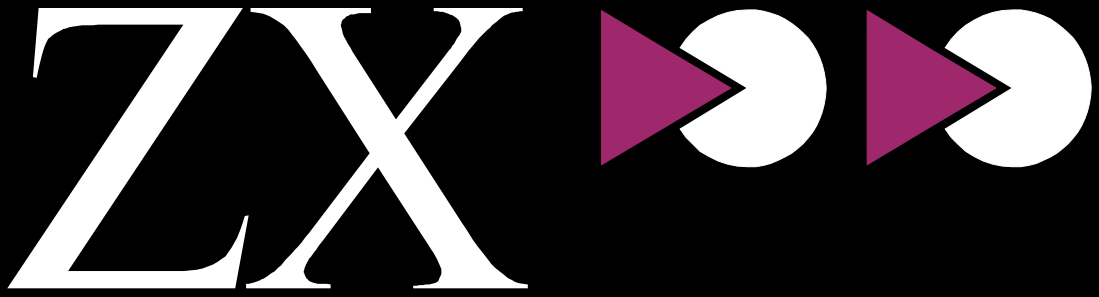


POZNÁMKY

ROM

Obsah:

Při tvorbě vlastních programů ve strojovém kódu, nebo při analýze již existujících programů je výhodné znát možnosti, které Spectrum nabízí jednak ve formě systémových proměnných, jednak ve formě podprogramů v ROM. Proto se v tomto metodickém materiálu pokusíme blíže vysvětlit využití hlavních systémových proměnných a některých podprogramů ROM.

U systémových proměnných je to poměrně jednoduché, poněvadž jich není tolik. Obtížnější je to s podprogramy ROM, kterých je více a jsou složitější. Ten, kdo se chce vážné zabývat programováním ve strojovém kódu, by si měl bezpodmínečně opatřit a podrobně prostudovat některý z komentovaných výpisů ROM Spectra. Zatím nejlepší je **Disassembler ROM** Basic, vydaný jako metodický materiál **666.ZO Svazarmu (PS 64, Praha 6 16900).**

Každá číselná hodnota se dá vyjádřit více způsoby. Čísla můžeme v podstatě rozdělit do dvou skupin. Jsou to čísla typu integer t. j. celé číslo, a číslo typu floating point, t. j. číslo s pohyblivou řádovou čárkou (neboli reálná čísla).

Čísla tu integer

U čísel typu integer je to jednoduché. Jsou to čísla, se kterými se denně setkáváme: -23, 5, 1187, 23689... Počítač je ukládá do jednotlivých paměťových míst ve formě 8 bitů, tvořících 1 bajt (jednotlivá paměťová místa jsou vlastně osmibitové registry schopné uchovávat hodnoty od %00000000 do %11111111).

Maximální hodnota kterou můžeme uložit do 1 bajtu či paměťového místa je tudíž $FF = 255 = %1111111. Větší čísla se ukládají do 2 bajtů v pořadí nižší bajt, vyšší bajt. U dvoubajtového čísla $5BFF je nižším bajtem $FF a vyšším bajtem $5B. V tomto pořadí jsou také bajty uloženy do paměťových míst A a A+1. Maximální hodnota dvoubajtového čísla může být $FFFF = 65535. Decimální hodnotu dvoubajtového čísla dostaneme tak, že k decimální hodnotě prvního bajtu přičteme 256 násobek decimální hodnoty druhého bajtu. Opačně t. j. při převádění decimální hodnoty čísla většího než 255, podělíme dané číslo 256, integer (t. j. celočíselnou část výsledku) uložíme do vyššího bajtu a zbytek čísla (t. j. číslo mínus 256krát integer, nikoliv zbytek dělení) uložíme do nižšího bajtu.

Je-li například obsah paměťového místa A **X**7E a paměťového místa A+1 **X**B2. Potom převodu na decimální hodnoty je to 126 a 178 a obsah 2 bajtů je 126+256*178=45694.

Naopak decimální číslo 23521 uložíme do 2 bajtů paměti takto:

23551|256=91+255

výsledek po dělení 91 uložíme do bajtu A+1, zbytek, t.j. 23551-256*91=255d=**X**FF uložíme do bajtu A. Hexadecimální vyjádření čísla 23521 je pak **X**5BFF a jeho uložení v paměti bude **X**FF **X**5B.

Čísla typu floating point

S čísly typu floating point je to poněkud složitější. Každé číslo můžeme vyjádřit ve tvaru:

A*B↑n.

A .. mantisa
B .. základ
n .. exponent

Zvolíme-li za základ číslo 2 a položíme-li podmínku 0.5= A 1, můžeme každé číslo vyjádřit jako **A*2↑n.**
Mantisu A v našem výrazu můžeme vyjádřit ve tvaru:

A1*2↑-1+A2*2↑-2+A3*2↑-3+. . . . +An*2↑-n

A1 až **An ..** jednotlivé bity binárního zlomku

Například:
0.75d=0.50+0.25=2↑-1+2↑-2. Binárně 1100.

Jakékoliv číslo převedeme do tvaru A*2↑n tak, že ho podělíme nejbližší vyšší mocninou dvou. Výsledek dělení je pak mantisa A a mocnina se objeví v exponentu n. Pro číslo 12d bude: 12/16=0.75 můžeme psát 12=0.75*2↑4 (16=2↑4, kontrola je 0.75*2↑4=0.75*16=12). Pro číslo 425.725 je nejbližší vyšší mocnina dvou 512=2↑9. Bude pak 425.725/512=0.8314941. . . . a můžeme psát 425.725=0.8314941*2↑9.

Mantisu převedeme z decimálního tvaru do binárního zlomku tak, že od daného decimálního čísla postupně odečítáme záporné mocniny dvou. Je-li v čísle daná mocnina dvou obsazena, bude v odpovídajícím bitu 1, není-li mocnina obsazena, bude v bitu nula.

V našich dvou příkladech to bude:

a) číslo 0.75

0.75	decimální mantisa
-0.50	(2na-1) - je obsazeno: .1
0.25	mezisoučet
-0.25	(2na-2) - je obsazeno: .11
0.00	mezisoučet roven nule, nejsou žádné další mocniny 2, v binárním zlomku budou následovat nuly.

b) číslo 0.8314941

```
 0.8314941 decimální mantisa
-0.5000000 (2na-1) - je obsazeno: .1
 0.3314941 mezisoučet
-0.2500000 (2na-2) - je obsazeno: .11
 0.0814941 mezisoučet
-0.1250000 (2na-3) - je obsazeno: .110
 0.0814941 opakování mezisoučtu
-0.0625000 (2na-4) - je obsazeno: .1101
 0.0189941 mezisoučet
-0.0312500 (2na-5) - je obsazeno: .11010
 0.0189941 opakování mezisoučtu
-0.0156250 (2na-6) - je obsazeno: .110101
 0.0033691 mezisoučet
```

atd. podle toho, s jakou přesností chceme dané číslo vyjádřit. Čím větší počet desetinných míst (bitů), tím přesnější vyjádření čísla. Ve většině případů je však výsledkem dělení (t. j. mantisa v decimálním tvaru) číslo s nekonečným počtem desetinných míst a musíme pak náš převod někde zastavit. Dopouštíme se tím určité nepřesnosti, která odvisí od počtu bitů, které máme k dispozici pro vyjádření binárního zlomku. Ve Spectru je vyhrazeno 32 bitů.

Pokud zvolíme systém, ve kterém mantisa může nabývat hodnot v intervalu 0.5,1) víme, že mantisa je vždy větší než 0.5, t.j., že v mantise je vždy obsazena mocnina 2↑-1, t.j., že hodnota prvního bitu v binárním zlomku je vždy 1. Ve Spectru se této skutečnosti využívá tak, že první bit binárního zlomku určuje znaménko mantisy. Je-li roven 0, je mantisa kladná, je-li roven 1, je záporná.

Vnitřní uložení čísel

Spectrum ukládá čísla do celkem 5 bajtů. Jednak jsou v této formě uložena čísla v programu a v tabulce proměnných, a jednak s nimi pracuje kalkulátor.

Reálná čísla

Reálná čísla jsou zapsána v exponenciálním tvaru:

číslo=mantisa*2↑exponent,

kde **mantisa** je v intervalu 0.5 až 1
exponent je v rozsahu -127 až 127

Hodnota je uložena v pěti bajtech:

exp	ma	nt	is	a
1.	2.	3.	4.	5.

První bajt obsahuje hodnotu exponentu, zvětšenou o ↑80=128, následující 4 bajty pak 32 bitů binárního zlomku. Jelikož je mantisa vždy vetší než 0.5, je první bit využit pro vyjádřeni znaménka mantisy.

Několik příkladů:

01111110 00000000 00000000 00000000 00000000

číslo=(1/2)*2↑(126-128) 0.125

10000000 01001100 11001100 11001100 11001100

číslo=(1/2+1/4+1/32+1/64+.....)*2↑(128/128) 0.8

Je zajímavé, že číslo 0.5 mažeme vyjádřit dvě odlišnými způsoby:

0.5=1*2↑-1 nebo 0.5=0.5*2↑0

To například způsobuje chybné vyhodnocení výrazu 1/2=0.5, který by měl nabýt hodnoty 1.

Celá čísla Číslo v intervalu -65535 až 65535. Je uloženo na pěti bajtech následovně:

1.	2.	3.	4.	5.
00	00/FF	nižší	vyšší	00

První bajt obsahuje 0, druhý bajt znaménko (0-kladné, ↑FF-záporné). Ve třetím a čtvrtém bajtu je uložena číselná hodnota (v pořadí nižší bajt, vyšší bajt). Pokud je číslo záporné, potom je uloženo v doplňku: 65536-ABS(x). V pátém bajtu je uložena nula.

Několik příkladů:

```
00 00 00 00 00  =  ↑0000      0
00 00 01 00 00  =  ↑0001      1
00 00 FF 00 00  =  ↑00FF    255
00 00 3A 30 00  =  ↑303A  12346
00 FF FF FF 00  = ↑-0001     -1
00 FF FF FE 00  = ↑-0100   -256
```

V prvních dvou bajtech je číslo řádku v pořadí vyšší, nižší bajt. Je to výjimka ze zavedeného pravidla opačného ukládání bajtu a je to proto, poněvadž nejvyšší číslo řádku může být 9999=▮3915. Podle hodnoty prvního bajtu je-li nižší rovno ▮39-pozná počítač, že se jedná o programový řádek. Je-li vyšší než ▮39, znamená to pro počítač, že se již nejedná o programový řádek, ale o některou proměnnou.

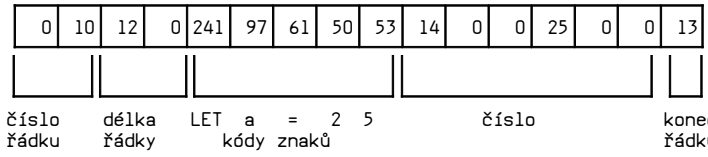
V následujících dvou bajtech je celkový počet bajtů programového řádku včetně závěrečného ENTER. Přičtením této hodnoty k adrese začátku programového řádku vypočte počítač adresu začátku dalšího řádku.

V následujících bajtech jsou postupně uloženy kódy jednotlivých znaků v programovém řádku (klíčová slova a ostatní "tokens" t.j. slova vkládaná stlačením jediné klávesy, mají vždy jen jeden kód).

Je-li v programu číslo, je uloženo jak ve znakovém tvaru, tak i ve vnitřní, pětibajtové formě. Za znakovou reprezentací následuje číslo 14, signalizující, že následuje pětibajtová série.

Na konci řádku je číslo 13, což je kód ENTER.

Příklad: Programový řádek 10 LET a=25 je v paměti počítače (t.j. v oblasti PROG) uložen následovně (dekadicky) :



4. Uložení proměnných v basicu.

Proměnné se po spuštění programu (po přiřazení hodnoty do proměnné příkazy jako LET, INPUT, READ...) přenesou do oblasti VARS, kde tvoří tzv. tabulku proměnných (proměnné zůstávají ve VARS i když po spuštění programu vymažeme všechny programové řádky; nikoliv však příkazem NEW).

Prvním a u řetězcových proměnných také jediným znakem musí být písmeno. Aby počítač poznal, o jaký druh proměnné se jedná, je první bajt, přesněji řečeno jeho první 3 bity, následovně zakódovány:

Typ proměnné	první tři bity	první bajt
řetězec	010	$41 až $5A
proměnná označená jediným písmenem	011	$61 až $7A
číslicové pole	100	$81 až $9A
proměnná označená více písmeny	101	$A1 až $BA
řetězcové pole	110	$C1 až $DA
řídící proměnná v příkazu FOR-NEXT	111	$E1 až $FA

Ve všech případech zůstává v posledních 5 bitech prvního bajtu kód písmene malé abecedy mínus $60, což znamená, že pro a (nebo A - u proměnných se nerozlišují malá a velká písmena) bude v posledních 5 bitech hodnota %00001 = 1, která se bude postupně zvyšovat až dosáhne hodnotu %11010 = $1A = 26 pro z. Nejnižší hodnota prvního bajtu řetězcové proměnné a$ je pak %01000001=$41 a podle toho, že je to víc než $39, pozná počítač, že opustil oblast, v níž je uložen program a nachází se v tabulce proměnných.

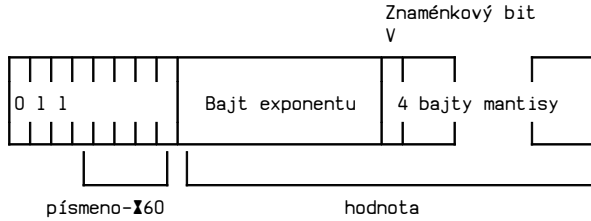
Toto kódováni provádí počítač tak, že ke kódu písmen malé abecedy přičítá určité hodnoty. Jsou to:

-20 u řetězců
0 u proměnných s jednopísmenovým názvem
20 u číslicových polí
40 u proměnných s vícepísmenovým názvem
60 u řetězcových polí
80 u řídících proměnných.

Ještě stručně ke způsobu uložení jednotlivých typů proměnných v tabulce proměnných.

Proměnná s jednopísmenovým názvem (a):

délka	6
1. bajt	kód názvu
2.-6. bajt	pětibajtové vyjádření hodnoty



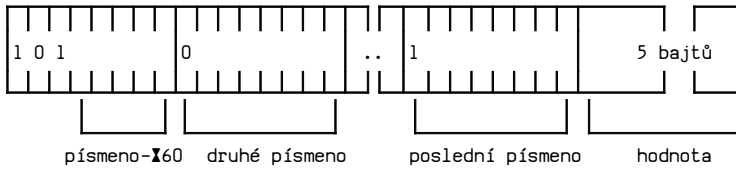
Proměnná s vícepísmenovým názvem (Adam):

délka počet bajtů názvu + 5

1.bajt kód prvního písmene názvu + X40

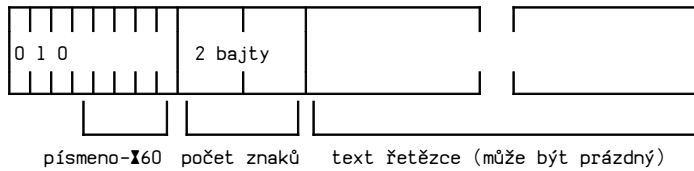
Další bajty až do celkové délky názvu jsou kódy jednotlivých písmen názvu. Kód posledního písmene je zvětšeno X80, což je znamení počítači, že končí název a začíná hodnota.

Posledních pět bajtů je pětibajtové vyjádření hodnoty proměnné.



Řetězec (a$):

délka	počet znaků v řetězci + 3 (bez $ a uvozovek).
1.bajt	kód názvu řetězce - &H20
2. a 3.bajt	počet znaků v řetězci (bez uvozovek) v pořadí nižší bajt, vyšší bajt.
	V dalších bajtech jsou kódy jednotlivých znaků řetězce.



Řídící proměnná FOR-NEXT smyčky (FOR a = ... TO):

délka 19

1. bajt kód názvu řídící proměnné + &80

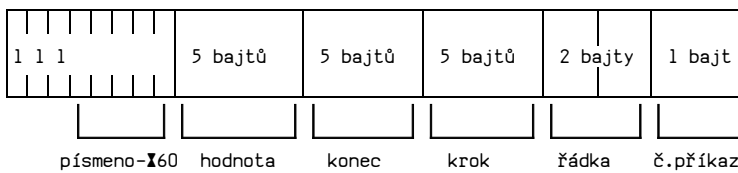
2.-6. bajt pětibajtové vyjádření počáteční hodnoty řídící proměnné. Při každém průchodu smyčkou se hodnota zvyšuje o velikost kroku, až dosáhne konečné hodnoty, t.j. hodnoty uvedené po TO, případně nejvyšší vyšší.

7.-11. bajt pětibajtové vyjádření konečné hodnoty řídící proměnné.

12.-16. bajt pětibajtové vyjádření hodnoty kroku (1, není-li krok udán).

17.-18. bajt číslo řádku na který se smyčka vrací po příkaze NEXT v obvyklém pořadí, t. j. nižší bajt, vyšší bajt.

19. bajt číslo příkazu v daném řádku, na který se smyčka vrací po NEXT.



Číslicové pole (A(M, N)):

délka — počet prvků (M*N)*5 + počet rozměrů (v našem případě 2) * 2 + 4.

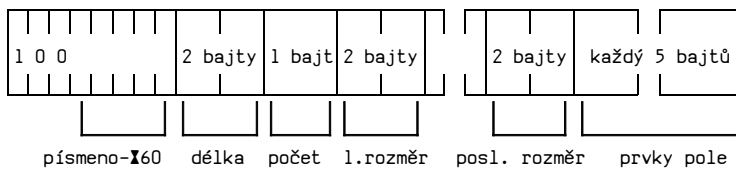
1. bajt — kód názvu pole + ⧗20

2. a 3. bajt — počet bajtů počínaje 4. bajtem do konce uložení pole (t. j. celkový počet bajtů - 3).

4. bajt — počet rozměrů, t. j. v našem případě 2.

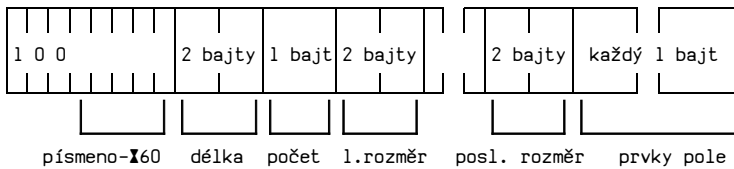
V dalších dvojicích bajtů jsou hodnoty jednotlivých rozměrů, t. j. hodnoty M a N.

V dalších M * n * 5 bajtech jsou hodnoty jednotlivých prvků pole v pětibajtovém vyjádření. V prvních pěti bajtech je hodnota prvku s indexem 1, 1, následuje prvek s indexem 1, 2 atd. až 1, N, pak následuje série prvků s indexy 2, 1, 2, 2 až 2, N a pak další série až po konečnou s indexy M, 1, M, 2 až M, N.



Řetězcové pole (A$(M, N))

délka	počet prvků (M*N)+počet rozměrů (2)*2+4
1. bajt	kód názvu pole + ▮60
2. a 3. bajt	počet bajtů počínaje 4. bajtem do konce uložení pole (t. j. celkový počet bajtů - 3).
4. bajt	počet rozměrů (2)
	V dalších dvojicích bajtů jsou hodnoty jednotlivých rozměrů, t. j. hodnoty M a N.
	V dalších bajtech jsou kódy jednotlivých znaků řetězce v pořadí 1. řetězec, 2. řetězec až M-tý řetězec.



V oblasti PROG, t.j. při ukládání programových řádků používá počítač po znaku 14 v pětibajtovém uložení hodnot všude tam, kde je to možné, hodnot typu integer, pokud není daná hodnota integer, pak nastupuje vyjádřeni typu plovoucí řádová čárka.

V oblasti VARS, t. j. v tabulce proměnných, počítač používá zásadně vyjádření typu plovoucí řádová čárka (floating point), i když se jedná např o číslo 1. Některé počítače mají možnost již při vložení čísla určit, jde-li o číslo typu integer. Počítač pak tomuto číslu vyhradí pouze 2 bajty. Spectrum pro jakékoliv číslo vyhradí vždy 5 bajtů. Zvětšuje se tím tabulka proměnných a prodlužuje doba na její prohledávání.

Ještě poznámku ke kalkulátoru (viz kapitola o ROM, adresa #38). Kalkulátor má pseudokód (příkaz) #34, kterým se uloží do zásobníku kalkulátoru float. point číslo v pětibajtovém tvaru. Aby nemusely být vypisovány všechny nulové bajty, je počet bajtů takového čísla zakódován v prvním (exponentovém) bajtu čísla, přičemž číslo musí mít nejméně dva bajty (další nuly doplní kalkulátor).

Hodnotu upraveného prvního bajtu dostaneme odečtením X50 a přičtením X40*(počet bajtů-2). Např. číslo 2 má v pětibajtovém vyjádření tvar X82 X00 X00 X00 X00 (exponent a pět nul). Do tvaru stravitelného pro kalkulátor ho upravíme následovně:

1. bajt = X82 - X50 + X40 * (2 - 2) = X32
2. bajt = 00 (nemění se).

Konečný tvar je X32 X00 (t. j. pouze 2 bajty). Číslo 0.8 je pětibajtově vyjádřeno: X80 X4C XCC XCC XCD , t. j. má všech pět bajtů. Úprava pro kalkulátor:

1. bajt = X80 - X50 + X40 * (5 -2) = XF0.
Další bajty X4C XCC XCC XCD se nemění.

Konečná podoba: XF0 X4C XCC XCC XCD . V obou případech si kalkulátor vypočte správnou hodnotu prvního bajtu, t. j. X80, resp. X82.

5. Členění paměti.

ROM

ROM znamená "Read Only Memory" t. j. paměť, ze které je možno pouze vybírat informace. Tyto informace jsou do ROM vloženy při výrobním procesu a není možno je později měnit. Znamená to, že pro kteroukoliv adresu v ROM můžeme použít PEEK, ale POKE nejde.

Celkem má ROM Spectra 16384 bajtů, proto se označuje jako 16k ROM a je pro obě verze (t. j. 16k RAM a 48k RAM) stejná.

Další informace jsou uvedeny v kapitole PODPROGRAMY ROM.

RAM

RAM je zkratka anglického názvu Random Access Memory, což přeloženo do češtiny znamená paměť s náhodný přístupem. Znamená to, že do jednotlivých paměťových míst můžeme libovolně vkládat informace a také z nich informace vybírat. Pro RAM platí několik základních pravidel.

Po zapnutí počítače jsou jednotlivá paměťová místa zaplněna zcela náhodnými hodnotami od 0 do FF. Počítač proto v první fázi provede tzv. inicializaci systému, spočívající ve vložení nul do všech paměťových míst, vymezení hranic jednotlivých oblastí RAM a vložení počátečních hodnot do systémových proměnných. To je ta časová prodleva než se po zapnuti počítače objeví na obrazovce Copyright.

Další zásadou je to, že po zapsání jakékoliv hodnoty do paměťového místa tam tato hodnota zůstává tak dlouho, dokud není přepsána další hodnotou, pokud ovšem při jejím výběru nedojde k okamžitému vymazání některým podprogramem ROM.

Spectrum v 16k verzi má celkem 16384 paměťových míst v RAM, 48k verze má v RAM celkem 49152 paměťových mást. RAM je rozdělena do jednotlivých oblastí, kam se ukládají vždy určité typy informací. V obou variantách Spectra začíná RAM na adrese 16384 a první čtyři oblasti RAM mají pevné adresy. Poslední pevnou adresou v RAM je 23734, kde končí oblast systémových proměnných a hranice všech dalších oblastí jsou pohyblivé a mění se podle množství informací v jednotlivých fázích práce počítače dokonce hranice některých oblastí splývají, t.j., některé oblasti neexistují, resp. mají nulový obsah. Adresy těchto pohyblivých hranic oblasti jsou uloženy v systémových proměnných.

Název oblasti	začátek (dek.) délka (dek.)	(hex.) (hex.)	název a adresa systémové proměnné		
ROM	00000 16384	₭0000 ₭4000			
Bitová mapa	16384 6144	₭4000 ₭1800			
Atributy	22528 768	₭5800 ₭0300			
Vyrovnávací paměť tiskárny	23296 256	₭5B00 ₭0100			
Systémové proměnné	23552 182	₭5C00 ₭00B6			
(Dodat. syst. proměnné)	23734	₭5CB6 ₭003A			
(Mapy microdrive)	23792	₭5CF0			
Kanálové informace+80h	(23734	₭5CB6)	CHANS	23631	₭5C4F
Program v basicu	(23755	₭5CCB)	PROG	23635	₭5C53
Tabulka proměnných+80h			VARS	23627	₭5C48
Vložený řádek+ENTER +80h			E-LINE	23641	₭5C59
Vstupní data+ENTER			WORKSP	23649	₭5C61
Pracovní prostor					
Zásobník kalkulátoru			STKBOT	23651	₭5C63
Volná paměť			STKEND	23653	₭5C65
Zásobník			SP-registr mikroprocesoru		
Zásobník GOSUB					
3Eh	65367	₭FF57)	RAMTOP	23730	₭5CB2
Definované znaky	65368 168	₭FF58) ₭00A8)	UDG	23675	₭5C7B
Konec RAM	(65535	₭FFFF)	P-RAMT	23732	₭5CB4

Bitová mapa

Oblast od adresy 16384 po adresu 22527 (16384-16383) je tzv. Bitová mapa, čili oblast pro uložení obrazové informace. Bitová mapa proto, že každému bodu na obrazovce odpovídá jeden bit v paměti. Tato oblast zaujímá celkem 6144 bajtů. Proč zrovna 6144? Obrazovka má 24 řádků po 32 sloupcích, celkem 768 znaků. Každý znak je uložen v osmi bajtech, tudíž pro uložení všech znaků obrazovky je třeba 768 * 8 = 6144 bajtů. A nebo si můžeme říci, že obrazovka má 192*256 bodů, což je 49152 bitů. Počet bajtů spočítáme jednoduše: 49152/8=6144. Teď se musíme podrobněji podívat na tuto oblast.

Každý z 8 bajtů tvořících znak vyjadřuje jeden řádek znaku na obrazovce. Např. písmeno "a" je vytvořeno následovně:

```
            BIN          HEX
........   0000 0000      00
........   000C 0000      00
..XXX...   0011 1000      38
.....X..   0000 0100      04
..XXXX..   0011 1100      3C
.X...X..   0100 0100      44
..XXXX..   0011 1100      3C
........   0000 0000      00
```

V BIN vyjádření znamená 0 barvu papíru a 1 barvu inkoustu (papír a inkoust viz. odstavec Atributy).

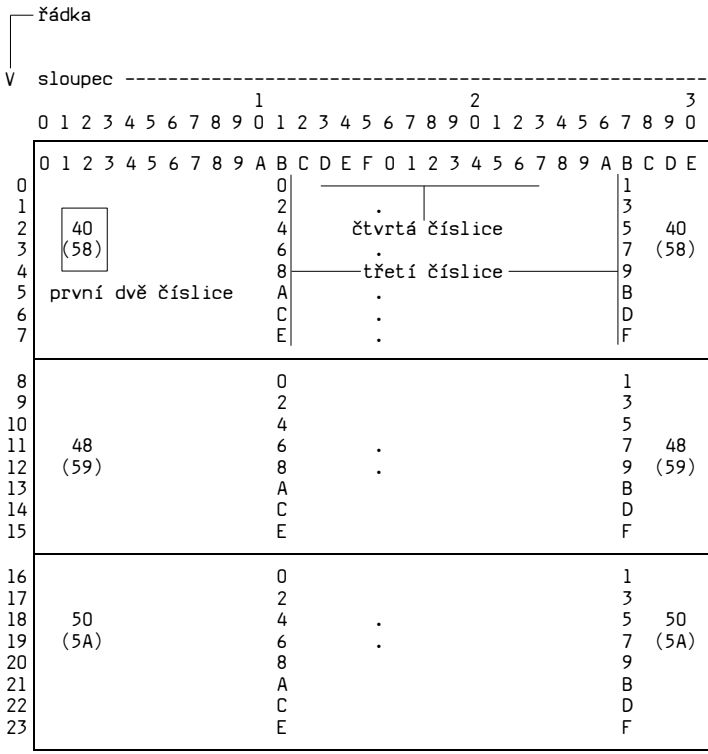
Použijeme-li pro adresování jednotlivých bajtů v bitové mapě registr DE, pak platí: Je-li v DE adresa prvního bajtu některého znaku, pak adresy dalších bajtů tohoto znaku dostaneme postupným opakováním INC D, zatímco adresu 1. bajtu následujícího znaku dostaneme INC DE. Toto INC DE platí ovšem jen v jednotlivých třetinách obrazovky. Při přechodu z jedné třetiny do druhé musíme použít následující sekvenci strojových instrukcí:

```
RR   D
RR   D
RR   D
INC  DE
RL   D
RL   D
RL   D
```

Je to proto, že adresa prvního bajtu posledního znaku řádku 7 je X40FF a adresa prvního bajtu prvního znaku řádku 8 je X4800 (nikoliv X4100, což je adresa 2. bajtu prvního znaku). Uvedená sekvence strojových instrukcí dá v každém případe správnou adresu prvního bajtu následujícího znaku. Adresu prvního bajtu libovolného znaku na obrazovce určíme pomocí tabulky (čísla v závorkách platí pro adresy atributů). Do jednotlivých adres bitové mapy můžeme pomocí POKE vkládat libovolné hodnoty. Zkuste např:

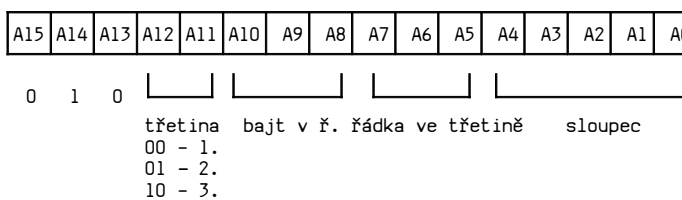
```
LD    HL,X40A4
LD    (HL),XFF
RET
```

Po spuštění tohoto programu se na obrazovce v místě odpovídajícím PRINT AT 5, 4 objeví tenká čárka. Je to proto, poněvadž adresa X40A4 odpovídá podle tabulky: X40-první třetině obrazovky, XA-5. řádek, 4-4. sloupec. Určení adresy v bitové mapě:



Pro určení atributů platí pro první dvě číslice hodnoty v závorkách, další dvě číslice podle vzoru nahoře.

Adresu v bitové mapě je možné znázornit i následovně:



Atributy

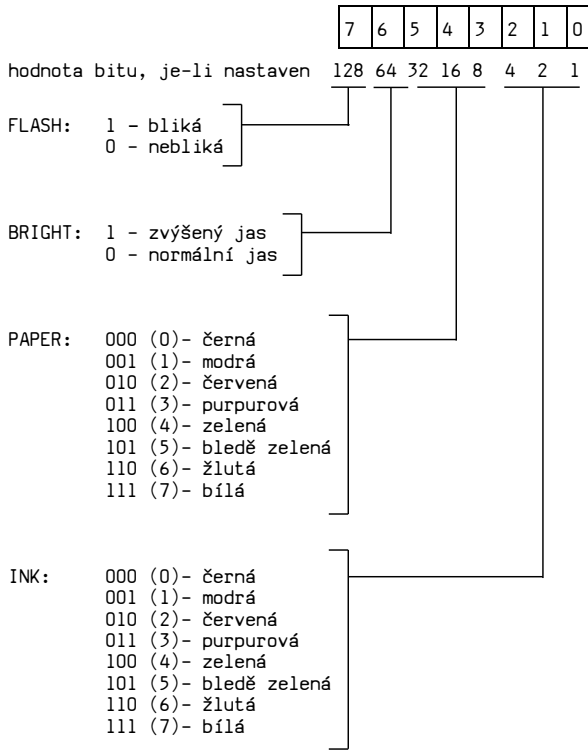
Oblast od adresy 22528 po adresu 23295 (#58C0 - #5AFF) je vyhrazeno pro uložení atributů znaků zobrazovaných na obrazovce. Zavedené termíny pro atributy: paper (papír), ink (inkoust), bright (jas) a flash (blikání).

Každému znaku přísluší jeden bajt v pořadí 1. znak prvního řádku, 2. znak prvního řádku až poslední znak prvního řádku. Pak následují stejným způsobem postupné všechny ostatní řádky. (Píší-li znaky první řádky, jsou tím míněny znaky řádku 0). Tato skutečnost je důvodem, proč Spectrum nemá "plnohodnotnou" bodovou barevnou grafiku (nemůže např. zobrazit horní polovinu znaku jinou barvou než jeho spodní polovinu), ale celý znak musí mít stejnou barvu (znakem rozumíme vždy 8 bajtů vyjadřujících znak, takže znakem je i mezera), ať už se jedná o inkoust nebo papír. Adresy, v nichž jsou uloženy atributy odpovídající jednotlivým pozicím na obrazovce, zjistíme z tabulky na předchozí straně.

Vlastnosti atributů jsou v jednotlivých bajtech zakódovány následujícím způsobem:

bajt=128*FLASH+64*BRIGHT+8*PAPER+INK.

I když uvedený způsob kódování vypadá na první pohled složitě, je vlastně velice jednoduchý a nejlépe je vidět na následujícím schématu. Bajty atributů jednotlivých znaků (je jich celkem 24*32 =768) obsahují:



Nastavením jednotlivých bitů, t.j. vložením odpovídajících hodnot do bajtu atributu můžeme libovolně nastavit atributy kteréhokoliv znaku na obrazovce. Maximální hodnota bajtu atributu může být

128*1+64*1+8*7+7 = 128+64+56+7 = 255 (#FF).

Všechny barvy, které může Spectrum generovat na obrazovce jsou kombinace tři základních barev barevného spectra: modré (001), červené (010) a zelené (100). Jejich sečtením vznikne bílá barva (111), sečtením zelené a červené vznikne žlutá (110) atd. V tříbitovém vyjádření je možno použít pouze 8 různých kombinací, což je důvodem, proč Spectrum může generovat pouze 8 barev.

Vyrovnávací paměť tiskárny

Oblast od adresy 23296 po adresu 23551 (X5B00 až X5BFF) je vyhrazena pro vyrovnávací registr tiskárny. Tato oblast RAM má 256 bajtů a je určena pro dočasné uložení nekompletního řádku určeného pro tisk tiskárnou ZX Printer. Tiskárna totiž musí vytisknout celý řádek naráz, i když je např. příkazy TAB nebo AT rozdělen do několika programových řádků a na obrazovce se vypisuje postupně podle stádií programu.

Pokud není požadováno použití tiskárny ZX Printer a program ve strojovém kódu není delší než 256 bajtů, je možno této části RAM využít pro uložení programu. Je zde totiž chráněn před přepsáním basicovým programem.

Systémové proměnné

Oblast od adresy 23552 po adresu 23733 (X5C00 až X5CB5) je vyhrazena pro uložení systémových proměnných. Horní hranice, t. j. adresa 23733, ovšem platí pouze pro případ, že ke Spectru není připojen Interface 1, což je přídavné zařízení, pomocí něhož je uživateli k dispozici lokální síť, rozhraní RS232 a rychlá vnější paměť (Microdrive). Je-li připojen a aktivován, posune se horní hranice oblasti systémových proměnných na adresu 23791 (X5CEF). Je to proto, poněvadž Interface 1 potřebuje pro svoji práci některé další systémové proměnné, které jsou pak připojeny za konec normálních systémových proměnných.

Systémové proměnné obsahují informace nutné pro práci počítače a mají přesně vymezený význam. Mimo jiné je v nich uložena adresa začátku pohyblivých oblastí RAM. Poněvadž některé z nich můžeme výhodně použít v programech ve strojovém kódu, bude jim věnována samostatná kapitola.

Mapy microdrive

Je-li ke Spectru připojen Interface 1 s Microdrive, začíná od adresy 23792 (▮5CF0) oblast map Microdrive. Její velikost závisí na počtu připojených Microdrive (maximálně 8), poněvadž pro každý aktivovaný Microdrive je vytvořena samostatná mapa o délce 32 bajtů. Do této mapy se ukládají informace o obsazení jednotlivých sektorů Microdrive.

Kanálové informace

Není-li ke Spectru připojen Interface 1, je od adresy 23734 po adresu 23754 (▮5CB6 až ▮5CCA) vyhrazeno 21 bajtů pro uložení tzv. kanálových informací. Do těchto bajtů se během inicializace přenesou z ROM určité hodnoty, sloužící pak k nalezení adresy příslušného kanálu. Pokud není připojen Interface 1, jsou stále otevřené 4 kanály: klávesnice, obrazovka, tiskárna a pracovní prostor. Je-li připojen Interface 1 spolu s Microdrive, jsou vytvořeny další kanály, do kterých se ukládají informace, které mají být přeneseny do jednotlivých sektorů Microdrive. Každý z těchto kanálů pak zabírá 595 bajtů.

Pro programy ve strojovém kódu je z kanálů důležité vědět to, že je-li otevřen kanál K, znamená to tisk na spodní část obrazovky. Chceme-li psát na horní část obrazovky, musíme otevřít kanál S. (K-Keyboard = klávesnice, S-Screen = obrazovka). V případě, kdy ke Spectru není připojen Interface 1, neexistuje v RAM oblast "Mapy microdrive" a hned za kanálovými informacemi následuje oblast, ve které je uložen basicový program.

Začátek oblasti kanálových informací je v systémové proměnné CHANS. Tato oblast je uzavřena bajtem 80h.

Program v basicu

Tato oblast začíná adresou, která je uložena v systémové proměnné PROG, a podle ní se někdy celá oblast označuje jako PROG. Jednotlivé programové řádky jsou v této oblasti uloženy ve tvaru uvedeném v předchozí kapitole. Pokud není připojen Interface 1, začíná oblast PROG na adrese 23755 (▮5CCB).

Tabulka proměnných

Za oblastí PROG následuje oblast, v níž jsou uloženy hodnoty jednotlivých proměnných. Její začátek je v systémové proměnné VARS a podle toho bývá někdy celá oblast nazývaná VARS. Tato oblast se vytvoří až po spuštění programu a jsou do ní přeneseny hodnoty proměnných z programových řádků v předchozí oblasti. Hodnoty jednotlivých proměnných jsou uloženy ve tvarech uvedených v předchozí kapitole a ukládají se postupně podle toho, jak následují v Basicovém programu. Příkaz vytvoří potřebný počet bajtů, do nichž jsou pak ukládány jednotlivé hodnoty později. Poslední bajt této oblasti má hodnotu ▮80. Hodnoty proměnných zůstávají v této oblasti zachovány i po vymazání všech programových řádků a můžeme s nimi i pak operovat. Vymaže je příkaz NEW, LOAD, CLEAR a RUN.

Vložený řádek

Za oblastí PROG následuje oblast, ve které jsou uloženy příkazy nebo řádky basicu před stlačením ENTER. Jsou to příkazy nebo řádky, které počítač vypisuje do dolní části obrazovky. Po stlačení ENTER se buď provede příkaz (včetně přenesení hodnot proměnných do oblasti VARS) nebo se programový řádek přenese do oblasti PROG. Do této oblasti se také přenese programový řádek příkazem EDIT a zde je možno provádět změny v programovém řádku, ať už právě vkládaném nebo editovaném. Začátek oblasti je uložen v systémové proměnné E-LINE a poslední bajt v této oblasti má hodnotu ▮80. Pokud nemá programový řádek číslo řádku, je po stlačení ENTER vymazán a nikam se nepřenese.

Vstupní data

Do další oblasti, začínající na adrese uložené v systémové proměnné WORKSP, se ukládají data vložená z klávesnice po příkazu INPUT. Po stlačení ENTER se přenesou do oblasti VARS.

Pracovní prostor

Na horním konci oblasti "Vstupní data" (po ENTER) je část označená jako pracovní prostor (WORKSPACE). V této oblasti provádí počítač operace s řetězci. Obě oblasti se překrývají, poněvadž do prostoru mezi WORKSP a STKBOT buď ukládají "Vstupní data", která se po ENTER přenesou do dalších oblastí a oblast WORKSP-STKBOT se vynuluje, nebo během provádění programu se v této oblasti provádí operace s řetězci. Nikdy neprobíhají obě činnosti současně.

Zásobník kalkulátoru

Další oblast začínající na adrese uložené v systémové proměnné STKBOT je "Zásobník kalkulátoru". Sem ukládá kalkulátor jednotlivé hodnoty, se kterými pracuje. Zásobník kalkulátoru narůstá od nižších adres k vyšším, t. j. jeho vrchol je na nejvyšší adrese, dané systémovou proměnnou STKEND.

Volná paměť

Dál následuje oblast která by se dala označit jako rezerva. Jsou to volné bajty RAM, na jejichž úkor se rozšiřují ostatní oblasti. Začátek je na adrese uložené v systémové proměnné STKEND a konec je na adrese uložené v registru SP. Chceme-li znát počet volných bajtů, které máme ještě k dispozici, musíme od obsahu registru SP odečíst hodnotu systémové proměnné STKEND.

Zásobník

Pod RAMTOPem je poslední oblast RAM, tzv. Stack. Někdy se označuje jako zásobníková paměť, uživatelský zásobník, zásobník návratových adres, nebo pouze zásobník. Do zásobníku ukládá Z80 jednak návratové adresy při provádění instrukce CALL a RST, jednak obsahy registrů při provádění instrukce PUSH. Naopak ze zásobníku Z80 vybírá návratové adresy při provádění instrukce RET a obsahy registrů při provádění instrukce POP. Poslední informace se ukládá na vrchol zásobníku. Označení vrchol zásobníku není příliš vhodné, poněvadž zásobník narůstá odshora dolů, t. j. od vyšších adres k nižším adresám. Vrchol zásobníku je tudíž nejnižší adresa. Další zvláštnost je v tom, že adresa vrcholu zásobníku není uložena v žádné systémové proměnné, ale přímo v registru SP Z80 (SP znamená Stack pointer = ukazatel zásobníku).

Vybírání informací ze zásobníku se děje v opačném pořadí než ukládání, t. j. informace se vybírá z vrcholu zásobníku . Jako první se vybere poslední vložená informace, jako druhá pak předposlední atd. Pokud se do zásobníku ukládají dvoubajtová čísla, děje se ukládání v obvyklém pořadí nižší bajt, vyšší bajt, t. j. vyšší bajt je na vrcholu zásobníku. Vybírání se děje v pořadí vyšší bajt, nižší bajt. Do zásobníku se též ukládají návratové adresy příkazu GOSUB.

Definované znaky

Při inicializaci systému se vytvoří oblast pro uložení definovaných grafických znaků GRAPH-a až GRAPH-u (kód 144 až 164). Tato oblast začíná na adrese uložené v systémové proměnné UDG. U 16k je to 32000 (#7F58), u 48k je to 65368 (#FF58) a je chráněna před přepsáním basicovým systémem (pokud ovšem nedefinujete vlastní grafické znaky) a před příkazem NEW.

6. Systémové proměnné

Systémové proměnné obsahují informace o okamžitém stavu systému a řadu z nich můžeme, nebo dokonce musíme, při sestavování strojových programů použít.

Jak již bylo uvedeno, systémové proměnné zaujímají jeden nebo dva bajty, výjimečně i více bajtů (KSTATE, FRAMES a MEMBOT). Některé z nich mohou být měněny pomocí POKE, některé ne. Ze všech však mohou být pomocí PEEK získávány informace. Místo POKE a PEEK je možno použít strojové instrukce LD.

Chceme-li získat decimálně vyjádřený obsah jednobajtové systémové proměnné, stačí

PRINT PEEK adresa pramenné.

U dvoubajtových proměnných je třeba použít

PRINT PEEK adresa+256*PEEK (adresa+1).

Chceme-li změnit obsah jednobajtové systémové proměnné, postačí

POKE adresa proměnné, nová decimální hodnota.

Při měnění hodnoty dvoubajtové systémové proměnné musíme použít

POKE adresa,hodnota-256*INT(hodnota/256)
POKE adresa+1,INT(hodnota/256).

Nyní k jednotlivým systémovým proměnným (pro lepší přehlednost je uvedu v abecedním pořadí).

ATTR-P
23693
x5C8D

8 bajtů této systémové proměnné obsahuje stále běžné barevné informace. Používá je například CLS a podobné příkazy.

ATTR-T
23695
x5C8F

Barevná informace pro právě tisknutý znak, například při RST **x**10.

BORDCR
23624
x5C48

Barevná informace pro spodní část obrazovky. Její změnou můžeme docílit, aby vkládaný řádek ve spodní části obrazovky měl zvýšený jas (BRIGHT) nebo blikal (FLASH), pro což Spectrum normálně příkazy nemá. Také můžeme dosáhnout toho, aby vkládaný řádek nebyl vidět, t. j. aby byla stejná barva PAPER a INK, což normálně nemůže nastat. Použijeme-li POKE, pak musí následovat CLS. Nový stav pak trvá až do dalšího příkazu BORDER nebo NEW.

B-REG
23655
x5C67

Tuto systémovou proměnnou používá kalkulátor jako počítadlo. Bližší viz kapitola o ROM, část RST **x**28.

CHANS
23631-23632
x5C4F-**x**5C50

V systémové proměnné je uložena adresa začátku oblasti kanálových informací. Bez připojeného Interface 1 je v proměnné adresa 23734.

CHARS
23606-23607
I5C36-**I**5C37

Obsahuje adresu začátku tabulky znaků mínus 256 (**I**100). Normálně je v ní hodnota 15360 (**I**3C00). Této systémové proměnné můžeme použít, potřebujeme-li více, než 21 definovaných grafických znaků. Můžeme si totiž vytvořit ve volné části RAM novou tabulku všech znaků a změnou hodnoty v proměnné CHARS získat přístup k této nové tabulce (v CHARS musí být hodnota o 256 nižší, než je začáteční adresa nové tabulky). Zpět do původní tabulky ROM se vrátíme pomocí:

```
POKE 23606,0 : POKE 23607,60.
```

CH-ADD
23645-23646
I5C5D-**I**5C5E

Systémová proměnná obsahuje adresu znaku který bude jako příští přenesen do Bitové mapy a jejím prostřednictvím vypsán na obrazovku..

COORDS
23677-23678
I5C7D-**I**5C7E

Obsahuje souřadnice X (v 23677) a Y (v 23678) posledního bodu zobrazeného příkazem PLOT nebo DRAW. Můžeme měnit, čímž měníme polohu bodu na obrazovce.

Zvolíme-li například:

```
LET XO=23677 : LET YO=23678
```

pak můžeme použít

```
DRAW A-PEEK XO, B-PEEK YO
```

a dostaneme přímku z bodu, jehož souřadnice jsou v COORDS, do bodu o souřadnicích A, B.

CHURCHL
23633-23634
I5C51-**I**5C52

Obsahuje adresu z oblasti kanálových informací, příslušnou pro právě otevřený kanál.

DATADD
23639-23640
I5C57-**I**5C58

Obsahuje adresu čárky za poslední položkou v řádku DATA, která byla přečtena příkazem READ v programu.

DEFADD
23563-23564
I5C0B-**I**5C0C

Obsahuje adresu argumentu definované funkce DEF FN.

DEST
23629-23630
I5C4D-**I**5C4E

Adresa proměnné při jejím vyhodnocování, jinak 0.

DEF-CC
23684-23685
I5C84-**I**5C85

Obsahuje adresu prvního bajtu vypisovaného na obrazovku.

DFCCL
23686-23687
I5C86-**I**5C87

Totéž jako DF-CC, ale pro spodní část obrazovky (DF-CC pro horní část obrazovky).

DF-SZ
23659
x5C6B

Obsahuje počet řádků ve spodní části obrazovky - normálně 2. Může být měněna hodnotami od 0 do 24, přičemž ale POKE 23659,0 znamená, že počítač nemá kam vypisovat sdělení a při první potřebě vypsat sdělení (například SCROLL?) dojde ke zhroucení systému. Podobně při INPUT nebo SAVE a při stlačení klávesy BREAK (po BREAK následuje sdělení). Toto je možné využít při ochraně programu s autostartem proti zastavení pomocí BREAK. Pokud se do takového programu vloží POKE 23659,0, pak první BREAK vede ke zhroucení systému. Rovněž se nesmí použít příkazy CLS, CLEAR, INPUT, SAVE... Je-li obsah DF-SZ roven jedné, je možné použít PRINT AT 22,xx, je-li však obsah DF-SZ roven nule, nefunguje PRINT AT 23,xx. Chceme-li přesto psát na poslední řádek, je lépe použít PRINT AT 22, 31 nebo PRINT **x** 0.

ECHO-E
23682-23683
x5C82-**x**5C83

Obsahuje adresu ve spodní části obrazovky, za kterou nejde dál posunout kurzor směrem doprava.

ERR-NR
23610
x5C3A

Obsahuje kód sdělení (chyby) mínus 1.

ERR-SP
23613-23614
x5C3D-**x**5C3E

Obsahuje návratovou adresu na kterou se vrátí program po návratu z podprogramu RST **x**08, který vyvolává sdělení, tedy například adresu, kterou počítač pokračuje po zodpovězení dotazu SCROLL?.

E-LINE
23641-23642
#5C59-#5C5A

Obsahuje adresu začátku oblasti "Vložený řádek".

E-PPC
23625-23626
#5C49-#5C4A

Obsahuje adresu běžného řádku, tedy řádku u které stojí programový kursor. Ale pozor! Interně Spectrum používá jiného způsobu číslování řádku na obrazovce než v basicu. Pro interní číslováni platí vzorec:

řádka na obrazovce=24-interní číslo

FLAGS
23611
#5C3B

Různé příznaky. Je-li nastaven bit 1, znamená to, že je použita tiskárna, je-li nulován bit 5. znamená to, že možno vyhodnotit další stlačenou klávesu, atd.

FLAGS2
23658
#5C6A

Další příznaky

FLAGX
23665
#5C71

Další příznaky.

FRAMES
23672-23674
#5C78-#5C7A

Při každém přerušení, což se děje 50 krát za sekundu, je o jedničku zvýšena hodnota systémové proměnné FRAMES. Počáteční hodnota při zapnutí počítače je 0. FRAMES tak funguje jako počítadlo délky provozu počítače a může být použito k měření času (buď průběžně nebo stopky). Hodnotu FRAMES zjistíme pomocí:

PRINT PEEK 23672+256*PEEK 23673+65536*PEEK 23674.

Max. hodnota, kterou může počítač uložit do 3 bajtů FRAMES je 2^24-1=16777215, což odpovídá době 3 dny, 21 hod. , 12 min. a 24. 3 sec. Během provádění příkazu BEEP, operací s kazetovým magnetofonem a tiskárnou nedochází ke zvětšení hodnoty FRAMES, což je třeba brát v úvahu, chceme-li FRAMES používat k přesnému měření času.

KSTATE
23552-23559
#5C00-#5C07

Systémovou proměnnou používá počítač při vyhodnocování stlačené klávesy, nebo při vyhodnocování dvou stlačených kláves a při opakování. 8 bajtů je rozděleno do dvou čtveřic pro dvě stlačené klávesy. V prvních bajtech je #FF, není-li stlačena žádná klávesa, nebo #C0, dojde-li ke stlačení klávesy. Druhé a třetí bajty jsou nezajímavé, ve čtvrtých bajtech je kód stlačené klávesy bez ohledu na současně stlačenou SHIFT klávesu. Znamená to, že v těchto bajtech je vždy kód bílého znaku v levé horní části klávesy (písmene A až Z a číslice 1 až 0).

K-CUR
23643-23644
#5C5B-#5C5C

Adresa kursoru.

K-DATA
23565
▮5C0D

Do systémové proměnné se ukládá barevná informace vložené klávesnicí před jejím dalším zpracováním.

LAST-K
23560
▮5C08

Obsahuje kód poslední stlačené klávesy s ohledem na současné stlačení SHIFT klávesy (t. j. rozlišuje např. mezi "A" a "a"). V LAST-K se objeví kód 15 při současném stlačení CAPS SHIFT a 9 (GRAPHICS). Obdobně je kód TRUE VIDEO (CAPS SHIFT a 3) 4 a kód INV VIDEO (CAPS SHIFT a 4) 5.

LISTSP
23615-23616
▮5C3F-▮5C40

Adresa návratu z automatického výpisu.

MASK-P
23694
▮5C8E

Obsahuje masku, kterou se přenáší barevná informace ze systémové proměnné ATTR-P (MASK-P AND ATTR-P). Je-li bit v MASK-P nastaven, použije se hodnota odpovídajícího bitu v ATTR-P. Můžeme použít například chceme-li omezit počet barev na obrazovce, nebo chceme-li vyloučit BRIGHT nebo FLASH.

MASK-T
23696
▮5C90

Totéž jako MASK-P, ale pro ATTR-T.

MEM
23656-23657
I5C68-**I**5C69

Adresa začátku zásobníkové paměti kalkulátoru (obvykle MEMBOT, ale ne vždy).

MEBOT
23698-23727
I5C92-**I**5CAF

Používá kalkulátor jako zásobníkovou paměť pro uložení hodnot, se kterými pracuje.

MODE
23617
I5C41

Obsahuje hodnoty 0 až 2 podle typu kurzoru na obrazovce. 0 = K, L a C, 1 = E a 2 = G. Je možno použít ke změně režimu klávesnice. Například POKE 23617,2 vyvolá režim G.

NEWPPC
23618-23619
I5C42-**I**5C43

Obsahuje číslo řádku, na který má skočit příkaz GO TO nebo GO SUB. Změnou můžeme vynutit skok na danou řádku.

NESPPC
23620
I5C44

Obsahuje číslo příkazu v řádku, na který má skočit GOTO nebo GOSUB. Chceme-li využít skok na n-tý příkaz v řádku L, musíme provést :

```
POKE 23618,L-256*INT(L/256)
POKE 23619,INT(L/256)
POKE 23620,n
```

Použijeme-li POKE 23618 a POKE 23619, musí vždy následovat POKE 23620.

NMIREG
23728-23729
I5CB0-**I**5CB1

Adresa ošetření nemaskovaného přerušení. Pro chybu v obslužném programu v ROM není tato proměnná využívána.

NXTLIN
23637-23638
I5C55-**I**5C56

Začáteční adresa dalšího programového řádku, který má být programem proveden.

OLDPPC
23662-23663
I5C6E-**I**5C6F

Číslo řádku, kterým program pokračuje po CONTINUE.

OSPCC
23664
I5C70

Číslo příkazu v řádku, kterým pokračuje CONTINUE.

PIP
23609
I5C39

Délka trvání zvukového signálu při stlačení klávesy. Po zapojení obsahuje nulu, což se vlastně rovná 256 a odpovídá délce signálu cca 1/3 sec. Je možné použít hodnoty od 0 do 256, čímž se délka signálu zkracuje.

PPC
23621-23622
I5C45-**I**5C46

Číslo právě prováděného řádku.

PROG
23635-23636
x5C53-**x**5C54

Adresa začátku oblasti pro uložení programu v basicu. Bez připojeného Interface 1 je to 23755.

PR-CC
23680
x5C80

Nižší bajt příští pozice tisku při LPRINT (ve vyrovnávací paměti tiskárny - vyšší bajt je **x**5B).

P-FLAG
23697
x5C91

Různé příznaky.

P-POSN
23679
x5C7F

Číslo sloupce pozice LPRINT.

P-RAMT
23732-23733
x5CB4-**x**5CB5

Adresa posledního fyzického bajtu RAM. Pro 16k RAM je to 32767 (**x**7FFF) a pro 48k RAM je to 65535 (**x**FFFF).

RAMTOP
23730-23731
x5CB2-**x**5CB3

Adresa posledního bajtu systému basic. V systémové proměnné RAMTOP je uložena hodnota o jedničku menší, než hodnota systémové proměnné UDG. Pro 16k versi je to po zapnutí počítače hodnota 32599 (**x**7F57) a pro 48k versi je to hodnota 65367 (**x**FF57). Je to poslední paměťové místo basicového systému. Snížením RAMTOP si můžeme vytvořit oblast pro uložení strojových programů.

Spectrum má příkaz CLEAR s možností připojit k tomuto příkazu argument. CLEAR bez argumentu vymaže oblast proměnných. CLEAR s argumentem (například CLEAR 32499), sníží RAMTOP na hodnotu danou argumentem a vytvoří nad RAMTOPem odpovídající počet volných bajtů, které jsou chráněny před přepsáním basicovým programem i před příkazem NEW (CLEAR 32499 vytvoří od adresy 32500 sto volných bajtu pro uložení programů ve strojovém kódu - po 32600 je UDG u 16k verse. Začátek volného prostoru je uložen v systémové proměnné RAMTOP (první volný bajt=RAMTOP+1). Pokud nedojde ke snížení RAMTOP, je tato oblast tvořena jediným bajtem.

RASP
23608
x5C38

Délka varovného tónu při vkládání znaku do řádku delšího, než 2 spodní řádky. Normálně obsahuje hodnotu 64 a může být měněn od 0 do 255. Pro urychlení vkládání dlouhých řádků můžete zkusit POKE 23608,0.

REPDEL
23561
x5C09

Obsahuje dobu, po kterou musí být stlačena klávesa, aby došlo k opakování (v 1/50 s). Normální hodnota je 35, ale může být měněna od 0 do 255 (0 = 256).

REPPER
23562
#5C0A

Délka intervalu opakování při stlačené klávese.

SCR-CT
23692
#5C8C

Obsahuje počet řádků+1, které budou rolovány na obrazovce nahoru, než se objeví sdělení SCROLL?. Pro eliminování sdělení SCROLL? můžeme použít POKE 23692,255.

SEED
23670-23671
#5C76-#5C77

Pomocí systémové proměnné SEED generuje počítač náhodná čísla. Při každém vyvolání funkce RND se hodnota v SEED změní podle následujícího programu:

```
LET SEED =75*(SEED+1)
LET SEED=SEED-65537*INT(SEED/65537)
```

Funkce RANDOMIZE přenese do SEED hodnoty ze dvou nižších bajtu systémové proměnné FRAMES, která pak tvoří nový základ pro RND. Vložíme-li RANDOMIZE N, pak je v SEED hodnota N.

STKBOT
23651-23652
#5C63-#5C64

Adresa dna zásobníku kalkulátoru.

STKEND
23653-23654
#5C65-#5C66

Adresa vrcholu zásobníku kalkulátoru a začátek volné paměti.

STRLEN Délka právě vyhodnocovaného řetězce.
23666-23667
#5C72-#5C73

STRMS Adresy kanálů připojených k jednotlivým proudům. Během
23568-23605 inicializace systému se do prvních 14 bajtu přenesou
#5C10-#5C35 hodnoty z ROM pro stále otevřené kanály K, S, P a R.

SUBPPC Číslo právě vykonávaného příkazu v basicovém řádku.
23623 Můžeme použít ve spojení s ERR-NR, chceme-li do sdělení
#5C47 vložit číslo řádku a příkazu, u kterého došlo k zastavení programu. Je-li L číslo řádku a n číslo příkazu v řádku, bude:

```
POKE 23621,L-256*INT(L/256)   číslo řádku
POKE 23622,INT(L/256)         do PPC
POKE 23623,n                  číslo příkazu do SUBPPC
```

S-POSN Obsahuje číslo sloupce (v 23688) a číslo řádku (v
23688-23689 23689) běžné pozice PRINT (t. j. právě vypisovaného
#5C88-#5C89 znaku) v interním číslování:

číslo sloupce=33-interní číslo sloupce
číslo řádky=24-interní číslo řádky.

SPOSNL Totéž Jako S-POSN, ale pro spodní část obrazovky.
23690-23691
#5C8A-#5C8B

S-TOP Obsahuje číslo řádku, který bude zobrazen jako první na
23660-23661 obrazovce při automatickém výpisu.
#5C6C-#5C6D

TVDATA Obsahuje řídící znaky pro AT a TAB a informaci o barvě.
23566-23567
#5C0E-#5C0F

TVFLAG Příznaky pro obrazovku. Je-li bit 0 nastaven, znamená
23612 to spodní část obrazovky, je-li nulován, znamená to
#5C3C horní část obrazu.

T-ADDR Adresa příští položky v syntaktické tabulce.
23668-23669
#5C74-#5C75

UDG Adresa začátku oblasti pro definované grafické znaky.
23675-23676 Normálně obsahuje 32600 (#7F53) nebo 65368 (#FF58) pro
#5C7B-#5C7C 16/48k RAM.

VARS
23627-23628
I5C4B-**I**5C4C

První adresa oblasti, ve které jsou uloženy hodnoty proměnných. Délku basicového programu dostaneme odečtením hodnoty v systémové proměnné PROG od hodnoty v systémové proměnné VARS.

WORKSP
23649-23650
I5C61-**I**5C62

Obsahuje adresu začátku oblasti "Vstupní data".

X-PTR
23647-23648
I5C5F-**I**5C60

Adresa prvního znaku, ve kterém je syntaktická chyba, t. j. znaku následujícího po "?".

V ROM je uložen tzv. operační systém Spectra. T.j. všechny nutné informace pro procesor Z80. Bez těchto informací v ROM by byl počítač sbírkou jednotlivých součástek, neschopnou jakékoliv činnosti.

Rozdělení ROM Teprve ten, kdo detailně ovládá ROM, je absolutním pánem Spectra a může plně využít všech výhod strojového kódu. Ale právě detailní znalost ROM a jejich jednotlivých podprogramů je nejnáročnější a nutně vyžaduje detailní prostudování některého komentovaného výpisu ROM. V této stručné příručce můžeme dát jen nejzákladnější informace o ROM a jejich podprogramech.

Rozdělení ROM je přibližně následující:

0000-0094 restarty a návazné malé podprogramy

0095-028D tabulky všech symbolů uvedených na klávesnici

028E-03B4 podprogramy pro test klávesnice a vyhodnocení stlačené klávesy

0385-04A9 podprogramy pro tónový generátor

04AA-04C1 krátký nevyužitý podprogram ze ZX81

04C2-09F3 podprogramy pro obsluhu kazetového magnetofonu

09F4-0F2B podprogramy pro výpis znaku na obrazovce

0F2C-11B6 podprogramy pro editování basicového řádku nebo příkazu, t. j. jeho vkládání a změny včetně INPUT

11B7-12A1 inicializační podprogram, který po zapojení počítače provede test RAM, nastaví veškeré počáteční hodnoty systémových proměnných a vypíše na obrazovku sdělení "1982 Sinclair Research Ltd. "

12A2-1A47 hlavní prováděcí podprogram, který přenese vložené řádky do oblasti PROG a vypíše jejich seznam na obrazovce

1A48-1B16 tabulky pro interpretaci basicových příkazu a jejich parametrů pro syntaktickou kontrolu

1B17-24FA hlavní interpretační podprogram jednotlivých basicových příkazů a provádění syntaktického testu

24FB-32C4 podprogramy pro vyhodnocení výrazu v jednotlivých příkazech a funkcích a podprogramy pro matematické operace a přípravu dat pro kalkulátor

32C5-386D podprogram kalkulátoru včetně tabulky odskoků po RST 28

386E-3CFF nevyužitá oblast (pouze hodnoty FF)

3000-3FFF tabulka bajtů jednotlivých ASCII znaků, které mohou být vypsány na obrazovku

A teď trochu podrobněji k jednotlivým částem ROM.

0000
START
RST X0

Start systému po jeho zapojení. Bezprostředně vyvolá inicializační podprogram START/NEW na adrese X11CB.

0008
ERROR-1
RST X8

Podprogram je vyvolán vždy, je-li důvod k vypsání sdělení ve spodní části obrazovky. RST X08 můžeme použít při návratu z programu ve strojovém kódu do basicu místo instrukce RET. Po instrukci RST X08 musí ale následovat bajt XFF, čímž se ukončí program ve strojovém kódu a přejde na pokračování basicového programu. Je možné i vynucené sdělení, když po RST X08 následuje bajt příslušného sdělení (tabulka od adresy X1391 v ROM).

0010
PRINT-A-1
RST X10

Tento podprogram vypíše na obrazovku znak, jehož kód je v registru A (do místa běžné pozice PRINT). Je to jeden z nejuniverzálnějších podprogramu ROM. Lze ho použít ke vložení řídících znaků obrazovky nebo ke změně kanálu. Vzhledem k jeho univerzálnosti se u něj zdržím trochu déle a předběhnu ostatní částí ROM.

Napíšeme-li následující program, vložíme do počítače a spustíme, objeví se na prvním řádku spodní části obrazovky písmeno A:

```
LD   A,X41     ; kód "A" do reg. A
RST  X10       ; vypiš znak
RET            ; zpět do basicu
```

Proč znak A je snad jasné. Ale proč ve spodní části obrazovky? Je to proto, že normálně má Spectrum po zahájení činnosti otevřen kanál K, který je určený pro spodní část obrazovky. Chceme-li náš znak vypsat do horní části, musíme nejdříve otevřít kanál S, příslušný pro tuto část obrazovky (kanál P je pak pro tiskárnu). Provedeme to následujícím programem:

```
LD   A,2       ; číslo proudu do A
CALL ▮1601     ; otevři kanál S
```

CALL 1601 vyvolá podprogram na adrese 1601, který otevře kanál S. Stejného účinku dosáhneme následujícím programem:

```
XOR  A         ; nuluje A
LD   (▮5C3C),A ; 0 do syst. prom. TVFLAG
```

což je vlastně výsledek předešlého programu.

Napíšeme-li nyní:

```
LD   A,2
CALL ▮1601     ; kanál S
LD   A,"A"     ; znak A
RST  ▮10       ; výpis
RET            ; zpět Basic
```

objeví se A v levém horním rohu obrazovky. Levý horní roh proto, že je to v tomto případě běžná pozice PRINT (při prázdné obrazovce).

Rychlost strojového kódu můžete vyzkoušet následujícím doplněním:

```
     LD   A,2
     CALL #1601      ; kanál S
OPAK LD   A,"A"      ; znak A
     RST  #10        ; výpis
     JR   OPAK       ; opakování od ld A, 41
     RET             ; zpět Basic
```

a porovnáním tohoto programu s Basicovým:

```
     1 PRINT "A";
     2 GO TO 1
```

Většinou však nechceme psát pouze do levého horního rohu obrazovky, ale na určité místo (např. na řádek 10 a sloupec 15). Máme pak víc možností. Jednou z nich je využít řídících znaků obrazovky. Použijeme řídící znak AT, jehož kód je #16. Napíšeme-li pak program:

```
LD   A,2       ;
CALL ▮1601     ; kanál S
LD   A,▮16     ; řídící znak AT
RST  ▮10       ;proveď
LD   A,10      ; řádek 10d
RST  ▮10       ; proveď
LD   A,15      ; sloupec 15d
RST  ▮10       ; proveď
LD   A,"A"     ; znak A
RST  ▮10       ; proveď
RET            ; Basic
```

objeví se znak A na požadovaném místě obrazovky.

Máme však i další možnost. Můžeme použít podprogram ROM na adrese ▮0DD9, který na obrazovku nastaví pozici PRINT podle parametru v registrovém páru BC, přičemž v B je číslo řádku a v C číslo sloupce (interní číslování - řádek ▮0E a sloupec ▮12). Pak náš program pro výpis na obrazovku dostane tvar:

```
LD   B,#0E      ; basic řádek 10
LD   C,#12      ; basic sloupec 15
CALL 0DD0       ; PRINT AT
```

a pokračuje normálně:

```
LD   A,"A"      ; znak A
RST  #10        ; proveď
RET             ; Basic
```

Jako první musí samozřejmě být:

```
LD   A,2
CALL #1601      ; kanál S
```

Místo dvou instrukcí LD B, 0E a LD C, 12 můžeme napsat

```
LD    BC,#0E12
```

Pokud napíšeme program v následujícím pořadí:

```
LD   A,2
CALL ▮1601
LD   A,"A"
LD   BC,▮E12
CALL ▮DD9
RST  ▮10
RET
```

nebude fungovat, resp. na obrazovce se objeví něco jiného. Je to proto, poněvadž podprogram CALL ▮DB9 přepíše hodnotu v registru A. Musíme proto před vyvoláním CALL ▮DD9 původní hodnotu v reg. A uchovat. Nejjednodušší způsob je PUSH AF a pak POP AF. Program pak bude vypadat následovně:

```
LD   A,2
CALL ▮1601     ; kanál S
LD   A,"A"     ; znak A
LD   BC,▮E12   ; řádek a sloupec
PUSH AF        ; uchování hodnot v A
CALL ▮DD9      ; PRINT AT
POP  AF        ; původní hodnoty do A
RST  ▮10       ; výpis
RET            ; Basic
```

Člověk je ale náročný a požaduje vypsání více znaků na obrazovce najednou. Můžeme pokračovat osvědčeným způsobem, poněvadž RST ⧗10 vypíše nový znak na následující pozici PRINT, t.j. za předchozí znak.

Vypadalo by to následovně:

```
LD   A,2
CALL ⧗1601
LD   A,"A"     ; znak A
RST  ⧗10
LD   A,"H"     ; znak H
RST  ⧗10
LD   A,"O"     ; znak O
RST  ⧗10
LD   A,"J"     ; znak J
RST  ⧗10
RET
```

Je to ale zdlouhavé a "nemotorné" a zabírá zbytečně mnoho místa. Nic nám ale nebrání, použít podprogramu PR-STRING na adrese ⧗203C, určeného pro výpis řetězců. Před vyvoláním podprogramu musí být v DE adresa prvního znaku řetězce a v BC počet znaků v řetězci, a samozřejmě od dané adresy jednotlivé kódy znaků v řetězci. Pak nám pro vypsání "AHOJ" na obrazovku postačí následující program. (zvolil jsem jeho uložení do vyrovnávací paměť tiskárny):

```
5B00 DEFB #11,6,#10,2    ; paper=6, ink=2
     DEFB #12,1          ; flash=1
     DEFM "AHOJ"
5B0A LD    A,2
     CALL #1601          ; kanál S
     LD    DE,#5B00      ; adresa 1. znaku
     LD    BC,10         ; 10 znaků
     CALL #203C          ; podprg. PR-STR.
5B1B RET                 ; basic
```

Podprogram CALL #203C není nic jiného než smyčka, která za vás provede opakovaně LD A,N a RST #10 z předchozího příkladu, postupně pro všechny hodnoty uložené v bajtech od adresy v registrovém páru DE.

Pomocí CALL #0DD9 je možno určit pozici tisku na obrazovce, musíme dávat ale pozor na přepsání registrů a použít PUSH/POP. Není snad třeba zdůrazňovat, že řetězec se znaky může být na zcela jiném místě paměti, než vlastní prováděcí program, který začíná obvyklým LD A,2. Znamená to, že uvedený program musíme odstartovat pomocí RAND USR 23306, nikoliv RAND USR 23296. Naše znaky, které chceme vypsat na obrazovku, včetně řídicích znaků obrazovky, můžeme dokonce uložit do řetězcové proměnné (např. B$).

Pak ovšem musíme před vyvoláním programu zjistit adresu začátku naší řetězcové proměnné v oblasti VARS, zvýšit tuto hodnotu o 3 a přenést do DE. V bajtech VARS+1 a VARS+2 je délka řetězce, kterou přeneseme do BC.

V předchozím podprogramu jsme v jednotlivých bajtech na začátku programu (X5B00) definovali barvu pro PAPER a INK a blikání nápisu. Můžeme si to dovolit proto, že pomocí RST X10 můžeme realizovat veškeré řídící znaky na obrazovky, nikoliv pouze AT, jak již bylo uvedeno. Zásada je, že po řídicím znaku musí následovat RST X10 a pak příslušné parametry řídícího znaku, následované vždy RST X10.

Je třeba ještě upozornit, že RST X10, podobně jako CALL X0DD9, přepíše všechny registry. Používáme-li BC jako počítadlo při opakování, musíme před každým vyvoláním RST X10 pomocí PUSH BC obsah BC uložit do zásobníku a pak vrátit pomocí POP BC. Pochopitelně že pro všechny uvedené řídící znaky obrazovky můžeme použít podprogramu PR-STRING podle příkladu na předchozí straně.

0018
SET-CHAR
RST X18

Podprogram uloží do A kód aktuálního znaku, t.j. znaku adresovaného systémovou proměnnou CH-ADD

0020
NEXT-CHAR
RST X20

Podprogram uloží do A kód znaku následujícího po znaku, který by do A uložil RST X18.

Co je to ale aktuální a následující znak? Nejlépe je to vidět, vložíme-li následující krátké programy:

```
RST 18    a    RST 20
RST 10         RST 10
RET            RET
```

a odstartujeme je PRINT USR XXXX,A (kde XXXX je adresa začátku programu). První program vypíše na obrazovku "," a druhý vypíše "A". Je to proto, že prvním znakem na který počítač po provedení PRINT USR XXXX narazí, je právě čárka a dalším znakem je A.

Pomocí smyčky tak na obrazovku můžeme vypsat i více znaků, následujících po příkazu PRINT USR XXXX. Je přitom ale důležité mít na paměti, že RST 20 zanedbává mezery (a samozřejmé přepisuje registry). Napíšeme-li smyčku ve tvaru:

```
ZAC :     RST  20        ; přísun následujícího znaku
          RST  10        ; výpis znaku
          JR   ZAC       ; znovu
```

a odstartujeme PRINT USR XXXX, ABCDE, pak program sice vypíše na obrazovku požadované "ABCDE", ale pak si dělá co chce, poněvadž mu chybí RET pro návrat do basicu. Můžeme tomu odpomoci tím, že si zvolíme návratový znak, při němž se program vrátí. Může to vypadat třeba následovně:

```
ZAC :     RST  20        ; znak do A
          CP   "0"       ; je v A znak "0" ?
          RET  Z         ; ano, zpět do basicu
          RST  10        ; ne, vypíše znak
          JP   ZAC       ; opakování
```

Když pak program spustíme PRINT USR XXXX, ABCDE0, vypíše nám na obrazovku "ABCDE". Znak 0 pak znamená konec a návrat do basicu pomocí instrukce RET Z. Zkusíme-li tento program vyvolat PRINT USR XXXX,AB CD E0, pak se na obrazovce objeví stejně jako předtím "ABCDE" bez mezer.

Chceme-li nápis umístit do jiného místa obrazovky, musíme podobně jako v předchozí části, nastavit kanál a pozici PRINT na obrazovce.

Řetězec znaku, které chceme zobrazit, můžeme také umístit do jiného místa v paměti a změnit adresu aktuálního znaku v systémové proměnné CH-ADD.

0028
FP-CALC
RST **1**28

Tento podprogram je sice sám o sobě velice jednoduchý, má pouze 3 bajty, ale vzápětí vyvolá podprogram CALCULATE na adrese **1**335B, což je kalkulátor. Jeho pomocí je možno provádět veškeré funkce včetně rozhodování a skoku. Poněvadž se jedná o velice užitečný podprogram, zase přeskočím a uvedu ho zde celý.

Podprogram kalkulátoru vyvoláváme instrukcí RST **1**28, po níž následuje kód požadované operace kalkulátoru. Jednotlivé kódy a jimi vyvolané operace jsou:

kód příslušná činnost

00 je-li výrok pravdivý, relativní skok daný hodnotou následujícího bajtu (jako jr, DIS)

01 výměna poslední a předposlední hodnoty na vrcholu zásobníku

02 vynuluje vrchol zásobníku

03 odečte hodnotu uloženou pod vrcholem zásobníku od hodnoty na vrcholu zásobníku a výsledek uloží na vrchol zásobníku (-)

04 vynásobí dvě hodnoty uložené na vrcholu zásobníku a pod vrcholem zásobníku a výsledek uloží na vrchol zásobníku (*)

05 vydělí vrchol hodnotou pod vrcholem a výsledek uloží na vrchol zásobníku (:)

06 umocní vrchol hodnotou pod vrcholem a výsledek uloží na vrchol zásobníku (x na y)

07 provede funkci X OR Y, výsledek je na vrcholu zásobníku a je X je-li y=0, jinak 1 (OR) . . X je na vrcholu, Y pod

08 provede X AND Y, výsledek na vrcholu je X, je-li Y 0, jinak 0 (AND)

09 provede <> pro čísla, výsledek na vrcholu je 0-nepravda, 1 -pravda (<>)

0A provede < pro čísla, výsledek jako 09 (<)

0B provede porovnání řetězců $<$, výsledek jako 09 ($<$)

0C provede =< pro čísla, výsledek viz 09 (=<)

0D provede >= pro čísla, výsledek viz 09 (>=)

0F sečte vrchol a hodnotu pod vrcholem a výsledek uloží na vrchol zásobníku (+)

10 provede $ AND číslo, výsledek na vrcholu je x$, je-li Y 0, jinak prázdný řetězec (X$ AND Y)

11 provede $=< , výsledek viz 09

12 provede $>= , výsledek viz 09

13 provede $<> , výsledek viz 09

14 provede $> , výsledek viz 09

15 provede $< , výsledek viz 09

16 provede $= , výsledek viz 09

17 provede X$ + Y$, výsledek uloží na vrchol

18 provede VAL$, výsledek uloží na vrchol

19 provede USR$, výsledek uloží na vrchol

1A provede READ IN, t. j. přenese číslo z vrcholu do registru A, a současně změní na číslo typu integer

1B provede změnu znaménka hodnoty na vrcholu (NEG)

1C CODE

1D VAL

1E LEN

1F SIN

20 COS

21 TAN

22 ARC SIN = ASN

23 ARC COS = ACS

24 ARC TAN = ATN

25 LN

26 EXP = E↑X

27 INT

28 SQR

29 SGN

2A ABS

2B PEEK

2C IN

2D provede USR N, vyvolá program ve strojovém kódu na adrese dané hodnotou na vrcholu zásobníku, po ukončení tohoto podprogramu se na vrchol uloží obsah BC

2E STR$

2F CHR$

30 provede NOT, uloží na vrchol 1, byla-li předchozí hodnota na vrcholu 0, jinak dá na vrchol 0

31 provede DUP, okopíruje vrchol na nový vrchol zásobníku

32 provede dělení N modulo M, t. j. vydělí číslo N číslem M a výsledek uloží následovně: INT(N/M) na vrchol a zbytek pod vrchol, před vyvoláním je M na vrcholu a N pod

33 provede nepodmíněný relativní skok o velikosti dané následujícím bajtem (jr DIS)

34 uloží na vrchol číslo typu floating point, jehož hodnota je dána následujícími pěti bajty

35 provede relativní skok daný následujícím bajtem a sníží hodnotu v syst. proměnné BREG o 1, pokračuje, dokud není BREG = 0 (jako djnz), do BREG dáme předem počet opakování 36 provede 0 vrcholu zásobníku, výsledek na novém vrcholu bude 1, je-li splněno, jinak 0

37 provede 0, ostatní viz 36

38 konec výpočtu kalkulátoru

39 provede redukci argumentu SIN a COS v rozmezí -0.5 a 0.5

3A provede oddělení celočíselné části se zaokrouhlením

3B FP-CALC-2, je to podprogram pro provedení jediné aritm. operace, jejíž kód je v syst. proměnné BREG

3C převede číslo ve tvaru xEn na floating point (n na vrcholu, x pod)

3D převede číslo z vrcholu typu integer na floating point

86 generuje polynomický rozvoj pro výpočet trigonometrických funkcí, následuje 6 konstant float. p. 88 viz 86, ale 8 konstant

8C viz 86, ale 12 konstant

A0 uloží na vrchol 0

A1 uloží na vrchol 1

A2 uloží na vrchol 0. 5

A3 uloží na vrchol PI/2

A4 uloží na vrchol 10

C0 přenese číslo typu floating point z vrcholu do zásobníkové paměti č. 0

C1 viz C0, ale do paměti 1

C2 viz C0, ale do 2

C3 viz C0, ale do 3

C4 viz C0, ale do 4

C5 viz C0, ale do 5

E0 přenese číslo ze zásobníkové paměti č. 0 na vrchol

E1 viz E0, ale z paměti 1

E2 viz E0, ale z 2

E3 viz E0, ale z 3

E4 viz E0, ale z 4

E5 viz E0, ale z 5

Jako zásobníkovou paměť používá kalkulátor 30 bajtů systémové proměnné MEMBOT. Při přenosech ze zásobníku do zásobníkové paměti zůstává hodnota na vrcholu zásobníku zachovaná, při přenosu ze zásobníkové paměti na vrchol zásobníku se přenesená hodnota umístí na vrchol zásobníku a původní hodnota na vrcholu zásobníku se přesune pod vrchol.

Tím jsem vyčerpal všechny funkce kalkulátoru. Zůstává otevřená otázka, jak přenést počáteční hodnoty na vrchol zásobníku, poněvadž instrukce RST 28 předpokládá, že na vrcholu zásobníku je již daná hodnota, se kterou se bude provádět předepsaná funkce. V ROM Spectra je pro danou úlohu několik podprogramů:

2D28 přenese hodnotu z registru A na vrchol zásobníku a
STACK-A převede ho na číslo typu floating point

2D2B přenese BC viz call 2D28
STACK-BC

2AB1 přenese B na vrchol a pak CDEA pod vrchol
STACK-ST-0

24FB vyhodnotí výraz a výsledek uloží na vrchol
SCANNING

Pro opačný postup, t. j. přesunutí hodnot z vrcholu zásobníku do registrů jsou v ROM následující podprogramy:

2DD5 **FP-TO-A** — přenese hodnotu z vrcholu zásobníku do registru A

2314 **STK-TO-A** — stejné, jako call 2DD5, navíc v C 1, je-li A kladné, nebo FF je-li záporné

2DA2 **FP-TO-BC** — přenese z vrcholu hodnotu do BC a současně uloží do A nižší bajt výsledku

2307 **STK-TO-BC** — provede totéž, jako call 2DA2, navíc po DE uloží znaménko výsledku (1, FF viz výše)

2BF1 **STK-FETCH** — uloží 5 bajtů z vrcholu do registrů BCDEA

2DE3 — vypíše na obrazovku poslední hodnotu t. j. vrchol

Můžeme také použít kód 34, jehož pomocí lze na vrchol zásobníku uložit číslo v pětibajtovém vyjádření.

Chceme-li vypočítat SQR(sin x↑2+cos x↑2), můžeme postupovat následovně:

```
LD   A,X  ;                  hodnota x do A
CALL 2D28 ;x                 přenese hodnotu na vrchol
RST I28   ;x                 zásobníku a začne výpočet
DEFB 31   ;x,x               zdvojí vrchol zas. (DUP)
DEFB 1F   ;x,sin x           vypočte sin x
DEFB 31   ;x,sin x,sin x     zdvojí sin x
DEFB C4   ;x,sin x↑2         vypočte sin x * sin x
DEFB 01   ;sin x↑2,x         přehodí x a sin x↑2
DEFB 20   ;sin x↑2,cos x     vypočte cos x
DEFB 31   ;sin x↑2,cosx,cosx zdvojí cos x
DEFB C4   ;sin x↑2,cos x↑2   vypočte cos x * cos x
DEFB CF   ;sin x↑2+cos x↑2   sečte sin x↑2 a cos x↑2
DEFB 28   ;SQR sinx↑2+cosx↑2 vypočte odmocninu
DEFB 38   ;--- // --------    konec výpočtu
CALL I2DA2;                  přenese hodnotu do BC
RET       ;                  zpět do Basicu

narůstání zásobníku ---------->
```

Pro výpočet můžeme použít i například zásobníkové paměti kalkulátoru. Zde pozor, protože při výpočtu trigonometrických a jiných funkcí používá kalkulátor zásobníkové paměti č. 0, 1 a 2. První řádek je sice jednoduchý, ale málo univerzální. Můžeme použít například RST I20; přenese hodnotu do registru A. Zde musíme program spustit PRINT USR XXXX,x, kde x je naše číslo.

Prostřednictvím kalkulátoru můžeme provádět podmíněné i nepodmíněné skoky, vyvolávat další podprogramy, manipulovat s řetězci, provádět logické operace apod.

0030
BC-SPACES
RST I30

Podprogram vyvolá další podprogram, jehož prostřednictvím vytvoří BC volných bajtů v oblasti pracovního prostoru (workspace). BC = obsah registrového páru BC.

0038
MASK-INT
RST #38

50x za vteřinu vyvolá ULA obvod maskované přerušení, během něhož provádí Z80 test klávesnice a vyhodnocuje stlačenou klávesu. Přerušení vyvolá RST #38. Každé přerušení zvětší počítadlo v systémové proměnné FRAMES a vyvolá podprogram testující stav klávesnice, tj. byla-li stl. klávesa a která.

Je třeba ještě se zmínit o způsobu, jak je prováděn test klávesnice. Celá klávesnice je rozdělena do 8 bran, jejichž adresy jsou v následující tabulce.

adr. Segmentu		klávesnice	adr. segmentu	
F7FE	63486	1 2 3 4 5 6 7 8 9 0	EFFE	61438
FBFE	64510	Q W E R T Y U I O P	DFFE	57342
FDFE	65022	A S D F G H J K L en.	8FFE	49150
FEFE	65278	CS Z X C V B N M sssp	7FFE	32766

Podprogram na adrese #028E testuje postupně jednotlivé segmenty pomocí instrukce IN A,(C), přičemž adresu segmentu uloží do BC. Výsledkem tohoto podprogramu jsou hodnoty v DE. Není-li stisknuta žádná klávesa, DE=#FFFF, v opačném případě obsahuje reg. E hodnotu mezi 00 a 27, které jsou přiřazeny jednotlivým klávesám.

Tabulka hodnot:

1-5...	24	1C	14	0C	04	03	0B	13	1B	23	... 6-0
Q-T...	25	1D	15	0D	05	02	0A	12	1A	22	... Y-P
A-G...	26	1E	16	0E	06	01	09	11	19	21	... G-en
cs-V..	27	1F	17	0F	07	00	08	10	18	20	... B-sp

Dojde-li ke stlačení současně s jednou "shift" klávesou, obsahuje reg. D hodnotu "shift" klávesy. Dojde-li ke stlačení obou "shift" kláves, obsahuje D hodnotu Caps Shift a E hodnotu Symbol Shift. Je-li stlačena klávesa "A", je v DE hodnota #FF26, jsou-li současně stlačeny SS a "A", pak je v DE #1826. Caps Shift a Symbol Shift dají #2718. Někdy je možno tohoto podprogramu a hodnot v DE použít v programech ve strojovém kódu pro testování stlačené klávesy.

Po ukončení popsaného podprogramu pokračuje provádění dalšího podprogramu od adresy #02C3, jehož výsledkem je vypočtení kódu stlačené klávesy (včetně všech "shiftu") a jeho uložení do systémové proměnné LAST-K. Tím je vlastně ukončena činnost RST #38.

Adresou #0038 končí podprogramy tzv. nulté stránky vyvolávané instrukcí RST a začínají podprogramy vyvolávané instrukcí CALL.

0066
RESET

Zde začíná obsluha nemaskovaného přerušení. Spectrum při své normální činnosti NMI nepoužívá, ale je možné jej tímto podprogramem obsloužit, kdyby nebylo obráceného testu obsahu systémové proměnné.

028E
KEY-SCAN

Podprogram pro test klávesnice (popsaný již u RST #38), vyvolaný podprogramem Keyboard.

02BF
KEYBOARD

Podprogram Keyboard, vyvolaný RST 38H.

03B5
BEEPER

Od této adresy začínají podprogramy tónového generátoru. Zabudovaný reproduktor je aktivován bitem 4 portu FEH. Následuje-li po OUT FEH bit4=0, je mikrofon aktivován, při bit4=1 je mikrofon vypnut. Rytmem zapínání a vypínání mikrofonu je dána frekvence tónu.

Podprogram na adrese 03B5H lze využít pro generování tónu následujícím programem:

```
LD    DE,délka tónu ; např. 0105H pro 1 sec.
LD    HL,výška tónu ; např. 066AH pro střední "C"
CALL 03B5H          ; tón
RET                 ; Basic
```

Obdobně lze vypočítat hodnoty pro jiné tóny, i když rychlejší asi bude metoda zkusmého dosažení hodnot.

Stejného výsledku dosáhneme použitím podprogramu pro funkci BEEP na adrese 03F8H. Před voláním musí být hodnoty pro dobu i výšku tónu na vrcholu zásobníku. Tyto hodnoty se však od předchozích liší.

Je zřejmé, že podprogram 03B5H přepisuje všechny registry, které je tudíž nutné uložit instrukcí PUSH. Během provádění též nejdou hodiny.

04C2
SA-BYTES

Zde začínají podprogramy pro obsluhu magnetofonu, tj. SAVE, LOAD, VERIFY a MERGE.

Hlavní podprogram začíná na adrese ¤0605 a podle hodnoty systémové proměnné TADDR vyvolá příslušnou funkci.

```
(TADDR) - ¤E0 = 0   -  SAVE
                1   -  LOAD
                2   -  VERIFY
                3   -  MERGE
```

Jednotlivé funkce vyvoláme pomocí:

```
LD   A,0 až 4        ;podle požadované fce
CALL ¤060B
```

Podprogramy pro jednotlivé fce začínají na adresách:

```
¤07CB - VERIFY
¤0808 - LOAD
¤08B6 - MERGE
¤0970 - SAVE
```

Vlastní nahrávání pro LOAD, VERIFY a MERGE na adr. ¤0556 a při SAVE na adr. ¤04C2.

09F4
PRINT-OUT

Na této adrese začíná program pro vypsání znaku uloženého v reg. A buď na obrazovku nebo tiskárnu. Tento podprogram je vyvolán podprogramem RST ℣10 poté, co je do HL přenesena ze systémové proměnné CURCHL velikost odskoku pro výpis na obrazovku. Tento program končí tím, že přenese na PRINT-pozici bitové mapy jednotlivé bajty znaku, který má být vypsán. Tyto bajty jsou uloženy v tabulce znaků, začínající na adrese ℣3D00. Bitová mapa je zobrazena na obrazovce pomocí obvodu ULA.

PRINT pozici určuje podprogram na adr. ℣0B03 v závislosti na obsahu systémových proměnných SPOSN a DF CC a podle nastavení bitu 1 v systémové proměnné FLAGS (nastavení bitu znamená tisk tiskárnou) a dále podle nastavení bitu 0 v systémové proměnné TVFLAG (je-li nulován je tisk v horní části obrazovky, je-li nastaven, pak v dolní části - v tom případě se používají hodnoty ze syst. proměnných SPOSNL a DFCCL). Hodnota ze SPOSN je v BC a hodnota z DF CC je v HL.

Pro výpis na horní část obrazovky můžeme tento podprogram upravit:

```
LD   BC,číslo řádku a sloupce ;řádek v B, sloupec v C
LD   HL,adresa znaku v bitové mapě ;první bajt
XOR  A
LD   (5C3C),A  ;nastaví kanál 5 pro horní část obr.
CALL 0B10
```

0D6B
CLS

Podprogram CLS vymaže obrazovku.

0E44
CL-LINE

Podprogram pro vymazání B spodních řádků obrazovky. B je hodnota v registru B.

0E00
CL-SCROLL

Podprogram pro rolování obrazovky. Počet rolovaných řádků v registru B.

0E88
CL-ATTR

Podprogram vypočte k dané adrese bajtu v Bitové mapě adresu příslušného atributu v oblasti atributů. Adresa bajtu je v HL, výsledek v BC.

0F2C
EDITOR

Zde začíná podprogram pro editování řádku buď při jeho vkládání z klávesnice, nebo při editaci řádku z obrazovky. Též obsluhuje vkládání dat příkazem INPUT.

0FA9
ED-EDIT

Příkaz EDIT. Přenese do spodní části obrazovky kurzorem označený řádek.

10A8
KEY-INPUT

Podprogram vyhodnotí kód stlačené klávesy a uloží do systémové proměnné LAST K.

11B7
NEW

Od této adresy začíná inicializace systému po příkazu NEW (NEW =CALL 11B7).

11CB
START/NEW

Od této adresy pak inicializace systému po zapnutí počítače.

1219
RAM-SET

Základní nastavení systémových proměnných. Inicializace končí vypsáním "Copyrigthu" na obrazovce.

15D4
WAIT-KEY

Podprogram čeká na stlačení libovolné klávesy a pak pokračuje v provádění následující části programu. Používá se například u LOAD.

15EF
OUT-CODE

Podprogram pro konečnou fázi vypsání znaku na obrazovku, tj. přenesení příslušného bajtu z tabulky znaku umístěné od adresy #3D00 do Bitové mapy, odkud jsou již znaky dále přenášeny pomocí ULA.

1601
CHAN-OPEN

Podprogram pro otevření kanálu. V registru A musí být číslo odpovídajícího proudu. Je to #FD, 0 nebo 1 pro kanál K (spodní část obrazovky), #FE pro kanál S (horní část obrazovky), #FF pro kanál R (pracovní prostor) a 3 pro kanál P (tiskárna). Výsledkem otevření jednotlivých kanálů je pak následující nastavení bitu v systémových proměnných:

```
kanál K: set 0, TVFLAG ; spod. část obrazovky
         res 5, FLAGS  ; je možno klávesu
         set 4, FLAGS2 ; kanál K
         res 1, FLAGS  ; ne tiskárna

kanál S: res 0, TVFLAG ; celá obrazovka
         res 1, FLAGS  ; ne tiskárna

kanál P: set 1, FLAGS  ; tiskárna použita
```

1655
MAKE-ROOM

Podprogram vytvoří potřebný počet volných bajtů. Počet bajtů v BC. HL ukazuje na l. pozici nad nově vytvořenou zónou volných bajtů. Po skončení podprogramu je v HL adr. bajtu před začátkem nově vytvořené zóny, v DE poslední adr. této zóny, čili volné bajty jsou od HL+l do DE.

16DC
INDEXER

Podprogram k prohledávání tabulek v ROM. V HL je adresa začátku tabulky a v C je kód hledaného znaku. Nalezení znaku ohlásí nastavení indikátoru přenosu (SCF).

1855
OUT-LINE

Vypíše na obrazovku basicový řádek, jehož počáteční adresa (tj. adresa l. bajtu čísla řádku) je v HL.

196E
LIN-ADDR

Vyhledá adresu počátku basicového řádku, jehož číslo je v HL. Po skončení podprogramu je v HL adresa začátku hledaného řádku, případně adresa následujícího (není-li v podprogramu zadaný), v DE je začátek předchozího řádku.

1A1B
OUT-NUM-1

Vypíše na obrazovku obsah registrového páru BC (dekadicky).

NĚKTERÉ BASICOVÉ PŘÍKAZY

17F9-LIST	1E27-DATA	1EED-GO SUB
17F5-LLIST	1E42-RESTORE	1F23-RETURN
1B8A-RUN	1E4F-RANDOMIZE	1F3A-PAUSE
1BB2-REM	1E5F-CONTINUE	1F60-DEF FN
1CEE-STOP	1E67-GO TO	1FC9-LPRINT
1CF0-IF	1E7A-OUT	1FCD-PRINT
1D03-FOR	1E80-POKE	2089-INPUT
1DAB-NEXT	1EA1-RUN	2294-BORDER
1DEC-READ	1EAC-CLEAR	

1F1A
FREE-MEM

Vypočte adresu prvního volného místa v paměti. Celkový rozsah volné paměti pak získáme pomocí:

PRINT 65536-USR 7962 ; ▮1F1A=7962

nebo
PRINT 32768 - USR 7962 pro 16k.

1F54
BREAK-KEY

Test klávesy BREAK. CY=0 byl BREAK, CY=1 nebyl break.

22AA
PIXEL-ADD

Vypočte adresu bodu jemné grafiky na obrazovce. V BC je adresa bodu (x, y)(u PLOT, DRAW a CIRCLE se souřadnice ve strojovém kódu shodují s basicovými, tj. levý dolní roh (0, 0), pravý horní (255, 175)). Po skončení podprogramu je v HL adresa příslušného bajtu v Bitové mapě a v A je pozice bitu.

22CB
POINT-SUB

Před vyvoláním musí být na zásobníku kalkulátoru souřadnice bodu (x, y), po skončení je poslední hodnota na zásobníku 1, je-li bod=barva ink, nebo 0, je-li bod=barva paper.

22DC
PLOT

Před vyvoláním na zásobníku (x, y) - poslední hodnota.

22E5
PLOT-SUB

Totéž jako ₭22DC, rozdíl je v zadání souřadnic bodu:v B je y a v C je x. Bod o souřadnicích x=100 y=50 zobrazíme pomocí:

```
LD    A,02
CALL ₭1601          ; kanál S
LD    BC,₭3264      ; y=50, x=100
CALL ₭22E5          ; podpr. PLOT
RET                 ; zpět Basic
```

2320 Začátek podprogram CIRCLE. Vlastní podprogram začíná na adrese #232D. Před vyvoláním musí být na zásobníku kalkulátoru hodnoty x, y a r (poslední hodnota). Basicovému CIRCLE 100, 100, 50 bude odpovídat program ve strojovém kódu:

```
LD   A,2
CALL #1601          ; kanál S
LD   A, 100         ; x=100 do A
CALL #2D28          ; A na zásobník
LD   A,100          ; y=100 do A
CALL #2D28          ; A na zásobník
LD   A,50           ; r=50d do A
CALL #2D28          ; A na zásobník
CALL #232D          ; kružnice
RET
```

Pokud tento program nabude správně fungovat, vložte před LD A,100 ještě:
EXX
PUSH HL EXX
a za CALL #232D:
EXX
POP HL
EXX.

Je to uchování obsahu reg. páru HL .

24B7
DRAW-LINE Rovnou přímku z bodu x0, y0 do bodu x, y zobrazíme buď pomocí CALL #24B7, když jsme před tím uložili na zásobník kalkulátoru hodnoty x, y a do syst. proměnné COORDS hodnoty x0, y0.

Lze použít i podprogram X24BA, což je totéž jako X24B7, jen jiné zadání souřadnic: v B je ABS y, v C je ABS x, v D je SGN y, v E je SGN x, COORDS jako v předchozím . Přímku z bodu 50,50 do bodu 100,100 vykreslíme pomocí:

```
LD   A,2
CALL X1601     ;kanál S
EXX
PUSH HL
EXX            ;uchování HL
LD   A,50      ;x0=50 do A
LD   (X5C7D),A ;A do COORDS-x /pokud je x y
LD   (X5C7E),A ;A do COORDS-y   je třeba LDA, N/
LD   BC,X6464  ;x=y=100 do BC
LD   DE,X0101  ;x i y kladné hodnoty
CALL X24BA     ;podprogram DRAW
EXX
POP  HL
EXX            ;do HL' původní hodnota
RET            ; zpět Basic
```

2394 Zobrazení části kruhu (basicový DRAW x, y, a). Hodnoty x, y, a (poslední hodnota)jsou na zásobníku kalkulátoru , x0 a y0 v COORDS.

24FB
SCANNING

Od této adresy začínají podprogramy vyhodnocení výrazu. Sám podprogram ▮24FB je velmi důležitý. Vyhodnotí výraz a výsledek uloží na vrchol zásobníku kalkulátoru, odkud si pak vyhodnocené hodnoty vybírají další podprogramy pro příkazy. K tomuto podprogramu malý příklad: Chceme vyhodnotit (vypočíst) hodnotu výrazu PI*3+2. Mohli bychom použít kalkulátoru, ale následujícím programem to jde rychleji.

```
RST ▮20          ;přísun následujícího znaku
CALL ▮24FB       ;SCANN-vypočte hod. výr. a uloží na SP
CALL ▮2DE3       ;vypíše obsah vrcholu SP na obrazovku
RET              ;Basic
```

Tento podprogram musíme vyvolat: PRINT USR xxxxx,PI*3+2

Obdobně můžeme vyhodnotit pro Spectrum přípustné výrazy včetně výrazu se řetězci a pod. Jedná-li se o výrazy s řetězci, pak podpr. CALL ▮24FB uloží na zásobník 5 bajtů, v nichž jsou následující hodnoty:

1. bajt není definován
2. a 3. bajt =adresa začátku řetězce
4. a 5. bajt =délka řetězce

2D3B
INT-TO-FP

Podprogram Integer to Floating point. Změní číslo typu integer uložené na SP na tvar floating point a uloží ho tamtéž.

2D4F
E-TO-FP

Začátek aritmetických podprogramů používaných kalkulátorem. Sám podprogram na adrese ₭2D4F změní číslo typu xE↑n, tj. desetinné exponenciální číslo na typ float. point, které uloží na SP.

335B
CALCULATE

Na této adrese začíná podprogram kalkulátoru a dál následují jednotlivé operace kalkulátoru vyvolané pomocí kódu za RST ₭28.

3D00

Na této adrese začíná tabulka znaků, které může Spectrum zobrazit na obrazovce (nejedná se o tzv. definované znaky) Každý znak sestává z 8 po sobě následujících bajtů. Adresa prvního bajtu znaku je B*kód znaku+₭3000. Např. kód "a" je ₭61, první bajt znaku "a" v tabulce je na adrese 8*₭61+₭3000=₭3FC8.

V index reg. IY je hodnota ▮5C3A

index IY	HEX	DEC	název	stručná charakteristika
C6	5CD0	23552	KSTATE	- vyhodnocení stlačené klávesy
CE	5C08	23560	LAST K	- kód posl. kl. s ohledem na shifty
CF	5C09	23561	REPDEL	- doba k REPEAT (normal. 35h=0, 7s)
D0	5C0A	23562	REPPER	- délka intervalu v REPEAT (0, 1s)
D1	5C0B	23563	DEFADD	- adr. argumentu DEF FN
D3	5C00	23565	K DATA	- barev. inf. z kl. před dalším zprac.
D4	5C0E	23566	TVDATA	- řídicí zn. AT a TAB, inf. o barvě
D6	5C10	2356B	STRMS	- 33bajtů-adr. kanálu přip. k proudům
FC	5C36	23606	CHARS	- adr. tab. znaku mínus 256
FE	5C38	23608	RASP	- délka varov. tónu při překr. řádky
FF	5C39	23609	PIP	- délka pípnutí kl.
00	5C3A	23610	ERR-NR	- kód sdělení mínus 1
01	5C3B	23611	FLAGS	- návěští(D1-tiskárna, D5-další kl.)
02	5C3C	23612	TVFLAG	- návěští obrazovky(D0- 0 dol. část)
03	5C30	23613	ERR SP	- návratová adr. po RST ▮08
05	5C3F	23615	LISTSP	- návr. adr. po aut. výpisu
07	5C41	23617	MODE	- 0=K, L, C; 1=E; 2=G
08	5C42	2361B	NEWPPC	- číslo řádku pro GOTO a GOSUB
0A	5C44	23620	NESPPC	- číslo příkazu v NEWPPC
0B	5C45	23621	PPC	- číslo právě prováděného řádku
0D	5C47	23623	SUBPPC	- číslo právě vyk. přík. v PPC
0E	5C48	23624	BORDCR	- barev. inf. pro spod. část obrazovky
0F	5C49	23625	E PPC	- adr. řádku, kde stojí kurzor
11	5C4B	23627	VARS	- poč. adr. oblasti basic. proměnných
13	5C40	23629	DEST	- adr. prom. při vyhodnoc. (jinak 0)
15	5C4F	23631	CHANS	- zač. obl. kanál. informací

17	5C51	23633	CHURCHL	- adr. právě otevřeného kanálu
19	5C53	23635	PROG	- adr. zač. Basic. programu
1B	5C55	23637	NXTLIN	- adr. následujícího progr. řádku
1D	5C57	23639	DATADD	- adr. čárky za posl. přečtenou pol.
1F	5C59	23641	E LINE	- adr. EDIT v RAM
21	5C5B	23643	K CUR	- adr. kursoru
23	5C5D	23645	CH ADD	- adr. zn. , který bude přenesen do DF
25	5C5F	23647	X PTR	- adr. l. zn. se synt. chybou
27	5C61	23649	WORKSP	- adr. prac. prostoru
29	5C63	23651	STKBOT	- adr. zač. zásobníku kalkulátoru
2B	5C65	23653	STKEND	- adr. vrcholu zas. kal. -zac. vol. pam.
2D	5C67	23655	B REG	- počítadlo kalkulátoru
2E	5C68	23656	MEM	- obvykle (ne vždy) MEMBOT
30	5C6A	23658	FLAGS2	- příznaky
31	5C6B	23659	DF SZ	- počet ř. ve spod. části obr. norm. 2
32	5C6C	23660	S TOP	- č. r. , zobr. jako l. po aut. list.
34	5C6E	23662	OLDPPC	- č. r. po CONTINUE
36	5C70	23664	OSPPC	- č. přík. v OLDPPC
37	5C71	23665	FLAGS	- příznaky
38	5C72	23666	STRLEN	- délka vyhodnocovaného řetězce
3A	5C74	23668	T ADDR	- adr. příští položky v syntakt. tab.
3C	5C76	23670	SEED	- určuje RND a RAND. (2bajte FRAMES)
3E	5C78	23672	FRAMES	- 3bajty zvětšené každých 20ms
41	5C7B	23675	UDG	- adr. zač. graf. znaku
43	5C7D	23677	COORDS	- souřad. x, y pro PLOT a DRAW
45	5C7F	23679	P POSN	- č. sl. LPRINT
46	5C80	23680	PR CC	- spodní bajt příští pozice tisku
47	5C81	23681	rezerva	
48	5C82	23682	ECHO E	- posl. adr. kurz. spod. č. obrazovky
4A	5C84	23684	DF CC	- adr. l. bajtu psaného na obr.
4C	5C86	23686	DFCCL	- dtto, ale pro spodní část
4E	5C88	23688	S POSN	- souřadnice pozice PRINT (tab. 5)
50	5C8A	23690	SPOSNL	- dtto pro dol. část
52	5C8C	23692	SCR CT	- počet ř. +1 k rolování
53	5C8D	23693	ATTR P	- běžné barev, informace
54	5C8E	23694	MASK P	- maska pro ATTR P
55	5C8F	23695	ATTR T	- barev. inf. o vydávaném znaku

56	5C90	23696	MASK T	- maska pro ATTR T
57	5C91	23697	P FLAG	-
58	5C92	23698	MEMBOT	- 30 bajtů-zásobník kalk.
76	5CB0	23728	NMIREG	- volné pro vektor NMI
78	5CB2	23730	RAMTOP	- adr. posl. bajtu Basic. programu
7A	5CB4	23732	P RAMT	- adr. posl. bajtu RAM

Vytiskla tiskárna ÚSNV Pankrác,Praha 4

TISK SNV zak. č. 290/89

Elektronická verzia: 12.11.2011
Peter Turányi alias Softhouse
http://softhouse.speccy.cz

Ďakujem
PSL za skeny