OSOBNÍ POČÍTAČ

SORD M5

PŘÍRUČKA K PROGRAMOVACÍMU JAZYKU

BASIC-F

B A S I C - F

1. ZÁKLADNÍ VLASTNOSTI

2. PŘENOS DAT

3. PŘERUŠENÍ

4. GRAFIKA

5. PROGRAMOVÁNÍ

6. DOPLŇKY

Programovací jazyk BASIC-F výrazně zvyšuje užitnou hodnotu počítače SORD M5 a umožnuje ho využívat i pro profesionální účely. Příručka je značně upraveným a doplněným překladem originálního anglického manuálu. Navazuje přímo na "Návod k použití pro základní sestavu s programovacím jazykem BASIC-I". Uživatelé, kteří budou používat pouze jazyk BASIC-F, nemusí probírat části 3.04, 3.08, 4.03, 4.06 a 5.01, nebot BASIC-F používá v těchto částech jiné příkazy, než které jsou popsány pro BASIC-I. Na rozdíl od anglického originálu jsou vypuštěny texty a popisy příkazů týkajících se diskové jednotky. Pro diskovou jednotku bude v případě jejího prodeje napsán samostatný návod k použití.

Cílem příručky nebylo podrobně popsat všechny příkazy, funkce a možnosti jazyku BASIC-F. Tím by se její rozsah zvýšil nad únosnou mez. Hlavním účelem příručky je seznámit uživatele přehledně se základními vlastnostmi programovacího jazyku BASIC-F a na několika demonstračních programech naznačit jeho další možnosti. Pro dokonalé seznámení se všemi příkazy a funkcemi je potřeba si projít originální anglický manuál.

České příručky pro programovací jazyky BASIC-I a BASIC-F by nemohly vzniknout bez vzorného přepsání rukopisů, což zajistili ve vynikající kvalitě paní Hana Doležalová /BASIC-I/ a moje manželka Irma /BASIC-F/. Oběma chci aspoň tímto způsobem projevit svoji vděčnost.

Ing. Libor Štolc, duben 85

1.01 Využití paměti

Programový modul BASIC-F obsahuje čtyři paměťové integrované obvody- 2 paměti ROM o kapacitách 16 KB a 4 KB a 2 paměti RAM o výsledné kapacitě 4 KB. Společně s monitorem o kapacitě 8 KB, který je uvnitř počítače, je celková kapacita paměti ROM 28 KB. Dalších 8 KB je paměť RAM, z níž je 768 bytů vyhraženo pro systémové tabulky a proměnné monitoru. Podrobný popis této oblasti paměti i všech podprogramů monitoru je ve velice přehledné formě v příručce Monitor Handling Manual. Přídavným paměťovým modulem EM-5 je možno paměť RAM zvětšit na 36 KB a tím vlastně vyčerpat přímé adresové možnosti použitého mikroprocesoru Z80A. Paměť pro uchování informací o obrazovém výstupu o kapacitě 16 KB je zapojena jako vnější periferie a její adresace se proto neprovádí po adresové sběrnici.

Překladač jazyku BASIC-F potřebuje určitou část paměti RAM pro své systémové proměnné a pracovní oblasti. Informace o těchto systémových proměnných ani komentovaný výpis jazyku BASIC-F v jazyku symbolických adres zatím nejsou k dispozici. Na konci paměti je po zapnutí počítače vyhraženo 256 bytů jako pracovní oblast, která se využívá pro ukládání souřadnic bodů při použití příkazu PAINT a jako pracovní oblast pro práci s řetězcovými proměnnými.

Pod pracovní oblastí je uživatelská oblast pro ukládání programů a dat. Funkce FRE/X/ nám pro X od 0 do 4 přímo vypisuje informace o paměti RAM, v níž pracuje BASIC-F. Uvedené hodnoty platí po zapnutí počítače:

	BASIC-F	BASIC-F+EM-5	
PRINT FRE/0/	256	256	- maximální velikost pracovní oblasti
PRINT FRE/1/	5899	34571	- zbývající uživatelská oblast
PRINT FRE/2/	246	246	- zbývající pracovní oblast
PRINT FRE/3/	6145	34817	- zbývající uživatelská a pracovní oblast
PRINT FRE/4/	36863	65535	- poslední adresa pro BASIC-F

Nejdůležitějším údajem pro programátora je zbývající uživatelská oblast. Toto číslo nám říká kolik bytů máme ještě k dispozici pro program i pro uložení proměnných v průběhu programu. Jednořádkovým programem
10 INPUT N: DIM A/N/:PRINT FRE/1/ však zjistíme, že hodnota FRE/1/ nikdy nemůže být menší než 100. To znamená, že skutečně využitelná paměť RAM je 5799 při použití pouze modulu BASIC-F a 34471 při použití modulu BASIC-F s pamětovým modulem EM-5. Hodnoty FRE/0/ a FRE/4/ si můžeme programově změnit pomocí příkazu CLEAR, jestliže potřebujeme rezervovat blok paměti na jejím konci pro uložení programu ve strojovém kódu nebo jako zásobníkový blok paměti pro přesuny paměti VRAM. Prvním parametrem příkazu CLEAR je maximální velikost pracovní oblasti jako celé násobky 256 a druhým parametrem je poslední adresa, kterou ještě používá jazyk BASIC-F. Zde je vhodné připomenout, že i v takto rezervované oblasti paměti je možno s jazykem BASIC-F pracovat. Normálně zde fungují příkazy PEEK, POKE, CALL, SAVE a OLD.
V příkazu CLEAR, který můžeme využívat v režimu okamžitého výpočtu i v programu, není nutné zadávat vždy oba parametry. Stačí zadat jeden a druhý zůstane beze změny. Minimální hodnota velikosti pracovní oblasti musí být vždy 256. Jinak se ohlásí chybové hlášení číslu 5. Příkazem CLEAR se zlikvidují všechny proměnné, dokonce i pole v příkazu DIM.

1.02 Reálné a celočíselné proměnné, uložení v paměti

Základní odlišností jazyku BASIC-F od jazyků BASIC-I a BASIC-G je možnost pracovat s reálnými čísly s pohyblivou desetinou tečkou při aritmetických

operacích i při počítání s funkcemi. Celočíselné proměnné ale zůstaly zachovány i u jazyku BASIC-F. Pro odlišení jejich názvů od názvů reálných proměnných je název každé celočíselné proměnné nebo pole doplněn na konci o znak %.

Důležitými informacemi před vlastním počítáním s reálnými proměnnými je použitelný číselný rozsah a maximální dosažitelná přesnost. Tyto parametry jsou přímo dány způsobem uložení hodnot reálných proměnných v paměti.

Reálné proměnné/v dalším textu většinou pouze jako proměnné/ jsou uloženy v osmi bytech. První byte obsahuje informaci o znaménku a celočíselný exponent E v rozsahu -64 až +63. V dalších sedmi bytech může být uloženo $2^{M}-1$ různých čísel M, jejichž hodnota je určena součtem příslušných převrácených celočíselných mocnin čísla 2. Nejmenší hodnota čísla M může být 2^{-B} a maximální se blíží jedničce, takže pro M platí $2^{-B} \leq M < 1$. Hodnota proměnné P je potom určena výrazem $P=M^{B}$, kde B je 16. Princip uložení hodnot proměnných v paměti je tedy stejný jako u většiny počítačů. V následující tabulce jsou pro srovnání uvedeny parametry některých větších počítačů:

pocitac	pocet bitu M	B	-E	+E
SORD M5	56	16	-64	63
Hewlett Packard HP-45	10	10	-98	100
PDP-11	24	2	-128	127
Univac 1108	27	2	-128	127
IBM 360 (370)	14	16	-64	63
Control Data 6600	48	2	-976	1070

Názornou představu o uložení hodnot proměnných Vám poskytne program ULCISEL:

```
10 rem ULCISEL
20 rem *******
30 dim B$(16),H(8),L(8),D(8),C(56)
40 cls:print "ROZKLAD HODNOT REALNYCH CISEL":print
50 for I=1 to 56:C(I)=2^-I:next I
60 data 0000,0001,0010,0011,0100,0101,0110,0111
70 data 1000,1001,1010,1011,1100,1101,1110,1111
80 for I=1 to 16:read B$(I):next I
90 input "VYRAZ ";A$:C=calc(A$):if C=0 then stop else A=varptr(C)
100 print:B=0:K=1
110 for I=1 to 8:D(I)=peek(A+I-1)
120 if I>1 then goto 140 else W=D(1)
130 F=1:if D(1)<128 then W=W-64 else W=W-192:F=-1
140 D=D(I):DH=int(D/16):DL=D mod 16:C$=B$(DH+1)+B$(DL+1):print C$,D
150 if I=1 then goto 180
160 for J=1 to 8:W$=mid$(C$,J,1):if W$="1" then B=B+C(K)
170 K=K+1:next J
180 next I:D=16^W
190 print:if F=-1 then print C;"= -";B;"*";D else print C;"=";B;"*";D
200 print B*D*F:print:goto 90
```

Po zadání libovolného čísla nebo aritmetického výrazu následuje na osmi řádkách výpis jednotlivých bytů v binárním i dekadickém tvaru. Současně dochází k postupnému načítání převrácených mocnin dvou, které jsou předem vypočítány a uloženy v poli M/56/, do proměnné M v případě, že příslušný bit má hodnotu 1. Dále se vytiskne zadané číslo případně hodnota zadaného výrazu a o řádek níž hodnota určená programově. Obě čísla musí být stejná. Vpravo se vytisknou hodnoty jednotlivých složek, z nichž se hodnota zadaného čísla nebo vý-

razu skládá, v pořadí znaménko, základ M a exponent E. Při praktickém ověřování činnosti programu zjistíme, že minimální hodnota základu M je 0.065, což je možno dokázat po krátké úvaze i teoreticky. Následující tabulka, v níž jsou vytištěny výstupní údaje programu ULCISEL pro některá čísla a některé výrazy, je určena pro demonstraci výše uvedeného popisu pro uživatele počítače M5 i pro ostatní zájemce.

```
SIN(120)          COS(120)          PI                123456789         1
 0.8660254037844 -0.5               3.14159265359     123456789          1
 1                -1                1                 1                  1
 0.8660254037844   0.5              0.1963495408494   0.459912378341     0.0625
 1                 1                16                268435456          16
01000000 64       11000000 192      01000001 65       01000111 71        01000001 65
11011101 221      01111111 127      00110010 50       01110101 117       00010000 16
10110011 179      11111111 255      01000011 67       10111100 188       00000000 0
11010111 215      11111111 255      11110110 246      11010001 209       00000000 0
01000010 66       11111111 255      10101000 168      01010000 80        00000000 0
11000010 194      11111111 255      10001000 136      00000000 0         00000000 0
01100101 101      11111111 255      01011010 90       00000000 0         00000000 0
10011110 158      10001011 139      00110001 49       00000000 0         00000000 0
```

Na závěr této části ještě dvě poznámky k programu ULCISEL. Jistě jste sami poznali, že pro určení počáteční adresy proměnné slouží funkce VARPTR /název proměnné/. Jejím argumentem může být i název celočíselné proměnné nebo řetězcové proměnné. Pro výpočet hodnot aritmetických výrazů zadaných ve formě dané hodnoty řetězcové proměnné slouží funkce CALC /řetězcová proměnná nebo konstanta/. Funkce CALC funguje úplně stejně jako funkce VAL u počítačů Sinclair.

1.03 Rozsah, vyjádření s exponentem a přesnost proměnných

Vzhledem k tomu, že z předchozí části již známe způsob uložení proměnných v paměti, můžeme si nyní prakticky vyzkoušet mezní hodnoty číselného rozsahu proměnných. Dále uváděná pořadí bitů jsou počítána zleva ve shodě s výstupem programu ULCISEL. Maximální kladné číslo bude mít první bit prvního bytu nulový a ostatní bity všech bytů budou jedničky. Maximální záporné číslo bude mít všech osm bytů zaplněno jedničkami. Obě tato čísla nám vytiskne krátký program:

```
10 rem MAXIMUM
20 rem *******
30 C=0:A=varptr(C):input "+MAX=1, -MAX=0 ";F:poke A,255-F*128
40 for I=A+1 to A+8:poke I,255:next I:print C
```

Nejmenší kladné číslo bude mít všechny bity nulové vyjma posledního bitu prvního bytu a čtvrtého bitu druhého bytu. Nejmenší záporné číslo bude mít navíc znaménkový první bit prvního bytu. Hodnotu obou čísel nám vytiskne opět krátký program:

```
10 rem MINIMUM
20 rem *******
30 C=0:A=varptr(C):input "+MIN=0, -MIN=1 ";F:poke A,1+F*128
40 poke A+1,16:for I=A+2 to A+8:poke I,0:next I:print C
```

Mezi nejmenším kladným a záporným číslem je ještě nula. Má-li proměnná hodnotu nula, potom je všech osm bytů nulových.

Uložení čísel není rovnoměrné. Čísla jsou nahuštěna z obou stran k nule. Mezi nejmenším kladným číslem a číslem 0.0625 může být v počítači M5 uloženo stejné množství čísel, jako mezi číslem 0.0625 a největším číslem záporným.

Také mezi sousedními celými kladnými nebo zápornými mocninami šestnácti je stejný počet čísel, teoreticky 2 -1. Takže např. počet čísel v intervalu 1 až 16 je stejný jako v intervalu 16 až 256 nebo 1048576 až 16777216. Při výpočtech s velkými čísly se proto může stát, že výsledek nemůže být přesně uložen v paměti a musí být potom zaokrouhlen na nejbližší číslo, které je možno v paměti uložit.

Jistě jste si již všimli, že výsledky aritmetických operací nebo hodnoty funkcí jsou zobrazeny na 13 platných číslic. Jestliže číslo potřebuje na své vyjádření více než třináct číslic, provádí se výstup čísla v podobném tvaru, který je používán při výpočtech pomocí logaritmů. Tzv. mantisa má však hodnotu od jedné do deseti.V tomto tvaru se mohou číselné hodnoty používat v režimu okamžitého výpočtu nebo jako vstupní hodnota v příkazu INPUT. V následující tabulce jsou v levém sloupci ukázány všechny způsoby zadání hodnot proměnných nebo konstant v tvaru s exponentem. V prvním sloupci je demonstrován přechod na výstup čísla v tvaru s exponentem.

```
PRINT 1E4       10000              PRINT 1E13      1.E 13
PRINT 1.E5      100000             PRINT 1E14      1.E 14
PRINT 1.0E6     1000000            PRINT 1E15      1.E 15
PRINT 1E 7      10000000           PRINT 1E16      1.E 16
PRINT 1.E 8     100000000
PRINT 1.0E 9    1000000000         PRINT 1E-12     0.000000000001
PRINT 1E+10     10000000000        PRINT 1E-13     0.0000000000001
PRINT 1.E+11    100000000000       PRINT 1E-14     1.E-14
PRINT 1.0E+12   1000000000000     PRINT 1E-15     1.E-15
```

Jistě Vás nyní napadne otázka, jestli je maximální počet platných číslic hodnot proměnných třináct, jak je zobrazováno počítačem, nebo více. Odpověď Vám dá další program, který snad nepotřebuje komentáře. Jeho výstupní hodnoty následují hned za programem.Je z nich názorně vidět, že v paměti počítače jsou hodnoty proměnných uloženy až na šestnáct platných číslic. Poslední číslice je samozřejmě zaokrouhlena.

```
10 rem PRESNOST
20 rem ********
30 A=9999999999999
40 for I=0 to 5
50 for J=1 to 9:B=J/10^I
60 print B;(A+B)-A
70 next J:print:if I=5 then goto 90
80 if inkey$="" then goto 80
90 next I
 1  1                    0.1  0.099853515625      0.01  0.009765625
 2  2                    0.2  0.199951171875      0.02  0.019775390625
 3  3                    0.3  0.2998046875        0.03  0.02978515625
 4  4                    0.4  0.39990234375       0.04  0.039794921875
 5  5                    0.5  0.5                 0.05  0.0498046875
 6  6                    0.6  0.599853515625      0.06  0.059814453125
 7  7                    0.7  0.699951171875      0.07  0.06982421875
 8  8                    0.8  0.7998046875        0.08  0.079833984375
 9  9                    0.9  0.89990234375       0.09  0.08984375
0.001  0.0009765625      0.0001  0                 0.00001  0
0.002  0.001953125       0.0002  0                 0.00002  0
0.003  0.0029296875      0.0003  0.000244140625    0.00003  0
0.004  0.00390625        0.0004  0.000244140625    0.00004  0
0.005  0.0048828125      0.0005  0.00048828125     0.00005  0
0.006  0.005859375       0.0006  0.00048828125     0.00006  0
0.007  0.0068359375      0.0007  0.00048828125     0.00007  0
0.008  0.0078125         0.0008  0.000732421875    0.00008  0
0.009  0.0087890625      0.0009  0.000732421875    0.00009  0
```

Při velkém počtu aritmetických operací dochází k chybám, které jsou způsobeny tím, že příslušný mezivýsledek leží mezi dvěma čísly, která se dají uložit přesně v paměti a leží na "číselné ose počítače" těsně za sebou. Vzhledem k tomu, že výsledné hodnoty mají minimálně třináct platných číslic, je možné i po zaokrouhlení na menší počet platných číslic dosáhnout značné přesnosti, která je u většiny osobních počítačů nedosažitelná. Následující program Vám umožní názorně sledovat,jak chyby vznikají. Na začátku programu se vytvoří čtvercové pole A/N,N/, jehož dimenzi zadáváte pomocí náhodných čísel. Pole A/N,N/ tvoří čtvercovou matici, která se v jednom cyklu dvakrát invertuje. Vzhledem k tomu, že podprogram na inverzi matice provádí inverzi do původního pole, musíme teoreticky dostat po dvojím invertování původní pole A/N,N/. Počet těchto cyklů s dvojím invertováním je dalším vstupním údajem. Při každém cyklu se výsledné pole vypisuje na obrazovku, kde je stále vidět původní pole, takže je možno podtupně sledovat, jak dochází k zvětšování chyb nebo k chybám dalším. Při praktickém zkoušení programu nezadávejte rozměr pole A větší než osm.

1.04 Rychlost výpočtů

Zajímavým parametrem, který lze použít pro porovnání osobních počítačů s jazyky BASIC, je jejich rychlost. Komplexní porovnání je možné pouze na řadě stejných delších programů pro všechny srovnávané počítače. Pro orientační a rychlé srovnání se používá v anglických časopisech při testech a popisech nových počítačů osm krátkých programů. Výpisy těchto programů jsou v článku " Srovnání rychlosti některých osobních počítačů", uveřejněném v časopisu Sdělovací technika 1/85. V článku je i srovnávací tabulka s dobami všech osmi testovacích programů u deseti počítačů. Pro zajímavost byly změřeny časy běhů testovacích programů i u počítače M5 s těmito výsledky: 1 - 2s, 2 - 3.6s, 3 - 14s, 4 - 15.2s, 5 - 15.8s, 6 - 24.6s, 7 - 44.2s, 8 - 302s. Porovnáním s tabulkou ve výše citovaném článku zjistíme, že v prvních sedmi testovacích programech je počítač M5 rychlejší než všech deset počítačů. Z počítačů tuzemských to jsou počítače TNS /JZD Slušovice/ , IQ-151 /ZPA Nový Bor/, PMD-85 /Tesla Pieštany/, PP-01 /VÚVT Žilina/, SAPI 1 /Tesla Liberec/. Zahraniční počítače jsou zastoupeny typy HP-85A /Hewlett Packard/, MCS-80 /Intel/, ZX Spectrum /Sinclair Research/ a TI-99/4A /Texas Instruments/. Pomalejší než devět počítačů je M5 v posledním osmém testu, který používá funkce LOG/X/ a SIN/X/. Většina z uváděných počítačů má však přesnost pouze šesti platných číslic. Velice rozšířený počítač ZX Spectrum má přesnost osm platných číslic a doba posledního testu je 239.3s proti 302s počítače M5, takže i v tomto testu je při přepočtu na jednu platnou číslici počítač M5 rychlejší /302/13 proti 239.3/8/. V průměrné době ze všech osmi testů předčí M5 dobou 52.675s vedle počítače ZX Spectrum /55.74s/ i počítače SM 50/40 /62.13s/ a PP-01 /55.15s/.

Rychlost výpočtů v jazyku BASIC lze zvýšit výrazně pouze použitím překladače jazyku BASIC. Počítač SORD M5 je navržen tak, že je možné technicky i programově jej přizpůsobit potřebám uživatele. Je tedy reálná naděje, že jednou bude vedle dalších programovacích jazyků k dispozici překladač jazyku BASIC-F. Ovšem k tomu by určitě byla nutná i přídavná paměť 32 KB RAM.

1.05 Řetězcové proměnné

Řetězcové proměnné mají po zapnutí počítače nastavenu délku osmnácti znaků. Platí to i pro pole řetězcových proměnných, o čemž se přesvědčíme krátkým programem:

```
10 dim A$(10)
20 for I=1 to 10:print varptr(A$(I)):next I
```

Z rozdílů adres začátků jednotlivých prvků pole A$/10/ale vidíme, že jejich délka je devatenáct znaků. První byte, jehož adresa je přímo určena hodnotou funkce VARPTR/A$/I//, totiž obsahuje počet znaků daného prvku pole. V případě všech prázdných prvků pole po příkazu DIM A$/10/ budou hodnoty všech prvních bytů prvků pole A$/10/ nulové.

Pole A$/10/ je jednorozměrné. Počet dimenzí, to znamená počet indexů v závorce za názvem pole může být až 255.

Příkazem LEN D si můžeme délku prvků pole A$/10/ nastavit od jednoho znaku do 255 znaků. Stačí doplnit do výše uvedeného programu řádek
5 INPUT "DELKA PRVKU POLE A$/10/"; D: LEN D
Příkaz LEN D nastavuje maximální délku všech řetězcových proměnných, které jsou použity po příkazu LEN poprvé, až do dalšího příkazu LEN D. Stejnou zkratku má i funkce LEN/ / pro zjištění délky řetězcové proměnné, jejímž argumentem je jméno řetězcové proměnné nebo jméno řetězcového pole i s příslušnými indexy.

Názvy všech řetězcových i číselných proměnných musí začínat vždy nečíselným znakem a mohou mít maximální délku 32 znaků. To je pouze doplnění pro ty zájemce a uživatele, kteří si ještě nepřečetli základní příručku pro BASIC-I.

1.06 Číselná pole, přesuny do paměti VRAM

Při použití polí reálných nebo celočíselných proměnných je jediným omezením při jejich použití velikost volné paměti RAM. Maximální počet dimenzí je totiž 255. Maximální počet prvků v jedné dimenzi může být teoreticky až 65535 a může to být konstanta nebo proměnná.

Jestliže nepoužíváme obrazovkový režim GII, je prvních 8 KB paměti VRAM nevyužito. Tuto paměť proto můžeme používat pro přechodné uchování polí číselných i řetězcových a tím si zvětšujeme využitelnou paměť pro data. Pro rychlý přesun reálných a celočíselných polí jsou dále uvedeny čtyři podprogramy:

1. - $RPVRAM - přesun pole reálných proměnných do paměti VRAM
2. - $VRAMRP - přesun pole reálných proměnných z paměti VRAM
3. - $CPVRAM - přesun pole celočíselných proměnných do paměti VRAM
4. - $VRAMCP - přesun pole celočíselných proměnných z paměti VRAM

Vstupní parametry všech čtyř podprogramů jsou stejné:
PP$ - označení prvního prvku pole
N - počet prvků pole
Jedinou vnitřní proměnnou je proměnná A, která označuje počáteční adresu paměti VRAM, od které se číselné pole ukládá. Její hodnota je nulová, takže je možno do paměti VRAM přesouvat pouze jediné pole. Jestliže proměnnou A vypustíme z podprogramů a šikovně ji používáme jako další vstupní parametr, můžeme do paměti VRAM přesunovat i několik číselných polí. Uživatelé si jistě následující podprogramy upraví i pro přesuny řetězcových polí.

```
1000$RPVRAM
1010 A=0:exe("Z=VARPTR("+PP$+")"):call &0E61,,N*8,A,Z:return
1020$VRAMRP
1030 A=0:exe("Z=VARPTR("+PP$+")"):call &0E7D,,N*8,Z,A:return
1040$CPVRAM
1050 A=0:exe("Z=VARPTR("+PP$+")"):call &0E61,,N*2,A,Z:return
1060$VRAMCP
1070 A=0:exe("Z=VARPTR("+PP$+")"):call &0E7D,,N*2,Z,A:return
```

Příkazy EXE a CALL budou vysvětleny později v příslušných částech příručky. Použití dvou uvedených podprogramů si ukážeme na krátkém demonstračním programu, jehož činnost je komentována přímo v průběhu programu:

```
10 rem PRESUN
20 rem ******
30 dim A(500)
40 print "":fcol 1:bcol 14
50 print "GENEROVANI POLE A(500)"
60 for I=i to 500:A(I)=I*I:next I
70 print "KONEC GENEROVANI POLE A(500)"
80 print "ZACATEK PRESUNU DO VRAM"
90 PP$="A(1)":N=500:gosub $RPVRAM
100 print "KONEC PRESUNU DO VRAM"
110 clear:dim B(50),A(500):print "VYNULOVANI POLE A(500)"
120 print "ZACATEK PRESUNU Z PAMETI VRAM"
130 PP$="A(1)":N=500:gosub $VRAMRP
140 print "KONEC PRESUNU Z PAMETI VRAM"
150 print "ZACATEK TESTU SPRAVNOSTI PRESUNU"
160 for I=1 to 500:if A(I)<>I*I then stop else next I
170 print "KONEC TESTU - PRESUN V PORADKU":stop.
```

1.07 Přehled matematických funkcí

Vedle aritmetiky s plavoucí desetinnou tečkou je další výraznou změnou jazyku BASIC-F proti jazykům BASIC-1 a BASIC-G možnost použití všech základních matematických funkcí. Uložení hodnot všech dále uváděných funkcí je stejné jako u reálných proměnných, to znamená v osmi bytech.

U goniometrických funkcí je možno příkazem THETA volit práci v radiánech nebo ve stupních /THETA 0 - radiány, THETA 1 - stupně/. Po zapnutí počítače se pracuje s radiány. Ke goniometrickým funkcím SIN, COS, TAN, ATN je možno počítat i konstantu PI.

Dalšími všeobecně známými a používanými funkcemi jsou:
ABS - absolutní hodnota
EXP - exponenciální funkce
INT - celočíselná část argumentu
LN - přírodní logaritmu /základ e/
LOG - desítkový logaritmus /základ 10/
RND - náhodné číslo
SQR - druhá odmocnina

1.08 Současné použití celočíselných a reálných proměnných

Názvy celočíselných proměnných se na konci doplňují znakem %. Platí to i o celočíselných konstantách. Použití celočíselných proměnných přináší v programech výraznou úsporu paměti i zrychlení programů. Následující dva programy názorně demostrují zrychlení průběhu prázdného cyklu při použití řídící proměnné I%. Doba běhu programu TEST1 je dvanáct vteřin a doba běhu programu TEST2 čtyřicet sedm vteřin, to znamená zrychlení na čtvrtinu původní doby.

```
10 rem TEST1
20 rem *****
30 T=time:for I%=1% to 30000%:next I%:print time-T

10 rem TEST2
20 rem *****
30 T=time:for I=1 to 30000:next I:print time-T
```

Při použití celočíselných proměnných a konstant v aritmetických výrazech společně s reálnými proměnnými a funkcemi platí dvě základní omezení:
a/ hodnota celočíselné proměnné nebo velikost celočíselné konstanty musí být v rozsahu od -32768 do +32767
b/ nelze samostatně nebo jako součást aritmetického výrazu použít dělení dvou celočíselných proměnných
Ve všech ostatních případech lze celočíselné a reálné proměnné i konstanty používat současně. Výslednými hodnotami jsou vždy reálné proměnné.

Na straně 104 originálního manuálu je uvedena funkce CDBL, která je zcela zbytečná. Převod na reálnou proměnnou se totiž provede mnohem jednodušeji vynásobením celočíselné proměnné nebo konstanty jedničkou /např. A%=1000%:A=1xA%/. Na straně 109 je uvedena funkce CINT pro převod reálných čísel na celá čísla se zaokrouhlením. Tato funkce je velice užitečná, pouze se musí při jejím použití dát pozor na překročení číselného rozsahu celočíselných proměnných.

2.01 Přehled periferních zařízení

K počítači M5 je možno přímo připojit několik periferních zařízeních. Nejpoužívanějším zařízením vedle TV přijímače je kazetový magnetofon. Pro výpis programů a dat je možno použít buď tiskárnu PT-5 firmy SORD nebo kteroukoliv jinou s paralelním stykem podle standartu Centronics. Přes paralelní stykový modul PI-5 je možno připojit jednu nebo dvě diskové jednotky FD-5, které jsou také výrobkem firmy SORD. Pro usnadnění programování přenosu dat s těmito periferními zařízeními používá počítač M5 přenos dat v jednotné formě pojmenovaných souborů. Tento způsob se používá prakticky u všech větších profesionálních počítačů. Pro rozlišení jednotlivých periferií se používají následující zkratky:

	výstupní zařízení	vstupní zařízení
CNS /Console/	znaková obrazovka	klávesnice
GRP /Graphic/	grafická obrazovka	
PRT /Printer/	znaková tiskárna	
PRI /Image/	grafická tiskárna	
CMT /Cassette/	kazetový magnetofon	
FX /File/	disková jednotka	

Vzhledem k tomu, že tato příručka nepopisuje práci s diskovou jednotkou, nebude zde ani vysvětlen způsob vytváření souborů pro disketu, jak je stručně objasněno v originálním manuálu.

2.02 Otevření souboru pro přenos dat

Z předchozí části již víme, že každý soubor musí mít svůj název. Jeho maximální délka je devět znaků. Pro přesné určení periferie je třeba příslušnou zkratku spojit nějakým způsobem s názvem souboru. To se udělá velice jednoduše tak, že se za zkratku periferie napíše dvojtečka a za ní název souboru. Informaci o názvu souboru a příslušné periferii si počítač zapamatuje při tzv. otevření souboru pomocí příkazu OPEN. V příkazu OPEN musí být zadán ještě další důležitý parametr, kterým je číslo programového přenosového kanálu o rozsahu 0 až 15. Číslo kanálu totiž umožnuje současnou práci s několika soubory. Pohyb datového souboru z hlediska počítače může být dvojí; buď se jedná o výstupní soubor nebo o soubor vstupní. I tato informace musí být vyjádřena v příkazu OPEN. V následujících typických případech si názorně ukážeme otevírání souborů:

a/ OPEN "CMT:DATA" FOR OUTPUT AS 1 - otevření souboru pro záznam dat na kazetový magnetofon
b/ OPEN "PRT:VYPIS" FOR OUTPUT AS 3 - otevření souboru pro výpis dat na tiskárně
c/ OPEN "CMT:SEZNAM" FOR INPUT AS 5 - otevření souboru pro čtení dat z kazetového magnetofonu

Pro zjednodušení práce s obrazovkou a klávesnicí je pro obrazovku a klávesnici vyhraženo trvale číslo kanálu 0 a není nutno při běžných výpisech a tiscích na obrazovku provádět otevírání souboru příkazem OPEN.

2.03 Výstup a vstup dat

Pro výstup a vstup dat se používají příkazy PRINT a INPUT doplněné označením čísla příslušného kanálu. Za klíčovými slovy PRINT nebo INPUT s číslem kanálu následuje seznam proměnných. V seznamu příkazu PRINT mohou být i konstanty. Při čtení dat příkazem INPUT musí seznam přesně odpovídat

seznamu v příkazu PRINT, pomocí něhož byla data na vnější periferii /kazetový magnetofon, disková jednotka/ zaznamenána. Jednotlivé položky seznamu v příkazu PRINT mohou být odděleny čárkami nebo středníky. Při záznamu dat na pásku je výhodnější používat středníky, neboť při použití čárek se zbytečně na pásku zaznamenávají i mezery. Vzhledem k jednotnému způsobu přenosu dat se i na pásku magnetofonu zaznamenávají číselná data po jednotlivých číslicích. Každá číslice potom zabírá jeden přenosový byte, v němž je reprezentována svým ASCII kodem. Po skončení přenosu dat je potřeba oznámit počítači jeho ukončení. To se provede tzv. uzavřením souboru příkazem CLOSE s parametrem, který označuje číslo příslušného kanálu. Jako názorná ukázka je dále program na záznam kvadrátů celých čísel od jedné do sta na pásku a jejich zpětné čtení s kontrolním výpisem na obrazovku:

```
10 rem DATA 1
20 rem ******
30 dim A(100):open "CMT:DATADEMO1" for output as#1
40 for I=1 to 100:print#1 I*I:next I
50 close#1
60 print "KONEC NAHRAVANI, PRETOCTE MAGNETOFON":stop
70 open "CMT:DATADEMO1" for input as#1
80 for I=1 to 100:input#1 A(I):next I
90 close#1
100 for I=1 to 100:print I,A(I):next I
```

Pro šťastné uživatele počítače M5 s připojenou tiskárnou následuje demonstrační program pro tisk malé násobilky. Program se dá velice jednoduše upravit pro tisk na obrazovku nebo pro záznam na magnetofonovou pásku.

```
10 rem DATA 2
20 rem ******
30 open "PRT:" for output as#1
40 for R=1 to 10:print#1 R;
50 for S=1 to 10:print#1 tab(5*S+2);R*S;:next S
60 print#1:next R:close#1
```

2.04 Současný přenos dat na více periferií

V části 2.02 již byla zmínka o současném přenosu dat mezi počítačem a několika periferními zařízeními. Jedná se spíše o přenos střídavý, neboť v určitém časovém okamžiku se může provádět přenos dat pouze mezi počítačem a jedním periferním zařízením. Následující program Vám tuto možnost názorně předvede. Jedná se o poněkud upravený program DATA1 tak, aby se data zaznamenávaná na magnetofonovou pásku a potom z ní opět čtená vypisovala průběžně na obrazovce. Zde je třeva poznamenat, že pro vyzkoušení tohoto programu mohou být použity pouze magnetofony s vyvedeným dálkovým ovládáním. Záznam i čtení dat totiž probíhá po jednotlivých blocích, mezi nimiž dojde k zastavení pásku, aby bylo možno provést výpis na obrazovku.

```
10 rem DATA 3
20 rem ******
30 open "CMT:DATADEMO1" for output as#1
40 for I=1 to 100:print#1 I*I:print I,I*I:next I
50 close#1
60 print "KONEC NAHRAVANI, PRETOCTE MAGNETOFON":stop
70 open "CMT:DATADEMO1" for input as#1
80 for I=1 to 100:input#1 A:print I,A:next I
90 close#1
```

Majitelé počítače M5 s připojenou tiskárnou si mohou program DATA3 doplnit podle následujícího výpisu pro tisk dat. Navíc je doplněn přesun ukazatele při tisku otazníků na druhou obrazovku.

```
10 rem DATA 4
20 rem ******
30 open "PRT:" for output as#2
40 open "CMT:DATADEMO1" for output as#1
50 for I=1 to 100:A=I*I
60 Print#1 A:print I,A:Print#2 A;:next I:Print#2
70 close#1
80 Print "KONEC NAHRAVANI, PRETOCTE MAGNETOFON":stop
90 open "CMT:DATADEMO1" for input as#1
100 for I=1 to 100:Print "■":input#1 A:Print "■";I,A
110 Print#2 A;:next I:Print#2
120 close#1
130 close#2
```

2.05 Záznam číselných a alfanumerických polí na pásku

Vzhledem k tomu, že seznam v příkazu PRINT může být libovolný, je možno současně zaznamenávat na magnetofonovou pásku čísla i alfanumerické řetězce. Přitom se může délka řetězců měnit. Názorně si to můžeme ukázat na dalším programu, ve kterém dochází společně s čísly k záznamu dvaceti šesti řetězců různé délky. Každý řetězec obsahuje opakovaně pouze jedno písmeno abecedy.

```
10 rem DATA 5
20 rem *****
30 open "CMT:DATADEMO2" for output as#1
40 len 100:print#1 26:for I=1 to 26:A=int(rnd(100)+1)
50 W$=rpt$(A,chr$(64+I))
60 print I,A:print W$
70 print#1 I,A,W$
80 next I:close#1
90 print "KONEC NAHRAVANI, PRETOCTE MAGNETOFON":stop
100 open "CMT:DATADEMO2" for input as#1
110 input#1 N:for I=1 to N:input#1 K,L,W$
120 print K,L:print W$:next I:close#1
```

Při záznamu číselných polí mohou zabírat jednotlivé prvky pole třináct i více přenosových bytů. Vedle třinácti platných číslic může mít číslo ještě desetinnou tečku, záporné znaménko případně i exponent. Přitom v paměti počítače jsou číselné proměnné uloženy pouze v osmi bytech. To znamená, že při záznamu číselných polí se může délka záznamu až zdvojnásobit proti záznamu bloku paměti s číselným polem. V těchto případech je potom výhodnější provést přesun číselného pole do paměti VRAM a na pásku zaznamenat přímo příslušný blok paměti VRAM. Po zpětném přečtení příkazem OLD je třeba provést zpětný přesun číselného pole na původní místo v paměti RAM. Příslušné podprogramy jsou uvedeny v části 1.06 a jsou také použity v demonstračním programu PRESUN. Pro záznam bloku paměti VRAM slouží příkaz VSAVE se třemi parametry - názvem souboru, první adresou bloku, poslední adresou bloku /např. VSAVE "POLE A",0,799/. Blok paměti RAM s číselným polem by bylo možné zaznamenat na pásku přímo z paměti RAM /pomocí příkazu SAVE se stejnými parametry jako VSAVE/. Při zpětném čtení by se ale blok paměti objevil na stejném místě jako před záznamem a to pravděpodobně nebude souhlasit s umístěním číselného pole. Proměnné a pole jsou totiž v paměti uloženy až za programem a neustále dochází k jejich posunu.

2.06 Výpis programů na periferní zařízení

Příkaz LIST provádí výpis programu na obrazovku znaky malé abecedy. Příkaz CLIST provádí výpis velkými písmeny. Při delších výpisech je možno výpis zastavit stisknutím tlačítka SPACE. Po dalším stisknutí výpis pokračuje. Při přidání jednoho nebo dvou čísel řádků do příkazu LIST dochází pouze k výpisu části programu jako v jazyku BASIC-I. Jestliže chceme provádět výpis na tiskárnu, musíme do příkazu LIST přidat zkratku tiskárny. Příkazem LIST "PRT:" dostaneme výpis na běžné znakové tiskárny bez inverzních řídících znaků. Ty se objeví pouze po příkazu LIST "PRI:" na tiskárně PT-5 nebo na jiné tiskárně s možností grafického výstupu. Použijeme-li v příkazu LIST zkratku pro kazetový magnetofon a název programu, dostaneme výpis programu ve znakové podobě na magnetofonovou pásku /např. LIST "CMT:PROG 1"/. Příkazem OLD "PROG 1" se program nahraje zpět do paměti počítače. Doba záznamu i nahrávání je však výrazně delší než při nahrání programu příkazem SAVE. Velkou výhodou programů zaznamenaných na pásce ve znakové podobě je to, že při postupném nahrávání těchto programů příkazem OLD nedochází k jejich vymazání z paměti počítače. To umožnuje sestavovat program v paměti počítače z programů dílčích.

2.07 Rychlost záznamu na pásku

Průměrná rychlost záznamu na pásku je 1900 bitů za sekundu. Průměrná proto, že bit o hodnotě nahrává jednou periodou poloviční frekvence než bit o hodnotě 1. Rychlosti 1900 bitů za vteřinu potom odpovídají frekvence 2850 Hz a 1425Hz. Rychlost záznamu je určena hodnotou systémové proměnné STDLY /ACMT baud rate factor/ na adrese &7019 /28697/. Po zapnutí počítače je na adrese &7019 hodnota 33. Její změnou je možno rychlost záznamu na pásku měnit. Následující program vypočítává průměrné přenosové rychlosti pro hodnoty systémové proměnné STDLY od 2 do 33:

```
10 rem ZAZN RYCH
20 rem ********
30 for N=13 to 33
40 B=(1/(0.00003536+N*0.000008929)+2/(0.00004971+2*N*0.000008929))/3
50 print N,cint(B)
```

Výrobce doporučuje používat rychlost záznamu od 1170 bitů za sekundu do 4000 bitů za sekundu. Podle jeho informací jsou běžné kazetové magnetofony schopné bezpečně zaznamenat programy a data v rozmezí rychlostí 1170 až 2830 bitů za sekundu. Každý uživatel si musí na svém magjetofonu sám otestovat optimální maximální rychlost záznamu. Stačí postupně snižovat hodnotu na adrese &7019 podle požadovaných rychlostí záznamu a pokaždé několikrát nahrát delší program nebo blok paměti. Testování správnosti záznamu i zpětného čtení se provede příkazem VERIFY nebo OLD. Při zpětném nahrávání rychleji nahraných programů nebo dat není třeba zadávat příslušnou hodnotu na adrese &7019, neboť počítač si sám tuto hodnotu zjistí a použije ji při nahrávání. Na adrese &7019 přitom nedojde ke změně.

2.08 Struktura záznamu na pásku

Stejně jako řada dalších technických a programových vlastností počítače M5, odpovídá i záznam na pásku svojí strukturou a vlastnostmi spíše větším profesionálním počítačům než běžnému osobnímu počítači. Podrobně je popsán v šesté

části příručky Monitor Handling Manual. Záznam se skládá z identifikačního bloku a z datových bloků o délce 256 bytů. Na závěr záznamu je blok EOF /end of file/. Na závěr každého datového bloku je součet všech bytů bloku, který slouží pro kontrolu správnosti zpětného čtení do počítače. Před identifikačním blokem je zaznamenána konstantní frekvence, která slouží pro nastavení automatiky magnetofonu při nahrávání a pro zjištění rychlosti záznamu při zpětném čtení. Konstantní frekvence je i v mezerách mezi jednotlivými datovými bloky. Délka těchto mezer je při záznamu dat příkazem PRINT větší než při běžném záznamu programů a pamětových bloků. Je to proto, aby se mohly číst jednotlivé datové bloky. Následující program nebo spíše podprogram slouží k přečtení jednoho identifikačního bloku souboru na pásce a k výpisu nejdůležitějších informací z něho.

```
10 rem HLAVICKA
20 rem ********
30 len 32:A$=rpt$(32,"0")
40 A=varptr(A$)+1
50 call &1598,,,,A
60 F%=peek(A)
70 L%=F% and 224%
80 if L%=128% then print "BASIC-F"
90 if L%=96% then print "BASIC-G"
100 if L%=64% then print "FALC"
110 if L%=32% then print "BASIC-I"
120 M%=F% and 8%
130 if M%=0% then print "RAM" else print "VRAM"
140 print "JMENO SOUBORU ";
150 print mid$(A$,2,9):print
160 print "NAHRAVACI ADRESA ";peekw(A+10):print
170 print "DELKA SOUBORU ";peekw(A+12):print
```

3.01 Přerušení programu

Počítač M5 je jedním z mála osobních i profesionálních počítačů, u kterých je možno pracovat s tzv. přerušením. Řada majitelů počítače M5 bude jistě něco vědět o přerušení v mikropočítačových systémech, kde se požívá hlavně pro urychlení přenosu dat mezi mikropočítačem a vnějšími zařízeními vstupu. U těchto systémů funguje přerušení tak, že po oznámení vnějšího zařízení pomocí aktivního signálu na vstupu pro indikaci přerušení se v co nejkratší době dokončí vykonání právě prováděné instrukce a program si odskočí na podprogram pro obsloužení příslušné periferie. Při tomto odskočení se do zásobníkové paměti uloží adresa následující instrukce hlavního programu, aby program mohl opět pokračovat po skončení podprogramu pro obsloužení přerušení. U jednoúčelových mikropočítačových systémů pracujících podle programu ve strojovém kódu se přerušení používá celkem běžně. Přerušení ve vyšším programovacím jazyku jako je BASIC se začalo v oblasti osobních počítačů používat až u počítačů MSX v roce 1984, u nichž slouží podobně jako v jazyku BASIC-G počítače SORD M5 pro efektivnější a jednodušší programování počítačových her.

Princip přerušení spočívá v tom, že na nějaký vnější podnět /klávesnice, ovladače/ nebo z počítače /chyba, časový interval, setkání dvou sprajtů/ se program na okamžik přeruší a vykoná se předem zadaný podprogram. Po jeho skončení pokračuje hlavní program dál.

3.02 Přerušení při chybě

Dojde-li při práci s textem na obrazovce v režimu okamžitého výpočtu nebo v průběhu programu k jakékoliv chybě z hlediska systému počítače, okamžitě se zastaví provádění posledního příkazu a na obrazovku se vypíše chybové hlášení. Chybové hlášení obsahuje kód chyby a číslo řádku programu, ve kterém k chybě došlo. Kód chyby se současně uloží do proměnné ERR a číslo řádku s chybou do proměnné ERRL. Jestliže na začátku programu použijeme příkaz ON ERROR GOSUB s určením začátku podprogramu /číslem řádku nebo návěštím/, nedojde k zastavení programu a k výpisu chybového hlášení, ale k odskoku na námi určený podprogram. V tomto podprogramu můžeme využít hodnoty ERR a ERRL pro rozhodnutí jak pokračovat v programu dál podle druhu chyby. Místo příkazu RETURN je výhodnější používat příkaz RESUME, který má dva významy. Při jeho provedení se jednak nastavují hodnoty ERR a ERRL na nuly, čímž vlastně přestává chyba pro počítač existovat, a současně se může provést skok na libovolný řádek programu, je-li v příkazu RESUME uveden. Jestliže číslo řádku nebo návěští není uvedeno, pokračuje se dalším příkazem a pro návrat z podprogramu je třeba použít příkaz RETURN.

Velice vhodný příklad na použití příkazů ON ERROR GOSUB a RESUME je na straně 59 originálního manuálu. K příkladu stačí pouze dodat to, že chyba 25 nastane tehdy, jestliže se při vstupu dat v příkazu INPUT stiskne pouze tlačítko RETURN. Využití přerušení při chybě umožnuje programátorovi rychlejší odladování programů i některé nové postupy při tvorbě aplikačních programů. Z praktických zkušeností je možno doporučit využívání druhé obrazovky zvláště při práci s jemnou grafikou v režimu GII na první obrazovce.

3.03 Přerušení po časovém intervalu

Díky použitému časovači Z80 CTC na základní desce počítače je možno programově využívat přerušení po zadaném časovém intervalu. Příkazem ON EVENT GOSUB se určuje číslo řádku nebo návěští obslužného podprogramu. Časový interval se zadává příkazem EVENT, který má dva parametry. Prvním parametrem

je jednobytová hodnota /0 až 255/, která určuje násobek 1/50 vteřiny jako délku časového intervalu. Minimální délka je tedy 1/50 vteřiny a maximální délka 256/50 vteřin. Nulová hodnota totiž představuje hodnotu 256. Druhým parametrem je dvoubytová hodnota /0až 32767/. Hodnota 0 představuje podobně jako u prvního parametru hodnotu 32768. Toto číslo je násobek 1/50 vteřiny jako délka časového intervalu, který uplyne od příkazu EVENT do prvního přerušení. Pro povolení časového přerušení je třeba použít ještě příkaz EVENT ON. Příkazem EVENT OFF se přerušení zakazuje. Jednou z jednoduchých aplikací časového přerušení představuje následující program, který simuluje digitální stopky s průběžným měřením časů:

```
10 rem STOPKY
20 rem ******
30 rem (PROGRAM BYL NAPSAN POUZE PRO DEMOSTRACNI UCELY BEZ O
VERENI PRESNOSTI)
40 cls:Print "ZACATEK MERENI PO STISKNUTI":Print:Print "LIBO
VOLNE KLAVESY"
50 if inkey$="" then goto 50
60 event 5,50:on event gosub #DESETINA:event on
70 cls:T=9:R=1:N=1
80 if inkey$="" then goto 80
90 T1=T:Print cursor(10,R);N;T1/10:if R<20 then R=R+1 else P
rint "■"
100 N=N+1
110 goto 80
120#DESETINA
130 T=T+1:Print cursor(0,0);T/10:return
```

4.01 Úvod ke grafickým příkazům

Základní grafické příkazy jazyku BASIC-F jsou stejné jako grafické příkazy jazyku BASIC-I. Není tedy nutné v této příručce opakovat to, co je napsáno v příručce pro BASIC-I. Proto je třeba, aby uživatelé před dalším pokračováním již znali systém uložení grafických informací v paměti VRAM a všechny grafické příkazy jazyku BASIC-I. Týká se to i sprajtů. V příloze příručky pro BASIC-I je řada obrázků, které mají sloužit k názornějšímu vysvětlení a pochopení grafiky počítače M5. O grafických možnostech počítače M5 by se dalo napsat mnoho stránek stejně jako by se dalo uvést i mnoho demonstračních programů. Z časových i kapacitních důvodů to nebylo možné udělat přímo v této příručce.

V následujících částech jsou stručně vysvětleny další inicializační a výkonné grafické příkazy jazyku BASIC-F. Vzhledem k tomu, že autor této příručky neměl pro vyzkoušení činnosti grafických příkazů barevnou televizi, nejsou uvedeny všechny funkční podrobnosti některých příkazů.

4.02 Obarvení znakových výstupů na obrazovku

Pomocí tří příkazů /BCOL, COLOR, FCOL/ je možno programově měnit barvy znakových výstupů na obrazovku. Nejčastěji budou asi používány příkazy BCOL a FCOL v textovém režimu, kdy umožní nastavit barvu znaků i pozadí jako celku. Parametrem obou příkazů je kod barvy v rozsahu 0 až 15. Příkaz FCOL /forward color/ nastavuje barvu znaků a příkaz BCOL /backward color/ barvu pozadí obrazovky. Po zapnutí počítače a přepnutí do textového režimu je barva pozadí černá /kod barvy 1/ a barva znaků šedá /barva 14/. Pro řadu uživatelů bude pravděpodobně příjemnější provést inverzi, kdy na šedém podkladě budou černé znaky jako např. u počítačů Sinclair. To se provede velice jednoduše po přepnutí do textového režimu dvěma příkazy z klávesnice: FCOL 1:BCOL 14. Podle potřeby si můžeme text i pozadí obarvit libovolnými kombinacemi barev. Pouze je třeba dát pozor na to, aby kody barev v obou příkazech nebyly stejné. Potom totiž nebude nic vidět. U některých kombinací barev může dojít k patrnému chvění na rozhraní obou barev, což snižuje čitelnost textu. Není to ale ani vada počítače, ani vada Vaší televize.

V režimu GI je třeba použít pro obarvení znaků příkaz COLOR se dvěma parametry. Prvním parametrem je ASCII kod znaku. Druhý parametr obsahuje informaci o barvě znaku /horní čtyři bity/ a barvě pozadí daného znaku /spodní čtyři bity/. Vzhledem k tomu, že v režimu GI je rezervováno pro tabulku barev pouze 32 bitů, obarví se vždy osm znaků stejným způsobem, jestliže obarvíme příkazem COLOR pouze jeden znak. Pro jednotné obarvení všech znaků i jejich po pozadí je třeba použít krátký program:

```
10 rem BARVY GI
20 rem ********
30 input "BARVA ZNAKU ";BZ:input "BARVA POZADI ";BP:B=BP+16*BZ
40 for I=0 to 248 step 8:color I,B:next I
```

l v režimu GI je možno použít příkaz BCOL. Jestliže ho použijeme přímo po zapnutí počítače nebo po přepnutí do režimu Gl z jiného zobrazovacího režimu, nastaví se nám barva pozadí celé obrazovky jako v režimu textovém. Použijeme-li ho však po provedení programu BARVY GI, nastaví se nám pouze barva rámečku mezi aktivní plochou a okrajem obrazovky.

Program na obarvení znaků v režimu GII je skoro stejný jako pro režim GI. Pouze cyklus pro I musí být od 0 do 255 s krokem 1, nebot je možno si každý znak obarvit zvlášť. Příkaz BCOL funguje úplně stejně jako v režimu GI.

4.03 Inicializace a základní vlastnosti jemné grafiky

Po přepnutí do obrazovkového režimu GII pomocí tlačítek CTRL+R se skoro nic ve srovnání s režimem GI nezmění. Pouze pro obarvení znaků je třeba trochu jiný program. Pro využití plochy obrazovky na jemnou grafiku v bodovém rastru 256x192 je zapotřebí příkazem GINIT provést její inicializaci. Po příkazu GINIT je již možno používat všechny příkazy pro jemnou grafiku. Příkaz GMODE a jeho dva parametry určují výslednou barvu nebo způsob zobrazení jednotlivých bodů ve vztahu k původnímu grafickému obsahu obrazovky. O příkazu GMODE bude ještě zmínka v jedné z dalších částí.

Všechny grafické příkazy PLOT, GMOVE, DRAW a PAINT používají jako parametry grafické souřadnice X,Y. Obecně mohou být grafické souřadnice X,Y libovolná celá čísla v rozsahu -32768 až 32767. Na obrazovce se ale zobrazí pouze body, jejichž souřadnice X je v intervalu 0 až 255 a souřadnice Y v intervalu 0 až 191. Grafické souřadnice 0,0 přísluší levému hornímu rohu obrazovky. Při práci s výkonnými grafickými příkazy tedy není zapotřebí stále testovat, zda již nejsme mimo obrazovku. Následující demonstrační program nás o tom přesvědčí:

```
10 rem GRAFIKA 1
20 rem **********
30 Print "■■"
40 ginit
50 draw 0,0,255,0:draw 255,0,255,191
60 draw 255,191,0,191:draw 0,191,0,0
70 for Y=0 to 95 step 5
80 draw-255,Y,511,Y+95
90 next Y
100 Print "■"
110 if inkey$="" then goto 110
120 Print "■"
```

Příkazem DRAW jsou na řádku 80 kresleny úsečky z bodů, jejichž souřadnice X je -255 do bodů o souřadnici Y 511. Všechny počáteční i koncové body úseček tedy leží mimo obrazovku. Poslední tři řádky programu se budou vyskytovat prakticky ve všech dalších demostračních programech s jemnou grafikou. Na řádku 100 dojde k přesunu ukazatele na obrazovku 2. Jestliže by program touto řádkou končil, potom po skončení programu zůstane grafický obsah obrazovky zachován. Na řádce 110 se testuje, zda se stiskne libovolné znakové tlačítko klávesnice. Po stisknutí se provede řádek 120, kterým se provede výměna obrazovek 1 a 2. Opět máme k dispozici ukazatel a můžeme si napsat příkaz pro výpis programu nebo program opravit či doplnit. Po stisknutí tlačítek CTRL+Y nastane opět výměna obrazovek a uvidíme původní grafickou obrazovku. Po dalším stisknutí tlačítek CTRL+Y se opět dostaneme na program. Na řádku 30 se řídícím znamek inverzní U nastavuje obrazovka 1. Je to proto, aby se umožnil opakovaný běh programu. Po skončení programu jsme totiž na obrazovce 2 a není možno na obou obrazovkách nastavit režim GII.

4.04 Příkaz PLOT pro obarvení jednoho bodu

Základním grafickým výkonným příkazem je příkaz PLOT. Jeho parametry jsou grafické souřadnice X,Y bodu, který chceme obarvit. Barva bodu se nastaví příkazem FCOL před příkazem PLOT. Pozadí obrazovky se kdykoliv může obarvit příkazem BCOL. Následující demonstrační program náhodně obarvuje nahodnými barvami zadaný počet bodů grafické obrazovky:

```
10 rem GRAFIKA 2
20 rem *********
30 Print "■■■■":inPut "POCET BODU ";N
40 ginit:bcol 0
50 for I=1 to N:fcol rnd(14%)+1
60 Plot rnd(255%),rnd(191%)
70 next I
80 Print "■"
90 if inkey$="" then goto 90
100 Print "■"
```

Jednou z mnoha aplikací příkazu PLOT je kreslení křivek zadaných různými funkčními závislostmi. Jako názorný příklad je dále uveden program pro kreslení grafu libovolné zadané funkce s nezávisle proměnnou X. Program je sice trochu delší, ale přesto je tak jednoduchý, že nepotřebuje žádného komentáře.

```
10 rem GRAF FUN
20 rem ********
30 dim F(256):Print "■■■■":fcol 1:bcol 14:len 255
40 inPut "FUNKCE PROMENNE X ";F$:L=len(F$):Print
50 inPut "RADIANY=0, STUPNE=1 ";T:theta(T):Print
60 inPut "MIN. X (LEVA MEZ) ";X1:Print
70 inPut "MAX. X (PRAVA MEZ) ";X2
80 D=(X2-X1)/255:Print:Print "KROK D =";D:Print "VYPOCET FUN
KCNICH HODNOT"
90 Print:for I=1 to 256:X=X1+D*(I-1):F(I)=calc(F$):next I
100 Print:Print "HLEDANI MINIMA A MAXIMA"
110 MIN=F(1):MAX=F(1):for I=2 to 256
120 if F(I)<=MAX then goto 130 else MAX=F(I):goto 140
130 if F(I)<MIN then MIN=F(I)
140 next I
150 Print:Print "MIN.=";MIN;"     MAX.=";MAX
160 if inkey$="" then goto 160
170 if X2>0 then goto 180 else M=0:goto 200
180 if X1<0 then goto 190 else M=0:goto 200
190 M=fix(-255*X1/(X2-X1))
200 if MAX>0 then goto 210 else MAX=0:N=0:goto 230
210 if MIN<0 then goto 220 else MIN=0:N=191:goto 230
220 N=fix(191*MAX/(MAX-MIN))
230 E=(MAX-MIN)/191
240 Print "■■■■":ginit:bcol 14:fcol 1:draw 0,N,255,N
250 if M<>0 then draw M,0,M,191
260 for I=1 to 255:Y=(F(I)-MIN)/E
270 Plot I-1,191-Y:next I
280 if inkey$="" then goto 280
290 Print "■":stoP
```

Většina osobních počítačů s jemnou grafikou má příkaz CIRCLE pro kreslení kružnic, elips a pravidelných mnohoúhelníků. Příkaz CIRCLE má i jazyk BASIC-G počítače M5. Do paměti ROM jazyku BASIC-F se už asi nevešel. Kružnice a elipsy se ale často při programování jemné grafiky používají. Pro ulehčení programování je užitečné si vytvořit podprogram pro kreslení obecně zadaných kružnic pomocí příkazu PLOT. Následující program nakreslí uprostřed obrazovky několik soustředných kružnic. Pro vlastní kreslení kružnice se používá podprogram $KRUZNICE. Pro urychlení kreslení kružnic se na začátku programu vypočítají souřadnice 91 bodů čtvrtiny kružnice o poloměru 255 a uloží se pro použití podprogramem $KRUZNICE do polí X%/91%/ a Y%/91%/. Podprogram $KRUZNICE by šel vylepšit optimalizací kroku na řádku 100 /zatím je krok jeden stupeň/ podle velikosti zadaného poloměru. Kreslení malých kružnic nyní trvá stejně dlouho jako kreslení kružnic velkých a při optimalizaci kroku by se výrazně zrychlilo.

```
10 rem KRUZNICE1
20 rem *********
30 dim X%(91%),Y%(91%):theta 1:MX=191:MY=255
40 for I%=0% to 90%:X%(I%+1%)=cint(MX*sin(I%)):Y%(I%+1%)=cin
t(MY*cos(I%)):next
50 Print "■":ginit:gmode 4:SX%=128%:SY%=91%
60 for M=0.025 to 0.2 step 0.025
70 gosub $KRUZNICE:next M:Print "■":goto 200
90$KRUZNICE
100 for S%=1% to 91%
110 X%=cint(M*X%(S%)):Y%=cint(M*Y%(S%))
120 Plot SX%+X%,SY%+Y%:Plot SX%+X%,SY%-Y%
130 Plot SX%-X%,SY%+Y%:Plot SX%-X%,SY%-Y%
140 next S%:return
200 if inkey$="" then goto 200
210 Print "■"
```

4.05 Příkazy GMOVE a DRAW pro kreslení úseček

V části 4.03 již byla zmínka o příkazu DRAW pro kreslení spojnice dvou bodů. Jako parametry příkazu DRAW byly použity dvě dvojice grafických souřadnic oddělených čárkou. V příkazu DRAW ale můžeme použít pouze jednu dvojici grafických souřadnic. Příkazem DRAW se potom nakreslí daný přírustek od bodu, který se určí příkazem GMOVE. Příkazem GMOVE se nastavuje poloha tzv. grafického ukazatele. Je to bod grafické obrazovky, který se nezobrazí, ale od kterého začíná kreslení úsečky. Po nakreslení úsečky přírustkovým příkazem DRAW se poloha grafického ukazatele změní na grafické souřadnice koncového bodu nakreslené úsečky. K původní poloze grafického ukazatele se přičte přírustek zadaný v příkazu DRAW. Poloha grafického ukazatele se nezmění, jsou-li v příkazu DRAW dvě dvojice grafických souřadnic odděleny čárkou /kreslení spojnic dvou bodů/.

Jedním příkazem DRAW můžeme nakreslit i lomenou čáru. Stačí jako parametry zadat grafické souřadnice jednotlivých bodů /konstanty nebo proměnné/ oddělené středníky. Souřadnici počátečního bodu lomené čáry je nutno zadat v příkazu GMOVE. Poslední poloha grafického ukazatele je uložena mezi systémovými proměnnými na adresách &70B2 a &70B0. Následující program Vám předvede základní možnosti použití příkazů DRAW a GMOVE.

```
10 rem GRAFIKA 3
20 rem *********
30 Print "■":ginit:gmode 4
40 for Y%=0% to 191% step 3%
50 draw 0%,0%,255%,Y%:draw 255%,191%,0%,191%-Y%
60 next Y%:Print "■"
70 if inkey$="" then goto 70
80 ginit:gmode 4
90 for I%=1% to 100%
100 X%=rnd(230%):Y%=rnd(170%):L%=8%+rnd(100%)
110 gmove X%,Y%:draw X%+L%,Y%:X%+L%,Y%+L%:X%,Y%+L%:X%,Y%
120 next I%:Print "■"
130 if inkey$="" then goto 130
140 ginit:gmode 4
150 for I%=1% to 21%
160 X%=I%*3:Y%=I%*2:L%=191%-I%*6:gmove X%,Y%
170 draw L%,Y%:draw L%,L%:draw X%,L%:draw X%,Y%
180 next I%:Print "■"
190 if inkey$="" then goto 190
200 Print "■"
```

4.06 Obarvení a mazání jemné grafiky

V předchozích demonstračních programech pro jemnou grafiku jsme zatím s vyjímkou programu GRAFIKA 2 nepoužívali více než dvou barev. Barva grafických bodů byla totožná s barvou znaků a barva pozadí byla stejná jako barva pozadí před nastavením režimu GR. Z programu GRAFIKA 2 je již asi zřejmé, jakým způsobem se mohou kreslit různobarevné čáry. Stačí před příkazem DRAW použít příkaz FCOL s parametrem určujícím kód požadované barvy. Barva pozadí se nastavuje příkazem BCOL. Před kreslením většího počtu různobarevných čar, které se navzájem protínají, je třeba si uvědomit další základní vlastnost barevné jemné grafiky, která zde bude ještě zopakována, přestože byla již vysvětlena v příručce BASIC-1. Grafický programový systém i vlastnosti použitého video procesoru umožnují obarvit každou z osmi řádek základního znakového rastu 8x8 bodů pouze dvěma barvami. Proto může dojít při křížení dvou různobarevných čar ke vzniku dalších barevných bodů kolem průsečíku. Příkazem GMODE je snad možné to částečně vylepšit. To si ale musí majitelé barevných televizorů zkusit sami.

Jestliže používáme při kreslení pouze jedné barvy, využijeme příkaz GMODE při mazání. Pro kreslení použijeme GMODE 4 a před vymazáním je třeba parametr čtyři změnit na sedmičku. Demonstrační program Vám to předvede:

```
10 rem GRAFIKA 4
20 rem *********
30 print "■":ginit:gmode 4
40 for Y%=0% to 191% step 3%
50 draw 0%,0%,255%,Y%:draw 255%,191%,0%,191%-Y%
60 next Y%:gmode 7
70 for Y%=0% to 191% step 6%
80 draw 0%,0%,255%,Y%:draw 255%,191%,0%,191%-Y%
90 next Y%:print "■"
100 if inkey$="" then goto 100
110 print "■"
```

4.07 Vybarvování uzavřených tvarů příkazem PAINT

Velice užitečným při praktických aplikacích jemné grafiky je příkaz PAINT, který rychle vybarví zadanou plochu uzavřenou již nakreslenou křivkou. Příkaz PAINT může mít ve své základní formě dva nebo tři parametry. První dva parametry jsou grafické souřadnice bodu na obrazovce, který musí být uvnitř ohraničené plochy, kterou chceme vybarvit. Jestliže není třetí parametr zadán, obarví se plocha barvou, která byla naposledy dána příkazem FCOL. Křivka, tvořící hranici plochy dostane stejnou barvu jako vybarvená plocha i v tom případě, že byla původně nakreslena jinou barvou. Třetí zadaný parametr v příkazu PAINT nahrazuje příkaz FCOL. Je to tedy kód barvy, kterou chceme plochu obarvit. Jediný příkaz PAINT můžeme použít i pro obarvení více ploch. Potom musí být trojice čísel /grafické souřadnice a kód barvy/ odděleny středníkem. Následující program ukazuje použití příkazu PAINT:

```
10 rem GRAFIKA 5
20 rem *********
30 print "■":ginit:bcol 0
40 for I%=0% to 9%:fcol 14
50 X%=I%*8:Y%=I%*3:L%=191%-I%*12
60 gmove X%,Y%
70 draw L%,Y%:draw L%,L%
80 draw X%,L%: draw X%,Y%
90 if I%=0% then goto 100 else fcol I%:paint X%-2%,Y%
100 next I%:print "■"
110 if inkey$="" then goto 110
120 print "■"
```

4.08 Tisk čísel nebo textu do jemné grafiky

V předchozích částech jste se seznámili s používáním příkazů pro jemnou grafiku a měli by jste být schopni tyto příkazy používat v nejrůznějších praktických aplikacích. Jednou z aplikací může být například grafické znázornění nějakého časového průběhu ve formě sloupcového grafu. Pro popis grafu je ale zapotřebí doplnit krátký text a hlavně popsat obě osy. Běžně používaný příkaz PRINT pro textovou obrazovku zde nemůžeme používat. Je zapotřebí pracovat s grafickou obrazovkou jako s výstupním grafickým zařízením se zkratkou GRP, jak bylo uvedeno v části 2.01. Při tisku čísel je výhodnější si číselné hodnoty převést na znakové řetězce, neboť při tisku čísel se navíc objeví mezera před číslem. V následujícím demonstračním programu dochází k vymazání jemné grafiky v místech tisku znaků. Použitím příkazu GMODE s vhodnými parametry by možná šlo tisknout bez mazání grafických bodů.

```
10 rem GRAFIKA 6
20 rem *********
30 Print "■R■":ginit
40 for I=0 to 191 step 7
50 draw 0,0,255,I:draw 255,191,0,191-I
60 next I
70 open "GRP:" for output as#1
80 theta 1:for I=0 to 22
90 A$=num$(I):B$=num$(sin(I))
100 Print#1 cursor(6,I):right$(A$,len(A$)-1)
110 Print#1 cursor(11,I):right$(B$,len(B$)-1);
120 next I:close#1:Print "■"
130 if inkey$="" then goto 130
140 Print "■"
```

4.09 Záznam a čtení grafických obrazovek přes magnetofon

Příkaz VSAVE umožňuje zaznamenat na magnetofonovou pásku libovolný blok paměti VRAM zadaný počáteční a konečnou adresou. Navíc je možno si tento blok pojmenovat. Zpětné přehrání se provede příkazem OLD doplněným případně jménem pamětového bloku pro jeho vyhledání. Kompletní barevnou grafickou obrazovku v režimu GII je nutno nahrát jako dva samostatné bloky paměti VRAM. Každý z bloků má kapacitu 6 KB. První blok od adresy &0000 do &17FF obsahuje informace o barvách. Druhý blok od adresy &2000 do &37FF potom informace o zobrazených bodech a tvarech. Při nahrávání zpět do paměti VRAM se přímo na obrazovce postupně objevují jednotlivé části grafické obrazovky. Následující dva krátké podprogramy jsou určeny pro záznam a čtení grafických obrazovek přes magnetofon:

```
1000$ZAZNAMGII
1010'**********
1020 rem N$ JE ZADANY NAZEV SOUBORU
1030 rem OBRAZOVKA MUSI BYT V REZIMU GII
1040 Print "■"
1050 vsave "CMT:"+N$+"B",0,&17FF
1060 vsave "CMT:"+N$+"T",&2000,&37FF
1070 Print "■":return

1000$NAHRANIGII
1010'***********
1020 Print "■R■":ginit:Print "■"
1030 old "CMT:":old "CMT:":return
```

4.10 Nástin dalších grafických možností počítače M5

V předchozích částech jste se seznámili s dalšími grafickými příkazy jazyku BASIC-F, jejichž základní použití bylo demonstrováno řadou krátkých programů. Při praktických aplikacích je ale možno využívat grafické možnosti počítače M5 v mnohem větší míře. Každý uživatel by si hlavně měl stále uvědomovat, že má k dispozici dvě obrazovky, na nichž může pracovat současně. Jestliže se nepracuje v režimu GII, může mít každá obrazovka svůj vlastní generátor znaků zcela předefinován.

Zajímavou aplikační oblastí je tvorba a využití programů pro zpracování textových informací. U profesionálních systémů se používá na jedné řádce 64 až 80 znaků. U počítače M5 je maximální počet znaků 40, což je pro praktické využití málo. V režimu GII by jistě šlo programově zajistit zobrazení dvou užších znaků na jedné původní znakové pozici. Doplněním čárek a háčků by se získal základ pro kvalitní a hlavně užitečný program pro zpracování textů, který by se výborně uplatnil například i při psaní této příručky.

Další aplikační oblasti jako výukové programy, počítačové hry, počítačové grafické umění a návrh počítačem /CAD/ snad nepotřebují dalšího komentáře.

Při práci na grafické obrazovce můžeme využívat rastr 256x192 bodů. Jednobarevná grafická obrazovka potom potřebuje 6 KB paměti VRAM. Při použití přídavné paměti 32 KB máme k dispozici asi 34 KB paměti RAM. Okamžitě se nabízí myšlenka využít paměť RAM i pro uchování grafických informací ve větším rastru než je původní. Grafická obrazovka by potom sloužila jako pracovní výřez, se kterým by se dalo libovolně pohybovat ve větší grafické oblasti. Pro přesuny dat by se musely použít podprogramy ve strojovém kódu. Pro efektivní využití takto vytvořeného grafického systému by byla ale zapotřebí tiskárna s grafickým výstupem a případně i disková jednotka. To už se ale dostáváme od čistě amatérského využívání počítače M5 k jeho použití v profesionální praxi.

Na závěr této části je ještě třeba upozornit uživatele na možnost používání sprajtů a zvukových efektů. Příkazy pro práci se sprajty jsou stejné jako v jazyku BASIC-I. Využitím vhodných podprogramů monitoru by se daly pohybové možnosti sprajtů výrazně vylepšit. Časem by se i v jazyku BASIC-F mohlo pracovat se sprajty stejně jako v jazyku BASIC-G. Zvukový generátor použitý v počítači M5 umožnuje velice efektivní zpětné akustické působení. Uživatelé s hudebními buňkami jistě objeví brzy další možnosti jeho využití.

5. 01 Základní informace o uložení programů v paměti

Programové řádky zadává uživatel ve znakové formě. Po stisknutí tlačítka RETURN se znaková forma řádku převede do vnitřní systémové formy, která zabírá mnohem méně místa v paměti hlavně proto, že každé klíčové slovo jazyku BASIC-F se převede na jednobytový kód. Začátek každého programového řádku má standartní tvar:

1. byte - hodnota &FF /255/
2. byte - délka řádku v bytech
3. byte - počet mezer mezi číslem řádku a prvním příkazem při výpisu příkazem LIST
4. byte - kód prvního příkazu řádku
5. byte - spodní byte čísla řádku
6. byte - horní byte čísla řádku

Z výše uvedeného tvaru začátku programového řádku v paměti RAM je třeba si zapamatovat pouze to, že při psaní programových řádek můžeme za číslem řádku programu udělat pro zvýšení přehlednosti programu libovolný počet mezer, aniž se tím prodlouží uložení programu v paměti.

V programových řádcích se vedle příkazů a funkcí vyskytují číselné a řetězcové konstanty a proměnné. Zajímavým způsobem jsou v programu zakodovány číselné a řetězcové proměnné. Vedle jejich názvu jsou ještě rezervovány dva byty, které se při spuštění programu příkazem RUN doplní adresou, kde je hodnota příslušné proměnné uložena v paměti RAM.

Pro činnost interpretu jazyka BASIC-F je třeba řada tzv. systémových proměnných. Základními dvoubytovými systémovými proměnnými jsou:

a/ adresa začátku programu na adrese &726A
b/ adresa konce programu a začátku proměnných na adrese &726C
c/ adresa konce proměnných na adrese &726E

5.02 Přednosti programování v jazyku BASIC-F

Psaní programů v jazyku BASIC-F výrazně usnadňují a zefektivňují některé jeho vlastnosti, které jsou stručně popsány v následujících bodech:

a/ dlouhé názvy číselných a alfanumerických proměnných /až 32 znaků/
b/ použití alfanumerických návěští, což je asi vůbec největší přednost jazyku BASIC-F
c/ vyjádření řádků programu několika způsoby - v příkazech GOTO, GOSUB, RESTORE, ON GOTO, ON GOSUB a ON RESTORE se kromě čísel řádků nebo jejich návěští mohou používat číselné a řetězcové proměnné
d/ možnost použití smyčky REPEAT UNTIL - příkaz REPEAT indikuje, kam se program vrací, jestliže podmínka v příkazu UNTIL není splněna

5.03 Definování textů na funkčních tlačítkách

Při současném stisknutí tlačítka FUNC a některého ze znakových tlačítek A až Z se na obrazovce vypíše klíčové slovo jazyku BASIC-F, které je na příslušné klávese vyznačeno. Urychluje to psaní programů i příkazů v režimu okamžitého výpočtu. Vzhledem k omezenému počtu dvaceti šesti takto fungujících tlačítek je ale řadu příkazů stejně potřeba vyťukávat znak po znaku. Všechna klíčová slova na tlačítkách A až F jsou uložena v paměti ROM. Následující program umožňuje jejich výpis a zároveň ukazuje způsob jejich uložení:

```
10 rem VYPIS TL
20 rem ********
30 print "■■■":fcol 1:bcol 14
40 input "VYSTUP (0-CNS,1-PRT)";P:print
50 if P=1 then open "PRT:" as#1
60 FKN=peek(&70DC):FKA0=peek(&70DD)+256*peek(&70DE)
70 for I=1 to FKN:FKA=FKA0+2*(I-1)
80 A0=peek(FKA)+256*peek(FKA+1)
90 print#P I;tab(5);chr$(I+64);" ";FKA;" ";A0;"  ";
100 N=peek(A0):for A=A0+1 to A0+N:print#P chr$(peek(A));:nex
t A
110 print#P:next I
120 if P=1 then close#1
```

Změnou adresy uložené na adrese &70DD je možno využívat jinou textovou tabulku, ke které musí být vytvořena tabulka počátečních adres textu pro jednotlivá tlačítka. Další program umožňuje uživateli nadefinovat nové texty o délce až 255 bytů pro tlačítka A až Z:

```
10 rem DEF TEXTU
20 rem *********
30 if fre(4)>36863 then clear,&FFFF else clear,&8FFF
40 dim K%(26):pokew &70DD,&5D05:poke &70DC,26
50 input "ZADANI-0, NAHRANI-1 ";W:print:if W=0 then goto 80
60 input "NAZEV SOUBORU ";N$:old "CMT:"+N$:P=vpeek(0):A=vpeek(1)+256*vpeek(2)-3
70 print:print "POCET KLAVES -";P;"  CELKEM";A;"ZNAKU":print:goto 210
80 input "POCET KLAVES (1-26) ";P:vpoke 0,P:print
90 len 1:input "KLAVESA ";K$:if K$="0" then goto 160
100 K=ascii(K$)-64:if K>P then goto 90
110 if K%(K)=0 then goto 130
120 exe("PRINT "+chr$(K+64)+"$"):goto 90
130 len 255:input "TEXT ";T$:print
140 L=len(T$):len L:K%(K)=L
150 exe(chr$(K+64)+"$=T$"):goto 90
160 A=3:for K=1 to P:vpoke A,K%(K):A=A+1
170 if K%(K)=0 then goto 200
180 len K%(K):exe("T$="+chr$(K+64)+"$")
190 for J=1 to K%(K):vpoke A,ascii(mid$(T$,J,1)):A=A+1:next J
200 next K:AH=int(A/256):AL=A-256*AH:vpoke 1,AL:vpoke 2,AH:A=A-3
210 M2=fre(4)-A-2*P:clear 256,M2:M2=fre(4)+1:M=M2:P=vpeek(0):M1=M+2*P:len 4
220 C=3:K=0:len 4:poke &70DC,P
230 N=vpeek(C):poke M1,N:C=C+1
240 W$=hex$(M1):exe("POKEW M2,&"+W$)
250 M2=M2+2:M1=M1+1:K=K+1:if N=0 then goto 270
260 for J=1 to N:poke M1,vpeek(C):M1=M1+1:C=C+1:next J
270 if K<P then goto 230
280 W$=hex$(M):exe("POKEW &70DD,&"+W$)
290 input "ZAZNAM NA MAGNETOFON - 1 ";W:if W<>1 then stop
300 input "NAZEV SOUBORU ";N$:vsave "CMT:"+N$,0,C-1
```

Program se skládá ze tří částí. První část slouží pro zadání textů pro maximální počet dvaceti šesti tlačítek /A až Z/. V druhé části se texty přesunou do paměti VRAM. V třetí části se zmenší paměť RAM pro využití jazykem BASIC-F a do volného prostoru se přesunou z paměti VRAM texty ze současným vytvářením tabulky počátečních adres textů. V programu jsou i části umožňující uživateli zadané texty nahrát na pásku magnetofonu a později zpětně nahrát a po přesunu používat. Zadané nebo nahrané texty zůstanou v paměti i po vymazání programu a proměnných příkazem NEW a změnách zobrazovacích režimů.

Uživatelem definované texty mohou přinést další zefektivnění činnosti programátora, neboť se v nich mohou vyskytovat často používané kombinace příkazů nebo i podprogramy na jeden řádek. Dále se ještě uplatní i v aplikačních programech, neboť v příkazu INPUT se po stisknutí funkční klávesy objeví příslušný

definovaný text. V programu může být i několik tabulek textů pro funkční klávesy A až Z, které se programově mění pouhou změnou dvoubytové adresy na adrese &70DD.

5.04 Psaní a odlaďování programů

Psaním programu se v této části rozumí zadávání programů v textové formě uživatelem z klávesnice. Psaní programů urychluje automatické řádkování /zadává se příkazem AUTO/, původní nebo uživatelem definované texty na funkčních tlačítkách a možnost napsat mezery.

Při odlaďování programů uživateli výrazně pomáhají vedle chybových hlášeních dva příkazy - TRACE a STEP. Příkaz TRACE umožnuje výpis čísel právě prováděných řádků programu. Pro odlišení s ostatními čísly, která se při běhu programu mohou tisknout, jsou čísla řádků v hranatých závorkách. K jejich výpisu dojde po vykonání příkazu TRACE ON. Výpis se zastaví příkazem TRACE OFF. Příkazy TRACE ON a TRACE OFF se mohou zadat přímo z klávesnice v režimu okamžitého výpočtu, nebo se doplní do programu na začátku a na konci části programu, jejíž průběh nás zajímá. Jestliže potřebujeme detailnější pohled na průběh části programu, musíme použít příkaz STEP. Po příkazu STEP ON dojde k zastavení programu před každým řádkem programu, jehož číslo se vypíše na obrazovku. Jestliže chceme, aby program pokračoval dál, musíme použít příkaz CONT. Před pokračováním programu si ale můžeme zjistit hodnoty proměnných, které nás zajímají. Příkazem STEP OFF se krokování programu po programových řádcích ukončuje. Příkazy STEP ON a STEP OFF se zadávají přímo z klávesnice v režimu okamžitého výpočtu, nebo se doplní do programu.

Při úpravách programů se vedle mazání programových řádek příkazem DEL mohou programové řádky přečíslovat příkazem RENUM v celém nebo v části programu. Samotný příkaz RENUM bez dalších parametrů přečísluje celý program s konstantním krokem 10. První řádka přečíslovaného programu má také číslo 10. Při použití parametrů příkaz RENUM přečísluje všechny řádky programu nebo jenom část programu od zadaného řádku do konce. První parametr je nové číslo řádku programu, od kterého přečíslovaná část začíná. Druhý parametr je původní číslo řádku, od kterého má být program přečíslován. Třetím parametrem je nový krok mezi čísly řádků přečíslované části programu. Není-li třetí parametr zadán, nastaví se krok na deset.

6.01 Způsob popisu příkazů a funkcí v originálním manuálu

Podstatnou část originálního anglického manuálu tvoří abecední přehledy všech příkazů a funkcí jazyku BASIC-F. Vzhledem k tomu, že nebylo možné popsat v této příručce všechny příkazy a funkce jazyku BASIC-F, bude dále probrán způsob popisu příkazů a funkcí, aby i uživatelé s minimálními znalostmi angličtiny mohli manuál prakticky používat.

Každému příkazu nebo funkci je věnována jedna stránka, na které ve většině případů zbývá ještě dost místa pro případné poznámky uživatele. Pod klíčovým slovem označujícím příkaz nebo funkci je vyznačen formát, v jakém se příkaz nebo funkce mohou používat. Pro jednoznačný popis formátu se používá několika symbolů popsaných na začátku třetí části originálního manuálu. Zkratka CR ve složených závorkách je uvedena u většiny příkazů a u některých funkcí. Označuje nám, že se příkaz nebo fukce může používat přímo z klávesnice v režimu okamžitého výpočtu nebo v programovém řádku. Hranaté závorky uzavírají část, která může nebo nemusí být v příkazu použita. Týká se to hlavně příkazů s mnoha parametry, z nichž některé se mohou vynechat. Uvnitř hranaté závorky bývají často dvě nerovnítka, mezi nimiž je popsána jedna část příkazu. Je-li část příkazu pouze mezi dvěma nerovnítky, potom je povinná. Jestliže se vyskytnou v příkazu hranaté závorky s čárkou a několika tečkami, mohou se předchozí části v příkazu dále opakovat. Svislá čára je vždy mezi dvěma částmi, z nichž jedna musí být použita. Jako oddělovač parametrů nebo částí příkazů a funkcí se používá čárka a u některých příkazů i středník.

Dále je stručně vysvětlena činnost příkazu nebo funkce. Potom následuje komentář k výše popsané činnosti a krátký demonstrační program. Řada těchto programů není pro popis příkazu nebo funkce příliš vhodná.

6.02 Podprogramy ve strojovém kódu, příkazy CALL a REG

Příkaz CALL slouží k provedení podprogramu ve strojovém kódu, který začíná na adrese, uvedené jako první parametr příkazu CALL. Obrovskou výhodou příkazu CALL je možnost předávat vstupní parametry podprogramu ve strojovém kódu. Těmito parametry jsou čtyři dvoubytové proměnné nebo konstanty, které tvoří pokračování seznamu parametrů v příkazu CALL za počáteční adresou podprogramu. Před začátkem vykonání podprogramu se tyto parametry dostanou do dvojic registrů AF, BC, DE a HL mikroprocesoru Z80. Jako názorný příklad použití příkazu CALL je dále vypsán program na "rolování" grafické obrazovky po paměťových blocích o velikosti 2 KB směrem nahoru:

```
10 rem ROL VRAM
20 rem ********
30 clear 256,&87FF
40 print "[redacted]"
50 ginit:gmode 4:fcol 1:bcol 14
60 fcol 1:bcol 14
70 for I=1 to 191 step 5
80 draw 0,0,255,I
90 draw 0,I,255,0
100 next I
110 call &0E01,&1E00,&1800,,&00
120 gosub 150
130 A$=inkey$:if A$="" then goto 120
140 print "[redacted]":stop
150 call &0E7D,,&0800,&8800,&2000
160 call &0B81,,,&2000,&2800
170 call &0B81,,,&2800,&3000
180 call &0E61,,&0800,&3000,&8800
190 return
```

V programu se používají některé podprogramy monitoru pro přesuny bloků paměti. Současně se demonstruje rychlost podprogramů ve strojovém kódu.

Po skončení podprogramu ve strojovém kódu se ještě před návratem do jazyku BASIC-F uloží hodnoty ve dvojicích registrů AF, BC, DE, a HL do paměti RAM, odkud je možno si je později vytáhnout pomocí příkazu REG, jehož parametr 1 až 4 určuje, o kterou dvojici registrů se jedná.

Příkazy CALL a REG značně usnadňují práci s podprogramy monitoru i s vlastními podprogramy uživatele a umožnují rozšířit aplikační možnosti jazyku BASIC-F.

6.03 Náhrada uživatelem definovaných funkcí, funkce CALC a EXE

Funkce CALC nahražuje u počítače M5 uživatelem definované funkce, které jsou běžné v jiných programovacích jazycích BASIC. Její praktické použití bylo v demonstračních programech ULCISEL a GRAF FUN. Argumentem funkce CALC je řetězcová konstanta nebo proměnná, která obsahuje libovolný aritmetický výraz s funkcemi ve znakové formě. Při provádění funkce CALC se do výrazu dosadí okamžité hodnoty proměnných a výraz se vypočítá.

Funkce EXE se vůbec u osobních i profesionálních počítačů nevyskytuje. Jejím argumentem je stejně jako u funkce CALC řetězcová konstanta nebo proměnná. Funkce EXE funguje tak, že zajistí provedení jednoho nebo více příkazů v jazyku BASIC-F, které jsou v znakové formě v jejím argumentu. Praktické použití funkce EXE je dvakrát v programu DEF TEXTU. Další ukázkou může být následující krátký program:

```
100 len 50:A$="INPUT C:FOR I=1 TO C:PRINT I,RND(1000%):NEXT I"
110 exe(A$):B$="C=10"+right$(A$,len(A$)-7)
120 print:print B$:print:exe(B$)
```

Funkce EXE umožňuje zařazovat do programu programové řádky, jejichž příkazy nejsou v průběhu psaní programu známy. Tyto řádky se mohou doplnit později ve formě vstupních dat nebo si je dokonce může program doplnit sám. Možnosti příkazu EXE se zatím předem bez větších praktických zkušeností s programováním v jazyku BASIC-F nedají odhadnout.

6.04 Formátovaný tiskový výstup, příkaz NUM$

Řada osobních počítačů má příkaz PRINTUSING pro tzv. formátovaný tisk na obrazovku nebo na tiskárnu. Pomocí tohoto příkazu se velice jednoduše programují složité tiskové výstupy s kombinacemi číselných a alfanumerických údajů. Počítač M5 s jazykem BASIC-F příkaz PRINTUSING nemá. Pomocí několika příkazů NUM$ lze však příkaz PRINTUSING docela dobře nahradit. Příkaz NUM$ převádí číslo na alfanumerický řetězec. Jestliže použijeme další dva číselné parametry, zjistíme, že příkaz NUM$ umí převádět čísla do řetězců se zaokrouhlením na požadovaný počet desetinných číslic i doplňovat mezery před číslo. První parametr udává celkovou délku řetězce před desetinnou tečkou i s případnými mezerami. Druhý parametr potom udává celkový počet znaků za desetinnou tečkou i s desetinnou tečkou. Součet obou parametrů je celková délka řetězce. Následující program i jeho tiskový výstup Vám názorně předvede použití příkazu NUM$ pro výše popisovaný formátovaný tisk:

```
10 rem TISK TAB
20 rem ********
30 open "PRT:" for output as#1:P=1:theta 1
40 len 80:H$=rpt$(8,"1234567890"):print#1 H$;
50 for I=0 to 10 step 2
60 print#P num$(I,7,1);num$(sin(I),8,6);num$(cos(I),8,6);num$(exp(I),8,6);num$(2
^-I,8,6);num$(rnd(1500),8,6):next I:close#1
```

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      0.       0.00000       1.00000       1.00000       1.00000     316.98239
      2.       0.03490       0.99939       7.38906       0.25000     431.85914
      4.       0.06976       0.99756      54.59815       0.06250    1409.61968
      6.       0.10453       0.99452     403.42879       0.01563      32.85257
      8.       0.13917       0.99027    2980.95799       0.00391    1381.49037
     10.       0.17365       0.98481   22026.46579       0.00098     191.55908
```

6.05 Simulace číslicové klávesnice pro vstup čísel

Při využití počítače M5 pro práci s větším počtem číselných údajů, které je třeba vkládat ručně přes klávesnici zjistíme, že pro rychlý a bezchybný vstup čísel je stávající klávesnice s čísly v horní řadě tlačítek naprosto nevyhovující. Některé osobní počítače mají pro vstup čísel oddělenou číslicovou klávesnici od vlastní klávesnice alfanumerické. U počítače M5 je možno tento závažný nedostatek odstranit pomocí jednoduchého podprogramu, který je společně s hlavním krátkým programem na vstup čísel dále vypsán:

```
10 rem VSTUP CIS
20 rem *********
30 C$="45601230-0."
40 stchr "38444C5464443800" to 154,1
50 stchr "1030101010103800" to 149,1
60 stchr "3844040810207C00" to 150,1
70 stchr "7C081008044443800" to 151,1
80 stchr "081828487C080800" to 145,1
90 stchr "7C407804044443800" to 146,1
100 stchr "1820407844443800" to 147,1
110 stchr "0000000000303000" to 155,1
120 stchr "0000007C00000000" to 153,1
130 cls:for R=0 to 20 step 2:for S=0 to 16 step 16
140 locate S,R:gosub $VSTUP CISEL:locate S+8,R:print A;:next S,R:print:stop
150$VSTUPCISEL
160 A$="":poke &701A,146:input D$:poke &701A,145
170 L=len(D$):for I=1 to L
180 A=ascii(mid$(D$,I,1))
190 if A<100 then A$=A$+chr$(A)else A$=A$+mid$(C$,A-144,1)
200 next I:A=calc(A$):return
```

Při spuštění programu můžeme před vstupem čísel pozorovat změnu ukazatele na grafický vstupní režim. Na začátku programu jsou některé grafické znaky umístěné na tlačítkách pod číslicovými tlačítky 7, 8, 9 předefinovány na zbývající číslice, desetinnou tečku a znaménko minus. Kody předefinovaných grafických znaků jsou dále změněny na kody příslušných číslic a nový řetězec se převede na hodnotu číselné proměnné. Při praktickém využívání výše uvedeného podprogramu je vhodné si příslušné číslice a novou desetinnou tečku se znaménkem minus vyznačit na příslušných klávesách.