u ostatních 12-ti barev je kód barvy a kód palety totožný.

21.9.2 Zápis do PAL registrů

Zápis do PAL a SW se provádí zápisem na port 0F0H. Jednou instrukcí OUT se plní jeden registr. Struktura zapisovaného bytu je následující:



21-11 Palet register

B	Modrá
R	Červená
G	Zelená
I	Intenzita
SW	příznak zápisu do registru SW
SW0,SW1	zapisované bity SW registru

Pokud je SW bit nulový, zapíše se kód barvy (bity 0-3) do registru palet určeného adresou.

Pokud je SW bit jedničkový, nezapisuje se do PAL nic, ale zapisují se bity 0-1 do dvoubitového registru SW.

21.9.3 Praktické použití PAL

Kód barvy Y, přiřadíte kódu palety X instrukcemi:

```
LD      A,X*16+Y
OUT     (0F0H),A
```

Chcete-li aby rychle zčernala obrazovka, vykonáte instrukce:

```
        LD      A,00H
        OUT     (0F0H),A
        LD      A,10H
        OUT     (0F0H),A
        LD      A,20H
        OUT     (0F0H),A
        LD      A,30H
        OUT     (0F0H),A
```

Chcete-li, aby kód palety byl totožný s kódem barvy, vykonejte tuto posloupnost instrukcí:

```
        LD      A,55H           ; 1. registr bude SW
LOOP:   SUB     11H             ; a po něm všechny PAL registry
        OUT     (0F0H),A
        JR      NZ,LOOP
```

Pallet má naprosto přesnou analogii v MZ-800 BASICU. Posloupnost instrukcí pro naplnění registru palet:

```
        LD      A,x*16+y ; registr palety a hodnota
        OUT     (0F0H),A
```

je ekvivalentní s příkazem v jazyku BASIC:

```
        PAL     x,y
```

Nastavení SW registru instrukcemi:

```
        LD      A,40H+x ; nastavuje se SW
        OUT     (0F0H),A
```

Způsobí v Basicu příkaz:

```
        INIT    "CRT:M2Bx"
```

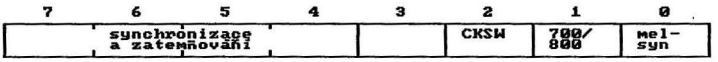
21.10 Border

Obrazovka MZ-821 je využita jen zčásti, na okraje obrazovky se data nazapisují. Tento okraj zvaný "Border" GDG zobrazuje v jedné specifikované barvě. Border se zadává přímo v kódu barvy, nikoli v pallete kódu na I/O adresu 06CFH např. instrukcemi:

```
        LD      BC,06CFH
        LD      A,kód barvy
        OUT     (C),A
```

21.11 Status obrazovky

Osmibitový status obrazovky se přečte instrukcí IN A,(0CEH) z portu 0CEH (DMD). Nepřečte se zapsaný mód, ale následující stavové slovo:



21-12 Status register

0.bit ..... Frekvence shodná s frekvencí čtenou z 0E008H v MZ-700 módu. (35 Hz - signál TEMP)

1.bit ..... Stav přepínače MZ-700 ON na zadním panelu počítače. ON = 0

2.bit ..... Signál CKSW, nastavuje se 7.bitem portu 7CFH. Jeho jediný pozorovaný efekt je posunutí obrazu na obrazovce asi o 2 cm doleva. V servisním manuálu mu říkají "superimpose bit".

3.bit ..... Nevyužito = 0

4.bit ..... Snímková synchronizace

5.bit ..... Řádková synchronizace

6.bit ..... Snímkové zatemňování

7.bit ..... Řádkové zatemňování

Bity 4 až 7 jsou aktivní v nule.

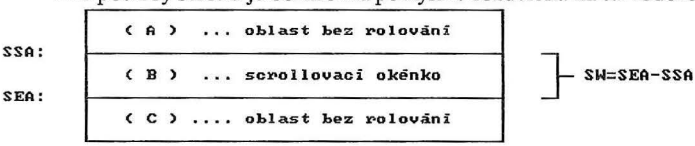
21.12 Scroll

Protože je kapacita VRAM dost vysoká a práce s ní je pomalejší než s RAM, není efektivní provádět rolování softwarově. Za tím účelem provádí GDG složité konverze adres do VRAM. Tím lze opticky posouvat text na obrazovce všemi čtyřmi směry, přitom se obsah VRAM nezmění.

Uvedené konverze provádí GDG nejen při zobrazování, ale i při zápisu a čtení dat (až na výjimky, na které bude upozorněno v příslušných odstavcích). Z uživatelského hlediska to vypadá, jako by se přímo přemístila data ve VRAM.

Scroll pracuje ve všech definovaných módech stejně, ani při přechodu z módů 640 x 200 do módů 320 x 200 a naopak není potřeba scroll inicializovat, jen je vhodné smazat obrazovku.

Pro potřeby scrollu je obrazovka pomyslně rozdělena na tři vodorovné pásy:



21 = 13 Rozdělení obrazovky

Oblasti (A) a (C) jsou vzhledem k rolování zamčeny, text se roluje jen v nastavitelném okénku (B).

21.12.1 Scroll registry

Scroll registry jsou čtyři registry videoprocesoru GDG, dostupné instrukcemi výstupu (OUT) na adrese 0CFH. Do kterého registru se bude zapisovat je určeno registrem B. Scroll registry nelze číst.

SSA .... "Scroll start adress" (7 bitů) - určuje začátek oblasti, kde se bude rolovat, tedy oblasti (B)

SEA .... "Scroll end adress" (7 bitů) - určuje konec oblasti (B), přesněji začátek oblasti (C)

SW ..... "Scroll width register" (7 bitů) - určuje výšku okénka (B). Musí nabývat hodnotu SW = SEA-SSA, nese tedy redundantní informaci

SOF .... "Scroll offset register" - Tento registr je registrem výkonným. Zatímco registry SSA a SEA určují kde se roluje, registr SOF určuje kam se roluje a o kolik se roluje. Jeho maximální hodnota je (SEA-SSA)*8.

21.12.2 Plnění Scroll registrů

Doporučujeme mít obraz Scroll registrů v paměti takto:

```
SREG:
SOF:   DEFW   0
SW:    DEFB   125
SSA:   DEFB   0
SEA:   DEFB   125
```

Nejdříve obsah registrů připravit a uložit do paměti, a potom teprve najednou plnit takto:

```
       LD     BC,6CFH
       LD     HL,SREG+4
       OTDR
```

Poznámky:

1. Nedoporučujeme plnit registry postupně, protože by se mohl poškodit obsah obrazovky, než všechny registry nabudou správné hodnoty.
2. Musíte mít na paměti, že před plněním musí být vždy SW = SEA-SSA, proto je nutné při každé změně SSA nebo SEA změnit i SW.
3. Po smazaní obrazovky doporučujeme nastavit registr SOF na nu lu.

21.12.3 Rolovací okéno

Rolovací okénko (dále jen okénko) je ta část plochy obrazovky, která bude podléhat rolování (B). Horizontálně zaujímá vždy celou šířku obrazovky, vertikálně je omezeno rozhraním s oblastí (A) a (C).

Rozhraní oblastí (A),(B) a (B),(C) je určeno registry SSA a SEA, jak je patrno z obrázku. Mohou nabývat pouze hodnot: 0, 5, 10, 15, ... 125, tedy číslo z intervalu [0..125] dělitelné pěti. Přitom z pochopitelných důvodů musí platit: SEA = SSA. Při SEA = SSA, okénko neexistuje.

Okénko nemůže začínat ani končit na libovolném mikrořádku (připomeňme, že mikrořádky číslujeme 0-199), ale jen na každém osmém, to znamená na jednom z 25-ti textových řádků (číslujeme je 0-24). Hodnotu registru dostaneme, vynásobíme-li pěti číslo toho textového řádku, kterým má okénko začínat resp. před kterým má okénko končit.

Pokud má okénko začínat na začátku obrazovky, bude SSA = 0, pokud má končit na konci obrazovky, bude SEA = 125.

Příklad:

Chceme mít na obrazovce 6 řádků před okénkem a 6 řádků za okénkem. Okénko začiná řádkem 5 (řádky se číslují od 0), SSA bude 5*6 = 30. Poslední řádek okénka je 18, to znamená, že první řádek oblasti (C) je řádek 19, SEA bude 5*19 = 95. SW se musí nastavit na SEA-SSA = 95-30 = 65.

21.12.4 Provedení scrollu

Doposud popsané registry pouze vymezují, kde se bude rolovat. Zbývá poslední ze čtyř registrů, a to je registr SOF, který určuje, kterým směrem se odroluje a o kolik se odroluje.

Rolování v okénku se provádí změnou (nikoli jen nastavením) registru SOF. Podle velikosti jeho přírůstku, resp. úbytku se rozlišuje mezi vertikálním nebo horizontálním scrollem.

Horizontálně lze rolovat i po mikrořádcích, vertikálně jen po 1/5 šířky obrazovky, t.j. 8 nebo 16 znaků v závislosti na módu.

Nebudeme se zabývat přesným mechanismem transformace adres, uvedeme jen praktické důsledky. Nejdříve uvádíme universální podprogram, který provede scroll zadaným směrem o zadaný počet bodů.

Před prvním spuštěním tohoto podprogramu je nutné samozřejmě určit okénko naplněním paměťových buněk SSA a SEA. SOF je vhodné nastavit na nulu, SW si podprogram spočítá sám.

```
TITLE 'Scroll'
;
; Posune obrázek v okénku
; na obrazovce podle obsahu DE registru
; (Přičte DE k SOF modulo (SEA-SSA)*8)
;
; Vstupní hodnoty v DE registru:
;
; Vertikálně:
```

```
;
; 40 ... nahoru o textový řádek
; -40 ... dolů o textový řádek
;  5 ... nahoru o mikrořádek
; -5 ... dolů o mikrořádek
;
; Horizontálně:
;
; -1 ... doleva o 1/5 šířky obrazovky
; 1 ... doleva o 1/5 šířky obrazovky
;
; Celočíselné násobky uvedených hodnot
; rolují naráz o příslušný počet prvků.
; Např: 120 roluje o 3 řádky nahoru
;
ROLL::
        LD      HL,(SOF) ; "scroll ofset register"
        ADD     HL,DE    ; zvětšit o co chceme rolovat
        EX      DE,HL    ; nový SOF do DE
        LD      HL,SEA   ; ukazatel na SEA
        LD      A,(HL)   ; SEA do A
        DEC     HL       ; ukazatel na SSA
        SUB     (HL)     ; SEA-SSA
        DEC     HL       ; ukazatel na SW
        LD      (HL),A   ; ulozit SW
        RET     Z        ; žádné okénko, není co rolovat
        LD      L,A      ; výšku okénka do HL
        LD      H,0      ; horní byte nulový
        ADD     HL,HL    ; * 2
        ADD     HL,HL    ; * 4
        ADD     HL,HL    ; * 8 = maximalní hodnota SOF
        EX      DE,HL    ; maximum do DE, aktuální do HL
ROLL2:OR        A        ; odčítej maximum do té doby
        SBC     HL,DE    ; než bude hodnota záporná
        JP      P,ROLL2
ROLL1:OR        A
        ADC     HL,DE    ; a teď přičítej maximum do té
        JP      M,ROLL1 ; doby, než bude hodnota kladná
        LD          (SOF),HL    ; ulož SOF, je zaručeně v intervalu:
        LD      HL,SREG+4       ; [ 0 , (SEA-SSA)*8 ]
        LD      BC,6CFH ; adresa portu scroll registru
        OTDR    ; naplň registry
        RET
; ulozeni scroll registru
```

```
SREG:
SOF:   DEFS   2        ; posunutí
SW:    DEFS   1        ; šířka okénka
SSA:   DEFB   1        ; začátek okénka
SEA:   DEFB   1        ; konec okénka
       END
```

Podprogram nejdříve spočte SW=SEA-SSA, potom přičte k registru SOF hodnotu zadanou v DE a nakonec upraví SOF tak, aby ležel ve správném intevalu, t.j. interval [0..SW*8]

21.12.5 Vertikální scroll

Verikální scroll je nejběžnější způsob rolování textu. MZ-800 dokáže rolovat obrazovku nahoru i dolů o libovolný počet mikrořádků. Provádí se změnou SOF o násobek pěti.

Příklady:

```
LD     DE,35    ; roluje o 7 mikrořádků nahoru
CALL   ROLL

LD     DE,-20   ; roluje o 4 mikrořádky dolů
CALL   ROLL

LD     DE,40    ; roluje o textový řádek nahoru
CALL   ROLL     ; (8 mikrořádků)
```

21.12.6 Horizontální scroll

Horizontální scroll je speciálním případem scrollu vertikálního, není však tak dobře vyřešen. Provedeme jej změnou SOF o 1 až 4, resp. -4 až -1. Roluje se po 1/5 šířky obrazovky, což je dost hrubé dělení. Navíc zároveň dochází i k vertikálnímu scrollu o mikrořádek v části obrazovky, proto využití vertikálního scrollu je poněkud diskutabilní.

21.13 Grafika

V této kapitole pojednáme o využití grafických obrazových schopností MZ-800 při použití grafiky. Při grafických operacích hrají roli dva aspekty:

1) Obrázek se obyčejně kreslí na klidnou obrazovku, rozumí se tím, že obrazovka neodrolovává. Proto můžeme nastavit všechny čtyři scroll registry na nulu a dále se scrolem nezabývat. V alternativním režimu by použití scrollu dokonce mohlo působit potíže.

2) Při kreslení je žádoucí, aby se body obrazovky, na které se právě nekreslí neměnily. (typickým příkladem je nakreslení bodu nebo přímky)

21.13.1 Mód pro grafiku

Z kapitoly 21.2 víme, že obrazovka může pracovat v několika různých módech, může mít 320 nebo 640 bodů na řádku, můžeme používat 2, 4 nebo 16 barev.

Pro grafické účely doporučujeme používání DMD 0,1,2, t.j. 320 x 200, protože tak dostaneme přibližně stejná měřítka v horizontálním i vertikálním směru. Pro princip práce s grafikou není režim rozhodující.

Volbu DMD mají jednodušší ti, co nemají rozšířenou VRAM, mohou pracovat jen v DMD 0 nebo 4. DMD čtyři je méně vhodný mód, proto jim rozhodování nebude činit vůbec žádné potíže.

Pokud ale disponujete rozšířenou VRAM, musíte se rozhodnout, jak s ní naložíte. Zde vám nabídneme některé možnosti, třeba sami přijdete na další. (Ještě připomeňme, že základní VRAM se označuje, jako "sada A", rozšířená jako "sada B")

1) Využije se jen sada A, ať už z důvodů kompatibility nebo proto, abychom mohli sadu B využít jiným způsobem, např. jako vyrovnávací paměť. (DMD = 0)

2) Bude se pracovat jen v sadě B. To je obdoba předchozího, a také vhodnějšího, případu.(DMD = 1)

3) Pracuje se v nejmocnějším módu, DMD 3. V tomto módu je k dispozici 16 různých barev naráz na obrazovce. (DMD = 2)

4) Alternující režim - DMD 0 a 1 se podle potřeby přepínají, takže je právě vidět jen obrázek ze sady A nebo obrázek ze sady B. Toho lze využít buď proto, aby nebyl vidět průběh kreslení složitějšího obrázku, a nebo když je potřeba střída vě pracovat na dvou obrazech.

21.13.2 Jakým způsobem se bude kreslit

Skutečnost, že grafika je úplná, dává možnost kreslit libovolné a libovolně umístěné grafické obrazce, bez ohledu na strukturu VRAM. VRAM je ale organizována bytově, proto při rozsvícení nebo zhasnutí jednoho bodu, by bylo nutné data z VRAM přečíst, bit odpovídající danému bodu maskovat a zpět data zapsat. Aby se nemusela data číst, používají se obvykle tyto WMD.

PSET ..... zápis bodu v určené barvě
OR ..... změna barvy bodu (logickou funkcí OR)
RESET ... mazání bodu v zadané barvě
EXOR .... změna barvy bodu (logickou funkcí XOR)

DMD se nastavují zápisem do WF instrukcí OUT (0CCH),A. Výše jmenované WMD (PSET, OR, RESET, EXOR) jsou použitelné pro grafiku díky následující vlastnosti:

Zápis v módech PSET, OR, RESET a EXOR:

a) Zápis bitu 1 způsobí zapsání takové barvy, která je specifikována ve WF registru.

b) Zápis bitu 0 **nezmění** obsah obrazovky

Nyní uvedeme na ukázku úsek programu, který na obrazovce rozsvítí jeden bod o souřadnicích X a Y. Bod [0,0] je v levém horním rohu obrazovky. Podprogram platí pro libovolný DMD.

```
        ;
WIH     EQU     40          ; 80 v módech 640 x 200
        ;
ADR:    DEFW    (Y * WIH) + (X div 8)
```

```
BIT:    DEFB    X mod 8
        ;
SETHLA:
        LD      A,barva a WMD
        OUT     (0CCH),A
        IN      A,(0E0H)
        LD      A,(BIT)          ; číslo bitu
        RLCA
        RLCA
        RLCA                     ; číslo bitu na patřičná místa
        OR      11000111B        ; ať je to instrukce SET x,A
        LD      (SETA+1),A       ; modifikovat instrukci
        XOR     A
SETA:   SET     0,A              ; v A bude nastaven jen 1 bit
        LD      HL,(ADR)         ; adresa bytu na obrazovce
        LD      (HL),A           ; rozsvítit bod
        IN      A,(0E1H)
        RET
```

21.13.3 Nastavení pracovní barvy a módu zápisu

Pracovní barva (to znamená barva, kterou se bude kreslit) se nastaví zároveň s WMD do WF registru. Její platnost trvá tak dlouho, dokud není změněna na jinou barvu.

Následující tabulky udávají, jak se musí nastavit WF registr v závislosti na DMD a na požadovaném WMD zápisu. Znakem "B" je označen bit, který určuje kód barvy, přesněji řečeno palety. Z jejich počtu je také patrné množství zobrazovaných barev.

WF registr pro grafické operace v módech 320 x 200

Zápis do sady:	4 barvy A	4 barvy B	16 barev A i B
PSET	11x0xxBB	11x1BBxx	11xxBBBB
OR	010000BB	0101BB00	010xBBBB
RESET	011000BB	0111BB00	011xBBBB
EXOR	001000BB	0011BB00	001xBBBB
	DMD = 0 nebo 1		DMD=2

WF registr pro grafické operace v módech 640 x 200

Zápis do sady:	2 barvy A	2 barvy B	4 barvy A i B
PSET	11x0xxxB	11x1xBxx	11xxxBxB
OR	0100000B	01010B00	010x0B0B

RESET 0110000B 01110B00 011x0B0B
EXOR 0010000B 00110B00 001x0B0B
DMD = 4 nebo 5 DMD = 6

B ..... bity určující barvu
x ..... nevýznamný bit
DMD ... display mód = obsah registru DMD (0CEH)

21.13.4 Popis grafických WMD (PSET, OR, RESET, EXOR)

Po nastavení barvy a módu zápisu do WF a po namapování VRAM instrukcí IN A,(0E0H), lze kreslit na obrazovku normálními instrukcemi pro práci s pamětí. Znovu připomínáme, že změna se týká jen těch bodů, do kterých je zapisována jednička, protože nula VRAM nemění.

O tom jaká bude barva bodu po zapsání jedničky, rozhodují tyto faktory:

1) Původní barva bodu před zápisem (značíme OLDC)
2) Barva, specifikovaná ve WF registru (značíme NEWC)
3) Mód zápisu specifikovaný ve WF registru (značíme WMD)

V dalším textu se snažíme vysvětlit činnost videoprocesoru a její využití. Uvádíme, co se děje ve VRAM, jakým způsobem se daného módu užívá v MZ-800 BASICu, a nakonec příklad nebo stručnou poznámku na vysvětlení pro bližší pochopení činnosti.

Mód zápisu PSET

Zapisuje se: NEWC

V Basicu: Využíván všemi grafickými příkazy, jejichž druhý parametr v hranatých závorkách je 0, např. SET[x,0]

Tento mód bude mít zřejmě nejširší využití, jako jediný ze čtyř uvedených módů nebere ohled na OLDC, je znamená, že rozsvítí bod v barvě NEWC.

Mód zápisu OR

Zapisuje se: OLDC or NEWC - logický součet zapisované barvy s původní barvou.

V Basicu: BASIC tento mód užívá při operaci typu SET[x,1].

Příklad:

Nechť je na obrazovce nakreslen plošný útvar, např. kruh v barvě s kódem palety 3, barva pozadí je 0. Úkolem je nakreslit přes celou obrazovku síťku v barvě 1. ale tak, aby byla za zobrazeným předmětem.

Při použití PSET by se musela číst VRAM a testovat, zdali na místo, kam chceme zapisovat není náhodou zobrazený předmět. Pokud ale použijeme mód OR, můžeme rastr nakreslit přes celou obrazovku. Na pozadí se síťka nakreslí, protože 0 OR 1 = 1, předmět si ale barvu ponechá, protože 3 OR 1 = 3.

Mód zápisu RESET

Zapisuje se: (not NEWC) and OLDC - logický součin komplementu zapisované barvy s původní barvou.

V Basicu: Využívají ho jen příkazy RESET[b,1] a BLINE[b,1]

Lze výhodně využít při mazání části obrázku, při zápisu stejnou barvou, jako byl obrázek, obrázek mizí. (Mění se na barvu 0, což je obyčejně barva pozadí)

Mód zápisu EXOR

Zapisuje se: NEWC xor OLDC - nonekvivalenci původní barvy a barvy zapisované.

V Basicu: V interpretru MZ-800 BASIC není tento WMD využit.

Lze využít, chceme-li zobrazit část obrázku v doplňkových barvách.

21.13.5 Čtení z VRAM při práci s grafikou

Protože máme dosti silný mechanismus grafického zápisu, je čtení méně potřebnou operací, přesto však v některých aplikacích (obrazová analýza, vybarvování ploch) je číst potřeba. Pro potřeby grafiky používáme SEARCH nebo SINGLE read. Ten nastavíme zápisem do RF registru na adresu (0CDH), jehož hodnotu určíme z následující tabulky:

RF registr při grafickém čtení v módech 320 x 200

	Čtení ze	RF registr	
4 barvy	sady A	Sxx000BB	--- DMD 0 nebo 1
	sady B	Sxx1BB00	_/
16 barev	sady A i B	SxxxBBBB	DMD 2

RF registr při grafickém čtení v módech 640 x 200

	Čtení ze	RF registr	
2 barvy	sady A	Sxx0000B	--- DMD 4 nebo 5
	sady B	Sxx10B00	/
4 barvy	sady A i B	Sxxx0B0B	DMD 6

B ... bity, určující čtenou barvu
x ... nevýznamné bity
S ... typ operace čtení
1 = "SEARCH read", selektivní čtení
0 = "SINGLE read", jednoduché čtení

Selektivní čtení

Selektivní čtení nedokáže určit, jaká je v daném bodě barva, ale jen jestli je tam určitá barva, kterou zadáváme prostřednictvím RF registru. Vrací se:

1 ... pokud je barva odpovídajícího bodu totožná se specifikovanou barvou.
0 ... v opačném případě.

Příklad:

Předpokládejme DMD = 2 (16 barev). V levém horním bodě obrazovky má osmice bodů barvy 2,6,4,3,6,6,7,0. Po provedení instrukcí:

```
LD      A,86H     ; zajímá nás barva 6
OUT     (0CDH),A  ; nastavení RF
IN      A,(0E0H)  ; namapuje VRAM
LD      A,(8000H) ; přečtení bytu z paměti
```

je obah registru A takový: 00110010B. Jedničky odpovídají barvě číslo 6. Pořadí bitů je opačné proto, že nultý bit se zobrazuje vlevo.

Jednoduché čtení

Jednoduché čtení lze využít jen ve speciálních prípadech, pokud je zapotřebí zjistit, jestli je na obrazovce určitá třída barev. Tato třída nemůže být bohužel zvolena libovolně. Třída je určena barvou, která je zapsána v RF registru. Pokud logický součet čtené barvy s barvou zapsanou v RW, dá opět čtenou barvu, vrací se jednička, pro ostatní barvy 0.

Jinými slovy: Kdyby se do WF registru dala stejná barva jako je v RF a mód OR, četla by se jednička tam, kde by se při zapisování jedničky nic nedělo.

Příklad:

Do RF registru dáme hodnotu: 6. Barvy, při kterých se vrací jednička jsou 6,7,14,15.

Příklad:

Je nastaven DMD = 0 (4 barvy). Skutečné barvy jsou kódům palety přiřazeny následovně (viz. 4.10):

0 ..... Černá (15)
1 ..... Světle žlutá (14)
2 ..... Zelená (4)
3 ..... Světle bílá (0)

Zajistěte, aby levý horní roh se stal černobílým.

```
CERNOBIL:
        IN      A,(0E0H) ; namapuje VRAM
        LD      A,00000001B      ; Single read barvy 1 (z roviny I)
        OUT     (0CDH),A ; uložit do RF registru
        LD      A,00000011B      ; SINGLE mód WMD
        OUT     (0CCH),A ; uložit do WF registru
        LD      HL,8000H ; adresa VRAM
        LD      DE,20
        LD      C,100    ; výška měněné oblasti
LOOP1:LD        B,20     ; šířka měněné oblasti
LOOP2:LD        A,(HL)   ; čti byte přes RF registr
        LD      (HL),A   ; zapiš byte přes WF registr
        INC     HL       ; další byte měněné oblasti
        DJNZ    LOOP2    ; vyčerpán mikrořádek
        ADD     HL,DE    ; na začátek následujícího
        DEC     C
        JR      NZ,LOOP1         ; další mikrořádek
        IN      A,(E1H)  ; odmapovat VRAM
        RET
```

Uvedený podprogram je v rozporu s tím, co bylo řečeno na začátku kapitoly o grafice. Vyjímečně je zde použit WMD mód SINGLE.

Dále je z podprogramu vidět, že software nedělá žádné konverze dat, o ty se stará videoprocesor GDG. Jeho činnost v tomto konkrétním případě si krátce vysvětlíme:

LD A,(HL) ... přečte data z VRAM s konverzí pomocí RF registru. Ten je nastaven tak, aby se četla data jen z roviny I, to znamená, že na místě žluté a bílé barvy vrátí 0, na místě černé a zelené barvy vrátí 1.

LD (HL),A ... Zapíše zpět data přečtená předcházející instrukcí, ale tentokrát přes konverzi pomocí WF registru. Zapisovaná jednička (původně barva bílá nebo žlutá) se zapíše jako kód 3 (bílá), zapisovaná nula (původně černá neo zelená) jako kód 0 (černá). Kdyby se místo REPLEACE použilo PSET, nula by se nezapisovala, a zelená barva by proto zůstala na obrazovce.

21.13.6 Alternující režim v grafice

Je režim v němž se střídavě přepínají DMD 0 a 1, resp. 4 a 5, to znamená, že jedna ze sad je zobrazovaná a druhá neviditelná. Zapisovat lze přitom do libovolné z nich. V módech OR, RESET a EXOR dokonce do obou současně.

Potíže by nastaly, kdyby se využíval scroll, což je pro grafiku málo obvyklé.

21.14 Obrazovka, jako textová periferie

Na textovou obrazovku jsou kladeny docela jiné požadavky, než na obrazovku grafickou:

1) Je žádoucí, aby se využíval scroll, to přináší mírné problémy v alternujícím režimu.

2) Při zobrazení znaku je potřeba, aby se smazalo pozadí znaku, proto se obvykle pracuje v módu REPLACE.

21.14.1 DMD pro textovou obrazovku

Pro textové účely je vhodnější mód 640 x 200, to znamená 80 znaků na řádku. Mód 320 x 200 dá jen 40 znaků na řádku. Pokud by to bylo dostačující, je výhodnější pracovat v MZ-700 módu, protože práce s VRAM je v něm rychlejší a pohodlnější. Navíc lze v MZ-700 módu užívat osmi barev na obrazovce.

Dále se budeme zabývat módem 640 x 200, čtenář si jistě vytvoří pro mód 320 x 200 patřičné analogie.

Není-li k dispozici sada B (rozšířená VRAM), pracujeme v DMD = 4 (resp. DMD = 0). Na obrazovce jsou jen dvě barvy, barva pozadí a barva písma. (resp. 3 barvy písma a barva pozadí)

Pokud jste se stali vlastníky rozšířené VRAM, máte podobné možnosti jako, jaké jsou popsány v odstavci 22.14.1:

1) Využívá se jen sada A, jakoby sada B vůbec neexistovala

2) Bude se pracovat jen v sadě B. To je obdoba případu 1), ovšem méně vhodná.

3) Pracuje se v módu DMD 6. V tomto módu jsou k dispozici 4 různé barvy naráz na obrazovce. 3 z nich se považují za barvy písma a ta čtvrtá s kódem 0 za barvu pozadí.

4) Alternující režim - Módy 0 a 1 se podle potřeby přepínají, takže je právě vidět jen text ze sady A nebo text ze sady B. V tomto případě je nutné si dát pozor na dvě věci:

1. Obrazovky jsou dvě, ale scroll registry jsou jen jedny. Proto při odrolování obrazovky, přeroluje i obrazovka neviditelná.

2. Při zápisu do neviditelné obrazovky se scroll vůbec ne uvažuje, proto vámi rozkreslený znak, se může objevit jinde, než kde je očekáván.

21.14.2 Nastavení barvy písma

Barva písma se nastavuje instrukcí OUT (0CCH),A do WF registru zároveň s módem zápisu WMD.

REPLACE mód ve WF registru v módu 320 x 200

	Zápis do	WF registr	
4 barvy	sady A	10x0xxBB	--- DMD
	sady B	10x1BBxx	/ 0 nebo 1
16 barev	sady A i B	10xxBBBB	DMD 2

REPLACE mód ve WF registru v módu 640 x 200

	Zápis do	WF registr	
4 barvy	sady A	10x0xxxB	--- DMD
	sady B	10x1xBxx	/ 0 nebo 1
16 barev	sady A i B	10xxxBxB	DMD 2

21.14.3 Nastavení barvy pozadí znaku

Barva pozadí znaku má vždy po zápisu v REPLACE módu kód palety 0. Je to proto, že zapsáním nuly se nulují příslušné bity ve všech uvažovaných rovinách.

Pokud chcete mít pozadí znaku jiné barvy než je barva pozadí, musíte pracovat v jiném WMD než v REPLACE (např. PSET), pole obrazovky pod znakem vybarvit požadovanou barvou a potom teprve znak rozkreslit. (V BASICu lze stejného efektu dosáhnout kombinací příkazů BOX a SYMBOL)

21.14.4 Rozkreslování znaku na obrazovku

Protože obrazovka je bodová, nestačí znak pouze zobrazit na určité místo ve VRAM, ale je ho zapotřebí do VRAM rozkreslit. šířka znaků je 8 bitů, z nichž nejméně jeden musí tvořit mezeru mezi znaky. Výška znaku je zcela závislá na vůli programátora. Tvar znaků bývá zpravidla uložen v tzv. generátoru znaků, jehož struktura a uložení také závisí na programátorovi.

Standardně je generátor umístěn v EPROM od adresy 1000H. Sestává ze dvou sad po 256-ti znacích. Jeden znak zaujímá 8 po sobě jdoucích bytů, výška znaku je 8 bodů, v módech 640 x 200 se na obrazovku vejde 80 takových znaků. Podrobně je standardní generátor popsán v kapitole 5.3. a ve třetím dílu této příručky.

Jak již bylo uvedeno, je nejvhodnější mód pro rozkreslování znaků mód REPLACE.

Zápis v módu REPLACE se provádí takto:

a) Zápis bitu 1 způsobí zapsání takové barvy, jaká je specifikována v WF registru

b) Zápis bitu 0 způsobí zapsání barvy s kódem nula, to znamená 0 do všech uvažovaných rovin.

Příklad:

Uvedený podprogram rozkreslí znak, jehož kód je v registru C a adresa umístění horního okraje znaku ve VRAM je v registru HL.

```
;
; Rozkreslení znaku na obrazovku
;
; B .... barva písma, pozadí je černé
; C .... kód znaku
; HL ... adresa horního okraje znaku na obrazovce
; lze ji spočítat například takto:
;
; HL = RADEK * 640 + SLOUPEC
;
RZKR: DI              ; slušnost, bude zneužíván zásobník
      IN    A,(0E0H)  ; namapovat VRAM + generátor
      LD    A,40H     ; mod REPLACE
      OR    B         ; pridat barvu
      OUT   (0CCH),A  ; do WF registru
      EX    DE,HL     ; schovat adresu ve VRAM do DE
      LD    L,C
      LD    H,0       ; kód znaku do HL
      ADD   HL,HL
      ADD   HL,HL
      ADD   HL,HL     ; vynásobit kód znaku osmi
      LD    BC,1000H  ; adresa generátoru znaků
      ADD   HL,BC     ; HL je adresa našeho znaku
      LD    (SPS),SP
      LD    SP,HL     ; adresa do generátoru v SP
      EX    DE,HL     ; obnovit adresu ve VRAM
      LD    DE,80     ; pocet znaků na řádku
      REPT  4         ; 4* se opakuje (kvůli rychlosti)
      POP   BC        ; 2 byty z generátoru
      LD    (HL),C    ; 1. byte uložit
      ADD   HL,DE     ; posunout ukazatel
      LD    (HL),B    ; 2. byte uložit
      ADD   HL,DE     ; posunout ukazatel
      ENDM
      LD    SP,(SPS)  ; obnovit SP
      IN    A,(0E1H)  ; odmapovat VRAM
```

```
        RETI    ; povolení přerušení a návrat
;
SPS:    DEFS 2  ; buňka na úschovu SP
```

21.14.5 Čtení v textovém režimu

Čtení v textovém režimu je ještě méně obvyklé než při grafických operacích. Jedině co může být užitečné, je zjistit jaký je znak na dané pozici. Lze to provézt jen vyhledáním příslušného obrazu v generátoru znaků.

Příklad:

```
        ;
        ; Prečte znak z VRAM. Znak může být v jakékoli barvě,
        ; pozadí znaku všk musí být celé v barvě pozadí.
        ;
        ; HL ... adresa horního okraje znaku ve VRAM
        ; HL = RADEK * 640 + SLOUPEC
        ; A .... přečtený kód znaku
        ;
RZNAK:
        LD      A,80H     ; SEARCH mód
        OUT     (0CDH),A  ; do RF registru
        IN      A,(0E0H)  ; mapuj RAM
        PUSH    HL        ; schovat adresu znaku ve VRAM
        LD      DE,1000H  ; adresa generátoru znaků
        LD      C,0       ; v generátoru je 256 znaků
LOOP1:POP       HL        ; obnovit adresu znaku ve VRAM
        PUSH    HL        ; a zase schovat
        LD      B,8       ; výška znaku
LOOP2:LD        A,(DE)    ; vyber byte znaku z generátoru
        CPL               ; testuje se písmo, ne pozadí
        CP      (HL)      ; porovnej s bytem ve VRAM
        JR      NZ,NESEDI       ; tak tento znak to zase není
        PUSH    DE
        LD      DE,80
        ADD     HL,DE     ; adresa další části znaku ve VRAM
        POP     DE
        INC     DE        ; další část znaku v generátoru
        DJNZ    LOOP2     ; a porovnávej další byte
        POP     HL        ; srovnat stack
        IN      A,(0E1H)  ; odmavopat VRAM
        LD      A,C       ; zde máme kód znaku
        OR      A         ; CY = 0
        RET               ; návrat s nalezeným znakem
NESEDI:
        INC     DE        ; upravit adresu do generátoru
```

```
DJNZ    NESEDI  ; aby ukazovala na další znak
INC     C       ; počítadlno znaků
JR      NZ,LOOP1        ; a testuj další znak
IN      A,(0E1H) ; odmapovat VRAM
SCF             ; všechny znaky vyčerpány
POP     HL      ; vrať se s příznakem CY = 1
RET             ; znak nenalezen
```

21.14.6 Alternující režim a scroll

Alternující režim je režim ve kterém se používají sady A a B jako dvě nezávislé sady. Jedna z nich je pomocí DMD vybrána k zobrazování. Zapisovat lze do zobrazované nebo i do v tomto okamžiku neviditelné sady, a tak si obrázek předem nachystat. Pokud se nepoužívá scroll, stačí nastavit správně registry, a nemohou vniknout žádné potíže.

Pokud je ale scroll užit (viz. 4.13), je nutné si dát pozor na to, že zapisováním do neviditelných rovin v módu PSET a REPLACE se scroll neuplatní !. Zapsaná data se po přepnutí objeví na obrazovce jinde, než kde jsou očekávána.

22 Závěr

ROM mikropočítače SHARP MZ-821 obsahuje značné množství chyb různých druhů. Především jsou to koncepční chyby, které byly pravděpodobně z větší části způsobeny snahou o maximální kompatibilitu se staršími systémy firmy SHARP, především s řadou MZ-700 a MZ-80K.

Celá konzola je napsána tak, že bohužel není příliš použitelná. Pracuje pod několika různými kódy, které jsou si částečně podobné a z nichž nelze ani jeden označit za standardní. "ASCII" je zde jen parodií na americký standard, malá písmena jsou volně rozseta po horní části kódu, kde se naprosto libovolně střídají s grafickými znaky a pochopitelně nejsou uspořádána podle abecedy. A přitom v generátoru znaků jsou tak, že jejich převod na ASCII je pouze otázkou přičtení konstanty.

Takzvaný display kód je odvozen z generátoru znaků, ale má dvě varianty lišící se v řídících znacích, pro KBD: a CRT:.

Tabulky dekódování klávesnice zakazují veškeré CTRL-znaky, kromě šesti, které jsou shodné s editačními klávesami. Klávesy TAB a F1-F5 nejsou vůbec čteny.

Není jasné, proč jsou v paměti ROM texty chybových hlášení, když vlastní interpret není rezidentní.

Logický formát záznamu není příliš dokonalý a těch pár dobrých vlastností co má stejně neumí existující software využít.

Existují dva zcela nezávislé monitory s minimálními schopnostmi a podprogramy pro zpracování příkazů jsou tedy v ROM dvakrát.

Poněkud méně početnou skupinu tvoří chyby v kódu. Sem patří jednak vysloveně překlepové chyby jako např instrukce:

```
LD HL,71H
```

která má správně vypadat takto:

```
LD (HL),71H
```

A také velké množství různých mezer ap., které byly pravděpodobně vytvořeny pomocí pseudoinstrukcí DEFS nebo ORG.

Nebo třeba]GOPGM, kde není respektována existence souborů s nulovou délkou, přesto, že všechny ostatní podprogramy s něčím takovým počítají. V tomto případě končí volání tohoto podprogramu instrukcí LDIR s nulou v BC registru. To znamená pád systému, nebo v nejlepším případě nesmyslnou inicializaci periferních obvodů.

Také se zde velmi pečlivě testuje, zda již program není uložen od správné adresy, ale díky špatně spočítanému relativnímu skoku se LDIR stejně provede.

22.1 Několik rad nakonec

- Nezapomínejte, že SP je standardně v oblasti od 10F0H. Pokud namapujete CGROM a provedete CALL, jistě se budete divit.

- Jestliže chcete používat program uložený v hlavičce a s délkou bloku 0 je nutné, aby jako ukládací adresa byla uvedena hodnota 1200H. Jako první akci musí tento program zavolat @INI55 pro obnovu správného nastavení I8255.

- Přerušení sice přijde jednou za dvanáct hodin, ale o to je záludnější. Poškození 1038H se tedy může projevit až za hodně dlouho a nejspíše úplným pádem systému.

- Přepínač MZ-700 ON/OFF nemá s nastavením módu počítače takřka nic společného, je pouze testován před spuštěním nahraného programu a podle jeho polohy je softwarově nastaven příslušný mód. Pokud je nastavení módu také na začátku zaváděného programu, na poloze přepínače nezáleží.

- Standardní monitor vždy nahrává program od adresy 1200H a pak ho přemístí na adresu správnou. To ocení každý, kdo v té oblasti měl data a nechtěl o ně přijít. V tomto případě je nutné provést G00AD a takto přejít do původního monitoru MZ-700. Povel L tohoto monitoru už ukládá program hned od správného místa.

- V každém módu nelze použít pouze příslušné mapování I/O portů. V módu MZ-700 se tedy nelze odkazovat např. na port 0D2H.

Přejeme Vám i sobě hodně pěkných programů.

Petr a Martin

23 Obsah

Petr Odehnal, Martin Veverka a kolektiv

Kurs základního programového vybavení mikropočítače SHARP MZ 800
Popis podprogramů monitoru

obálka: ak.mal. Karel Čapek

Vydalo (c) ZENITCENTRUM HZ ÚV SSM, Hostímská 1, 26601 Beroun jako součást souboru kursu základního programového vybavení mikropočítače SHARP MZ 800. Samostatně neprodejné.
JKPOV 735 342 41 1526