OSOBNÍ POČÍTAČ

SORD M5

NÁVOD K POUŽITÍ

PRO ZÁKLADNÍ SESTAVU S PROGRAMOVACÍM JAZYKEM

BASIC - I

Vážený spotřebiteli,

před prvým zapnutím Vašeho nového osobního počítače SORD M5 věnujte náležitou pozornost níže uvedeným zásadám pro provoz tohoto přístroje.

Osobní počítač SORD M5 bude pro většinu z Vás novým přístrojem v oblasti spotřební elektroniky. Pro jeho správné použití platí specifické zásady. Je proto velmi důležité, abyste před prvým uvedením počítače do provozu pečlivě prostudovali první tři části tohoto návodu k použití s důrazem na partie, kde jsou uvedeny základní pokyny pro zapojení počítače.

K nejdůležitějším zásadám správného použití počítače patří:

1. Před zapojením síťového zdroje do sestavy počítače musí být vypínač síťového zdroje v poloze vypnuto - OFF, to znamená kolébkový vypínač je zamáčknut na straně označené OFF /zapnutí zdroje se provádí stlačením kolébkového vypínače na straně ON/.

2. Výstupní kabel síťového zdroje nesmí být zapojen do počítače, do kterého není zasunut programový modul BASIC-I, BASIC-F, BASIC-G, případně další moduly.

3. Při každé výměně programového modulu musí být síťový zdroj vypnut - poloha vypínače - OFF.

4. Počítač se nesmí zapínat ani vypínat vidlicí přívodní šňůry síťového zdroje.

Počítač a síťový zdroj je navržen pro trvalý provoz v běžných kancelářských podmínkách při teplotě do cca 25°C . Proto jej chraňte před přímým slunečním a tepelným zářením.

Plastikový obal počítače a klávesnici je možno při znečištění otřít vlhkým hadrem s malou koncentrací běžných saponátových prostředků. Konektory systémové sběrnice a tiskárny lze čistit pouze za sucha vyfoukáním prachu.

Jestliže budete dodržovat výše uvedené zásady pro použití počítače, bude Vám počítač SORD M5 spolehlivě sloužit po řadu let.

Všem Vám, novým majitelům počítače SORD M5 , přejeme řadu hezkých chvil a pěkných zážitků při práci s tímto přístrojem. Věříme, že Vám osobní počítač SORD M5 přinese mnoho nových poznatků a znalostí z oboru výpočetní techniky a mikroelektroniky a stane se pro Vás nepostradatelným zařízením spotřební elektroniky ve Vaší domácnosti.

Osobní počítač SORD MS je řešen jako universální pro použití všemi členy rodiny.

Jeho uvedení do chodu a ovládání je tak jednoduché, že s ním mohou pracovat i děti. Vedle různých počítačových her bude k dispozici i řada programů výukových. Mládež školního věku využije počítač např. pro řešení školních úloh z matematiky. Předpokladem pro to je ale znalost programování. Podle zkušeností ze zahraničí i od nás je minimální věk pro výuku programování asi devět let. Vítaným pomocníkem bude počítač i studentům na středních a vysokých školách, neboť na počítači M5 je možno s modulem BASIC-F řešit úlohy, které se řeší na školních počítačích. Další možnosti využití se nabízejí při každodenním chodu domácnosti, kde může počítač M5 pomáhat např. při sestavování jídelníčku nebo sledovat rodinné finance. Dále může zastávat funkci jiných zařízení, jako hodin nebo budíku a při napojení na další vnější čidla může regulovat teplotu nebo hlídat byt. Další aplikace jsou závislé na schopnostech a fantazii uživatele.

Vzhledem k tomu, že originální manuál v angličtině neobsahuje podrobnější popis počítače a nepopisuje všechny možnosti jazyka BASIC-I, byl napsán česky manuál úplně znovu.

BASIC-I je programovací jazyk určený pro úplné začátečníky v programování. S pomocí této příručky by se měl každý naučit programovat v jazyku BASIC-I a účelně využívat počítač M5 pro svoji potřebu. Manuál je rozdělen do sedmi dílů. V prvním dílu se seznámíte s koncepcí počítače, jeho blokovou strukturou a naučíte se počítač uvést do chodu a používat programy na demonstrační kazetě. Druhý díl je krátký kurs základu programovacího jazyka BASIC-I. Po jeho zvládnutí by každý měl být schopen sám sestavit krátký program. Třetí díl vás seznámí s dalšími příkazy jazyka BASIC-I pro barevnou grafiku a zvukový výstup. Všechny grafickémožnosti počítače a práci s pamětí VRAM se naučíte ve čtvrtém dílu. Pro řadu uživatelů bude jistě zajímavý další díl, ve kterém se dozví, jak k počítači připojovat vnější výstupní a vstupní zařízení. Šestý díl obsahuje obrazovou přílohu k příslušným částem manuálu. V posledním dílu jsou výpisy několika zajímavých programů s podrobným popisem, které by měly sloužit jako inspirace pro vaši další programátorskou činnost.

Manuál je určen pro úplné začátečníky, kteří se dosud neseznámili s žádným počítačem ani s programováním. Předem je varujeme před tím, aby okamžitě začali ťukat programy z posledního dílu. První čtyři díly manuálu jsou sestaveny z jednotlivých obsahově uzavřených částí, které na sebe navazují. V každé části je nejprve krátký vysvětlující text a potom program pro demonstraci. Čtení manuálu bez současně zapnutého počítače není pro začátečníky vhodné. Všechny programy uvedené v prvních čtyřech dílech manuálu doporučujeme poctivě vyzkoušet i s případnými změnami. Teprve po důkladném prostudování a procvičení prvních dvou částí přistupte k části třetí a čtvrté.

Příručku doporučujeme prolistovat i pokročilým programátorům, kteří hodlají využívat hlavně programovací jazyky BASIC-F a BASIC-G. Samostatně bude možné přikoupit i Monitor Handling Manual, který obsahuje popis všech podprogramů paměti 8KB ROM. S využitím podprogramu ve strojovém kódu se výrazně zvětší možnosti počítače.

Každý majitel počítače SORD M5 se může stát členem Klubu uživatelů osobních počítačů při 602.ZO Svazarmu v Praze 6. Na přiloženém listu jsou všechny potřebné informace.

O B S A H

1. Základní seznámení s počítačem 1

1.1 Stručný popis počítače 1
1.2 Uvedení počítače do chodu 2
1.3 Použití programů na demonstrační kazetě 3

2. Základy programování v jazyku BASIC-I 5

2.01 Klávesnice a obrazovkový editor 5
2.02 Režim okamžitého výpočtu 6
2.03 Číselné a řetězcové proměnné 6
2.04 Struktura programu, příkazy RUN,NEW a LIST 7
2.05 Příkaz vstupu INPUT, mazání programových řádek 7
2.06 Příkazy cyklu FOR - NEXT 8
2.07 Příkaz tisku PRINT 9
2.08 Číselná pole, dimenzování polí příkazem DIM 10
2.09 Příkazy pro práci s magnetofonem SAVE, VERIFY,OLD a CHAIN 11
2.10 Čtení dat z programu příkazy READ a DATA 11
2.11 Generátor náhodných čísel, podmíněný příkaz IF - THEN - ELSE 13
2.12 Příkaz skoku GOTO 14
2.13 Podprogramy, příkazy GOSUB a RETURN 15
2.14 Práce s pamětí pomocí příkazů PEEK a POKE 16
2.15 Vyjádření znaků ASCII kódy, příkazy CHR$ a ACSII 16
2.16 Přerušení programu, příkaz CONT 17
2.17 Příkazy pro práci s řetězci LEN,LEFT$,MID$,RIGHT$ a RPT$ 17
2.18 Čtení klávesnice příkazem INKEY$ 18
2.19 Příkazy VAL a NUM$ pro převody mezi řetězci a čísly 19
2.20 Programové řádky s více příkazy 21
2.21 Tvorba programů s příkazy AUTO a DEL 21
2.22 Použití návěští v programech 22
2.23 Chybová hlášení, příkazy ERR, ERRL a ERRL$ 22

3. Barevná grafika a zvuk v jazyku BASIC-I 24

3.01 Generátor znaků, příkaz VPEEK 24
3.02 Vyjádření čísel v šestnáctkové soustavě, příkaz HEX$ 24
3.03 Definování nových znaků, příkazy STCHR a MOD 26
3.04 Nastavení barev, příkaz VPOKE 26
3.05 Sprajty - grafické bloky definované uživatelem 28
3.06 Definování sprajtů a jejich pozice, příkazy MAG,SCOD,SCOL a LOC 28
3.07 Technický popis zvukového generátoru 29
3.08 Programování zvukového generátoru, příkaz OUT 30
3.09 Programové naladění zvukového generátoru 31

4. Práce s pamětí VRAM 32

4.01 Čtyři pracovní obrazovkové režimy 32
4.02 Použití příkazu STCHR v režimu GII 33
4.03 Jemná grafika v režimu GII 33
4.04 Grafické možnosti režimu Multi-color 34
4.05 Práce v textovém režimu 35
4.06 Záznam bloků paměti VRAM a RAM na magnetofonech 35
4.07 Práce na dvou obrazovkách 35

5. Připojení a programování vnějších zařízení 36

5.01 Připojení tiskárny 36
5.02 Adresování bloků počítače, rozšíření paměti RAM 37
5.03 Využití vstupů pro ovladače, příkazy INP a TIME 37

6. Obrazová příloha

1. ZÁKLADNÍ SEZNÁMENÍ S POČÍTAČEM

1.1. Stručný popis počítače

Koncepce počítače SORD M5 je stejná jako u většiny osobních počítačů zahraničních i domácích výrobců. Pro vstup informací do počítače slouží klávesnice. Všechny vstupní informace se současně zobrazují na připojeném standardním TV přijímači, který je základním výstupním zařízením počítače. Připojení TV přijímače k počítači je zajištěno kabelem z počítače do antenních zdířek TV přijímače. Výstupní video signál je v počítači modulován na určitou frekvenci v pásmu pro 2. TV program. Pro trvalý záznam programu a dat se připojuje běžný kazetový magnetofon. Napájení počítače M5 je řešeno vnějším síťovým zdrojem. U počítače M5 jsou ještě výstupní konektory s video a zvukovým výstupem, výstupní konektor pro připojení paralelní tiskárny a dva vstupní konektory pro ruční ovládače.

Vnitřní zapojení počítače se vyznačuje použitím moderních integrovaných obvodů a možnosti rozšíření systému počítače uživatelem po stránce technické i programové. Jednoduché blokové schéma je pro názornější představu o vnitřním uspořádání počítače nakresleno na obr. 1.

Základem každého osobního počítače je mikroprocesor a paměti. Mikroprocesor pracuje podle programu uloženého v paměti. Činnost mikroprocesoru a všech ostatních částí počítače M5 je řízena a synchronizována impulsy frekvence 3.58 Mhz, které se získávají z krystalového generátoru 14.32 Mhz. Mikroprocesor Z-80A je nejpoužívanějším mikroprocesorem u osobních počítačů i u menších profesionálních stolních počítačů. Je programové i technicky slučitelný s mikroprocesorem TESLA MHB 8080A.

Paměti používané u osobních počítačů jsou dvojího druhu. V pamětech ROM /Read Only Memory/ je program uložen již při jejich výrobě a nezmizí ani při vypnutí napájení. U těchto pamětí dochází pouze k čtení informací a k jejich přesunu na datovou sběrnici. Datová sběrnice tvoří část systémové sběrnice počítače M5.

Systémová sběrnice počítače je soustava paralelních vodičů, pomocí niž dochází k řízení jednotlivých částí počítače a k přesunu informací mezi nimi. Dále obsahuje systémová sběrnice adresovou sběrnici, několik pomocných řídících signálů a napájení. Paměť ROM počítače M5 na kapacitu 8KB a obsahuje všechny podprogramy pro činnost jednotlivých bloků počítače a jejich vzájemnou spolupráci. Systémová sběrnice je vyvedena na přímý konektor v horní zadní části počítače. Do něho se zasouvají programové moduly.

Modul BASIC-1 obsahuje další paměť ROM o kapacitě 8KB s programem, který společně s využitím podprogramů v základní paměti 8KB ROM umožňuje psaní programu v jazyku BASIC-1 , jejich odladování a jejich činnost. Název programovacího jazyka BASIC vznikl jako zkratka z anglických slov Beginners All-purpose Symbolic Instruction Code /víceúčelový kód symbolických instrukcí pro začátečníky/. Jazyk BASIC je vlastně jediný programovací jazyk, který byl vytvořen pro normální lidi a ne pro specialisty programátory. Jazyk BASIC byl navržen jako jazyk interaktivní. To znamená, že uživatel je při tvorbě programu, při jeho odladování i po jeho spuštění v neustálém kontaktu prostřednictvím klávesnice s počítačem. BASIC se používá u všech osobních počítačů a u většiny stolních profesionálních počítačů. Od svého vzniku byla vytvořena řada variant jazyka BASIC. Jednou z nich je i BASIC-1 firmy SORD pro osobní počítač M5.

Paměť RAM /Random Access Memory/ je určena pro krátkodobé uchování dat během práce mikroprocesoru. Nejdříve musí dojít k přesunu dat z datové sběrnice do paměti a teprve potom se mohou data opět přes datovou sběrnici přečíst zpět do mikroprocesoru. Základní paměť RAM na kapacitu 4KB. Kapacita paměti nám říká, kolik informací můžeme do ní uložit. Všechny počítače pracují vnitřně ve dvojkové soustavě a proto všechny hodnoty vyjadřující nějaký parametr počítače jsou mocninou dvou. Přesun informací se děje u malých osobních počítačů po paralelní osmivodičové sběrnici. Na každém vodiči mohou nastat pouze dva stavy - vodič je bez napětí nebo na vodiči je napětí. U osmi vodičů může při všech možných kombinacích nastat 2↑8 = 256 různých stavů. Těmto stavům přiřazujeme dekadická čísla 0 - 255. Pamětové buňky se skládají z osmi dílčích částí, z nichž každá může mít také dva stavy. V jedné pamětové buňce mohou být tedy také uložena čísla 0 - 255. Anglický termín byte/bajt/ je základní jednotka kapacity paměti. Paměť o kapacitě 1KB obsahuje 2↑10 = = 1024 pamětových buněk.

Jazyk BASIC-I pracuje pouze s celými čísly. Každé číslo je uloženo ve dvou bytech. Jestliže v jednom bytu může být uloženo číslo o velikosti 0 až 2↑8 - 1, potom ve dvou bytech může být číslo o velikosti 0 až 2↑16-1 /0 - 65535/. Vzhledem k tomu, že je nutno pracovat i se zápornými čísly, je číselný rozsah - 32768 až + 32767. Adresová sběrnice je šestnáctivodičová takže může být adresováno 64KB pamětových buněk. V blokovém schéma, kde jsou v horní části jednotlivé pamětové bloky, je čárkovaně vyznačena i vnější paměť 32KB, kterou je možno k počítači připojit.

Pod mikroprocesorem je ve schema časovač, který pomáhá při synchronizaci činnosti celého počítače. Vedle něj jsou pod systémovou sběrnicí další bloky počítače, které slouží pro vstup a výstup informací. Každý počítač je zařízení na zpracování informací. Informace nebo data je nutno do počítače nějakým způsobem dostat a po jejich zpracování je převést do formy čitelné uživatelem. Pro vstup informací slouží klávesnice s ručními ovladači a kazetový magnetofon. Pro výstup je určen televizní přijímač, magnetofon a tiskárna. Kompletní televizní video signál vytváří video generátor s vlastní pamětí VRAM /Video RAM/ o kapacitě 16KB. Zvukový signál se vytváří v programovatelném zvukovém generátoru. V třetí až páté části manuálu naleznete další podrobnosti o video generátoru, zvukovém generátoru a o připojení tiskárny a dalších vstupních a výstupních zařízeních.

1.2. Uvedení počítače do chodu

Kartonová krabice, kterou jste obdrželi, obsahuje:
1. počítač SORD M5
2. síťový napájecí zdroj
3. programový modul BASIC-I
4. demonstrační kazety se třemi programy /2 hry a testovací program/
5. propojovací kabel pro připojení TV přijímače
6. propojovací kabel pro připojení magnetofonu
7. BASIC-I Manual
8. User's Guide

Dále jste dostali český manuál s výpisy programů, informaci o záruce a servisu a informaci o možnosti vaší spolupráce s 602.ZO Svazarmu v Praze 6. Informace o dalším prodávaném příslušenství vám sdělí pracovníci prodejny, ve které jste počítač koupili.

Pro uvedení počítače do chodu je nezbytný TV přijímač s možností příjmu II. TV programu. Konektor pro připojení kabelu pro TV přijímač je zcela vpravo

při pohledu na počítač zezadu a je označen zkratkou RF. Kabel je přizpůsoben pro vstupní impedanci TV přijímače 75 Ohmů a je možno jej přímo zasunout do anténního konektoru všech TV přijímačů vyráběných v posledních letech. U starších TV přijímačů se vstupem o impedanci 300 Ohmů bylo vyzkoušeno, že není potřeba převodu ze 75 Ohmů; pouze je třeba zajistit vodivý přechod z kabelu do anténního vstupu.

Dalším krokem je zasunutí programového modulu BASIC-I do konektoru v pravé horní části počítače. Konektor je pod odklopným krytem, který zabírá celou zadní horní plochu počítače nad klávesnicí. Kryt se otáčí nad zadní hranou počítače a zvedá se proto těsně nad klávesnicí. Na levé a pravé straně krytu jsou dva výstupky pro ulehčení otevření. Programový modul se zasunuje shora tak, aby označení modulu bylo čitelné ze směru od klávesnice. Jinak ani není možno modul zasunout.

Nyní zbývá připojit síťový zdroj. Přesvědčete se, zda je vypínač v poloze OFF /vypnuto/ a zasuňte síťovou šňůru s vidlicí do zásuvky s napětím 220 V. Druhý kratší a silnější kabel síťového zdroje je ukončen vícepólovým kulatým konektorem, který patří do vstupního konektoru počítače zcela vlevo při pohledu zezadu a je označen zkratkou POWER. Je důležité si zapamatovat, že je možno zapnout vypínač na síťovém zdroji pouze při zasunutém programovém modulu. Jinak může dojít k poškození počítače. Zapnutí zdroje je indikováno rozsvícením červené LED diody vpravo nad klávesnicí.

Na závěr zbývá zapnout TV přijímač a naladit ho na 36. kanál, kdy se na černé obrazovce objeví bílými písmeny napsáno BASIC-1, pod tím Ready a inverzní blikající L. U televizoru, který může přijímat zvukový doprovod i v normě CCIR, si nastavíme optimální hlasitost pro zvukovou indikaci stisku tlačítek klávesnice. Stačí stisknout libovolné znakové tlačítko, chvíli ho držet a současně nastavit hlasitost. Majitele TV přijímačů bez zvuku v normě CCIR mají možnosti získat zvukový signál z výstupního konektoru označeném AUDIO po zesílení nf zesilovačem.

1.3 Použití programů na demonstrační kazetě

Majitelé barevných TV přijímačů s normou PAL mají možnost si nastavit optimálně barvy pomocí prvního programu na demonstrační kazetě. Propojovací kabel pro magnetofon je na jedné straně ukončen vícepólovým kulatým konektorem, který se zasunuje na zadní straně počítače do konektoru označeného CASSETTE. Konektory od síťového zdroje a magnetofonu jsou různé, takže není třeba se obávat toho, že by se konektor síťového zdroje omylem zasunul do vstupního konektoru pro magnetofon. Na druhém konci jsou tři kolíkové konektory: bílý, červený a černý. Bílý je určen pro výstup magnetofonu s označením EAR nebo MON. Červený je pro mikrofonní vstup s označením MIC. Černý je pro dálkové zapínání a vypínání a patří proto do konektoru magnetofonu s označením REM. To platí pouze pro většinu běžných magnetofonů zahraniční výroby. U ostatních magnetofonů s DIN konektory je třeba si konec kabelu předělat. U kolíkových konektorů je společná zem na delší části kolíku. Kolík dálkového ovládání je připojen na kontakty spínacího relé.

Při nahrávání prvního programu na demonstrační kazetě postupujte následujícím způsobem:

1. propojte magnetofon s počítačem
2. zasuňte demonstrační kazetu do magnetofonu
3. nastavte hlasitost na maximum, případně tónovou clonu na maximální výšky

4. stiskněte klávesu PLAY nebo START magnetofonu
5. stiskněte na klávesnici tlačítka TAPE /na obrazovce se objeví nápis tape/ a tlačítko RETURN.

Magnetofon s dálkovým ovládáním se rozeběhne. U magnetofonu bez dálkového ovládání je vhodnější postup podle bodu 4 a 5 obrátit. Za chvíli se na obrazovce objeví nápis TVADJUST#.Po nahrání programu se magnetofon zastaví a na obrazovce se objeví 15 svislých barevných pruhů vhodných pro nastavení barevného TV přijímače. Po stisknutí libovolného tlačítka program pokračuje několika dalšími barevnými a tvarovými kombinacemi, až se vrátí na původní pruhy. Program je ve strojovém kódu a jediná možnost jak jej ukončit, je vypnout šítový zdroj.

Každý program je nahrán po blocích o délce 256 bytů. Na konci každého bloku je součet všech programových kódů bloku. Při nahrávání bloku se v počítači současně vytváří součet kódů, který se na konci bloku porovnává s nahraným součtem. Při nesouhlasu a tedy i při chybě během čtení bloku se čtení programu z pásky přeruší a na obrazovce se objeví chybové hlášení Err 18 in 0. Potom je nutno nahrávání programu opakovat. Při převíjení pásky je třeba vysunout kolík s dálkovým ovládáním.

Na demonstrační kazetě jsou dále programy BIORHYTHM DIAGNOSIS /Výpočet biorytmu/ a MUSIC TONE /Tóny z klávesnice/. Tyto programy jsou napsány v jazyku BASIC-1. Jejich výpisy s poznámkami o jejich činnosti jsou v programové části manuálu. Pro nahrání a spuštění obou programů se musí stisknout tlačítka CHAIN a tlačítko RETURN.

Program Výpočet biorytmu se skládá ze čtyř částí. První část je krátký program, který na obrazovce vytvoří název programu a pod ním grafický obrazec. Při nahrávání dalších dvou částí dojde k tvarovým i barevným změnám. Po nahrání poslední čtvrté části se na obrazovce objeví otázka "When is your birthday?". Uživatel musí nyní zadat datum narození ve tvaru RRRR,MM,DD. Při chybě v zadání je možno vymazat poslední znak současným stisknutím tlačítek CTRL a DEL. Tlačítko DEL je vpravo nahoře vedle tlačítka RESET. Po napsání data narození /např. 1960, 6, 28/ je potřeba stisknout tlačítko RETURN. Na obrazovce se objeví další otázka "When is your partner's birthday?" a uživatel musí zadat datum narození svého partnera stejným způsobem jako u sebe. Po další otázce "Biorhythms for what year and month?" se zadá rok a měsíc ve tvaru RRRR,MM a po stisknutí tlačítka RETURN dojde ke kreslení biorytmických křivek v pořadí zdravotní stav, citový stav a duševní stav. Na závěr se napíše určený vzájemný soulad. Dále můžete pokračovat třemi způsoby: vybrat si jiný rok a měsíc, zadat datum narození jiného partnera nebo začít úplně od začátku. Stačí stisknout jedno z tlačítek 1,2,3. Ukončení programu se provede současným stisknutím tlačítek SHIFT a RESET.

Program Tóny z klávesnice se skládá ze dvou částí. První část na obrazovce opět vytvoří název programu a grafický obrazec. Po nahrání druhé části se na obrazovce nakreslí část klávesnice, jejíž jednotlivé klávesy jsou označeny znaky klávesnice ze dvou řad. Při držení určitého tlačítka se generuje tón odpovídající příslušné klávese. Uživatel si nejdříve musí vybrat zda chce zvuk varhan nebo piána a dále si volit výškový rozsah zobrazené klávesnice. Pro výběr výškového rozsahu stačí kdykoliv mezi "hraním" stisknout jedno z tlačítek 1,2,3. Při změně barvy se musí nejdříve stisknout tlačítko RETURN. Ukončení programu se provede opět současným stisknutím tlačítek SHIFT a RESET.

2. ZÁKLADY PROGRAMOVÁNÍ V JAZYKU BASIC-1

2.01 Klávesnice a obrazovkový editor

Klávesnice je až na několik dalších tlačítek a chybějící mezerník totožná s klávesnicí psacího stroje. Proto budou mít určitou výhodu uživatelé, kteří umí alespoň trochu psát na psacím stroji. Stisknutí každého znakového tlačítka je indikováno zvukově a současně se na obrazovce objeví příslušný znak na pozici, na které je před stisknutím tlačítka blikající inverzní znak ve funkci ukazatele nebo tzv. kurzoru. Obrazovkový editor je součástí programového systému počítače a je určen pro práci s textem na obrazovce, zvláště pro psaní a opravu programu. Jeho základní funkce je nutno dobře zvládnout před vlastním programováním a proto vám doporučujeme vše, co bude v dalším textu vysvětleno, si okamžitě vyzkoušet přímo na počítači.

Všechny funkce obrazovkového editoru pracuji s opakováním. To znamená, že při delším držení určitého tlačítka se příslušná funkce automaticky opakuje. Držíme-li libovolnou znakovou klávesu stisknutou delší dobu, vidíme v řádce na obrazovce stále nové stejné znaky současně se zrychlenou zvukovou indikací. Po ukončení jednoho řádku se pokračuje na začátku řádku dalšího. Po zaplnění všech dvacetičtyř řádek dojde k posunu obsahu obrazovky o jeden řádek nahoru a pokračuje se stále na posledním řádku.

Po zapnutí počítače je kurzor vyznačen blikajícím inverzním L a na obrazovku se píší malé znaky abecedy nebo číslice a další speciální znaky. Tlačítka SHIFT v levém a pravém spodním rohu fungují jako zvednutí vozíku u psacího stroje pro psaní velkých písmen abecedy. Tlačítko SHIFT je nutno držet současně s tlačítkem znakovým. Tlačítko FUNC na levé straně mění zobrazovací režimy klávesnice. První tři tlačítka zleva v horní řadě slouží společně s tlačítkem FUNC k nastavení zobrazovacího režimu s malými znaky /letters/, s velkými znaky /capitals/ a s grafickými znaky /graphics/. Kurzor se mění na inverzní L, C a G. Ve všech třech režimech funguje i tlačítko SHIFT. Pomocí tlačítka FUNC je možno na jedno stisknutí tlačítka vypsat celý příkaz nebo funkci jazyku BASIC-1. Stačí držet tlačítko FUNC současně s tlačítkem označeným příslušným příkazem nebo funkcí.

Při psaní textu můžeme kurzor posunovat na libovolnou pozici obrazovky pomocí tlačítek; @ : / při současném držení tlačítka CTRL na levém okraji klávesnice. Původně napsaný text je potom možno přepsat. Mezera se zobrazí tlačítkem SPACE na pravém okraji klávesnice.

Pomocí tlačítka CTRL a dalších několika znakových tlačítek se vyvolávají další funkce obrazovkového editoru. Jestliže chceme do napsaného textu doplnovat znaky, je možno se přepnout po stisknutí CTRL+P do doplnovacího režimu, při němž dojde i ke změně kurzoru z velkých znaků L, C a G na malé l, c a g. Ke zpětnému přepnutí do původního přepisovacího režimu dojde po CTRL+O. Při mazání části textu je třeba použít po nastavení kurzoru na začátek části, kterou chceme vymazat, tlačítka CTRL+DEL. Tlačítko DEL je vpravo nahoře vedle RESET. Další funkce obrazovkového editoru vyvolávané tlačítkem CTRL jsou uvedeny v následujícím přehledu:

CTRL+B - posun kurzoru na začátek řádky
CTRL+C - posun obrazovky o jeden řádek dolů
CTRL+D - posun obrazovky o jeden znak doleva
CTRL+E - posun obrazovky o jeden řádek nahoru
CTRL+F - posun obrazovky o jeden znak vpravo
CTRL+I - tabulátor /8 mezer/

CTRL+K - posun kurzoru do levého horního rohu
CTRL+N - posun kurzoru na začátek další řádky
CTRL+X - vymazání textu do konce

Význam tlačítek RETURN, RESET a dalších s CTRL bude postupně vysvětlen v dalších částech.

2.02 Režim okamžitého výpočtu

Obrazovkový editor popsaný v předchozí části pracuje s textem v rozsahu jedné obrazovky. Text na obrazovce je zároveň uložen v paměti VRAM. Napíšeme-li na obrazovku libovolný text a stiskneme-li tlačítko RETURN, objeví se nám na následující řádce chybové hlášení Err 2 in 0. Tlačítko RETURN totiž přerušuje činnost obrazovkového editoru a způsobuje přesun textu z obrazovky a tím i z paměti VRAM pro zpracování počítačem. V případě nesrozumitelného textu pro počítač se objeví chybové hlášení. Texty pro zpracování počítačem při práci uživatele s obrazovkovým editorem mohou být pouze ve tvaru příkazu v režimu okamžitého výpočtu nebo ve tvaru programových řádek.

Příkazy v režimu okamžitého výpočtu jsou krátké povely uživatele pro počítač, které se po napsání a po stisknutí tlačítka RETURN okamžitě provedou. Jako příklad je možno uvést příkazy TAPE a CHAIN použité při nahrávání programu z demonstrační kazety. Dalším názorným příkladem je příkaz PRINT. Za mezerou může být za ním libovolný aritmetický výraz, jehož výsledek se objeví po stisknutí tlačítka RETURN na následující řádce. Jako příklad si můžete napsat PRINT ((2+3) * (8+2)) /10. Zde je třeba upozornit na to, že programovací jazyk BASIC-I pracuje pouze s celočíselnými proměnnými v rozsahu - 32768 až+ 32767. V režimu okamžitého výpočtu můžeme za sebou napsat několik příkazů, které ale musí být mezi sebou odděleny dvoutečkami. Pro mazání obrazovky je určen příkaz CLS. S dalšími povely pro práci v režimu okamžitého výpočtu se seznámíte později.

2.03 Číselné a řetězcové proměnné

Každé číslo je uloženo ve dvou paměťových buňkách, kde může nabývat libovolných hodnot v rozsahu -32768 až + 32767. Čísla, která mohou měnit svoji hodnotu, nazýváme číselné proměnné. Pro jednoznačné určení adresy paměťových buněk s číselnou proměnnou umožňuje jazyk BASIC-I přeřadit každé číselné proměnné její název. Název proměnné musí vždy začínat abecedním znakem. Ostatní znaky názvu mohou být i číslice a další speciální znaky. Posledním znakem nesmí být znak $. Maximální délka názvu proměnné je 32 znaků. S názvy proměnných se pracuje stejně jako s číselnými konstantami. Jako názornou ukázku si můžeme napsat následující řádky. Každý musí být ukončen stisknutím tlačítka RETURN:

```
DRAHA=200
CAS=4
RYCHLOST=DRAHA/CAS
PRINT RYCHLOST
```

Programovací jazyk BASIC se vyznačuje i tím, že může vedle čísel zpracovávat i texty. Proto je nutné se seznámit s pojmem řetězcové konstanty a řetězcové proměnné. Řetězcová konstanta je libovolná kombinace znaku uzavřená v úvozovkách. Nejčastěji se používá v příkazu PRINT. Zkuste napsat text PRINT "VÝSLEDEK NÁSOBENÍ 2 + 3 =";2+3 a stisknout tlačítko RETURN. Řetězcová

proměnná je libovolná kombinace znaku o maximální délce 18 znaků uložena v paměti RAM, které je pro její jednoznačné určení přiřazen název. Pro rozlišení názvů číselných a řetězcových proměnných musí být poslední znak názvu řetězcové proměnné znak $. Pro názornost si zkuste napsat:

```
TISK1$="VYSLEDEK "
TISK2$="NASOBENI "
PRINT TISK1$;TISK2$;"2 * 3 = ";2*3
```

Všechny informace a data, s nimiž se v jazyku BASIC-I pracuje, jsou ve formě jednoduchých nebo indexovaných číselných a řetězcových proměnných. Jednoduché číselné a řetězcové proměnné jsme právě stručně probrali. Věříme, že uživatel si sám udělá řadu příkladů na procvičení v režimu okamžitého výpočtu. S indexovanými proměnnými neboli poli se setkáme později.

2.04 Struktura programu, příkazy RUN, NEW a LIST

Nyní již můžeme přistoupit k vlastnímu programování, neboť umíme pracovat s klávesnicí a jednoduchými proměnnými v režimu okamžitého výpočtu. Před psaním programu je vhodné si vymazat obrazovku povelem CLS. Rychlejší je někdy na chvíli držet tlačítka CTRL+E, kdy se obsah obrazovky posune nahoru a kurzor zůstane na původním místě.

Program v každé variantě jazyka BASIC se skládá z očíslovaných programových řádek. Jako ukázku můžeme použít příklad použitý v předchozí části:

```
10 DRAHA=200
20 CAS=4
30 RYCHLOST=DRAHA/CAS
40 PRINT RYCHLOST
```

Po skončení čtvrtého řádku a po stisknutí RETURN se nám neobjeví číslo 50 jako v režimu okamžitého výpočtu. Jestliže ale použijeme příkaz CLS a příkaz LIST /výpis programu/, dostaneme výpis programu ve tvaru:

```
10 LET DRAHA=200
20 LET CAS=4
30 LET RYCHLOST=DRAHA/CAS
40 PRINT RYCHLOST
```

Přiřazovací příkaz LET je totiž nepovinný pro uživatele, ale ve výpisu se vždy objeví.

Ke spuštění programu slouží příkaz RUN. K vymazání programu i hodnot všech proměnných použijeme příkaz NEW. V povelu LIST můžeme použít dva parametry, které nám určují první a poslední vypisovanou řádku. Příkaz LIST 20,30 nám vypíše pouze řádky 20 a 30. Pro vypsání jediné řádky musíme číslo řádku napsat dvakrát /LIST 20,20/. Výpis programu je možno zastavit tlačítkem space a přerušit tlačítky SHIFT a RESET.

2.05 Příkaz vstupu INPUT, mazání programových řádek

U našeho prvního programu z předchozí části se sice pracuje s proměnnými DRAHA a CAS, ale přiřazujeme jim konstantní hodnoty. Jestliže změníme řádky 10 a 20, můžeme hodnoty proměnných DRAHA a CAS zadávat z klávesnice:

```
10 INPUT DRAHA
20 INPUT CAS
```

Příkaz INPUT je určen pro vstup hodnot číselných nebo řetězcových proměnných uživatelem z klávesnice. Pro označení názvu proměnné můžeme její název doplnit do příkazu INPUT:

```
10 INPUT "VELIKOST DRAHY ";DRAHA
20 INPUT "VELIKOST CASU ";CAS
```

Pomocí jednoho příkazu INPUT můžeme zadávat i více vstupních hodnot, které musíme oddělovat čárkou:

```
10 INPUT "VELIKOST DRAHY, VELIKOST CASU ";DRAHA,CAS
```

Řádek 20 potom ale musíme vymazat. To se provede napsáním čísla řádku a stisknutím tlačítka RETURN. Jistě jste si všimli, že se řádky 10 a 20 stále mění. Stačí je psát stále znovu. Druhou možností je přímo ve výpisu dělat úpravy textu programové řádky. Po skončení opravy je nutno stisknout tlačítko RETURN. Jinak se opraví pouze text na obrazovce ale ne text řádku v paměti RAM. Při chybě při vstupu v příkazu INPUT můžeme mazat znaky stisknutím tlačítek CTRL+DEL. Příkaz INPUT provádí kontrolu druhu proměnné i jejich počtu. Zkuste zadat místo čísla znaky nebo menší počet čísel oddělených čárkou, než je uvedeno v příkazu INPUT.

Při použití příkazu INPUT pro vstup hodnot řetězcových proměnných je v příkazu INPUT seznam názvu řetězcových proměnných:

```
10 INPUT "JMENO, PRIJMENI, ROK NAROZENI ";J$,P$,R$
20 PRINT "JMENO ";J$
30 PRINT "PRIJMENI ";P$
40 PRINT "ROK NAROZENI ";R$
```

Příkaz vstupu INPUT je jedním z nejdůležitějších příkazů jazyku BASIC, který se vyskytuje prakticky ve všech programech, neboť pomocí něho se zadávají vstupní informace nebo data uživatele z klávesnice.

2.06 Příkazy cyklu FOR - NEXT

S dosavadními znalostmi již umíme sestavit krátký program, který má všechny potřebné základní části: vstup dat, zpracování dat a výstup zpracovaných dat. Program začíná na první programové řádce a postupně se provádějí další řádky. V řadě programů je potřeba opakovat vícekrát některou část programu. Potom se používají tzv. cykly. Opakovaná část programu se nazývá tělo cyklu. Před tělem cyklu musí být hlavička ve tvaru FOR I=P TO K STEP S, kde

I - ŘÍDÍCÍ PROMĚNNÁ CYKLU
P - POČÁTEČNÍ HODNOTA ŘÍDÍCÍ PROMĚNNÉ CYKLU
K - KONEČNÁ HODNOTA ŘÍDÍCÍ PROMĚNNÉ CYKLU
S - KROK, O NĚJŽ SE ZVYŠUJE NEBO SNIŽUJE HODNOTA ŘÍDÍCÍ PROMĚNNÉ

Za tělem cyklu musí být příkaz konce cyklu NEXT I. Řídící proměnná cyklu I musí být v hlavičce i v příkazu konce cyklu stejná. Při vynechání konce hlavičky /STEP S/ se automaticky bere krok jedna. Počáteční hodnota řídící proměnné, konečná hodnota řídící proměnné a krok mohou být konstanty, proměnné nebo aritmetické výrazy. Jejich hodnoty e však v průběhu cyklu nesmí měnit.

Použití příkazu cyklu si ukážeme na krátkém programu pro zopakování malé násobilky. Před psaním programu nezapomente vymazat předchozí program příkazem NEW.

```
10 CLS
20 INPUT "ZAKLAD (1 - 10) ";N
30 FOR I=1 TO 10
40 PRINT I,I*N
50 NEXT I
```

2.07 Příkaz tisku PRINT

Bez většího vysvětlování jsme používali již několikrát příkaz PRINT. Příkaz tisku PRINT je také jeden z často používaných příkazů, neboť slouží k zobrazení hodnot číselných a řetězcových proměnných na obrazovce TV přijímače. Základní varianta příkazu tisku obsahuje za slovem PRINT seznam názvů proměnných, konstant a aritmetických výrazů oddělených mezi sebou středníkem nebo čárkou. Při oddělení středníkem dochází k tisku jednotlivých položek seznamu těsně za sebou. Čísla jsou oddělená pro lepší čitelnost dvěma mezerami. Při oddělení čárkou dochází k tzv. zonovému tisku, kdy jsou začátky tisku jednotlivých položek seznamu od sebe vzdáleny osm znaků. S použitím příkazu cyklu si můžeme základní funkce příkazu PRINT vyzkoušet na následujících programech:

```
10 CLS
20 FOR I=1 TO 20
30 PRINT I;I*10;I*100
40 NEXT I
```

```
10 CLS
20 FOR I=1 TO 20
30 PRINT I,I*10,I*100
40 NEXT I
```

V příkazu PRINT můžeme také nastavit začátek tisku na libovolnou pozici žádku s použitím další varianty příkazu PRINT ve tvaru PRINT TAB (S), kde S je konstanta nebo proměnná, případně aritmetický výraz označující sloupec 0 - 31 se začátkem tisku. K vyzkoušení použijte následující krátký program:

```
10 CLS
20 FOR I=0 TO 10
30 PRINT TAB(I);I*100;TAB(2*I);I*10;TAB(3*I);I
40 NEXT I
```

Jistě jste si všimli, že všechny řádky s příkazem PRINT nemají na konci středník nebo čárku. Potom dochází k dalšímu tisku na následující řádce. To znamená, že při tisku se pokračuje stále dolů bez možnosti návratu směrem nahoru. I při použití středníku nebo čárky na konci řádku s příkazem PRINT se bude pokračovat po zaplnění jednoho řádku na řádek následující. V některých programech budeme ale potřebovat tisknout na kterémkoliv místě obrazovky zadaném číslem řádku a číslem sloupce. K tomu slouží poslední varianta příkazu PRINT ve formě PRINT CURSOR(S,R) a kde S a R jsou konstanty nebo proměnné, případně aritmetické výrazy označující sloupec 0 - 31 a řádek 0- 23 začátku tisku. Následující program vám vytiskne celou malou násobilku. Program se vyznačuje použitím jednoho cyklu uprostřed těla cyklu druhého.

```
10 CLS
20 FOR R=1 TO 10
30 FOR S=1 TO 10
40 PRINT CURSOR((S-1)*3,R*2);S*R
50 NEXT S
60 NEXT R
```

2.08 Číselná pole, dimenzování poli příkazem DIM

Vedle jednoduchých číselných proměnných můžeme pracovat i s číselnými proměnnými ve formě poli. Proměnná pole vedle názvu pole potřebuje k svému jednoznačnému určení i pořadové číslo. Protože číselná pole mohou zabírat v paměti poměrné velkou kapacitu, je potřeba před použitím pole v programu provést tzv. dimenzování pole pomocí příkazu DIM. Jestliže chceme pracovat např. s jednorozměrným polem s názvem POLE1 o velikosti dvěstě číselných prvků, musíme na začátku programu použít řádek:

```
20 DIM POLE1(200)
```

Provedením tohoto řádku při běhu programu se vytvoří v paměti prostor pro pole s názvem POLE1 a všechny prvky pole dostanou nulové hodnoty. To si můžeme ověřit po napsání dalších řádek a po spuštění programu:

```
30 FOR I=1 TO 200
40 PRINT POLE1(I);
50 NEXT I
```

Následující program je už poněkud delší a je to také první program, který je užitečný. Provádí řádkové a sloupcové součty tabulky o deseti řádkách a pěti sloupcích. Na programu si můžeme všimnout několika nových věcí.

Poprvé se používá příkaz REM. Je to příkaz nevýkonný, který je určen pro poznámky programátora. Při delších programech je dobré si funkci kratších částí programu označit právě v příkazu REM. Každý program je také vhodné označit jménem. Jméno programu je uvedeno v řádce 10.

Na řádce 20 vidíme, že v příkazu DIM můžeme dimenzovat více polí.

Zajímavý je řádek 50, kde nám příkaz PRINT nastavuje polohu kurzoru pro vstup dat příkazem INPUT.

Jinak by vám měl být program zcela jasný.

```
10 REM *** TABULKA ***
20 DIM A(10,5),R(10),S(5)
30 REM VSTUP TABULKY, VYPOCET A TISK SOUCTU RADEK
40 CLS
50 FOR R=1 TO 10
60 FOR S=1 TO 5
70 PRINT CURSOR((S-1)*5,R*2-2);
80 INPUT A(R,S)
90 R(R)=R(R)+A(R,S)
100 NEXT S
110 PRINT CURSOR(27,R*2-2);R(R)
120 NEXT R
130 REM VYPOCET A TISK SOUCTU SLOUPCU A CELKOVEHO
140 FOR S=1 TO 5
150 FOR R=1 TO 10
160 S(S)=S(S)+A(R,S)
170 NEXT R
180 PRINT CURSOR(S*5-4,21);S(S);
190 M=M+S(S)
200 NEXT S
210 PRINT CURSOR(27,21);M;
```

2.09 Příkazy pro práci s magnetofonem SAVE, VERIFY, OLD a CHAIN

Náš poslední program je už poněkud delší a také se může někdy hodit. Bylo by jistě škoda si ho vymazat. Proto si ho zaznamenáme na magnetofonovou kazetu. Současně se na něm naučíme všechny příkazy potřebné pro práci s magnetofonem.

Za předpokladu, že máme v magnetofonu čístou kazetu přetočenou na začátek aktivní části pásku a že je magnetofon propojen s počítačem, můžeme program TABULKA nahrát na pásku. Pro záznam programu slouží příkaz SAVE následovaný po mezeře povinným názvem programu v úvozovkách o maximální délce devíti znaků. Před příkazem SAVE musíme zapnout magnetofon pro nahrávání. Magnetofon s dálkovým ovládáním se rozeběhne až po stisknutí tlačítka RETURN po napsání SAVE "TABULKA". Po nahrání programu se magnetofon s dálkovým ovládáním zastaví a na obrazovce se objeví READY s kurzorem. Magnetofon bez dálkového ovládání musíme zastavit ručně. Většina magnetofonů je vybavena automatickým nastavením úrovně záznamu při nahrávání. Před vlastním programem je několik vteřin zaváděcí signál konstantní frekvence a úrovně určený právě pro nastavení automatiky. Proto by u běžného magnetofonu měl být záznam proveden kvalitně. Některé magnetofony mají nastavení úrovně záznamu ručně. Zde je potom potřeba si nastavit úroveň ručně podle návodu výrobce magnetofonu.

Pro ověření správnosti záznamu na kazetě je určen příkaz VERIFY . Tento příkaz čte program z pásku a provnává ho s programem v paměti pročítače. Před jeho použitím si musíme přetočit kazetu zpět při vysunutí kolíku pro dálkové ovládání. Některé magnetofony umožnují převíjení vpřed i vzad i při rozpojeném dálkovém ovládání. Další postup je stejný jako při nahrávání programu z demonstrační kazety. Místo příkazu TAPE nebo CHAIN musíme použít příkaz VERIFY nebo VERIFY "TABULKA". V prvním případě se ověřuje první program na pásce, který je nalezen. V druhém případě se ověřuje pouze program s názvem "TABULKA". Při chybě čtení programu z pásky se objeví chybové hlášení Err 18 in 0. Názvy programů před programem hledaným se vypisují postupně na obrazovce. Název programu hledaného se vypíše doplněný na konci hvězdičkou. Příkaz VERIFY doporučujeme používat po každém záznamu programu na pásku příkazem SAVE.

K nahrání programu z magnetofonu do počítače slouží poněkud nezvyklý příkaz OLD. Všechny ostatní varianty jazyků BASIC totiž používají příkaz LOAD. Příkazem OLD se vymaže program v paměti počítače a nahraje se první program na pásce. V případě použití jména programu např. OLD "TABULKA" se hledá daný program a názvy předchozích programů se opět vypisují.

Posledním příkazem pro práci s magnetofonem je příkaz CHAIN. Funguje stejně jako příkaz OLD, ale navíc provede spuštění programu od první programové řádky. Příkaz CHAIN jsme již používali u programu na demonstrační kazetě. Může se používat na konci programu pro tzv. řetězení programů.

Příkazem SAVE je možno nahrát na pásku i části paměti RAM nebo VRAM. S tím se však seznámíme později.

2.10 Čtení dat z programu příkazy READ a DATA

Příkaz INPUT slouží pro vstup číselných nebo řetězcových proměnných uživatelem z klávesnice. V některých případech však budeme potřebovat hodnoty proměnných přímo v průběhu programu. Jako příklad si uvedeme program na naplnění a tisk pole MESICE$ s názvy jednotlivých měsíců. Zároveň se naučíme pracovat s poli řetězcových proměnných.

```
10 CLS
20 DIM MESICE$(12)
30 MESICE$(1)="LEDEN"
40 MESICE$(1)="UNOR"
50 MESICE$(3)="BREZEN"
60 MESICE$(4)="DUBEN"
70 MESICE$(5)="KVETEN"
80 MESICE$(6)="CERVEN"
90 MESICE$(7)="CERVENEC"
100 MESICE$(8)="SRPEN"
110 MESICE$(9)="ZARI"
120 MESICE$(10)="RIJEN"
130 MESICE$(11)="LISTOPAD"
140 MESICE$(12)="PROSINEC"
150 FOR M=1 TO 12
160 PRINT TAB(1-M/10);".";TAB(7);MESICE$(M)
170 NEXT M
```

Řádky 30 až 140 zabírají zbytečně příliš místa, protože se u hodnoty každé proměnné v přiřazovacím příkazu musí uvést i jméno pole a pořadí proměnné. Stejnou funkci však bude mít i následující program:

```
10 CLS
20 DIM MESICE$(12)
30 FOR M=1 TO 12
40 READ MESICE$(M)
50 NEXT M
60 FOR M=1 TO 12
70 PRINT TAB(1-M/10);".";TAB(7);MESICE$(M)
80 NEXT M
90 DATA LEDEN,UNOR,BREZEN,DUBEN,KVETEN,CERVEN
100 DATA CERVENEC,SRPEN,ZARI,LISTOPAD,PROSINEC
```

Na řádce 40 se poprvé setkáváme s příkazem READ. Za příkazem READ následuje vždy název jedné nebo více proměnných oddělených čárkami. Proměnné mohou být číselné i řetězcové. S příkazem READ přímo souvisí příkaz DATA. Za příkazem DATA následuje seznam hodnot proměnných, které se příkazem READ přiřazují jednotlivým proměnným. Je celkem pochopitelné, že počet hodnot proměnných v příkazu DATA musí odpovídat počtu čtení příkazem READ. Také musí souhlasit druh proměnných. Do našeho programu si můžeme doplnit i počty dnů v jednotlivých měsících. Stačí napsat následující řádky pro doplnění programu:

```
20 DIM MESICE$(12),DNY(12)
40 READ MESICE$(M),DNY(M)
70 PRINT TAB(1-M/10);".";TAB(7);MESICE$(M);TAB(17);DNY(M)
90 DATA LEDEN,31,UNOR,29,BREZEN,31,DUBEN,30
100 DATA KVETEN,31,CERVEN,30,CERVENEC,31,SRPEN,31
110 DATA ZARI,30,RIJEN,31,LISTOPAD,30,PROSINEC,31
```

Příkaz DATA může být kdekoliv v programu, dokonce i před příkazem READ. V programátorské praxi se stalo dobrým zvykem umísťovat příkaz DATA až na úplný konec programu.

2.11 Generátor náhodných čísel, podmíněný příkaz IF - THEN - ELSE

Prakticky každá varianta jazyku BASIC na příkaz, který funguje jako generátor náhodných čísel. Náhodná čísla se výborně uplatnují v nejrůznějších počítačových hrách, ale jsou zároveň obrovskou zásobárnou čísel např. při ladění programu. Pojem ladění programu se už v předchozím textu vyskytl bez náležitého vysvětlení. Ladění programu je souhrn činností programátora, které vedou po napsání první varianty nového programu k odstranění všech chyb v programu až k variantě programu důkladně vyzkoušené a bez chyb.

Činnost generátoru náhodných čísel si objasníme na následujícím krátkém programu:

```
10 REM *** GENERATOR ***
20 CLS
30 INPUT "POCET NAHODNYCH CISEL ";N
40 INPUT "MAXIMALNI HODNOTA ";M
50 FOR I=1 TO N
60 PRINT RND(M);
70 NEXT I
```

Parametr M v příkazu RND může nabývat všech kladných hodnot. Generovaná náhodná čísla jsou v rozsahu 0 až M.

Náhodná čísla budeme používat při popisu některých dalších příkazů jazyku BASIC-I. Prvním z nich je podmíněný příkaz IF - THEN - ELSE. Jeho úplný tvar je IF podmínka, THEN příkaz 1, ELSE příkaz 2. První příkaz za THEN se provede při splnění podmínky. Druhý příkaz za ELSE se provede při nesplnění podmínky. Ve zkrácené formě má podmíněný příkaz pouze tvar IF podmínka THEN příkaz. Podmínka se skládá ze dvou částí, mezi nimiž je jeden z relačních operátorů =,<>,>,>=,<,<= . Každá z obou částí může být tvořena číselnou konstantou, číselnou proměnnou nebo aritmetickým výrazem. Později si ukážeme, jak používat v podmínce i řetězcových konstant a proměnných a jak se tvoří podmínky složené. Následující program funguje jako hrací kostka:

```
10 CLS
20 A=RND(5)+1
30 IF A=1 THEN PRINT "JEDNA"
40 IF A=2 THEN PRINT "DVE"
50 IF A=3 THEN PRINT "TRI"
60 IF A=4 THEN PRINT "CTYRI"
70 IF A=5 THEN PRINT "PET"
80 IF A=6 THEN PRINT "SEST"
```

V úplném tvaru podmíněného příkazu IF - THEN - ELSE může být jako první nebo i druhý příkaz také další podmíněný příkaz. Následující krátký program nám určuje, zda je žádané číslo kladné nebo záporné a sudé nebo liché:

```
10 CLS
20 INPUT A
30 IF A>0 THEN IF (A/2)*2=A THEN PRINT "+SUDE"
   ELSE PRINT "+LICHE" ELSE IF (A/2)*2=A
   THEN PRINT "-SUDE" ELSE PRINT "-LICHE"
```

2.12 Příkaz skoku GOTO

Ve všech dosud uvedených programech se programy prováděly po jednotlivých řádkách od řádky první. V některých programech může nastat potřeba při splnění určité podmínky některé řádky vynechat. K tomu slouží příkaz skoku GOTO, jehož jediným parametrem je číselná konstanta nebo návěští označující číslo řádku, kde program pokračuje po provedení příkazu skoku GOTO.

Uživatelé, kteří alespoň trochu ovládají anglický jazyk, si jistě již všimli, že zatím všechny probrané příkazy jsou vlastně anglická slova, jejichž význam přímo určuje činnost počítače při prováděném příkazu. Stejně je tomu u příkazu skoku GOTO, který vznikl sloučením dvou krátkých anglických slov.

Následující program, v němž je použito příkazu skoku, nám vypisuje prvočísla. Prvočísla jsou čísla, která nejsou dělitelná beze zbytku žádným číslem kromě jedničky. Při určování prvočísel je použita stará klasická metoda, na jejíž princip určitě brzy přijdete.

```
10 REM *** PRVOCISLA ***
20 CLS
30 INPUT "MAXIMUM PRVOCISEL (<1268) ";N
40 DIM A(N)
50 M=N/2
60 FOR I=3 TO N STEP 2
70 IF A(I)=1 THEN GOTO 130
80 PRINT I;
90 IF I>M THEN GOTO 130
100 FOR J=I TO N STEP I
110 A(J)=1
120 NEXT J
130 NEXT I
```

V programu PRVOČÍSLA se přeskakují programové řádky směrem dopředu na řádky s vyššími čísly. Řádky se mohou přeskakovat i směrem dozadu, jak je vidět na dalším programu. Je to program na třídění čísel podle velikosti nejjednodušší a také nejpomalejší metodou, kdy se porovnávají dvě následující čísla a v případě, že číslo s menším indexem je větší, dojde k jejich výměně.

Aby jste nemuseli zadávat čísla z klávesnice, je použito na řádcích 15 až 70 generování pole k třídění pomocí generátoru náhodných čísel. Tuto část programu je možno změnit pro zadávání čísel příkazem INPUT. Počet čísel ke třídení zadávejte zpočátku do stovky. Při větším počtu trvá třídění příliš dlouho.

```
10 REM *** TRIDENI ***
15 REM GENEROVANI POLE A(N)
20 INPUT "POCET CISEL K TRIDENI ";N
30 DIM A(N)
40 INPUT "MAXIMALNI HODNOTA ";M
50 FOR I=1 TO N
60 A(I)=RND(M)
70 NEXT I
75 REM TRIDENI POLE A(N)
80 F=0
90 FOR I=1 TO N-1
100 IF A(I+1)>A(I) THEN GOTO 150
110 A(I+1)=W
120 A(I+1)=A(I)
```

```
130 A(I)=W
140 F=1
150 NEXT I
160 IF F=1 THEN GOTO 80
170 REM TISK SETRIDENEHO POLE A(N)
180 FOR I=1 TO N
190 PRINT I,A(I)
200 NEXT I
```

2.13 Podprogramy, příkazy GOSUB a RETURN

Při tvorbě obsáhlejších programů často dochází k tomu, že určitá část programu se opakuje několikrát, což vede k zbytečnému prodlužování zdrojového textu programu. Takovou část programu je možno mít v programu pouze jednou ve formě tzv. podprogramu a v případě potřeby si podprogram vyvolat příkazem GOSUB. Jediným parametrem v příkazu GOSUB je číselná konstanta nebo návěští označující řádku, kde začíná podprogram. K ukončení podprogramu slouží příkaz RETURN. Jestliže se při provádění programového řádku narazí na příkaz GOSUB, provede se skok na začátek podprogramu a program pokračuje až k příkazu RETURN. Provedení příkazu RETURN znamená návrat zpět na následující příkaz za příkazem GOSUB. Následující program počítá celkový počet kombinací výběru několika prvků z více prvků. Každý si jistě vzpomene, že při výpočtu je třeba třikrát počítat faktoriál. A právě výpočet faktoriálu je zde jako podprogram.

```
10 REM *** KOMBINACE ***
20 CLS
30 INPUT "VYBIRANE PRVKY ";S
40 INPUT "Z PRVKU VYBERU ";C
50 IF S>=C OR S=1 THEN GOTO 30
60 N=C
70 GOSUB 200
80 C1=F
90 N=S
100 GOSUB 200
110 S1=F
120 N=C-S
130 GOSUB 200
140 PRINT C1/S1*F;"MOZNYCH KOMBINACI"
150 STOP
200 REM VYPOCET FAKTORIALU
210 F=1
220 FOR I=1 TO N
230 F=F*I
240 NEXT I
250 RETURN
```

K programu je třeba ještě uvést několik poznámek. Vstupní hodnoty volte menší než 8. Jestliže se vám objeví chybové hlášení Err 6 nebo je výsledek záporný, došlo při výpočtu faktoriálu k překročení číselného rozsahu. Na řádce 50 je složená podmínka v podmíněném příkazu, která se skládá ze dvou jednoduchých podmínek spojených logickým operátorem OR. Na řádce 150 je pro nás nový příkaz STOP, který způsobí zastavení programu a výpis čísla řádky na obrazovku. Příkaz STOP je zde použit proto, aby po tisku výsledku program nepokračoval na řádce 200 výpočtem faktoriálu.

2.14 Práce s pamětí pomocí příkazů PEEK a POKE

Prakticky všechny varianty jazyků BASIC umožňují práci s jednotlivými pamětovými buňkami. Příkazem PEEK se určí obsah dané pamětové buňky. Následující program vypíše na obrazovku obsah prvních dvaceti bytů paměti ROM:

```
10 CLS
20 FOR A=0 TO 19
30 PRINT A,PEEK(A)
40 NEXT A
```

Příkaz POKE slouží pro zápis čísla v rozsahu 0 až 255 do pamětové buňky s danou adresou. Příkaz se potom používá ve tvaru POKE A,C. První parametr je adresa a druhý zapsané číslo. Na příkaz POKE nebude uveden žádný ukázkový program. Příkaz se využívá hlavně pro zadávání programu ve strojovém kodu a pro změny systémových proměnných, které jsou popsány v příručce Monitor Handling Manual. Oba příkazy byly uvedeny pouze pro informaci a také proto, že ukončují soubor základních příkazů jazyku BASIC - 1.

2.15 Vyjádření znaku ASCII kódy, příkazy CHR$ a ASCII

V několika dalších částech manuálu se seznámíte se skupinou příkazů pro zpracování řetězcových konstant a proměnných. Již víme, že řetězcové konstanty nebo hodnoty řetězcových proměnných se skládají z jednotlivých znaků a že v jedné pamětové bunce /v jednom bytu/ může být uloženo libovolné číslo v rozsahu 0 až 255. Místo čísel však můžeme používat soubor 256 znaků, jestliže každému znaku bude přiřazen číselný kód. Ve světě se při zpracování textových informací používá mezinárodní kód ASCII /American Standard Code for Information Interchange/, který využívá kódy 0 až 127. Kódy 0 až 31 jsou určeny pro řídící účely při přenosu informací. Kódy 32 až 127 přísluší speciálním znakům, číslicím a velkým a malým písmenům. Počítač M5 využívá dalších 128 kódů pro grafické symboly a některé japonské znaky. Kompletní výpis všech znaků, s nimiž je možno pracovat, je uveden v příloze G původního anglického manuálu na straně 95. V levém sloupci je kód znaku v desítkové soustavě, uprostřed je v závorce kód znaku v šestnáctkové soustavě a ve třetím sloupci je přímo symbol znaku.

Pro vzájemný převod mezi kódy a jejich znakovým vyjádřením slouží příkazy CHR$ a ASCII. Příkaz CHR$, v němž jako parametr použijeme kód znaku, nám přiřadí řetězcové proměnné jeden znak nebo nám znak přímo vytiskne. Pomocí dalšího programu si můžeme vypsat všechny znaky:

```
10 CLS
20 FOR K=32 TO 255
30 PRINT CHR$(K);
40 NEXT K
```

Opačnou funkci má příkaz ASCII, který nám určí kód daného znaku, jak si můžeme vyzkoušet na dalším programu. Místo jednoho znaku můžeme zadat celé slovo. V tom případě se nám vytiskne kód prvního znaku.

```
10 CLS
20 INPUT "ZNAK ";A$
30 PRINT ASCII(A$)
40 GOTO 20
```

Po vyzkoušení programu vypněte počítač, za několik vteřin ho opět zapněte a věnujte se další části.

2.16 Přerušení programu, příkaz CONT

Na řádce 40 předchozího programu je příkaz skoku zpět na řádek 20. Vznikne tzv. nekonečná smyčka. Přerušení programu je možné dvěma způsoby. Při prvním musíme zadat řetězec o větším počtu znaku než osmnáct. Program se přeruší s chybovým hlášením ERR 15. Druhý způsob přerušení je možno použít u kteréhokoliv spuštěného programu. Při současném stisknutí tlačítek CTRL a RESET se program přeruší za posledním vykonávaným příkazem v okamžiku stisknutí obou tlačítek a na obrazovce se vypíše hlášení a zastavení programu, které je stejně jako u příkazu STOP.

Důležitou vlastností všech variant jazyků BASIC je to, že po zastavení výpočtu příkazem STOP nebo z klávesnice může program pokračovat dále po příkazu CONT zadaném z klávesnice. Před příkazem CONT můžeme pracovat s počítačem v režimu okamžitého výpočtu. Těchto možností se využívá hlavně při ladění programu, kdy si chceme během výpočtu zjistit nebo změnit hodnoty některých proměnných.

2.17 Příkazy pro práci s řetězci LEN, LEFT$, MID$, RIGHT$ a RPT$

Základním příkazem pro práci s řetězcovými proměnnými je příkaz LEN pro určení počtu znaků. K vyzkoušení použijeme opět nekonečnou smyčku, kterou musíme přerušit z klávesnice pro její ukončení:

```
10 CLS
20 INPUT A$
30 PRINT LEN(A$)
40 GOTO 20
```

Pro výběr části řetězcové proměnné jsou určeny příkazy LEFT$, MID$ a RIGHT$. Příkaz LEFT$ vybírá levou část řetězce, příkaz MID$ libovolnou část řetězce a příkaz RIGHT$ potom pravou část řetězce. Ve všech třech případech je jako první parametr řetězcová konstanta nebo název řetězcové proměnné. Příkazy LEFT$ a RIGHT$ mají jako druhý parametr počet znaků zleva nebo zprava. Příkaz MID$ má jako druhý parametr pořadové číslo vybíraného znaku. Třetím parametrem příkazu MID$ je počet vybíraných znaků. Funkce všech tří příkazů je demonstrována následujícím programem:

```
10 CLS
20 A$="POCITAC SORD M5"
30 FOR I=1 TO 15
40 PRINT LEFT$(A$,I)
50 NEXT I
```

```
60 INPUT "LIBOVOLNY ZNAK ";B$
70 FOR I=1 TO 7
80 PRINT TAB(15-I);MID$(A$,8-I,I*2+1)
90 NEXT I
100 INPUT "LIBOVOLNY ZNAK ";B$
110 FOR I=1 TO 15
120 PRINT RIGHT$(A$,I)
130 NEXT I
```

Řetězce lze i slučovat pomocí znaku + . Další program provádí opakovaný tisk s rotací zadaného textu doleva o jeden znak. Na řádkách 60 a 70 je zpomalovací cyklus. Volbou ukončovací hodnoty řídící proměnné cyklu se mění i rychlost posunu.

```
10 INPUT "TEXT ";A$
20 L=LEN(A$)
30 CLS
40 PRINT CURSOR(0,0);A$
50 A$=RIGHT$(A$,1)+LEFT$(A$,L-1)
60 FOR I=1 TO 200
70 NEXT I
80 GOTO 40
```

Zajímavý příkaz je RPT$, který zopakuje zadaný řetězec:

```
10 CLS
20 INPUT "TEXT ";A$
30 INPUT "POCET OPAKOVANI ";N
40 PRINT RPT$(N,A$)
```

2.18 Čtení klávesnice příkazem INKEY$

V druhém ukázkovém programu předchozí části je dvakrát použit příkaz INPUT, který slouží pouze pro přerušení programu. Po vstupu libovolného čísla program pokračuje dál. Místo příkazu INPUT by šlo použít příkaz STOP a pokračovat dál po příkazu CONT z klávesnice. Pro podobné případy, kdy je zapotřebí během programu zadat jeden znak z klávesnice bez stisknutí tlačítka RETURN, je přímo určen příkaz INKEY$. Jeho funkci objasní následující program:

```
10 CLS
20 PRINT "TISKNETE ZNAKOVA TLACITKA"
30 PRINT
40 A$=INKEY$
50 IF ASCII(A$)=0 THEN GOTO 40
60 PRINT A$;
70 GOTO 40
```

Příkaz INKEY$ musí být vždy na pravé straně přiřaďovacího příkazu LET. Řetězcová jednoznaková proměnná na levé straně mění svoji hodnotu podle znaku tlačítka, které bylo v průběhu provádění příkazu stisknuto. Jestliže žádné znakové tlačítko nebylo stisknuto, bude mít kód znaku proměnné A$ hodnotu 0. Kód 0 má tzv. prázdný znak, který se vyjadřuje dvěma úvozovkami těsně u sebe. Proto můžeme psát řádek 50 jednodušeji:

```
50 IF A$="" THEN GOTO 40
```

Jestliže chceme v průběhu programu jeho zastavení a další pokračování až po

stisknutí libovolné znakové klávesy, stačí zařadit jeden řádek programu navíc:

```
100 IF INKEY$="" THEN GOTO 100
```

Pomocí příkazu INKEY$ můžeme v programech využívat stejná tlačítka, na které jsme zvyklí v režimu okamžitého výpočtu při současném stisknutí tlačítka CTRL. Další program simuluje pohyb zadaného znaku po obrazovce:

```
10 REM *** POHYB ***
20 CLS
30 INPUT "ZNAK ";Z$
40 S=0
50 R=0
60 CLS
70 PRINT CURSOR(S,R);Z$;
80 S0=S
90 R0=R
100 A$=INKEY$
110 IF A$="" THEN GOTO 100
120 IF A$="@" THEN R=R-1
130 IF A$="/" THEN R=R+1
140 IF A$=":" THEN S=S+1
150 IF A$=";" THEN S=S-1
160 IF R=23 AND S=31 THEN GOTO 100
170 IF R<0 OR R>23 THEN R=R0
180 IF S<0 OR S>31 THEN S=S0
190 PRINT CURSOR(S0,R0);" ";
200 GOTO 70
```

Program nepotřebuje téměř komentáře. Na řádkách 160 až 180 se testuje, zda se znak nepohybuje mimo obrazovku nebo do pravého dolního rohu, kdy by došlo k posunu celé obrazovky o jeden řádek. Na řádku 190 dochází k mazání předchozí pozice. Jestliže tuto řádku vynecháme, můžeme potom kreslit na obrazovce jednoduché obrazce.

2.19 Příkazy VAL a NUM$ pro převody mezi řetězci a čísly

Jazyk BASIC-I pracuje s číselnými a řetězcovými konstantami a proměnnými, které nelze používat současně např. v aritmetických výrazech. U některých programů může nastat potřeba provádět výpočty s čísly, která jsou v paměti uložena jako řetězcové proměnné. V profesionální praxi k tomu dochází velice často např. při čtení číselných údajů z děrné pásky. Proto prakticky všechny varianty jazyku BASIC obsahují příkazy pro vzájemné převody mezi řetězcovými a číselnými proměnnými.

Příkaz VAL převádí řetězcové konstanty nebo proměnné na číselné proměnné:

```
10 CLS
20 INPUT "CISLO ";A$
30 A=VAL(A$)
40 PRINT A
```

Příkaz NUM$ zase převádí číselné konstanty nebo proměnné na řetězcové proměnné:

```
10 CLS
20 INPUT "CISLO ";A
30 A$=NUM$(A)
40 PRINT A$
```

Příkazy VAL a NUM$ společně s dalšími příkazy pro práci s řetězci umožňují naprogramovat např. výpočet základních aritmetických operací s čísly zadanými ve formě řetězců. Jako příklad je dále uveden obsáhlejší program na provedení součtu dvou čísel, za nimž následuje výpis jednotlivých proměnných s popisem jejich významu a stručný popis činnosti programu.

```
10 REM *** SOUCET ***
20 CLS
30 INPUT "1. CISLO ";A$
40 INPUT "2. CISLO ";B$
50 A=LEN(A$)
60 B=LEN(A$)
70 IF A=B THEN GOTO 90
80 IF A>B THEN B$=RPT$(A-B,"0")+B$ ELSE
81 A$=RPT$(B-A,"0")+A$
90 L=LEN(A$)
100 FOR I=L TO 1 STEP -1
110 C=VAL(MID$(A$,I,1))+VAL(MID$(B$,I,1))+Z
120 C$=NUM$(C)
130 C=VAL(RIGHT$(C$,1))
140 IF LEN(C$)=1 THEN Z=0 ELSE Z=VAL(LEFT$(C$,1))
150 D$=D$+RIGHT$(C$,1)
160 NEXT I
170 IF Z>0 THEN D$=D$+LEFT$(C$,1)
180 FOR I=LEN(D$) TO 1 STEP -1
190 E$=E$+MID$(D$,I,1)
200 NEXT I
210 PRINT
220 PRINT TAB(11);A$
230 PRINT TAB(11);B$
240 PRINT
250 PRINT TAB(11-Z);E$
```

Proměnné:

- A$ - první zadané číslo ve formě řetězce
- B$ - druhé zadané číslo ve formě řetězce
- A - počet číslic prvního čísla
- B - počet číslic druhého čísla
- L - počet znaků obou čísel po případném doplnění jednoho zleva nulami
- C - součet dvou odpovídajících si číslic zadaných čísel
- Z - zbytek při překročení řádu při součtu
- D$ - součet zadaných čísel ve formě řetězce s opačným pořadím znaků
- E$ - řetězcová proměnná D$ s opačným pořadím znaků a tím i součet obou zadaných čísel

Činnost programu:

- 10 - 40 vstup čísel
- 50 - 80 doplnění čísla s menším počtem znaků nulami
- 90 - 170 provedení součtu zadaných čísel do řetězce D$
- 180 - 200 obracení pořadí znaku řetězce D$ do E$
- 210 - 250 zobrazení zadaných čísel a jejich součtu

V programu součet jsou použity skoro všechny příkazy pro práci s řetězci. Všimněte si, jak je možno s využitím závorek nahrazovat název řetězcové proměnné v příkazu VAL pouze její části. Na konci programu je dvakrát použit příkaz PRINT bez tiskového seznamu, který způsobí tisk prázdného řádku.

2.20 Programové řádky s více příkazy

V následujících čtyřech částech budou vysvětleny vlastnosti a příkazy programovacího jazyku BASIC-I, které výrazně urychlují a zjednodušují tvorbu programů.

Všechny krátké demonstrační programy doposud uvedené v manuálu byly psány tak, že na každém řádku byl pouze jediný příkaz. Krátké programy jsou potom velice dobře přehledné, ale programy delší se stávají svojí délkou značně nepřehledné. Stejné jako většina variant jazyků BASIC pro nejrůznější počítače, může i počítač M5 mít na jedné programové řádce několik příkazů oddělených dvoutečkou. Jako názorný příklad si můžeme znovu napsat program KOMBINACE z části 2.13:

```
10 REM *** KOMBINACE ***
20 CLS:INPUT "VYBIRANE PRVKY ";S:INPUT "Z PRVKU
21 VYBERU ";C:IF S>=C OR S=1 THEN GOTO 20
30 N=C:GOSUB 100
40 C1=F:N=S:GOSUB 100
50 S1=F:N=C-S:GOSUB 100
60 PRINT C1/S1*F;"MOZNYCH KOMBINACI":STOP
100 REM VYPOCET FAKTORIALU
110 F=1:FOR I=1 TO N:F=F*I:NEXT I:RETURN
```

2.21 Tvorba programů s příkazy AUTO a DEL

Při psaní programů většina programátorů čísluje jednotlivé řádky programu s určitým pevným krokem. Dělají to proto, aby si při odladování programu mohli do programu doplňovat další programové řádky. U delších programů jsou čísla řádků tak velká, že jejich psaní a přičítání kroku zbytečně zatěžuje programátora. Proto mají některé osobní počítače příkaz pro automatické řádkování programu. U počítače M5 je to příkaz AUTO se dvěma parametry oddělenými čárkou. První parametr udává číslo řádku odkud automatické řádkování začíná a druhý parametr potom značí krok mezi jednotlivými řádky. Automatické řádkování umožňuje i opravu chyb zjištěných během psaní programu v předchozích řádkách. Ruší se tlačítkem RETURN ihned po zobrazení dalšího čísla řádku.

Při tvorbě programu potřebujeme někdy vymazat větší počet řádek programu. Nejčastěji je to tehdy, jestliže děláme program úpravami programu starého nebo potřebujeme ze starého programu nějaký podprogram použít do programu nového. Již víme, že jedna programová řádka se vymaže tak, že se napíše její číslo a stiskne se tlačítko RETURN. Pro vymazání souvislé části programu slouží příkaz DEL se dvěma parametry oddělenými čárkami. První parametr je číslo prvního vymazaného řádku a druhý parametr je číslo posledního vymazaného řádku.

Současně se vymažou všechny programové řádky mezi těmito dvěma.

2.22 Použití návěští v programech

Při vysvětlování funkce příkazů GOTO a GOSUB jsme si uvedli, že jediným parametrem je číselná konstanta nebo **návěští** označující řádek, na kterém pokračuje program nebo začíná podprogram. Při častějším používání podprogramů, což je správný způsob programování, začne být za chvíli program pro uživatele nepřehledný. Jednotlivé podprogramy jsou totiž označeny čísly, takže si programátor musí vedle činnosti každého podprogramu pamatovat při psaní programu i jeho číslo řádku. Jestliže se v různých programech začnou používat stejné podprogramy ale s různými počátečními čísly řádků, nepřehlednost bude ještě větší. Bylo by určitě pro uživatele výhodnější, aby si podprogram označil nějakým krátkým názvem vystihujícím jeho funkci a tento název používal místo čísla počátečního řádku podprogramu. Počítač M5 jako jeden z mála mezi osobními i profesionálními počítači tuto možnost má.

V příkazech GOTO a GOSUB se mohou používat i tzv. návěští ve formě řetězcové konstanty bez úvozovek. Návěští začíná vždy znakem $, za kterým následuje maximálně 32 znaků. Řádek, na kterém program pokračuje nebo na kterém začíná podprogram, se označí stejným návěštím. Za číslo řádku se bez mezery napíše dané návěští. Pro další zvýšení přehlednosti programu je vhodné na řádek označený návěštím napsat další výkonné příkazy. Jako názorný příklad použijeme opět program KOMBINACE.

```
10 REM *** KOMBINACE ***
20 CLS:INPUT "VYBIRANE PRVKY ";S:INPUT "Z PRVKU
21 VYBERU ";C:IF S>=C OR S=1 THEN GOTO 20
30 N=C:GOSUB $FAKTORIAL
40 C1=F:N=S:GOSUB $FAKTORIAL
50 S1=F:N=C-S:GOSUB $FAKTORIAL
60 PRINT C1/S1*F;"MOZNYCH KOMBINACI":STOP
100 $FAKTORIAL
110 F=1:FOR I=1 TO N:F=F*I:NEXT I:RETURN
```

2.23 Chybová hlášení, příkazy ERR, ERRL a ERRL$

Tvorbu a odladování programu výrazně ulehčují chybová hlášení, se kterými jste se již určitě setkali. O některých chybách již byla zmínka v předchozím textu. Chybové hlášení vypíše při chybě a po případném zastavení programu vždy číslo chyby a číslo řádku, na kterém k chybě došlo. U chyb při příkazech z klávesnice je číslo řádku nulové. Číslo chyby je potom uloženo v proměnné ERR a číslo řádku s chybou v proměnné ERRL. V proměnné ERRL$ je uloženo návěští řádku s chybou. Tyto proměnné se více využijí v programovacím jazyku BASIC-G nebo BASIC-F, kde je možnost přerušení programu při chybě a další pokračování programu podle čísla chyby nebo čísla řádku s chybou.

Všechny chyby, které počítač M5 hlásí, jsou dále přehledně uvedeny i s krátkým komentářem. Časem se význam všech čísel chyb naučíte asi zpaměti.

Err 1 - chyba v příkazech cyklu FOR - NEXT; chybí příkaz FOR nebo NEXT

Err 2 - nesrozumitelný příkaz pro počítač; zadaný příkaz nemá počítač uložen ve své paměti a proto jej nemůže vykonat

Err 3 - chyba volání podprogramu; v podprogramu byl použit příkaz CLEAR nebo byl použit příkaz RETURN aniž byl podprogram volán příkazem GOSUB

Err 4 - chyba v programovém čtení dat; neodpovídají si položky v příkazech READ a DATA

Err 5 - nesprávný druh proměnné; v příkazu je použita číselná proměnná místo řetězcové nebo naopak

Err 6 - překročení číselného rozsahu; výsledek aritmetické operace je menší nebo větší než číselný rozsah počítače /-32768 až + 32767/

Err 7 - konec paměti pro data; v programu je použito příliš mnoho dat a polí

Err 8 - chybějící číslo řádku; v příkazech GOTO nebo GOSUB je použito číslo řádku, které v programu není

Err 9 - chyba indexu; hodnota indexu proměnné pole je nula nebo je záporná

Err 10 - dvojnásobné dimenzování; jedno pole bylo dvakrát dimenzováno příkazem DIM

Err 11 - dělení nulou; v aritmetickém výrazu došlo k dělení nulou

Err 12 - špatné použití příkazu z klávesnice; použití příkazu CONT v nesprávném okamžiku

Err 13 - nesprávný druh konstanty; číselná konstanta byla použita místo řetězcové nebo naopak

Err 14 - konec paměti pro program; program je příliš dlouhý a nevejde se do paměti

Err 15 - řetězec znaků příliš dlouhý; maximální délka hodnoty řetězcové proměnné je 18 znaků

Err 16 - chyba dimenzování pole; použitá indexovaná proměnná bez úvodního dimenzování příkazem DIM

Err 17 - stejné návěští použito dvakrát jako označení řádku

Err 18 - chyba při čtení programu nebo dat z magnetofonu

Err 19 - nesprávně použitý obrazovkový režim

3. BAREVNÁ GRAFIKA A ZVUK V JAZYKU BASIC - I

3.01 Generátor znaků, příkaz VPEEK

Při zobrazování znaků na obrazovce TV přijímače každý osobní počítač používá tzv. generátor znaků. Generátor znaků je část paměti ROM, v níž jsou uloženy tvary všech znaků. U počítače M5 se po zapnutí vytvoří generátor znaků v paměti VRAM, což umožnuje programové změny tvaru znaků. Každý znak je vytvořen z jednotlivých bodů v rastru 8 x 8 bodu. Následující program názorně ukazuje bodovou strukturu zadaných znaků z klávesnice:

```
10 REM *** ZNAKY ***
20 DIM A(8)
30 DATA 128,64,32,16,8,4,2,1
40 FOR I=1 TO 8
50 READ A(I)
60 NEXT I
70 CLS
80 A$=CHR$(225)
90 INPUT "ZNAK ";Z$
100 PRINT
110 A0=10240+8*ASCII(Z$)
120 FOR A=A0 TO A0+7
130 N=VPEEK(A)
140 FOR I=1 TO 8
150 B=N/A(I)
160 IF B=1 THEN PRINT A$; ELSE PRINT ":";
170 N=N MOD A(I)
180 NEXT I
190 PRINT
200 NEXT A
210 PRINT
220 PRINT
230 GOTO 90
```

K přerušení programu je potřeba použít tlačítek CTRL+ RESET buď při postupném zobrazování znaků nebo před vstupem znaku příkazem INPUT, kdy dojde k zastavení na další řádce. Na řádce 130 je nový příkaz VPEEK pro určení obsahu jednoho bytu paměti VRAM. Adresa v příkazu VPEEK může nabývat hodnot 0 až 16383. Na řádkách 140 až 180 probíhá rozklad a zobrazení jednoho bytu do binárního tvaru na tzv. bity. Generátor znaků začíná na adrese 10240 paměti VRAM a obsahuje 256 znaků. Každý znak zabírá 8 bytů. Celý generátor znaků zabírá paměť o velikosti 2 KB. Prvních 32 znaků se zobrazuje jako inversní velká písmena a čtyři šipky. Pro jejich zobrazení pomocí předchozího programu je potřeba změnit řádky 90 a 110 a místo znaku Z$ zadávat přímo kody 0 až 31. O významu a použití těchto řídících kodů bude pojednáno v dalším díle manuálu.

3.02 Vyjádření čísel v šestnáctkové soustavě, příkaz HEX$

Vedle běžně používaných čísel v desítkové soustavě může počítač M5 pracovat s čísly vyjádřenými v tzv. šestnáctkové nebo hexadecimální soustavě. Hexadecimální čísla se běžně používají při programování mikropočítačů.

U čísel v desítkové soustavě se používá deseti číslic 0 až 9. Každé číslo lze potom vyjádřit jako součet součinů jednotlivých číslic s příslušnou mocninou deseti. Číslo 5421 se tedy může napsat jako 5+10↑3+4+10↑2+2+10↑1+1+10↑0. Stejné pravidlo platí i pro vyjádření čísel v šestnáctkové soustavě. Vzhledem k tomu, že je potřeba šestnácti číslic, používají se vedle číslic 0 až 9 také písmena A až F. Desítkově vyjádřené číslo v rozsahu 0 až 255, které může být uloženo v jedné paměťové bunce v jednom bytu , se potom napíše v šestnáctkové soustavě pomocí dvou šestnáctkových čísel. Pro převod číselných proměnných v desítkové soustavě do soustavy šestnáctkové se používá příkaz HEX$, jak se můžete přesvědčit následujícím programem:

```
10 CLS
20 INPUT "CISLO ";N
30 IF N<0 THEN STOP
40 A$=HEX$(N)
50 PRINT A$
60 GOTO 20
```

Pro zastavení programu slouží řádek 30.

Znalost hexadecimálních čísel nám bude prospěšná v dalších částech při definování nových znaků a grafických obrazců.

Zajímavou a užitečnou vlastností jazyku BASIC - I je možnost vyjadřovat číselně konstanty dekadicky i šestnáctkově. Šestnáctkové konstanty ale musí mít předřazen znak &.

3.03 Definování nových znaků, příkazy STCHR a MOD

Nejdůležitějším příkazem pro práci s barevnou grafikou je příkaz STCHR, který má dvě základní funkce. Základní funkcí je definování nových znaků v generátoru znaku. Vzhledem k tomu, že při práci s texty vystačíme s běžnými ASCII znaky, můžeme si dále programově udělat 128 nových znaků.

Příkaz STCHR má obecný tvar: STCHR řetězec TO K,P. Řetězec je buď řetězcová konstanta o délce 16 znaků, název řetězcové proměnné nebo pravá část přiřazovacího příkazu, jehož výsledkem je hodnota řetězcové proměnné. Délka okamžité hodnoty řetězcové proměnné při provádění příkazu STCHR musí být 16 znaků. Řetězec obsahuje informace o tvaru případně barvě definovaného znaku nebo grafického obrazce. Při definování znaků představují každé dva znaky řetězce hexadecimální číslo vyjadřující ve své binární podobě rozložení bodů v jedné vodorovné řádce znakového rastru 8 x 8 . Levá část obr. 4 ukazuje způsob vytváření řetězce podle tvaru uživatelova znaku.

Dalšími parametry příkazu STCHR je ASCII kód K nového znaku a konstanta P určující funkci příkazu STCHR. Při definování znaků uživatelem má parametr P hodnotu 1.

Nahradíme-li v programu POHYB z části 2.18 na řádku 30 příkaz INPUT příkazem STCHR "183C7EDBFF3C1824 " TO 128, 1 a na řádku 70 řetězcovou proměnnou Z$ výrazem CHR$(128), budeme moci pohybovat po obrazovce novým znakem z obr. 4, jímž je oblíbené demonstrační UFO.

Uživateli se možností vytváření svých vlastních znaků otvírá široké pole působnosti. Jako názorný příklad si ukážeme program na grafické porovnání dvaceti kladných čísel o maximální velikosti 200. Program je uveden záměrně bez komentáře nebo poznámek proto, aby si uživatel celý program sám prošel a pochopil.

```
10 REM *** GR.SROVN. ***
20 DATA 80,C0,E0,F0,F8,FC,FE,FF
30 FOR I=1 TO 8:READ P$:STCHR RPT$(7,P$)+"00"
   TO 127+I,1:NEXT I
40 CLS:FOR R=1 TO 20:D=RND(200)
50 DH=D/8:DL=D MOD 8:PRINT D;TAB(6);
60 IF DH=0 THEN GOTO 80
70 FOR I=1 TO DH:PRINT CHR$(135);:NEXT I
80 IF DL>0 THEN PRINT CHR$(127+DL) ELSE PRINT
90 NEXT R
```

Na řádce 50 je uveden v druhém příkazu nový aritmetický operátor MOD. Výsledek tohoto příkazu je zbytek po dělení hodnoty proměnné D konstantou 8.

Program lze upravit na libovolná čísla, která se graficky srovnávají s půlprocentní přesností. Vždy je ale zapotřebí nejdříve určit číslo maximální a provést přepočet na čísla od 0 do 200.

3.04 Nastavení barev, příkaz VPOKE

Počítač M5 může pracovat s barevným výstupem v šestnácti barvách, jestliže jako barvy uvažujeme černou barvu a žádnou barvu. Názvy barev a přiřazení čísel barev je na dalších řádkách:

0 - žádná barva	8 - červená
1 - černá	9 - světle červená
2 - zelená	10 - žlutá
3 - světle zelená	11 - světle žlutá
4 - tmavě modrá	12 - tmavě zelená
5 - světle modrá	13 - purpurová
6 - tmavě červená	14 - šedá
7 - blankytně modrá	15 - bílá

Po zapnutí počítače mají znaky šedou barvu /14/ a pozadí je černé /1/. Stejně tak je černý rámeček obrazovky mezi aktivní plochou obrazovky a jejím okrajem. Jazyk BASIC-1 nemá žádný příkaz pro přímé obarvení znaku nebo obrazovky jako jazyk BASIC-G nebo BASIC-F. Na obarvení bylo nutno použít některé informace obsažené v příručce Monitor Handling Manual.

V paměti VRAM je od adresy 15232 oblast o velikosti 32 bytů, které obsahují informace o barvě znaků a jejich pozadí. Vzhledem k tomu, že celkový počet znaků je 256, musí být vždy osm za sebou následujících znaků v generátoru znaku obarveno stejně. Následující program demonstruje obarvení skupiny osmi znaků. Program končí po zadání nulové barvy znaku tak, že se nastaví inverzní výstup s černými znaky /1/ na šedém pozadí /14/. Na řádku 20 první příkaz volá podprogram v paměti 8 KB ROM, který je potřebný pro nastavení barev.

```
10 REM *** BARVA 1 ***
20 CALL &0DD8:CLS
30 FOR I=32 TO 255:PRINT CHR$(I);:NEXT I
40 INPUT "BARVA ZNAKU (1-15) ";BZ
50 IF BZ=0 THEN GOTO 100
60 INPUT "BARVA POZADI (0-15) ";BP
70 INPUT "SKUPINA ZNAKU (4-31) ";S
80 VPOKE 15232+S,(16*BZ+BP)
90 GOTO 40
100 FOR A=15232 TO 15263:VPOKE A,225:NEXT A
```

Na řádce 80 je nový příkaz VPOKE pro přímý zápis dat do paměti VRAM. Prvním parametrem je adresa paměti VRAM v rozsahu 0 - 16383 a druhým potom zapisována hodnota v rozsahu 0 - 255.

Nyní již můžete obarvit výstup programu GR.SROVN. z předchozí části alespoň dvěma barvami. Jednou barvou tisk číselných hodnot a barvou druhou grafický pruh.

Jestliže si změníme barvy všech znaků pomocí tabulky barev od adresy 15232 do adresy 15263 v paměti VRAM, můžeme si nastavit i barvu rámečku obrazovky. Další program nastavuje najednou barvu všech znaků, barvu pozadí a barvu rámečku obrazovky.

```
10 REM *** BARVA 2 ***
20 INPUT "BARVA ZNAKU ";BZ
30 INPUT "BARVA POZADI ";BP
40 INPUT "BARVA RAMECKU ";BR
50 CALL &0DD8:B=16*BZ+BP
60 FOR A=15232 TO 15263:VPOKE A,B:NEXT A
70 POKE &709F,BR:CALL &0CA3
```

3.05 Sprajty - grafické bloky definované uživatelem

V předchozí části jsme se naučili definovat a obarvovat nové znaky, což je základní věc při práci s barevnou grafikou. K nejzajímavějším grafickým možnostem počítače M5 patří grafické bloky definované uživatelem. Anglický název těchto bloků je sprite /výslovnost sprajt/. Přesný krátký a výstižný termín v češtině zatím neexistuje a proto budeme používat v dalším textu výraz sprajt odvozený přímo z výslovnosti anglického originálního termínu. Stejně se postupovalo i u dalších termínů z oblasti výpočetní techniky.

Základní vlastnosti sprajtů jsou uvedeny v následujících bodech:

A/ uživatel může definovat 256 sprajtů a používat současně maximálně 32 sprajtů

B/ každý sprajt může mít jednu ze šestnácti barev

C/ sprajt se může pohybovat po obrazovce v rastru 192 x 256 bodů pouze změnou souřadnic

D/ při zobrazování několika sprajtů dochází k jejich překrývání od sprajtů číslo 0 až ke sprajtu číslo 31

E/ základní velikost sprajtů je totožná se znaky ale může být i dvojnásobná a čtyřnásobná.

Tyto základní vlastnosti sprajtů budou podrobně vysvětleny a demonstrovány v další části.

3.06 Definování sprajtů a jejich pozice, příkazy MAG, SCOD, SCOL a LOC

V části 3.01 jsme si vysvětlili funkci generátoru znaků a u části 3.03 jsme se naučili programovat vlastní znaky změnou znaků generátoru znaků příkazem STCHR. Generátor znaků zabírá oblast paměti VRAM o velikosti 2 KB. Na obrazovce se může zobrazit maximálně 768 znaků. Pro uložení kodů znaků na jednotlivých pozicích obrazovky začíná na adrese 14336 paměti VRAM oblast 768 bytů. O tom se můžeme přesvědčit jednoduchým programem:

```
10 A=14336:FOR I=0 TO 767:VPOKE A+I,I MOD 256:NEXT I
20 IF INKEY$="" THEN GOTO 20
```

Generátor znaků je u paměti VRAM zopakován ještě jednou pro definování tvaru sprajtů. Oba dva generátory znaků jsou na sobě zcela nezávislé. Dále je v paměti VRAM rezervováno 128 bytů pro uložení informací o kodu, barvě a pozici 32 sprajtů.

Následující program demonstruje použití všech příkazů pro práce se sprajty:

```
10 REM *** SPRAJTY 1 ***
20 INPUT "PRVNI KOD (0-224) ";K:INPUT "ZVETSENI (0-3) ";Z
30 MAG Z
40 FOR S=0 TO 31:SCOD S,K+S:NEXT S
50 FOR S=0 TO 31:SCOL S,12:NEXT S
60 S=0:FOR Y=0 TO 7:FOR X=1 TO 4
70 LOC S TO RND(247),Y*32:S=S+1
80 NEXT X:NEXT Y
90 A$=INKEY$:IF A$="" THEN GOTO 90
100 IF A$<>" " THEN GOTO 20
110 CALL &1387
```

Příkaz SCOD přiřazuje sprajtu S kód tvaru sprajtu z generátoru znaků pro sprajty. Příkaz SCOL přiřazuje sprajtu S jednu barvu ze šestnácti. V našem programu je pevně nastavena jedna barva; je ale i možno použít místo konstanty náhodně volené barvy pomocí RND /15/. Jestliže v těchto dvou příkazech jsou stejná čísla sprajtu S, dojde k inicializaci příslušného sprajtu, který se umístí nalevo mimo obrazovku. Příkaz LOC nám způsobí přesun sprajtu na zadanou pozici tak, že na dané souřadnici je levý horní roh sprajtu. Poslední příkaz CALL /&1387/ provádí mazání všech sprajtů. Zvětšení zadané příkazem MAG platí pro všechny sprajty.

Při další samostatné práci se sprajty je potřeba si zapamatovat, že vedle sebe mohou být ve vodorovném směru maximálně čtyři sprajty.

Pro tvorbu vlastních sprajtů se používá příkaz STCHR bez parametru P nebo s parametrem P rovným nule. Způsob vytváření sprajtů ze čtyř definovaných částí je znázorněn na obr. 4.

Nejzajímavější vlastností sprajtů je jejich překrývání, které je možno využít v mnoha aplikačních programech. Pro demonstraci překrývání sprajtů je určen další program:

```
10 REM *** SPRAJTY 2 ***
20 MAG 3:FOR S=0 TO 3:SCOD S,236+S:NEXT S
30 SCOL 0,2:SCOL 1,4:SCOL 2,7:SCOL 3,12
40 FOR X=0 TO 73:FOR S=0 TO 3
50 LOC S TO X*S,0
60 FOR P=1 TO 50:NEXT P
70 NEXT S:NEXT X
80 FOR S=0 TO 2:FOR X=73*S TO 240-7*S
90 LOC S TO X,0
100 FOR P=1 TO 50:NEXT P
110 NEXT X:NEXT S
120 IF INKEY$="" THEN GOTO 120
130 CALL &1387
```

Výše uvedené programy názorně předvedly vlastnosti sprajtů. Je třeba si uvědomit, že v obou programech se mění prostřednictvím několika příkazů pouze obsah 128 bytů paměti VRAM. Ostatní si již zařídí video procesor.

3.07 Technický popis zvukového generátoru

Použitý zvukový generátor SN76489AN firmy Texas Instruments výrazně zvyšuje užitnou hodnotu počítače M5. Pro jeho efektivní využití je potřeba dokonale znát jeho strukturu, činnost a programování. Proto byly navíc proti anglickému manuálu zařazeny následující tři části. V první části, která popisuje zvukový generátor po stránce technické, byl použit originální technický popis obvodu SN76489AN výrobcem.

Zvukový generátor SN76489AN byl navržen jako levný integrovaný obvod pro generování hudebních tónů a zvukových efektů v mikroprocesorových systémech. Jeho základní vlastnosti jsou: tři programovatelné tónové generátory, programovatelný generátor šumu, programovatelné zeslabení ve všech kanálech i šumu, mísení všech tří kanálů a šumu.

Struktura obvodu je znázorněna blokovým schématem na obr. 2. Základní řídící frekvence je nejdříve snížena šestnáctkrát. Informace z datové sběrnice jsou

vedeny přes stykovou část do jednoho ze tří tónových generátorů. Každý tónový generátor se skládá z desetibitového čítače směrem dolů ve spojení s jednoduchou logikou, který funguje jako dělič řídící frekvence. Pro výslednou frekvenci platí vzhah f=fo /(32*N), kde fo je řídící frekvence a N desetibitové číslo 0 - 1023 . Výstup děliče pokračuje do další části, kde se provádí čtyřstupňové zeslabení řízené čtyřbitovou hodnotou 0-15 . Zeslabení jednotlivých stupňů je 2 dB, 4 dB, 7,5 dB a 12,5 dB,

Generátor šumu má dva základní pracovní režimy, periodický šum a bílý šum. Frekvence šumového generátoru může být určena čtyřmi způsoby. U prvních tří způsobů je frekvence pevně nastavena na fo/512, fo/1024 a fo/2048. Při posledním je frekvence určena výstupní frekvencí tónového generátoru 2. Zeslabení generátoru šumu je totožné s tónovými generátory. Všechny čtyři výstupní signály z tónových generátorů a šumového generátoru jsou společné s vnějším zvukovým signálem smíseny a zesíleny.

Informace potřebné k řízení činnosti zvukového generátoru jsou uchovány v osmi řídících registrech. Po dvou registrech potřebuje každý zvukový generátor i generátor šumu.

Zvukový generátor se připojuje k mikroprocesorovému systému na osmibitovou datovou sběrnici DO- D7 . Přenos dat do generátoru se řídí pomocí třech signálů WE, CE a READY.

3.08 Programování zvukového generátoru, příkaz OUT

Před vlastním popisem programování zvukového generátoru je třeba vysvětlit, jakým způsobem je možno dostat informaci z programu v jazyku BASIC-1 přes datovou sběrnici do zvukového generátoru. V prvním díle manuálu jsme se seznámili na obr. 1 s blokovým schematem počítače M5. V horní části obrázku byly paměťové bloky a v dolní jednotlivá vstupní a výstupní zařízení. Již víme, že pomocí příkazů POKE, PEEK, VPOKE a VPEEK můžeme pracovat přímo s pamětmi ROM, RAM i VRAM. Stejně tak můžeme pracovat i se vstupními a výstupními zařízeními pomocí příkazů OUT a INP. Příkaz OUT obsahuje stejně jako POKE dva parametry, adresu a přenášenou hodnotu. Adresa v příkazu OUT může být v rozsahu 0 až 255. Adresa zvukového generátoru je 32.

Z předcházejícího technického popisu je zřejmé, že pro činnost zvukového generátoru je potřeba pomocí příkazu OUT přenášet do příslušných vnitřních registrů potřebné informace. Na obr. 3 je znázorněna bitová struktura přenášených informací společně s adresami vnitřních registrů. Jestliže je poslední bit 1 /čísla 128 /, potom následující tři bity jsou adresou vnitřního registru a první čtyři bity nesou informaci pro naplnění příslušného registru nebo jeho části. U čísel s nulovým posledním bitem /číslo 128 / má význam pouze prvních šest bitů, které tvoří posledních šest bitů desetibitového čísla určujícího frekvenci. Informace ve vnitřních registrech zůstávají tak dlouho, dokud nejsou přepsány. Stejně tak se pamatuje i adresa tónového generátoru, jehož adresa byla naposledy nastavena. To znamená, že při opakovaných změnách frekvence pouze jednoho tónového generátoru není třeba přenášet stále dva byty.

Pro programování zvukového generátoru je nutno si na základě obr. 3 a z technického popisu odvodit několik základních pravidel, pomocí nichž můžeme napsat program pro řízení činnosti zvukového generátoru. Začneme s určením frekvence. Řídící funkce fo= 3.579545 MHz, takže pro výslednou frekvenci platí vztah f=111 860/N, kde N je číslo v rozsahu 1 - 1023. Maximální frekvence pro malé hodnoty N již není slyšitelná, minimální frekvence je 111 860/1023= 109,3 Hz.

Dělící poměr N je potřeba rozdělit na dvě čísla N1 /0-16/ a N2 /0-63/ pomocí vztahů N2= N/=6 a N1= N MOD 16. Do prvního přenášeného bytu je třeba doplnit číslo kanálu K /0-2/ podle vztahu D1=128+32*K+N1. Druhý přenášený byte má přímo hodnotu N2. Pro zeslabení Z /0-15/ v kanálu K /0-2/ je třeba použít příkaz OUT 32, 144+32*K+Z. Na adrese &701A je uložena systémová proměnná, jejíž poslední bit určuje, zda bude zapnuta zvuková indikace stisku klávesy u kanálu 2. Následující program umožnuje vyzkoušení funkce tónových generátorů:

```
10 REM *** ZVUK ***
20 CLS:A=PEEK(&701A):POKE &701A,A-128
30 INPUT "KANAL (0-2) ";K
40 INPUT "DELICI POMER (1-1023) ";N
50 IF N=0 THEN GOTO 110
60 INPUT "ZESLABENI (0-15) ";Z
70 OUT 32,128+32*K+N MOD 16
80 OUT 32,N/16
90 OUT 32,144+32*K+Z
100 GOTO 30
110 POKE &701A,A
```

Zbývá nastavení generátoru šumu. Jestliže bude P pracovní režim /0 - periodický šum, 1 - bílý šum/ a F proměnná pro určení frekvence /0-fo/512, 1 - fo /1024,2 - fo /2048,3 - odvozena z frekvence kanálu 2/, potom použijeme příkaz OUT 32,228+4*P+F. Pro zeslabení Z /0 - 15/ generátoru šumu musíme použít příkaz OUT 32,240+Z.

S výše uvedenými znalostmi se již můžeme pokusit naprogramovat řadu zajímavých zvukových efektů.

V příručce Monitor Handling Manual je popsán celý programový systém tvořící hudební interpretr zadaných dat, který se využívá hlavně v příkazu PLAY jazyku BASIC-G.

3.09 Programové naladění zvukového generátoru

Frekvenční rozsah zvukového generátoru pokrývá sice sedm oktáv /110 Hz - -14080 Hz/, ale pouze v necelých prvních čtyřech oktávách tohoto rozsahu, /110 Hz - 1396.913 Hz/ je možno si určit dělící poměry tak, že výstupní frekvence odpovídají frekvencím běžného hudebního půltónového ladění. Vzhledem k omezeným výpočetním možnostem počítačem M5 s modulem BASIC-I není možno uvést příslušný program pro výpočet dělících poměrů, které jsou uvedeny v následující tabulce:

	0	1	2	3	4
C		855	428	214	107
CIS		807	404	202	101
D		762	381	190	95
DIS		719	360	180	90
E		679	339	170	85
F		641	320	160	80
FIS		605	302	151	
G		571	285	143	
GIS		539	269	135	
A	1017	508	254	127	
AIS	960	480	240	120	
H	906	453	226	113	

4. PRÁCE S PAMĚTÍ V R A M

4.01 Čtyři pracovní obrazovkové režimy

Největší předností počítače M5 jsou jeho možnosti práce s barevnou grafikou zajišťované hlavně použitím video procesoru TMS9929A, který může pracovat ve čtyřech režimech. Video procesor má osm registrů pro zápis řídících informací a jeden registr, z něhož si informace čte počítač. Po inicializaci registrů video procesoru se televizní video signál vytváří nezávisle na činnostech ostatních bloků počítače podle obsahu paměti VRAM.

Základním režimem, ve kterém zůstává počítač po zapnutí a se kterým jsme zatím pracovali, je režim GI /Graphics I mode/. Vlastnosti tohoto pracovního režimu již známe.

Pro plnou jemnou grafiku je určen režim GII /Graphics II mode/. Je v něm možno nadefinovat 768 různých znaků na 24 řádkách po 32 znacích. To odpovídá bodovému rastru 256 x 192. K tomu je potřeba oblast paměti VRAM o velikosti 6 KB. V každých osmi vodorovných bodech mohou být definovány dvě barvy ze šestnácti možných. Pro obarvení je proto potřeba také oblast paměti VRAM o velikosti 6 KB. Pro práci v režimu GII pomocí příkazu STCHR je obrazovka rozdělena na tři stejné části - horní, střední a spodní. Sprajty se mohou používat stejně jako v režimu GI.

Jako třetí v pořadí se uvádí Multi-color režim. Každá znaková pozice na 24 řádkách po 32 znacích je rozdělena na čtyři bloky a každý z těchto bloků může mít jednu ze šestnácti barev. Sprajty se též mohou používat.

Posledním režimem je režim textový, kdy je možno zobrazovat pouze text bez sprajtů na 24 řádkách po 40 znacích. Znaky jsou vytvořeny v rastru 6 x 8 bodů. Obarvit je možno pouze pozadí a text jako celek. Na základě praktických zkušeností se doporučuje používání tohoto režimu při tvorbě programu a při používání textových informací v programech.

Přepínání obrazovkových režimů je možno provádět přímo z klávesnice pomocí tlačítka CTRL a dalších čtyř tlačítek: GI-S, GII-R, Multi-color-Q a textový režim -T. Všechny funkce prováděné pomocí tlačítka CTRL ve spojení s dalším tlačítkem je možno použít i v programu. V příkazu PRINT se s nimi pracuje jako s částí řetězcové konstanty. Současně s tlačítkem CTRL ale musí být stisknuto tlačítko SHIFT a příslušné tlačítko s řídícím kodem. Na obrazovce se řídící kody zobrazují inverzně. Mohou být použity v příkazu PRINT i samostatně, ale vždy musí být uzavřeny v uvozovkách. Použití řídících kodů v programu je v programu UFO GAME v příloze A originálního manuálu na straně 77.

V následující tabulce jsou přehledně shrnuty základní vlastnosti výše popsaných pracovních obrazovkových režimů:

Režim	Řídící kod	Počet řádků a sloupců	Rastr grafiky	Rastr znaků	Počet znaků	Počet barev	Sprajty
GI	S	24 x 32	192 x 256	8 x 8	256	16	ano
GII	R	24 x 32	192 x 256	8 x 8	768	16	ano
M-color	Q	-	48 x 64	4 x 4	1	16	ano
textový	T	24 x 40	192 x 256	8 x 6	256	2	ne

V dalších částech bude podrobně popsána práce v režimech GII, Multi-color a Text.

4.02 Použití příkazu STCHR v režimu GII

V části 3.03 jsme si vysvětlili základní funkci příkazu STCHR při definování nových znaků v režimu GI. V režimu GI můžeme používat maximálně 256 znaků a u každých osmi znaků můžeme nastavit barvu znaku a barvu pozadí.

V režimu GII slouží příkaz STCHR pro definování tvaru znaku i jejich barvy ve třech nezávislých generátorech znaků pro horní, střední a spodní část obrazovky. Názorně je to demonstrováno následujícím programem:

```
10 REM *** GRAFIKA 1 ***
20 PRINT "R":REM INVERZNI R (RIDICI KOD)
30 A=14336:FOR I=0 TO 767:UPOKE A+I,I MOD 256:NEXT I
40 FOR P=1 TO 6
50 FOR Z=0 TO 255
60 K$="":FOR J=1 TO 8:K$=K$+RIGHT$(HEX$(RND(255)),2)
61 NEXT J
70 STCHR K$ TO Z,P
80 NEXT Z:NEXT P
90 IF INKEY$="" THEN GOTO 90
100 PRINT "TR":REM INVERZNI TR (RIDICI KODY"
```

Na řádce 20 se nám programově přepne zobrazení na režim GII. Další řádek 30 zobrazí znaky všech tří generátorů znaků režimu GII. Na řádce 40 začíná cyklus pro parametr P od jedné do šesti. Řádek 60 generuje náhodný řetězec K$ pro příkaz STCHR, který je použit na řádce 70 pro definování nového tvaru nebo barvy. Parametr P v příkazu STCHR má tento význam:

P = 1 - definování tvaru znaku pro horní část obrazovky
P = 2 - definování tvaru znaku pro střední část obrazovky
P = 3 - definování tvaru znaku pro spodní část obrazovky
P = 4 - definování barev znaku pro horní část obrazovky
P = 5 - definování barev znaku pro střední část obrazovky
P = 6 - definování barev znaku pro spodní část obrazovky

To znamená, že se nám nejdříve změní náhodně tvary všech znaků na obrazovce a potom se postupně nastaví jejich barvy. Při nastavení barev představuje každá dvojice znaků řetězce v příkazu STCHR hexadecimální kód barvy pozadí a bodu v jednom řádku základního znakového rastru 8x8. Jestliže na řádku 40 bude počáteční hodnota řídící proměnné cyklu 4 místo 1, dojde k náhodnému obarvení tří generátorů znaků.

Na řádku 100 se provádí inicializace původních tří znakových generátorů režimu GII programovým přepnutím do textového režimu a okamžitě zpět do režimu GII.

Rozdělení obrazovky do tří částí nabízí řadu využití v různých programech. V předchozí části již byla zmínka o možnosti použití sprajtů v režimu GII stejně jako v režimu GI.

4.03 Jemná grafika v režimu GII

Z předchozí části je zřejmé, že v režimu GII je možno nadefinovat 768 různých znaků tvarově i barevně. V demonstračním programu GRAFIKA 1 jsme na to používali náhodná čísla. Aby bylo možno využívat plně jemnou grafiku v rastru 192 x 256, musíme umět zobrazit jediný bod o souřadnicích X (0 - 255), Y (0 - - 191). V následujícím programu je použit krátký podprogram $PLOT, který toto zobrazení umí.

```
10 REM *** GRAFIKA 2 ***
20 PRINT "R":REM INVERZNI R (RIDICI KOD)
30 A=14336:FOR I=0 TO 767:VPOKE A+I,I MOD 256:NEXT I
40 FOR P=1 TO 3:FOR Z=0 TO 255
50 STCHR "0000000000000000" TO Z,P
60 NEXT Z:NEXT P
70 FOR I=2 TO 6:FOR X=0 TO 255
80 Y=X/I:GOSUB $PLOT:NEXT X:NEXT I
90 FOR I=2 TO 6:FOR X=255 TO 0 STEP -1
100 Y=(255-X)/I:GOSUB $PLOT:NEXT X:NEXT I
110 IF INKEY$="" THEN GOTO 110
120 PRINT "TR":REM INVERZNI TR (RIDICI KODY):STOP
130 $PLOT
140 A=8192+256*(Y/8)+8*(X/8)+Y MOD 8
150 B=VPEEK(A):D=2↑(7-X MOD 8)
160 VPOKE A,B OR D
170 RETURN
```

V úvodu programu je nejdříve provedeno vymazání obrazovky v režimu GII. Z průběhu programu je zřejmé, že podprogram $PLOT doplňuje body do již zobrazené grafiky. Jestliže by jsme chtěli použít podprogram $PLOT pro mazání, musíme na řádku 160 použít příkaz VPOKE A,B AND (255-D).

Na obr. 6 je rozdělení paměti VRAM při přepnutí do režimu GII. Podprogram $PLOT je potom možno si doplnit obarvením zobrazeného bodu.

Kombinace jemné grafiky s textem je možná pouze tak, že se jednotlivé znaky dávají dohromady přes generátor znaků s použitím podprogramu $PLOT nebo se nejdříve předefinují na příslušných místech obrazovky původní znaky příkazem STCHR na znaky potřebné a potom se již normálně používá podprogram $PLOT.

4.04 Grafické možnosti režimu Multi-color

Na obr. 8 je rozdělení paměti VRAM a její přiřazení pozicím na obrazovce v režimu Multi-color. Na základě těchto informací byl sestaven následující demonstrační program. Je v něm použit podprogram $BLOK pro zobrazení barevného bloku s barvou B (0-15)a souřadnicích X (0 - 63) a Y (0 - 47). Na začátku programu je obarvení obrazovky danou barvou B. Sprajty se v tomto režimu používají stejně jako v režimu GI.

```
10 REM *** GRAFIKA 3 ***
20 PRINT "Q":REM INVERZNI Q (RIDICI KOD)
30 FOR  R=0 TO 5:K0=32*R:FOR K=K0 TO K0+32
40 A1=14336+R*128+K MOD 32
50 FOR A=A1 TO A1+97 STEP 32:VPOKE A,K
60 NEXT A:NEXT K:NEXT R
70 B=12:B=16*B+B:FOR A=10240 TO 11755
80 VPOKE A,B:NEXT A
90 FOR Y=0 TO 47:FOR X=0 TO 63:B=RND(15)
100 GOSUB $BLOK:NEXT X:NEXT Y
110 IF INKEY$="" THEN GOTO 110
120 PRINT "TS":REM INVERZNI TS (RIDICI KODY):STOP
130 $BLOK
140 A=10240+256*(Y/8)+8*(X/2)+Y MOD 8
150 D=VPEEK(A):DL=D MOD 16:DH=D/16
160 IF X MOD 2=0 THEN DH=B ELSE DL=B
170 VPOKE A,16*DH+DL
180 RETURN
```

4.05 Práce v textovém režimu

Textový režim umožňuje větší hustotu psaného i zobrazeného textu, což je vhodné při psaní a odladování delších programů. Čtyřiceti znaků na řádce se dosáhlo tím, že znakový rastr má rozměr 8 x 6. Při definování nových znaků se opět používá příkaz STCHR, ale v řetězci pro zadání tvaru znaku nejsou u každého hexadecimálního čísla využity první dva bity.

Barvu znaků a barvu pozadí si nastavíme krátkým programem:

```
10 PRINT "T":REM INVERZNI T (RIDICI KOD)
20 INPUT "BARVA ZNAKU ";BZ
30 INPUT "BARVA POZADI ";BP
40 VPOKE &709F,16*BZ+BP:CALL &0CA3
```

Vzhledem k tomu, že textový režim neumožňuje práci se sprajty, je značná část paměti VRAM volná pro využití programátorem. Konkrétně je to oblast paměti VRAM od začátku do adresy 10239. Na obr. 7 je znázorněno obsazení paměti VRAM v textovém režimu.

4.06 Záznam bloků paměti VRAM a RAM na magnetofon

V části 2.09 jsme se seznámili s příkazem SAVE pro záznam programu na magnetofon. Příkaz SAVE umožňuje při přidání dalších parametrů zaznamenat na pásku i libovolné bloky paměti VRAM nebo RAM. V předchozí části jsme si uvedli, že při práci v textovém režimu je 10240 volných bytů v paměti VRAM k dispozici uživateli. Tuto oblast paměti je možno využít pomocí příkazů VPOKE a VPEEK pro uložení dat z programu. Příkazem z klávesnice SAVE "JMENO", 0,10239,1 se nám tato oblast paměti VRAM zaznamená na magnetofon. Za názvem souboru jsou další dva parametry označující začátek a konec paměťového bloku. Poslední parametr o hodnotě 1 udává, že se jedná o oblast paměti VRAM. V případě paměti RAM je poslední parametr nula nebo úplně chybí. Při další práce s takto zaznamenaným souborem se používají běžné příkazy VERIFY nebo OLD bez názvu nebo s názvem souboru. Informace o druhu a velikosti zaznamenaného souboru jsou již obsaženy v hlavičce souboru na pásce.

Záznam a zpětné čtení paměti VRAM se doporučuje hlavně při psaní programů, které používají větší počet sprajtů. Místo aby se sprajty definovaly přímo v programu, nahrají se rychle a bez nároku na paměť z magnetofonu.

4.07 Práce na dvou obrazovkách

Paměť VRAM o kapacitě 16 KB umožňuje uchovat značné množství informací sloužících pro výstup na obrazovce TV přijímače. V předchozích částech jsme se stručně seznámili se čtyřmi pracovními režimy použitého video procesoru. Ani jeden ze čtyř pracovních režimů nepotřebuje celou paměť VRAM. Dokonce se dva pracovní režimy svými paměťovými nároky do paměti VRAM pohodlně vejdou. To vedlo autory základního technického a programového vybavení počítače M5 k vytvoření systému, který by uživateli umožňoval pracovat současně na dvou obrazovkách, z nichž by ale pouze jedna byla pro uživatele viditelná.

Po zapnutí počítače se objeví obrazovka 0 (screen 0). Programově nebo přímo z klávesnice se po řídícím kódu V objeví obrazovka 1 (screen 1).

Řídící kód V provádí jejich vzájemnou výměnu. Řídící kód U vždy zobrazí obrazovku 0. Řídící kod Z přesune kurzor na obrazovku, která není vidět. Potom je možno na ni programově nebo přímo z klávesnice zapisovat informace. Zpětný návrat kurzoru je po dalším řídícím kodu Z. Řídící kod Y působí opačně ke kódu Z. Zobrazí se obrazovka, která nebyla vidět, ale kurzor zůstane na původní, na kterou je možno zaznamenávat informace. Dalším řídícím kódem Y nastane původní stav.

Práce na dvou obrazovkách, z nichž jedna je zobrazena, je výhodná hlavně proto, že na každé obrazovce může být nastaven různý pracovní režim. Pro psaní a odladování programu je nejvhodnější textový režim a pro grafiku režimy GI nebo GII. Jsou povoleny všechny kombinace režimu kromě dvakrát použitého režimu GII.

Z obrázku 5 a 6 je zřejmé, že každá obrazovka má své vlastní vyhražené oblasti v paměti VRAM. Na obou obrazovkách tedy můžeme např. nastavit režim textový se dvěma odlišnými generátory znaku.

Při práci se dvěma obrazovkami se může stát, že najednou není vůbec nic vidět. V případě běžícího programu program zastavíme tlačítky CTRL+RESET nebo SHIFT+RESET, stiskneme CTRL+U a zkušíme několik přepnutí mezi dvěma různými obrazovkovými režimy.

5. PŘIPOJENÍ A PROGRAMOVÁNÍ VNĚJŠÍCH ZAŘÍZENÍ

5.01 Připojení tiskárny

Na zadní straně počítače je konektor pro připojení paralelní tiskárny, jejíž činnost je řízena podle standardu Centronics. Veškeré informace pro připojení tiskárny jsou na obr. 9 . Vedle výstupu na tiskárnu lze konektor využít jako universální osmibitový výstup. Protikus do použitého konektoru se u nás nevyrábí. Je nutno si upravit konektor FRB jednoduchým způsobem:

A/ uřízne se patřičná délka konektoru FRB
B/ uprostřed se opatrně rozřízne naplocho na dvě části lupínkovou pilkou
C/ obě části se v místě řezu obrousí, aby je po přiložení k sobě bylo možno zasunout do konektoru
D/ obě části se v místě řezu a broušení slepí k sobě
E/ připojí se připojovací kablíky a označí se vršek konektoru

Výpis programu a tisk dat na tiskárnu se provádí pomocí příkazů LIST a PRINT, v nichž musí být uvedeno označení tiskárny jako výstupního zařízení. Tiskárna má číslo 2, takže příkazy budou mít např. tento tvar:

```
LIST #2,10,200
PRINT #2,RND(100);
```

Ve výpisech programu se místo inverzně značených řídících kódů objeví mezery.

Na obrazovku se vypisují programy nebo data po maximálně čtyřiceti znacích. Při tisku na tiskárnu je možno si předem počet znaků nastavit na adrese &705B. Na této adrese je uložena systémová proměnná PMXCLM, která označuje maximální počet znaků na řádku. Po zapnutí počítače je nastaven počet znaků na 80, jak se můžeme přesvědčit příkazem PRINT PEEK(&705B) . Při tisku je počet již vytištěných znaků na řádce průběžně ukládán na adrese &705C. Pro řízení tiskárny je důležitá ještě jedna systémová proměnná s názvem POUTFG /Output Charakteristic Flag / na adrese &705A, v níž mají význam pouze první dva bity. Má-li první bit hodnotu 1, potom se na tiskárnu vysílá po řídícím kódu CR (&OD)

kód LF (&0A). Má-li druhý bit hodnotu 1, potom se vysílá na tiskárnu při výpisu programu po vytisknutí maximálního počtu znaků řídící kód CR nebo CR a LF. Toto automatické řádkování pochopitelně funguje i při tisku dat. Po zapnutí počítače mají oba bity hodnotu 1.

5.02 Adresování bloků počítače, rozšíření paměti RAM

Modulová koncepce počítače M5 umožňuje jeho rozšiřování technické i programové. Pro připojování dalších vnějších zařízení je třeba znát rozdělení paměti počítače a adresování jednotlivých vstupních a výstupních bloků počítače. Na obrázcích 9 a 10 je nakresleno rozdělení paměti počítače, adresování bloků počítače a zapojení konektoru systémové sběrnice. Na systémové sběrnici jsou vedle běžných datových, adresových a řídících signálů mikroprocesorového systému Z80 další signály, jejichž funkce je dále stručně nastíněna:

ROM0 - výběr paměti ROM při čtení adres &0000-&0FFF
ROM1 - výběr paměti ROM při čtení adres &1000-&1FFF
ROM2 - výběr paměti ROM při čtení adres &2000-&5FFF
EXM - výběr paměti RAM při adresách &8000-&FFFF
EXIOA - výběr stykového obvodu pro periferii A
EXIOB - výběr stykového obvodu pro periferii B
ROMDS - blokování čtení z paměti ROM &0000-&1FFF

Součástí zvláštního příslušenství počítače M5 je modul paměti RAM o kapacitě 32 KB společně s modulem umožňujícím připojení pamětového modulu s modulem programovým BASIC-I, BASIC-F, BASIC-G na konektor systémové sběrnice. Vzhledem k pokrokovému řešení paměti 32 KB s vysokou spolehlivostí a nízkou spotřebou, není třeba jiný síťový napaječ.

5.03 Využití vstupů pro ovladače, příkazy INP a TIME

Na zadní straně počítače jsou dva konektory pro připojení dvou ovladačů. Každým ovladačem lze volit jeden z osmi směrů nebo mačkat akční tlačítko v různých programech. Zapojení konektoru i ovladačů je na obr. 11. Použité konektory nejsou v ČSSR k sehnání. Je možno si je amatérsky udělat s celkem slušným výsledkem. Stačí k tomu několik drátků od normálních miniaturních odporů, tvrdý a měkký papír, acetonové lepidlo a šikovné ruce.

Programovací jazyk BASIC-I nemá speciální příkaz pro zjišťování směru zadaného ovladačem jako jazyk BASIC-G. Musíme použít pro nás nový příkaz INP, který přiřadí číselné proměnné hodnotu v rozsahu 0 až 255 přečtenou ze zvoleného vstupního zařízení. Příkazem INP je možno také programově zjišťovat současně stisknutá tlačítka klávesnice.

Místo dvou ovladačů je možno připojit jiné vstupní periferní zařízení. Při využití počítače M5 s jinými periferními zařízeními se jistě využijí při tvorbě obslužných programů hodiny reálného času počítače M5. Okamžitá hodnota proměnné TIME totiž ukazuje počet vteřin, které uplynuly od zapnutí počítače. Při provozování počítače v konstantních teplotních podmínkách je přesnost vnitřních hodin počítače M5 značná.

Uvnitř počítače M5 je malé relé pro zapínání a vypínání motorku magnetofonu. Relé je na malé napětí a má zatěžovací proud několik set miliampér. Na kolík pro dálkové ovládání magnetofonu si však můžeme připojit další výkonový

spínač. Pro jeho ovládání se použijí dva podprogramy monitoru. Příkaz CALL &1776 relé sepne a příkaz CALL &177E relé rozepne.

Z ne zcela běžných periferních zařízení byla zatím k počítači M5 připojena tříoktávová klávesnice. Jednoduchý obslužný program ve strojovém kodu umožnuje tříhlasou hru v reálném čase.

DOPLŇKY K DRUHÉMU VYDÁNÍ PŘÍRUČKY /DUBEN 85/

V druhém vydání byly odstraněny některé nepřesnosti, byla podstatně změněna a doplněna programová příloha a navíc dopsána následující část. Jsou zde objasněny některé problémy, s nimiž se uživatelé setkali a vysvětleny další příkazy, které se do prvního vydání nedostaly.

Řada uživatelů měla potíže se záznamem programů na kazetový magnetofon a s jejich zpětným převodem do počítače. Podle našich zkušeností s několika různými typy zahraničních magnetofonů vybavenými zástrčkami MIC a EAR byly záznam a zpětné čtení naprosto bezchybné. Vstup MIC je určen pro běžné levné typy dynamických mikrofonů. Některé magnetofony vybavené pětikolíkovými zástrčkami DIN mohou mít vstupy dva. Správnost záznamu na magnetofon si ověříme zpětným přehráním přes reproduktor a srovnáním s demonstrační kazetou. Hlasitost obou záznamů by měla být stejná. V žádném případě by neměla být hlasitost vašeho záznamu menší než záznamu z demonstrační kazety. Výstup EAR je výstupem na vnější sluchátko nebo vnější reproduktor s minimální impedancí 4 ohmy. Některé magnetofony tento výstup vůbec nemají. Na výstupu pro nahrávání na další magnetofon nebo pro zesilovač je pro počítač M5 příliš malé napětí. V takovém případě je třeba si z magnetofonu vyvést odbočku od reproduktoru s přepínáním na reproduktor nebo počítač. Při převodu programů z magnetofonu je třeba, aby tónová clona byla na maximu stejně jako regulátor hlasitosti.

Při delším používání počítače není třeba se obávat přílišného zahřívání napájecího zdroje. Je to způsobeno tím, že zdroj je navržen pro napájení dvou modulů /BASIC-F nebo BASIC-G nebo FALC s pamětí 32 K/, a při použití modulu BASIC-1 s malou spotřebou se ztrácí část příkonu na regulačních prvcích ve formě tepelné energie. Vlivem tepla může dokonce dojít k zkrabatění samolepek na horní i spodní straně zdroje, což je pro funkci zdroje i počítače naprosto nezávadné. Při provozním testování několika počítačů bylo zjištěno, že v běžných kancelářských podmínkách je počítač schopen prakticky nepřetržitého provozu.

Výstupní signál počítače pro televizní přijímač je v mezinárodní normě PAL. Některé typy barevných televizorů mohou přijímat barevný signál pouze v normě SECAM. Týká se to prakticky všech sovětských televizorů, starších československých televizorů a i některých zahraničních televizorů. U těchto televizorů bude potom po připojení počítače M5 obraz černobílý. Jestliže by se občas objevil náznak barvy, je lepší pro kvalitnější obraz přijímač přepnout na černobílý obraz /např. stažením barev/.

Při zobrazení programu nebo čísel na obrazovce TV přijímače v režimu GI může u některých televizorů dojít k tomu, že není vidět poslední znak na řádku. V textovém režimu by k tomu nemělo docházet tak výrazně. Je to způsobeno pracovním nastavením obvodů ve video části počítače pro dosažení maximálního využití šířky televizního řádku. Není to tedy žádná vada počítače. U televizních přijímačů s vyvedenými ovládacími prvky pro nastavení rozměrů obrazu je možno celý obraz opatrně posunout doleva.

V prvním vydání příručky nebyly uvedeny příkazy VIEW, FIRE a CLEAR.

Příkazem VIEW se určuje nová pracovní oblast i s novým počátkem pro pohyb ukazatele po obrazovce. Příkaz VIEW má čtyři parametry pro určení nové hranice pracovní oblasti ukazatele. Po zapnutí počítače je obrazovka ve stavu jako po příkazu VIEW 0,0,31,23. Zrušení pracovní oblasti a přechod do původního stavu můžeme provést buď příkazem VIEW 0,0,31,23 nebo pouze příkazem VIEW. Funkci příkazu VIEW si ukážeme na krátkém programu:

```
10 CLS
20 FOR I=1 TO 767
30 PRINT CHR$(RND(95)+32);
40 NEXT I
50 VIEW 7,7,24,16
60 FOR I=1 TO 1000
70 PRINT I;
80 NEXT I
90 CLS
100 PRINT "VIEW:CLS"
```

Po skončení programu a stisknutí CTRL+K se dostaneme do počátku nové pracovní oblasti ukazatele. Po stisknutí tlačítka RETURN se provedou příkazy VIEW s CLS a dostaneme se do původního stavu s plnou pracovní oblastí ukazatele. Umístění sprajtů není příkazem VIEW nikterak omezováno.

Příkaz FRE s parametry 0 až 4 slouží pro zjišťování informací o velikosti pracovních oblastí. Příkazy PRINT FRE (4) zjistíme poslední adresu v paměti RAM, kterou ještě používá jazyk BASIC-I.

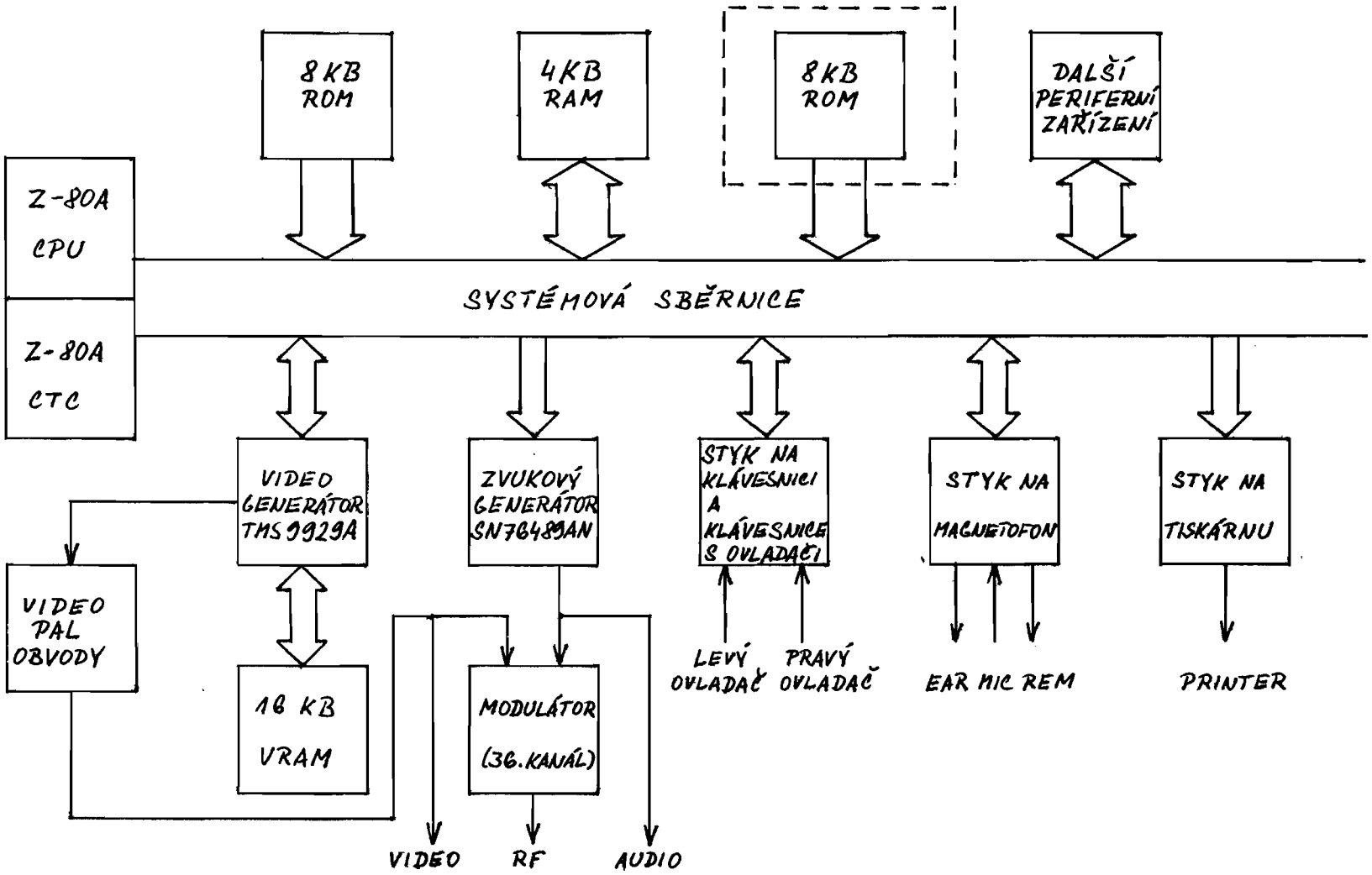
Příkazy PRINT FRE (1) udávají volnou paměť pro program a data. Zde je třeba připomenout, že hodnota FRE (1) nesmí klesnout pod sto. To znamená, že prakticky využitelná paměť pro program a data je pouze 2837 bytů. Zbytek do velikosti 4KB zabírají pracovní oblasti monitoru a jazyku BASIC-I s příslušnými tabulkami a systémovými proměnnými.

Příkaz CLEAR slouží k vymazání hodnot všech proměnných. Ruší se jím i všechny předchozí příkazy DIM.

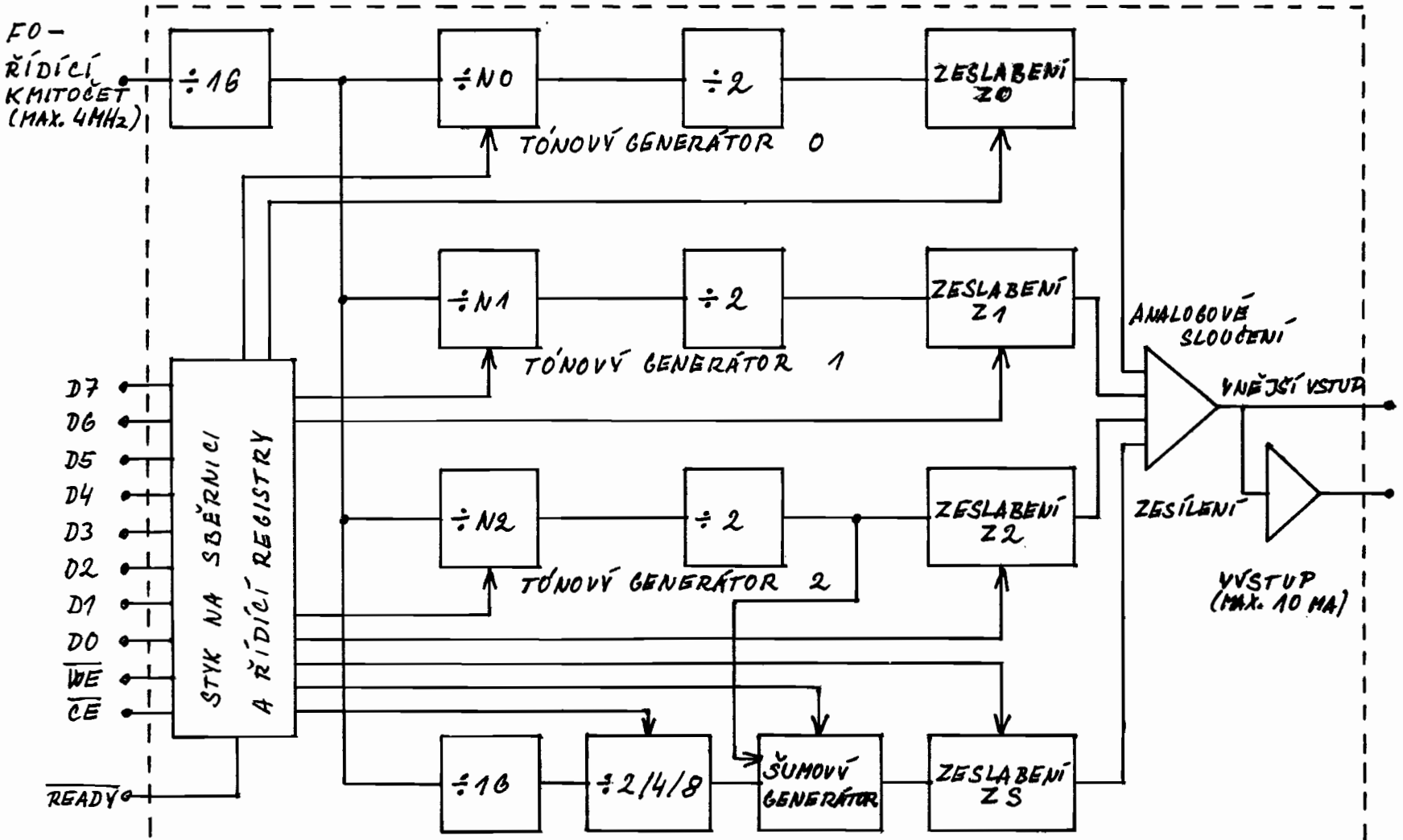
Na úplný závěr je vhodné zdůraznit, že tato příručka je určena i pro ty uživatele, kteří budou pracovat s programovacími jazyky BASIC-F a BASIC-G. Obě české verze příruček pro tyto jazyky předpokládají znalost příručky pro BASIC-I. Vzhledem k tomu, že v každé pracovní sestavě počítače M5 je vždy modul BASIC-I s touto příručkou, bylo by jistě zbytečné znovu doplnovat zde uváděné informace do příruček pro BASIC-F a BASIC-G.

Programovací jazyky BASIC-F a BASIC-G výrazně zvyšují užitnou hodnotu Vašeho počítače. BASIC-F je na úrovni programovacích jazyků BASIC pro profesionální stolní počítače a svojí přesností předčí většinu vyráběných osobních počítačů. BASIC-G Vás překvapí svými možnostmi při realizaci počítačových her a experimentů s barevnou grafikou a naučí Vás díky možnostem několikerého přerušení i trochu jinému myšlení při tvorbě programů.

OBR. 1 BLOKOVÉ SCHÉMA POČÍTAČE SORD M5



OBR. 2. BLOKOVÉ SCHEMA ZVUKOVÉHO GENERÁTORU



OBR. 3 ŘÍZENÍ ČINNOSTI ZVUKOVÉHO GENERÁTORU

1. NASTAVENÍ FREKVENCE TÓNOVÝCH GENERÁTORŮ 0 - 2

PRVNÍ PŘENÁŠENÝ BYTE

BIT	7	6	5	4	3	2	1	0
	1	R2	R1	R0	N3	N2	N1	N0
	ADRESA REGISTRU				N1			

KANÁL 0	0 0 0
KANÁL 1	0 1 0
KANÁL 2	1 0 1

DRUHÝ PŘENÁŠENÝ BYTE

BIT	7	6	5	4	3	2	1	0
	0	/	N9	N8	N7	N6	N5	N4
	N2							

N3	N2	N1	N0
N1			

DĚLICÍ POMĚR N (1-1023)

2. NASTAVENÍ ZESLABENÍ

PŘENÁŠENÝ BYTE

BIT	7	6	5	4	3	2	1	0	
	1	R2	R1	R0	Z3	Z2	Z1	Z0	
	ADRESA REGISTRU				ZESLABENÍ				
KANÁL 0	0	0	1	0	0	0	1		2 dB
KANÁL 1	0	1	1	0	0	1	0		4 dB
KANÁL 2	1	0	1	0	1	0	0		7 dB
GEN. ŠUMU	1	1	1	1	0	0	0		12.5 dB

3. NASTAVENÍ GENERÁTORU ŠUMU

PŘENÁŠENÝ BYTE

BIT	7	6	5	4	3	2	1	0
	1	1	1	0	/	P	F1	F0
	ADRESA REGISTRU							

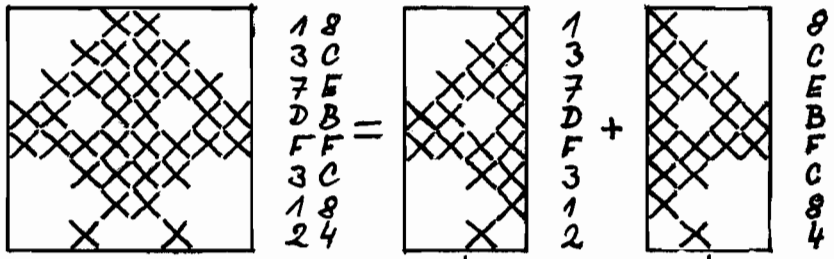
PRACOVNÍ REŽIM	
0 - PERIODICKÝ ŠUM	
1 - BÍLÝ ŠUM	

FREKVENCE		
0	0	F0/512
0	1	F0/1024
1	0	F0/2048
1	1	KANÁL 2

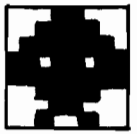
OBR. 4 DEFINOVÁNÍ TVARU ZNAKŮ A SPRAJTŮ, ZVĚTŠENÍ SPRAJTŮ

ZNAK UFO - STCHR "183C7EDBFF3C1824" TO 128,1:PRINT CHR$(128);

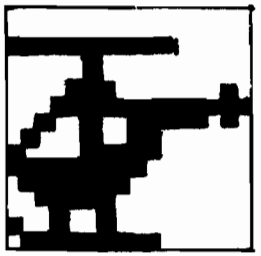
SPRAJT UFO - STCHR "183C7EDBFF3C1824" TO 128: SCOD 0,128:SCOL 0,12:LOC 0 TO X,Y

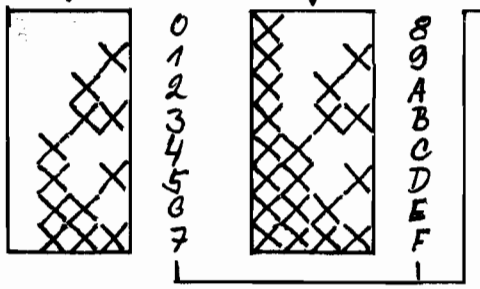


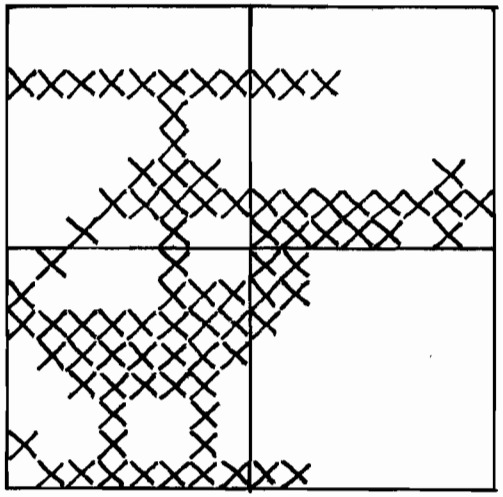

MAG 0 (8×8)


MAG 1 (16×16)


MAG 2 (16×16)


MAG 3 (32×32)





MAG 2, MAG 3

N (0-63), K=4*N, S (0-31)

K	K+2
K+1	K+3

SCOD S,K =

SCOD S,K+1 =

SCOD S,K+2 =

SCOD S,K+3

=

MAG 2 (MAG 3)

STCHR"0000FF04040E1F24" TO 124

STCHR"4487FF7F3E12027F"TO 125

STCHR "0000E0000002FFFA" TO 126

STCHR "C0C0800000000C0" TO 127

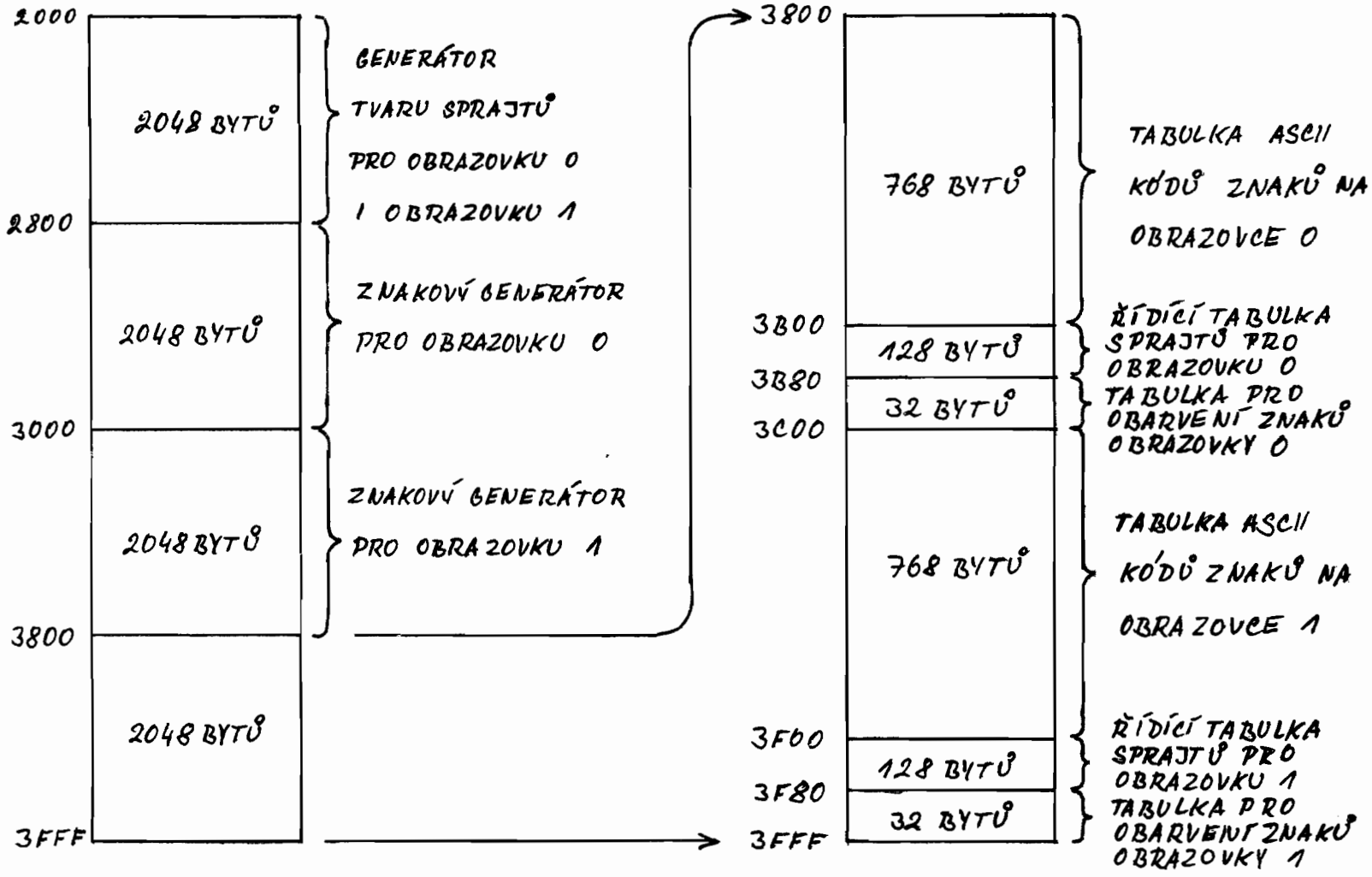
SCOD 0,124:SCOL 0,B B(0-15)

LOC 0 TO X,Y X(0-255)

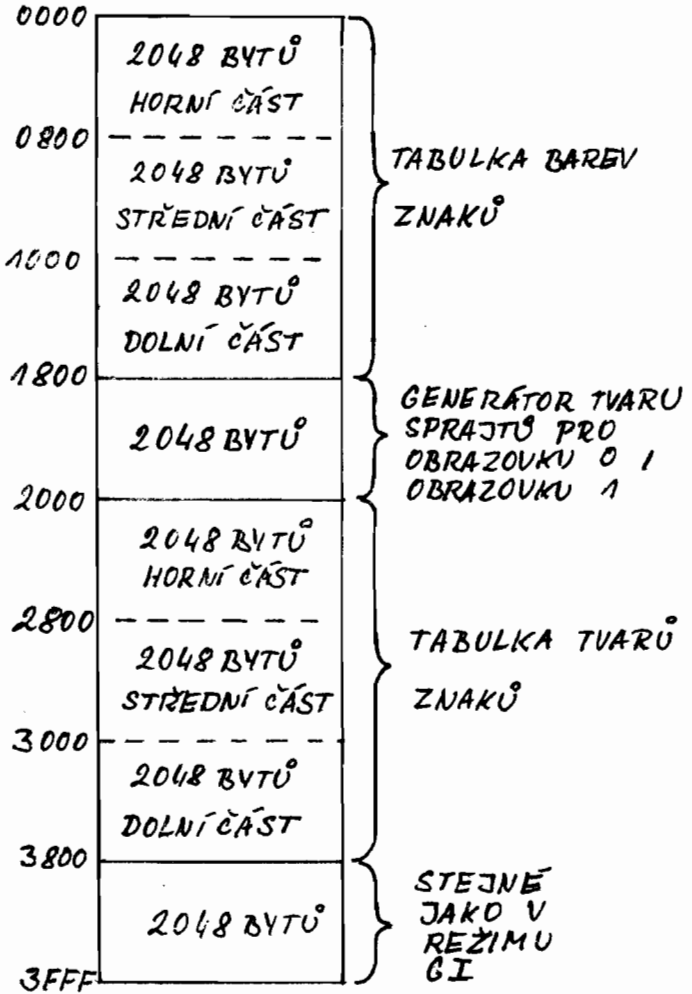
Y(0-191)

OBR. 5 ROZDĚLENÍ PAMĚTI VRAM V REŽIMU GI

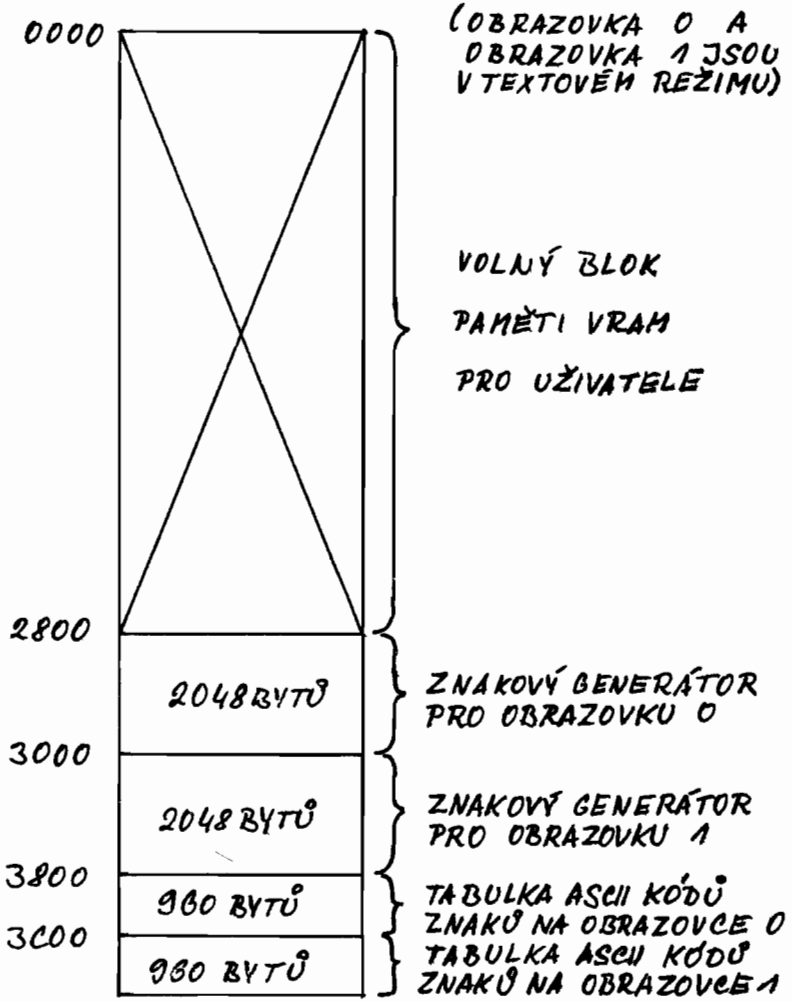
(PAMĚŤ VRAM DO ADRESY 1FFF JE NEVYUŽITA)



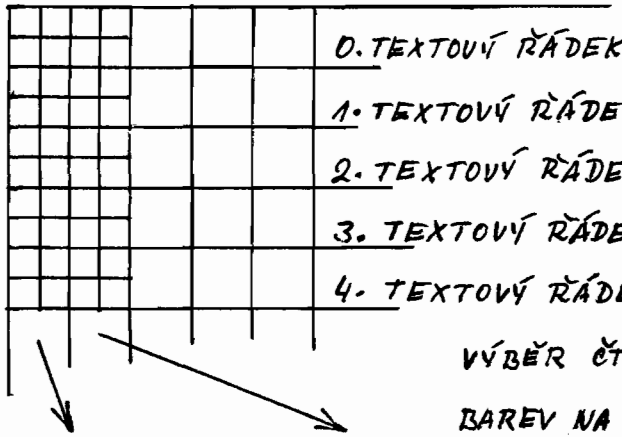
OBR. 6 ROZDĚLENÍ PAMĚTI VRAM V REŽIMU CII



OBR. 7 ROZDĚLENÍ PAMĚTI VRAM V REŽIMU TEXT



OBR.8 PAMĚŤ VRAM V REŽIMU MULTI-COLOR



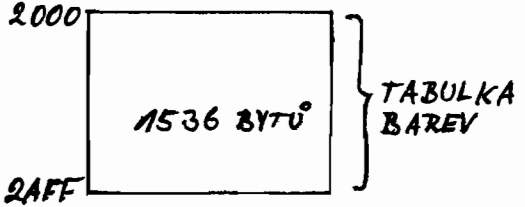
2000	A	B
2001	C	D
2002	E	F
2003	G	H
2004	I	J
2005	K	L
2006	M	N
2007	O	P

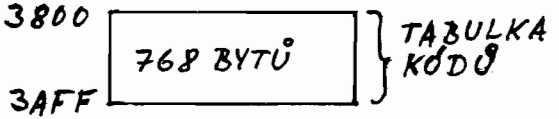
2008	A	B
2009	C	D
200A	E	F
200B	G	H
200C	I	J
200D	K	L
200E	M	N
200F	O	P

VÝBĚR ČTYŘ BAREV NA JEDNĚ ZNAKOVÉ POZICI JE ZÁVISLÝ NA KÓDU A NA VÝRAZU R MOD 4, KDE R JE ČÍSLO TEXTOVÉHO ŘÁDKU

TABULKA KÓDŮ PRO PROGRAM GRAFIKA 3

	0	1	2	…	30	31
0	0	1	2		30	31
1	0	1	2		30	31
2	0	1	2		30	31
3	0	1	2		30	31
4	32	33	34		62	63
5	32	33	34		62	63
6	32	33	34		62	63
7	32	33	34		62	63
8	64	65	66		94	95
9	64	65	66		94	95
26	128	129	130		158	159
27	128	129	130		158	159
28	160	161	162		190	191
29	160	161	162		190	191
30	160	161	162		190	191
31	160	161	162		190	191



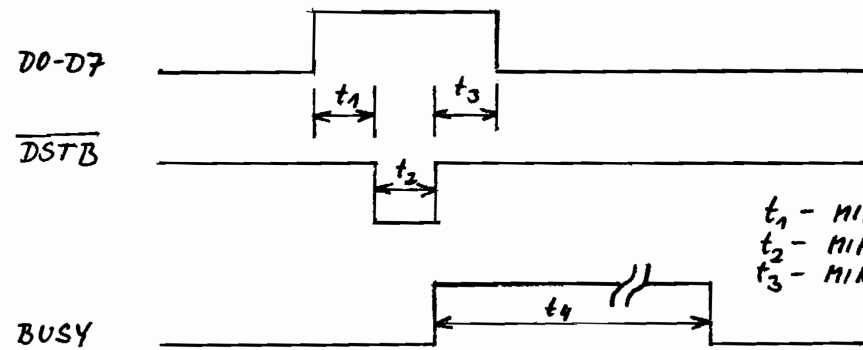


OBR. 9 PŘIPOJENÍ TISKÁRNY, SYSTÉMOVÁ SBĚRNICE

POHLED ZEZADU NA KONEKTOR TISKÁRNY

$\overline{DSTB}$	1	2	GND
D0	3	4	D1
D2	5	6	D3
D4	7	8	D5
D6	9	10	D7
X	11	12	BUSY
GND	13	14	GND
X	15	16	X

PRŮBĚH ŘÍDÍCÍCH SIGNÁLŮ PŘI VÝSTUPU DAT



t_1 - MIN. 1 μsec
t_2 - MIN. 1 μsec
t_3 - MIN. 1 μsec

t_4 - ZÁVISÍ NA PERIFERII

ŘÍZENÍ TISKÁRNY

40H OUTPUT [7|6|5|4|3|2|1|0] – DATA

50H OUTPUT [| | | | | | |0] – $\overline{DSTB}$

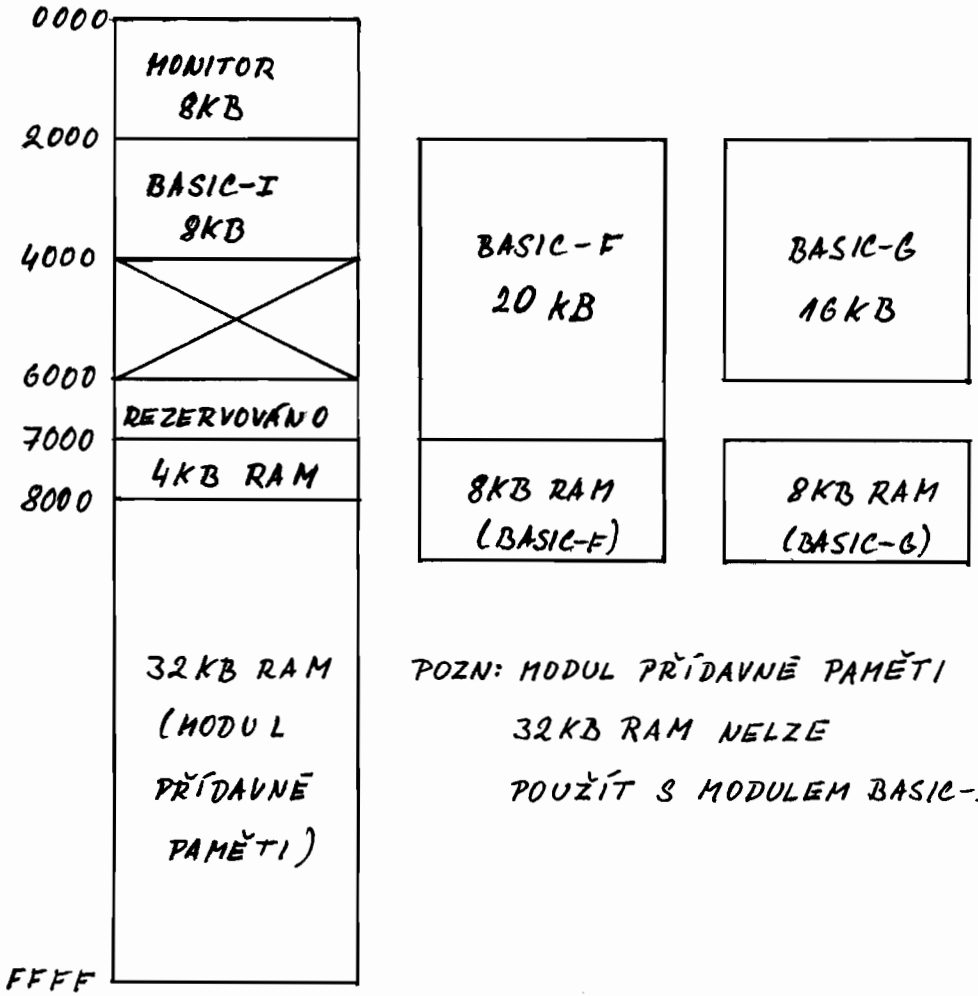
50H INPUT [| | | | | |1|] – BUSY

POHLED SHORA NA KONEKTOR

B		A
-12V	1	-12V
+12V	2	+12V
+12V	3	+12V
+5V	4	+5V
+5V	5	+5V
GND	6	GND
GND	7	GND
D4	8	D3
D5	9	D2
D6	10	D1
D7	11	D0
A8	12	A7
A9	13	A6
A10	14	A5
A11	15	A4
A12	16	A3
A13	17	A2
A14	18	A1
A15	19	A0
$\overline{MRQ}$	20	$\overline{WAIT}$
$\overline{RFSH}$	21	$\overline{ROMDS}$
$\overline{ROM1}$	22	$\overline{ROM0}$
$\overline{EXM}$	23	$\overline{ROM2}$
$\overline{MWR}$	24	$\overline{MRD}$
$\overline{IOWR}$	25	$\overline{IORD}$
$\overline{EXIOB}$	26	$\overline{EXIOA}$
EXCLK	27	$\overline{EXINT}$
Φ	28	$\overline{RST}$

OBR. 10 ADRESOVÁNÍ BLOKŮ POČÍTAČE

ROZDĚLENÍ PAMĚTI



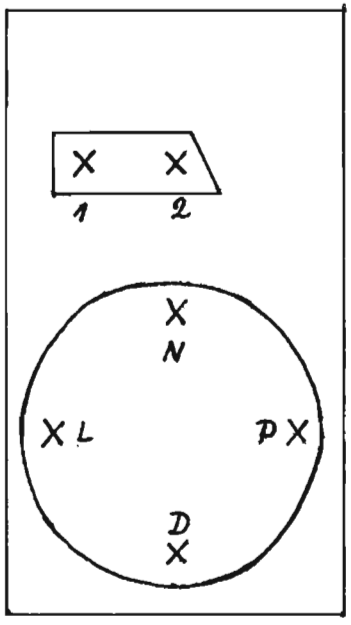
POZN: MODUL PŘÍDAVNÉ PAMĚTI 32KB RAM NELZE POUŽÍT S MODULEM BASIC-I

TABULKA OBSAZENÍ ADRES I/O BLOKŮ

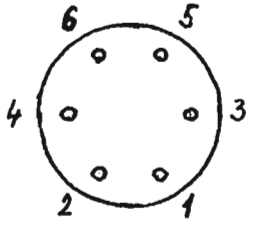
00	KANÁL 0	Z80A CTC
01	KANÁL 1	Z80A CTC
02	KANÁL 2	Z80A CTC
03	KANÁL 3	Z80A CTC
10		VIDEO PROCESOR
11		VIDEO PROCESOR
20	ZVUKOVÝ GENERÁTOR	
30	ŘADA 0	KLÁVESNICE
31	ŘADA 1	KLÁVESNICE
32	ŘADA 2	KLÁVESNICE
33	ŘADA 3	KLÁVESNICE
34	ŘADA 4	KLÁVESNICE
35	ŘADA 5	KLÁVESNICE
36	ŘADA 6	KLÁVESNICE
37	ŘÍZENÍ SMĚRU OVLADAČŮ	
40	TISKÁRNA	
50	TISKÁRNA, MAGNETOFON	
60 - 6F	VNĚJŠÍ PERIFERIE A (Z80 SIO)	
70 - 7F	VNĚJŠÍ PERIFERIE B (8255A)	
80 - FF	NENÍ VYUŽITO PRO SYSTÉM POČÍTAČE	

OBR. 11 INFORMACE O ZAPOJENÍ A PŘIPOJENÍ OVLADAČŮ

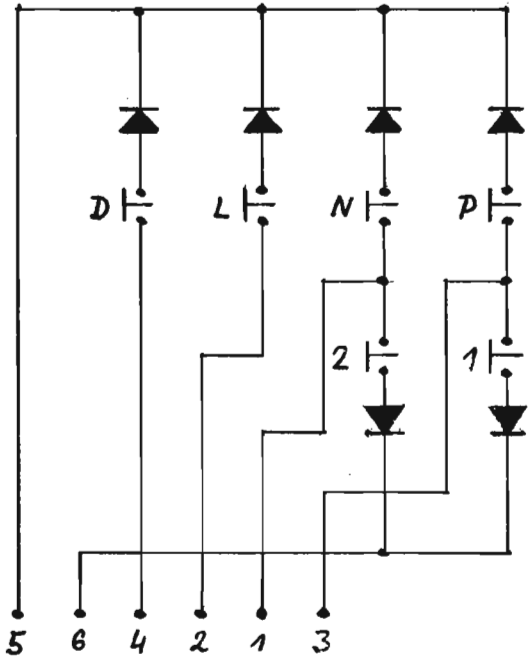
ROZLOŽENÍ TLAČÍTEK NA ZÁKLADNÍ DESCE OVLADAČE



POHLED NA KONEKTOR OVLADAČE V POČÍTAČI ZEZADU



SCHEMA VNITŘNÍHO ZAPOJENÍ OVLADAČE



ČTENÍ SMĚRU 37H INPUT

PRAVÝ				LEVÝ				
7	6	5	4	3	2	1	0	BIT
-Y	-X	Y	X	-Y	-X	Y	X	

ZAPOJENÍ KONEKTORŮ OVLADAČŮ

LEVÝ

1	2	3	4	5	6
c1	c2	c0	c3	R7	R1

PRAVÝ

1	2	3	4	5	6
c5	c6	c4	c7	R7	R1