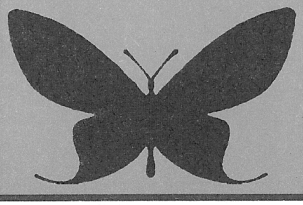
COMMPAS
vydavatelství, služby a distribuce
Dr. Ivan Pavlíček, Pavel Škvrně
p. o. box 80
140 00 Praha 4



COMMPAS
vydavatelství a služby
uživatelům mikropočítačů COMMODORE

KERNAL

přehled rutin KERNALU

COMMODORE 64

Obsah:

KERNAL je systém vyvinutý pro Commodore 64, který do značné míry usnadňuje práci programátorů ve strojovém kódu a ASSEMBLERU.

V určitém zjednodušení lze říci, že se jedná o standardní převodní (skokovou) tabulku pro vstup, výstup a záznam dat v operačním systému. Při rozvinutí systému se mohou jednotlivá paměťová místa v programu paměti ROM změnit.

Převodní tabulka KERNALU obsahuje 39 I/O rutin a další pomocné programy, které jsou prostřednictvím této tabulky přístupné. Tím je možné nejen ušetřit čas, ale i program přizpůsobit pro jiný počítač.

Převodní tabulka je umístěna na poslední stránce paměti. K jejímu použití je nutné zadat požadované parametry pro rutinu KERNALU, pak přejít příkazem JSR na příslušné místo převodní tabulky. Po skončení rutiny KERNAL předá řízení opět strojovému programu.

Proč se užívá převodní tabulky a proč se rovnou nepřechazí k příslušnému podprogramu KERNALU? Jednoduše proto, aby strojové programy pracovaly i při změně vlastních rutin KERNALU či BASICU.

V ostatních provozních systémech mohou být paměťová místa jednotlivých rutin na různých pozicích, převodní tabulka však pracuje stále správně.

Funkce KERNALU

1. Po připojení zdroje je nejdříve vráceno zpět ukazovátko délky a následně je vyřazen decimální mód.
2. Pak KERNAL přezkouší, zda na adrese $8000 HEX je v ROM připraven autostart, je-li tomu tak, pak bude obvyklá inicializace přerušena a řízení je předáno ROM na kartě. Není-li tento autostart v ROM přítomen, pak pokračuje obvyklá systémová inicializace.

3. Jako další inicializuje KERNAL všechny vstupy a výstupy včetně sériového. Oba integrované obvody CIA 6526 jsou uvedeny do pohotovosti pro práci s klávesnicí a je též aktivován časovací obvod 60 Hz. SID je uveden do výchozího postavení (vymazán). Je zvolena konfigurace ROM BASICU a vypnut motor datassette.
4. Dále KERNAL provede zkoušku paměti RAM, inicializuje kurzor a nultou stránku (zeropage) a připraví se také kazetová vyrovnávací paměť (buffer). Testovací rutina RAM je zkouška, kterou nelze vynechat. Začíná na adrese $0300 a postupuje vzestupně. Jestliže se dosáhne poslední adresy z RAM, je zde položena značka pro konec RAM. Spodní okraj vždy začíná na $0800, obrazovková na $0400.
5. Na závěr KERNAL provede následující: vektory vstupu a výstupu jsou uvedeny do standardního postavení, obrazovka a všechny proměnné jsou vymazány. Pro start BASICU je pak připravena nepřímá adresa $A000.

Rutiny KERNALU

	ADRESA		
JMÉNO	HEX	DECIM	FUNKCE
ACPTR	$FFA5	65455	Poslat byte sériovému portu
CHKIN	$FFC6	65478	Otevření kanálu pro vstup
CHKOUT	$FFC9	65481	Otevření kanálu pro výstup
CHRIN	$FFCF	65487	Vstup znaku
CHROUT	$FFD2	65490	Výstup znaku
CIOUT	$FFA8	65448	Výstup byte přes sér. sběrnici
CINT	$FF81	65409	Inicializace obraz. editoru
CLALL	$FFE7	65511	Uzavření všech kanálů a souborů
CLOSE	$FFC3	65475	Uzavření logického dat. souboru
CLRCHN	$FFCC	65484	Uzavření vstup. a výstup. kanálu
GETIN	$FFE4	65508	Čtení znaku z klávesnic. bufferu
IOBASE	$FFF3	65523	Zpětné hlášení adresy I/O zařízení

	ADRESA		
JMÉNO	HEX	DECIM	FUNKCE
IOINIT	$FF84	65412	Inicializace vstup/výstup
LISTEN	$FFB1	65457	LIST pro zařízení na sér. sběrnici
LOAD	$FFD5	65493	Načtení RAM z periferie
MEMBOT	$FF9C	65436	Spodní ukazov. paměti načíst/zadat
MEMTOP	$FF99	65433	Vrchní ukazov. paměti načíst/zadat
OPEN	$FFC0	65472	Otevření logického dat. souboru
PLOT	$FFF0	65520	X,Y-pozice kurzoru číst/zadat
RAMTAS	$FF87	65415	Inicializace RAM, tape-bufferu a obr. paměti $0400
RDTIM	$FFDE	65502	Čtení hodin
READST	$FFB7	65463	Čtení vstup/výstup stavového slova
RESTOR	$FF8A	65418	Navrácení I/O standard. vektoru
SAVE	$FFD8	65496	Obsah RAM zapsat na perif. zaříz.
SCNKEY	$FF9F	65439	Dotaz na klávesnici
SCREEN	$FFFD	65517	Vytvoření X,Y-souřadnic obrazovky
SECOND	$FF93	65429	Přenos sekundární adresy
SETLFS	$FFBA	65466	Zadání první a sekund. log. adresy
SETMSG	$FF90	65424	Řízení hlášení KERNALU
SETNAM	$FFBD	65469	Zadání jmen datových souborů
SETTIM	$FFDB	65499	Zadání času hodin
SETTMO	$FFA2	65442	Zadání časového omezení pro sériovou sběrnici
STOP	$FFE1	65505	Dotaz na tlačítko STOP
TALK	$FFB4	65460	Příkaz TALK pro sériovou sběrnici
TKSA	$FF96	65430	Přenos sekund. adresy po přík. TALK
UDTIM	$FFEA	65514	Zjištění času hodin
UNLSN	$FFAE	65454	Příkaz UNLISTEN pro sér. sběrnici
UNTLK	$FFAB	65451	Příkaz UNTALK pro sér. sběrnici
VECTOR	$FF8D	65421	Zaznamenání paměti RAM

Práce s KERNALEM

Při psaní programu ve strojovém jazyce doporučujeme užívat rutiny operačního systému. Ty totiž obsahují operace jak vstupu a výstupu, tak i přehled o paměťových místech a další funkce. Je zbytečné tyto rutiny stále opakovat. Pomocí přístupu do operačního systému je pak programování ve strojovém kódu rychlejší.

Jak již bylo uvedeno, je KERNAL skokovou tabulkou, jež je vlastně souhrnem JMP příkazů k rutinám operačního systému.

Abychom mohli některou z těchto rutin použít, je třeba nejdříve uskutečnit všechny přípravy pro její volání. Jestliže např. jedna rutina si vyžádá další rutiny KERNALU, pak je musíte sami vyvolat. Jestliže rutina žádá např. číslo do akumulátoru předem, musí toto číslo být také zadáno. Jestliže tyto podmínky nesplníte, pak vám samozřejmě jednotlivé rutiny nebudou pracovat.

Po provedení veškerých příprav vyvoláte zvolenou rutinu prostřednictvím příkazu JSR. Všechny přístupné rutiny KERNALU jsou utvořeny jako podprogramy, a proto končí příkazem RTS.

Oznámí-li vám rutina nějakou chybu, např. v akumulátoru, pak tyto chyby nenechte bez povšimnutí, jinak se vám vaše práce se strojovým programováním nemusí podařit.

KERNAL - popis rutin

B-1. ACPTR

Funkce: čtení dat ze sériové sběrnice
Startadresa: $FFA5(hex) 65445(dek)
Spojovací registr: A
Přípravná rutina: TALK, TKSA
Zpětné hlášení chyb: viz READST
Spotřeba stacku: 13

Použité registry: A, X
Popis: Tuto rutinu použijete, jestliže chcete načíst informace ze zařízení ze sériové sběrnice např. z diskety. Tato rutina vám přečte jeden datový byte ze sériové sběrnice, tento údaj je převeden do akumulátoru. Předem ale musíte vyvolat rutinu TALK. Přes ni dostane sériová sběrnice příkaz pro přenos dat. Jestliže si však příprava vyžádá sekundární příkaz, musí být přenesen před vyvoláním KERNAL příkazem TKSA. Chyby budou zpětně hlášeny stavovým slovem. Ke čtení stavového slova bude využito rutiny READST.
Postup:
0. Připravit zařízení na sériové sběrnici k přenosu dat (použít KERNAL rutiny TALK a TKSA).
1. Tuto rutinu vyvolat pomocí JSR.
2. Data zapsat do paměti nebo použít.

Příklad:

```
;GET A BYTE FROM THE BUS
JSR ACPTR
STA DATA
```

B-2. CHKIN

Funkce: otevření kanálu pro zápis
Startadresa: $FFC6(hex) 65478(dek)
Spojovací registr: X
Přípravné rutiny: OPEN
Zpětné hlášení chyb:
Spotřeba stacku: žádná
Použité registry: A, X
Popis: Každý logický datový soubor otevřený pomocí KERNAL rutiny OPEN může být přes tuto rutinu definován jako vstupní kanál, musí se ovšem jednat opravdu o vstupní kanál, jinak dojde k chybě a rutina bude přerušena.

Jestliže nejsou data zadávána prostřednictvím klávesnice, pak musí být vyvolána tato rutina před samotnou prací s příkazy Kernalu CHRIN nebo GETIN. Jestliže dojde k zadávání přes klávesnici a nejsou otevřené další vstupní kanály, pak nebude této rutiny i rutiny OPEN použito.

Jestliže použijete tuto rutinu pro přístroj připojený na sériovou sběrnici, pak bude TALK adresa (i sekundární adresa, pokud byla určena pomocí OPEN rutiny) automaticky přenesena.

Postup:

0. Otevřít logický datový soubor.
1. Naplnit registr X číslem použitého logického souboru.
2. Vyvolat CHKIN přes JSR.

Možné chyby:

#3: soubor neotevřen
#5: zařízení nepřipojeno
#6: soubor není vstupní

Příklad:

```
;PREPARE FOR INPUT FROM LOGICAL FILE 2
   LDX #2
   JSR CHKIN
```

B-3. CHKOUT

Funkce: otevření kanálu pro výstup
Startadresa: $FFC9(HEX) 65481(dek)
Spojovací registr: X
Přípravná rutina: OPEN
Zpětné hlášení chyb: 0, 3, 5, 7 (viz READST)
Spotřeba stacku: 4+
Použité registry: A, X
Popis: Číslo logického datového souboru, jež byl vytvořen pomocí rutiny OPEN, může být definováno jako výstupní kanál. Je

samozřejmé, že se musí jednat o výstupní zařízení, jinak dojde k chybě a rutina bude přerušena.

Chcete-li užít obrazovky jako výstupu a není ještě nadefinován žádný jiný výstupní kanál, pak tuto ani rutinu OPEN není třeba používat.

Při otevírání kanálu k sériové sběrnici přenáší tato rutina automaticky LISTEN-adresu vyvolanou OPEN-rutinou (rovněž případnou sekundární adresu).

Postup:

Nezapomeňte, že tato rutina se nepoužívá k přenosu dat na obrazovku.

0. Zadat prostřednictvím rutiny OPEN: číslo logického datového souboru, LISTEN-adresu a sekund. adresu (pokud je třeba).
1. Načíst do X registru číslo datového souboru užitého v příkazu OPEN.
2. Vyvolat rutinu pomocí JSR.

Příklad:

```
   LDX #3 ; definuje log. soubor 3 jako výst. kanál
   JSR CHKOUT
```

Možné chyby:

#3: Soubor není otevřen
#5: Zařízení nepřipojeno
#7: Není výstupní datový soubor

B-4. CHRIN

Funkce: vstup znaku
Startadresa: $FFCF(hex) 65487(dek)
Spojovací registr: A
Přípravná rutina: OPEN, CHKIN
Zpětné hlášení chyb: 0 (viz READST)
Spotřeba stacku: 7+
Použité registry: A, X

Popis: Pomocí této rutiny je čten jeden datový byte z kanálu již vytvořeného pro vstup pomocí rutiny CHKIN. Nebylo-li funkce CHKIN použito k definici dalšího vstupního kanálu, pak se vychází z toho, že veškerá data vstupují z klávesnice. Datový byte je tak přenesen do akumulátoru, po vyvolání zůstane kanál otevřen.

Zadávání dat prostřednictvím klávesnice je prováděno zvláštním způsobem. Nejprve je zřízen kurzor a bliká tak dlouho, než zadáte CR (RETURN). Všechny znaky (max. 88) jsou pak zapsány do vstupního bufferu BASICU. Tyto znaky můžete vyvolat i jednotlivě tak, že se tato rutina vyvolá pro každý znak zvlášť. Po zadání CR je pak zpracován celý řádek. Následující vyvolání této rutiny pak začíná opět od začátku, to znamená blikajícím kurzorem.

Postup:

Vstup z klávesnice:

1. Vyvolání datového byte prostřednictvím této rutiny.
2. Datový byte uložit do paměti.
3. Přezkoušet, jestli se jednalo o poslední datový byte (bylo CR?).
4. Není-li tomu tak, pak pokračovat bodem 0.

Příklad:

```
LDY #$00 ;PREPARE THE.Y REGISTER TO STORE THE
      DATA RD JSR CHRIN
STA DATA,Y ;STORE Yth DATA BYTE IN THE Yth
            ;LOCATION IN THE DATA AREA
INY
CMP #CR ;IS IT A CARRIAGE RETURN?
BNE RD ;NO,GET ANOTHER DATA BYTE
```

Příklad:

```
JSR CHRIN
STA DATA
```

Z jiných periférií:

0. Užít KERNAL-rutinu OPEN a CHKIN.
1. Vyvolat CHRIN pomocí JSR.
2. Data uložit do paměti.

Příklad:

```
JSR CHRIN
STA DATA
```

B-5. CHROUT

Funkce: výstup znaku
Startadresa: $FFD2(hex) 65490(dek)
Spojovací registr: A
Přípravné rutiny: CHKOUT, OPEN
Chybové hlášení: 9 (viz READST)
Spotřeba: sXTTAB ; (LOW BYTE)

```
LDA PROG+1
STA TXTTAB+1 ;(HIGH BYTE)
LDX VARTAB ;LOAD.X WITH LOW BYTE OF END SAVE
LDY VARTAB ;LOAD.Y WITH HIGH BYTE
LDA #TXTTAB ;LOAD AKUM. WITH PAGE 0 OFFSET
JSR SAVE
```

B-6. CIOUT

Funkce: výstup byte přes sériový port
Startadresa: $FFA8(hex) 65448(dek)
Spojovací registr: A
Přípravná rutina: LISTEN, (SECOND)
Zpětné hlášení chyb: viz READST
Spotřeba stacku: 5
Použité registry: žádné

Popis: Tato rutina slouží k informačnímu přenosu na zařízení připojenému na sériový port. Vyvoláním této rutiny dojde k přenosu jednoho datového byte na sériový port pomocí "handshake". Před vyvoláním je třeba vyvolat rutinu LISTEN pro přípravu přijímacího zařízení na sériové sběrnici pro příjem dat. Je-li třeba použít

sekundární adresy, pak musí být provedeno pomocí rutiny SECOND.

Do akumulátoru je pak zapsán jeden datový byte, který je přenesen přes sériový port. Musí být ale vykonána příprava, jinak nám stavové slovo ohlásí "timeout". Při této rutině je jeden znak vždy uložen do pomocné paměti. Jestliže se pro ukončení přenosu užije rutina UNLSN, pak bude znak uložený ve vyrovnávací paměti přenesen přes EOI a pak teprve dojde k přenosu příkazu UNLSN na zařízení.

Postup:

0. Použít rutinu LISTEN (popřípadě i SECOND).
1. Zapsat jeden datový byte do akumulátoru.
2. K přenosu datového byte vyvolat tuto rutinu.

Příklad:

```
LDA #(apostrof)X(afostrof) ;SEND AN X TO THE SER. BUS
JSR CIOUT
```

B-7. CINT

Funkce: inicializace obrazovkového editoru a video chipu 6567
Startadresa: $FF81(hex) 65409(dek)
Spojovací registr: žádný
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 4
Použité registry: A, X, Y
Popis: Touto rutinou dojde k inicializaci řídícího videochipu 6567 ve vašem COMMODORE 64 a obrazovkového editoru.
Postup: 1. Vyvolání rutiny pomocí JSR.
Příklad:

```
JSR CINT
JMP RUN ;BEGIN EXECUTION
```


B-8. CLALL

Funkce: uzavření všech datových souborů
Startadresa: $FFE7(hex) 65511(dek)
Spojovací registr: žádný
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 11
Použité registry: A, X
Popis: Pomocí této rutiny dojde k uzavření všech otevřených datových souborů. Při vyvolání ukazovátka v tabulce otevřených datových souborů budou vráceny do základního postavení a soubory uzavřeny. Automaticky dojde k vyvolání CLRCHN rutiny pro vracení vstupních/výstupních kanálů.
Postup: 1. Vyvolat rutinu pomocí JSR.
Příklad:

```
JSR CALL ;CLOSE ALL FILES AND SELECT DEFAULT I/O
         CHANNELS
JMP RUN ;BEGIN EXECUTION
```

B-9. CLOSE

Funkce: uzavření datového souboru
Startadresa: $FFC3(hex) 65475(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: 0, 240 (viz READST)
Spotřeba stacku: 2+
Použité registry: A, X, Y
Popis: Tato rutina slouží k uzavření datového souboru, kde všechny vstupy a výstupy byly již ukončeny. Rutinu je třeba vyvolat až po zapsání příslušného čísla datového souboru do akumulátoru (stejné číslo, které jsme použili při otevření souboru rutinou OPEN).

Postup:
1. Příslušné číslo datového souboru načíst do akumulátoru.
2. Vyvolat rutinu.

Příklad:

```
;CLOSE 15
   LDA #15
   JSR CLOSE
```

B-10. CLRCHN

Funkce: výmaz vstupních/výstupních kanálů
Startadresa: $FFCC(hex) 65484(dek)
Spojující registr: žádný
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: 9
Použité registry: A, X
Popis: Této rutiny se užívá k uzavření všech otevřených kanálů a k navrácení vstupních/výstupních kanálů do výchozího postavení (standardní hodnoty). Obvykle dojde k vyvolání této rutiny po otevření dalšího I/O kanálu nebo ukončení I/O přenosu. Standardní číslo pro vstupní zařízení je 0 (klávesnice), standardní výstupní zařízení má číslo 3 (obrazovka).

Jestliže je jeden z uzavíraných kanálů sériová sběrnice, pak je žádoucí nejprve použít pro vstupní kanál signál UNTALK a pro výstupní kanál signál UNLISTEN. Jestliže tuto rutinu nevyvoláme (tzn. zařízení připojená na sériovou sběrnici jsou stále schopna příjmu), mohou tato zařízení (více než jedno) přijímat data ve směru od počítače. Tím je třeba možné například využít tiskárnu pro výstup dat a disk pro uložení dat současně. Tak je možné přímo vytisknout disketový datový soubor.

Tato rutina je automaticky vyvolána při vykonávání rutiny CLALL.
Postup: 1. Vyvolat rutinu pomocí JSR.

Příklad:

```
   JSR CLRCHN
```

B-11. GETIN

Funkce: přečtení jednoho znaku
Startadresa: $FFE4(hex) 65508(dek)
Spojovací registr: A
Přípravné rutiny: CHKIN, OPEN
Zpětné hlášení chyb: viz READST
Stack: 7+
Použité registry: A (X, Y)
Popis: Jedná-li se u kanálu o klávesnici, pak tato rutina vezme jeden znak z vyrovnávací paměti klávesnice a přenese jej jako hodnotu ASCII do akumulátoru. Je-li vyrovnávací paměť prázdná, pak je do akumulátoru přenesena hodnota 00. Znaky jsou přenášeny automaticky pomocí rutiny pro zjištění ovlivnění klávesnice, kterou vyvolá rutina SCNKEY, a tyto znaky jsou řazeny do tzv. čekací řady. Ve vyrovnávací paměti klávesnice může být zaznamenáno pouze deset znaků. Je-li paměť naplněna, pak jsou další znaky ignorovány, než dojde k odstranění alespoň jednoho znaku z "čekací řady".

Jestliže se jedná o kanál k RS-232, užije se jen registru A a je vypsán jeden znak. K případnému přezkoušení viz READST.

Jedná-li se o kanál k sériové sběrnici, kazetě či obrazovce, pak je třeba vyvolat rutinu BASIN.
Postup:
1. Vyvolat rutinu příkazem JSR.
2. Přezkoušet, zda je akumulátor prázdný (naplněn 0).
3. Zpracovat data.

Příklad:

```
;WAIT FOR A CHARACTER
   WAIT JSR GETIN
```

```
CPM #0
BEQ WAIT
```

B-12. IOBASE

Funkce: vymezení I/O rozsahu paměti
Startadresa: $FFF3(hex) 65523(dek)
Spojovací registry: X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: 2
Použité registry: X, Y
Popis: Pomocí této rutiny dochází v X- a Y-registru k zaznamenání byte s vyšší a nižší hodnotou adresy té části paměti, kde se nachází I/O registr. Této adresy je pak možno užít k přístupu na I/O registr. Registr X obsahuje nižší adresový byte a registr Y vyšší adresový byte. Pomocí této rutiny se dosahuje kompatibility mezi C64, VC20 a jinými počítači.

Jestliže dojde k ustavení I/O registru pomocí této rutiny u strojových programů, pak co se týče KERNALU a BASICU dochází ke kompatibilitě mezi uvedenými počítači.

Postup:

1. Vyvolat rutinu pomocí JSR.
2. Registry X a Y zaznamenat na po sobě jdoucí paměťová místa.
3. Posun zaznamenat do registru Y.
4. Použít těchto I/O míst.

Příklad:

```
;SET THE DATA DIRECTION REGISTER OF THE USER PORT
        TO 0(INPUT)
JSR IOBASE
STX POINT ;SET BASE REGISTERS
STY POINT1
LDY#2
```



```
LDA #0 ;OFFSET FOR DDR OF THE USER PORT
STA (POINT),Y;SET DDR TO 0
```

B-13. IONIT

Funkce: inicializace I/O zařízení
Startadresa: $FF84(hex) 65412(dek)
Spojovací registr: žádný
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: žádný
Použité registry: A, X, Y
Popis: Pomocí této rutiny dojde k inicializaci všech I/O zařízení a rutin. Běžně je vyvolána jako část inicializace program. módu C64.
Příklad:

```
JSR IOINIT
```

B-14. LISTEN

Funkce: příkaz LISTEN pro zařízení připojené k sériové sběrnici
Startadresa: $FFB2(hex) 65457(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: viz READST
Stack: žádný
Použité registry: A
Popis: Touto rutinou je zařízení připojené na sériovou sběrnici připraveno k příjmu dat. Před vyvoláním je nutné uložit do akumulátoru číslo zařízení (0-31). Příkazem LISTEN je číslo bit po bitu přeměněno, lze též využít vazby a převést jako LISTEN-adresu. To je pak přeneseno jako příkaz přes sériovou sběrnici. Udané zařízení je tak připraveno k příjmu dat.
Postup:

1. Vybrané číslo zařízení zaznamenat do akumulátoru.

2. Vyvolat tuto rutinu příkazem JSR.

Příklad:

```
;COMMAND DEVICE #8 TO LISTEN
    LDA #8
    JSR LISTEN
```

B-15. LOAD

Funkce: načtení RAM z periferie
Startadresa: $FFD5(hex) 65493(dek)
Spojovací registr: A, X, Y
Přípravná rutina: SETLFS, SETNAM
Zpětné hlášení chyb: 0, 1, 2, 3, 4, 5, READST
Stack: žádný
Použité registry: A, X, Y
Popis: Touto rutinou jsou datové byte ze zvoleného zařízení načteny přímo do paměti počítače. Může se též použít k porovnání dat uložených v paměti počítače a dat vysílaných ze zařízení, přičemž data uložená v RAM zůstanou nedotčena (VERIFY).

K záznamu je do akumulátoru (A) načtena 0 a pro VERIFY pak 1. Jestliže zadáme OPEN pro výstupní zařízení se sekundární adresou (SA) 0, pak je adresa LOAD ignorována a registry .X a .Y musí obsahovat startadresu. Je-li sekundární adresa zvolena 1, 0 nebo 2, pak budou data načtena do počítače od adresy udané prostřednictvím pozice určené údajem výstupního zařízení. Tato rutina zprostředkuje také adresu konce LOAD v RAM.

Před vyvoláním rutiny je třeba vyvolat rutiny SETLFS a SETNAM.
Poznámka: Příkaz LOAD z klávesnice (0), RS-232 (2) nebo obrazovky (3) není možný.
Postup:
1. Vyvolat rutiny SETLFS a SETNAM. Jestliže je žádoucí použití přesunutého záznamu, pak je třeba užít rutinu SETLFS k přenosu sekundární adresy 0.

2. Registr A naplnit 0 pro čtení a 1 pro přezkoušení.
3. Je-li žádoucí čtení od určité adresy, je nutné registry X a Y uvést na tuto čtecí startadresu.
4. Vyvolat rutinu JSR.

Příklad:

```
;LOAD A FILE FROM TAPE
    LDA #DEVICE ;SET DEVICE NUMBER
    LDX #FILENO ;SET LOGICAL FILE NUMBER
    LDY CMD1 ;SET SECONDARY ADDRESS
    JSR SETLFS
    LDA #NAME1-NAME     ;LOAD .A WITH NUMBER OF
                        ;CHARACTERS IN FILE NAME
    LDX #-NAME ;LOAD .X AND .Y WITH ADDRESS OF
    LDY #-NAME ;FILE NAME
    JSR SETNAM
    LDA #0 ;SET FLAG FOR A LOAD
    LDX #$FF ;ALTERNATE START
    LDY #$FF
    JSR LOAD
    STX VARTAB ;END OF LOAD
    STY VARTAB+1
    JMP START
    NAME .BYT "FILE NAME"
    NAME 1 ;
```

B-16. MEMBOT

Funkce: nastavení ukazovátka pro spodní okraj paměti
Startadresa: $FF9C(hex) 65436(dek)
Spojovací registry: X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: žádný

Použité registry: X, Y
Popis: Rutiny je použito k nastavení spodního ukazovátka paměti. Je-li nastaven v akumulátoru přenosový bit (při vyvolání této rutiny), pak je ukazovátko spodního RAM byte vydáno v registrech X a Y.

U nerozšířeného COMMODORE 64 je hodnota začátku ukazovátka $0800 (2048 dek). Je-li při vyvolání této rutiny akumulátor prázdný (=0), pak jsou hodnoty X a Y registru přenášeny jako spodní a vrchní hranice ukazovátka RAM.

Postup:
Přečtení začátku RAM:
1. Nastavení přenosového Flagu.
2. Vyvolání rutiny.

Nastavení počátku paměti:
1. Vymazání přenosového Flagu.
2. Vyvolání rutiny.

Příklad:

```
;MOVE BOTTOM OF MEMORY UP 1 PAGE
   SEC ;READ MEMORY BOTTOM
   JSR MEMBOT
   INY
   CLC ;SET MEMORY BOTTOM TO NEW VALUE
   JSR MEMBOT
```

B-17. MEMTOP

Funkce: nastavení ukazovátka pro horní okraj paměti
Startadresa: $FF99(hex) 65433(dek)
Spojovací registr: X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: 2
Použité registry: X, Y

Popis: Touto rutinou je nastavena horní hranice RAM. Je-li při vyvolání této rutiny nastaven přenosový bit akumulátoru, pak jsou ukazovátka hranic RAM načtena do registru X a Y. Je-li přenosový bit akumulátoru vymazán (=0), pak jsou obsahy registru X a Y načteny do krajních ukazovátek paměti a tím změněny krajní pozice paměti.

Příklad:

```
;DELLOCATE THE RS.232 BUFFER
   SEC
   JSR MEMTOP ;READ TOP OF MEMORY
   DEX
   CLC
   JSR MEMTOP ;SET NEW TOP OF MEMORY
```

B-18. OPEN

Funkce: otevření logického datového souboru
Startadresa: $FFC0(hex) 65472(dek)
Spojovací registry: žádné
Přípravné rutiny: SETLFS, SETNAM
Zpětné hlášení chyb: 1, 2, 3, 4, 5, 6, 240, READST
Spotřeba stacku: žádná
Použité registry: A, X, Y
Popis: Pomocí této rutiny je možné zřídit logický datový soubor. Po jeho zřízení je jej možno užít k I/O práci. U většiny I/O rutin je tato rutina použita ke zřízení příslušného logického datového souboru. Pro použití této rutiny nejsou zapotřebí žádné parametry, je však třeba předtím vyvolat KERNAL rutiny SETLFS a SETNAM.

Postup:
1. Použít rutinu SETLFS.
2. Použít rutinu SETNAM.
3. Vyvolat tuto rutinu.

Příklad:

Toto je implementace basic příkazu: OPEN 15,8,15,"I/O"

```
LDA #NAME2-NAME ;LENGTH OF FILE NAME FOR
      SETLFS
LDY #+NAME ;ADDRESS OF FILE NAME
LDX #-NAME
LDA #15
LDX #8
LDY #15
JSR SETLFS
JSR OPEN
NAME .BYT"I/O"
NAME2
```

B-19. PLOT

Funkce: přečtení/zápis pozice kurzoru
Startadresa: $FFF0(hex) 65520(dek)
Spojovací registry: A, X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A, X, Y
Popis: Je-li při vyvolání této rutiny ustaven přenosový Flag akumulátoru, pak bude současná pozice kurzoru na obrazovce (koordináty X/Y) načtena do registru Y a X, kde Y je číslo sloupce (0-79) a X je číslo řádku (0-24). Je-li při vyvolání přenosový bit vymazán, pak se kurzor přemístí do pozice X/Y (odpovídající registrům Y a X).
Postup:
Přečtení pozice kursoru:
1. Přenosový Flag nastavit.
2. Vyvolat tuto rutinu.

3. Přečtení pozic x a y z registru X a Y.

Udání pozice kurzoru:
1. Přenosový Flag vymazat.
2. Do registru X a Y zapsat požadovanou pozici kurzoru.

Vyvolat tuto rutinu.
Příklad:

```
;MOVE THE CURSOR TO ROW 10,COLUMN 5(5,10)
    LDX #10
    LDY #5
    CLC
    JSR PLOT
```

B-20. RAMTAS

Funkce: test-RAM
Startadresa: $FF87(hex) 65415(dek)
Spojovací registry: A, X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Stack: 2
Použité registry: A, X, Y
Popis: Pomocí této rutiny dochází k testovaní RAM a zároveň je položeno horní, popřípadě i spodní ukazovátko paměti. Současně dojde k vymazání paměti $0000-$0101 a $0200-$03FF. Kromě toho dojde ještě k inicializaci kazetové vyrovnávací paměti a začátek obrazovky je nastaven na $0400. Běžně je tato rutina prováděna jako součást inicializace programového módu COMMODORE 64.
Příklad:

```
JSR RAMTAS
```

B-21. RDTIM

Funkce: přečtení systémového generátoru synchronizačních impulsů
Startadresa: $FFDE(hex) 65502(dek)
Spojující registry: A, X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A, X, Y
Popis: Tato rutina je užívána ke čtení systémového času. Rozlišení je na 1/60 s. Pomocí této rutiny dojde ke zjištění tří byte. Akumulátor pak obsahuje nejvyšší označený byte, registr X další označený byte a registr Y pak obsahuje nejnižší označený byte.
Příklad:

```
JSR RDTIM
STY TIME
STX TIME+1
STA TIME+2
 ...
TIME*=*+3
```

B-22. READST

Funkce: přečtení stavového slova
Startadresa: $FFB7(hex) 65463(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A
Popis: Pomocí této rutiny dojde ke zjištění současného stavu I/O zařízení. Tato rutina je běžně vyvolána novou spoluprací s I/O

zařízením. Uvádí informaci o stavu zařízení, případně chyby, které se objevily během I/O práce.

Byte přenášené do akumulátoru mají následující informace:

st. bit pozice	st. čísel. hodnota	čtení z kazety	sér. sběr. čtení/psaní	zkoušení kazety-verify +načtení (LOAD)
0	1		čas. omez. (timeout) zapsat	
1	2		čas. omez. (timeout) číst	
2	4	krátká věta		...
3	8	dlouhá věta		
4	16	neopravitelná chyba čtení		...
5	32	chyba verify		...
6	64	konec soub. EOI-vedení		
7	-128	konec pásky	nepřipoj. zařízení	konec pásky

Postup:
1. Vyvolat rutinu.
2. Programovou informaci dekódovat do registru A.

Příklad:

```
;CHECK FOR END OF FILE DURING READ
   JSR READST
   AND #64 ;CHECK EOF BIT (EOF=END OF FILE)
   BNE EOF ;BRANCH ON EOF
```

B-23. RESTOR

Funkce: uvedení systému do normálního stavu
Startadresa: $FF8A(hex) 65418(dek)
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2

Použité registry: A, X, Y
Popis: Pomocí této rutiny dojde ke zpětnému nastavení všech systémových vektorů použitých KERNAL a BASIC rutin na základní hodnotu. Přes rutinu VECTOR je možné jednotlivé systémové vektory číst a také upravovat.
Postup:
1. Vyvolat rutinu.

Příklad:

```
JSR RESTOR
```

B-24. SAVE

Funkce: přenos obsahu paměti na určené zařízení
Startadresa: $FFD8(hex) 65496(dek)
Spojující registry: A, X, Y
Přípravná rutina: SETLFS, SETNAM
Zpětné hlášení chyb: 5, 8, 9, READST
Spotřeba stacku: žádná
Použité registry: A, X, Y
Popis: Touto rutinou se obsah paměti zapíše na externí záznamové médium. Paměť je přenesena a zapsána počínaje akumulátorem udanou nepřímou adresou na stránce 0 až k adrese udané registry X a Y. Je přenášena jako datový soubor k I/O zařízení. Před použitím této rutiny je třeba použít rutiny SETLFS a SETNAM. K přenosu na zařízení 1 (Datassette) není potřeba uvádět jméno souboru. Naopak pokusíme-li se přenášet soubor bez jména na jiné zařízení, dojde k chybovému hlášení.
Připomínka: Není možné ukládat na zařízení 0 (klávesnici), 2 (rs232) a 3 (obrazovku), čímž by vznikla chyba a došlo k přerušení ukládání.
Postup:
0. Vyvolat rutiny SETLFS, SETNAM.
1. Dvě na sebe navazující paměťová místa nulté stránky (zero page) načíst s ukazovátkem, jež ukazuje na začátek přenášeného obsahu

(obvykle přichází u 6502 nejnižší byte jako první a následně pak vyšší byte).
2. Adresu ukazovátka z nulté stránky načíst do akumulátoru.
3. Nižší a vyšší byte koncové adresy přenášené paměti načíst do registru X a Y.
4. Vyvolat tuto rutinu.

Příklad:

```
LDA #1 ;DEVICE=1:CASSETTE
JSR SELFS
LDA #0 ;NO FILE NAME
JSR SETNAM
LDA PROG ;LOAD START ADDRESS OF SAVE
STA TXTTAB ;(LOW BYTE)
LDA PROG+1
STA TXTTAB+1 ;(HIGH BYTE)
LDX VARTAB ;LOAD.X WITH LOW BYTE OF END OF
    SAVE
LDY VARTAB ;LOAD.Y WITH HIGH BYTE
LDA #-TXTTAB ;LOAD AKUMULATOR WITH PAGE 0
    OFFSET
JSR SAVE
```

B-25. SCNKEY

Funkce: dotaz na klávesnici
Startadresa: $FF9F(hex) 65439(dek)
Spojovací registr: žádný
Přípravná rutina: IOINIT
Zpětné hlášení chyb: žádné
Spotřeba stacku: 5
Použité registry: A, X, Y
Popis: Pomocí této rutiny dochází k zjišťování, zda bylo použito klávesnice a jestliže ano, o které klávesy šlo. Je použita při každém

IRQ. Po každém ovlivnění klávesy dochází k záznamu hodnoty v ASCII do vyrovnávací paměti klávesnice.

Postup:

1. Vyvolat rutinu.

Příklad:

```
GET JSR SCNKEY ;SCAN KEYBOARD
    JSR GETIN ;GET CHARACTER
    CMP #0 ;IS IT NULL?
    BEQ GET ;YES SCAN AGAIN
    JSR CHROUT ;PRINT IT
```

B-26. SCREEN

Funkce: zajištění formátu obrazovky
Startadresa: $FFED(hex) 65517(dek)
Spojovací registr: X, Y
Přípravná rutina: žádná
Spotřeba stacku: 2
Použité registry: X, Y
Popis: Tato rutina nám opět zřídí formát obrazovky např. 40 sloupců v X a 25 řádek v Y. Může jí být užito i k určení, na kterém počítači běží program. Tato funkce byla pro C64 zavedena pro práci s jinými kompatibilními počítači.

Postup:

1. Vyvolat rutinu.

Příklad:

```
JSR SCRENEEN
STX MAXCOL
STY MAXROW
```

B-27. SECOND

Funkce: přenos sekundární adresy pro LISTEN

Startadresa: $FF93(hex) 65427(dek)
Spojovací registr: A
Přípravná rutina: LISTEN
Zpětné hlášení chyb: viz READST
Spotřeba stacku: 8
Použité registry: A
Popis: Touto rutinou dochází k přenosu sekundární adresy po vyvolání rutiny LISTEN na I/O zařízení. Zařízení je pak připraveno k příjmu. Této rutiny nelze využít k přenosu sekundární adresy při použití rutiny TALK. Sekundární adresy se obvykle užívá k přenosu doplňujících informací vstupu a výstupu.

Postup:

1. Přenášenou sekundární adresu zapsat do akumulátoru a upravit OR #$60.
2. Vyvolat tuto rutinu.

Příklad:

```
;ADDRESS DEVICE #8 WITH COMMAND (SECONDARY
        ADDRESS)#15
    LDA #8
    JSR LISTEN
    LDA #$60+15
    JSR SECOND
```

B-28. SETLFS

Funkce: definice logického datového souboru
Startadresa: $FFBA(hex) 65466(dek)
Spojovací registry: A, X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: žádná

Popis: Rutiny se užívá k definici čísla logického datového souboru, čísla zařízení a sekundární adresy (číslo příkazu) pro KERNAL rutiny, číslo logického datového souboru je systémem užito jako jakéhosi klíče pro datovou tabulku vytvořenou rutinou OPEN. Pro číslo zařízení je možno užít čísel 0-31. Tabulka udává kódy pro I/O zařízení připojitelná k C-64.

ADRESA	ZAŘÍZENÍ
0	Klávesnice
1	Datassette TM #1
2	RS-232C
3	Výst. obrazovka
4	Tiskárna na sériové sběrnici
8	Disketová jednotka na sér. sběrnici

Číslo zařízení s hodnotou 4 a vyšší se automaticky bere jako připojené k sér. sběrnici.

Příkaz pro zařízení je přenášen přes sériovou sběrnici jako sekundární adresa po přenosu čísla zařízení prostřednictvím sériové (HANDSHAKE) součinnosti přenosu. Jestliže není přenesena žádná sekundární adresa, pak je třeba indexový registr Y obsadit hodnotou 255.

Postup:

1. Načíst do akumulátoru číslo logického datového souboru.
2. Načíst do X číslo zařízení.
3. Načíst příkaz do Y.

Příklad:

```
FOR LOGICAL FILE 32,DEVICE #4,AND NO COMMAND:
   LDA #32
   LDX #4
   LDY #255
   JSR SETLFS
```


B-29. SETMSG

Funkce: výstup systémového hlášení
Startadresa: $FF90(hex) 65424(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A
Popis: Pomocí této rutiny dochází k ohlášení chybových a řídících hlášení řízených KERNALEM. Při vyvolání této rutiny jsou podle obsahu akumulátoru vydána chybová nebo řídící hlášení. Chybové hlášení je např. FILE NOT FOUND, řídící hlášení např. PRESS PLAY ON CASSETTE.

Bit 6 a 7 určuje, odkud hlášení přichází. Je-li bit 7 1, pak se jedná o chybové hlášení udané KERNALEM. Je-li nastaven bit 6, pak jsou udávána řídící hlášení.

Postup:

1. Načíst do akumulátoru vybranou hodnotu.
2. Vyvolat tuto rutinu.

Příklad:

```
LDA #$40
JSR SETMSG ;TURN ON CONTROL MESSAGES
LDA #$80
JSR SETMSG ;TURN ON ERROR MESSAGES
LDA #0
JSR SETMSG ;TURN OFF ALL KERNAL MESSAGES
```

B-30. SETNAM

Funkce: definice jména datového souboru
Startadresa: $FFBD(hex) 65469(des)
Spojovací registry: A, X, Y

Přípravná rutina: žádná
Spotřeba stacku: žádná
Použité registry: žádné
Popis: Pomocí této rutiny jsou definována jména datových souborů pro rutiny OPEN, SAVE a LOAD. Délka jména datového souboru je v akumulátoru, adresa jména datového souboru je v registru X a Y. Adresa může být libovolná systémová adresa paměti, od které je jméno datového souboru pak uloženo jako string. Jestliže jméno datového souboru není žádoucí, pak je akumulátor uveden na hodnotu 0. V tomto případě je možno registry X a Y uvést na libovolnou adresu paměti.
Postup:
1. Načíst do akumulátoru délku jména datového souboru.
2. Adresový byte s nižší hodnotou jména datového souboru načíst do index registru X.
3. Adresový byte s vyšší hodnotou načíst do registru Y.
4. Vyvolat tuto rutinu.

Příklad:

```
LDA #NAME2-NAME ;LOAD LENGTH OF FILE NAME
LDX #-NAME ;LOAD ADDRESS OF FILE NAME
LDY #+NAME
JSR SETNAM
```

B-31. SETTIM

Funkce: nastavení systémového času
Startadresa: $FFDB(hex) 65499(dek)
Spojovací registry: A, X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: žádné

Popis: Dochází k aktualizaci hodin (každou 1/60 sec=1 "jiffy") pomocí IRQ. Hodiny obsahují 3 byte, takže mohou ukazovat až 5184000 "jiffies" což je 24 hodin, pak dochází k nastavení na 0. Před vyvoláním rutiny je třeba načíst do akumulátoru první byte hodnoty, do registru X následující byte hodnoty a do Y poslední byte hodnoty výchozího nastavení (v jiffech).
Postup:
1. Načíst do akumulátoru a registrů X a Y hodnotu.
2. Vyvolat tuto rutinu.

Příklad:

```
;SET THE CLOCK TO 10 MINUTES=3600 JIFFIES
    LDA #0 ;MOST SIGNIFICANT
    LDA #+3600
    LDY #-3600 ;LEAST SIGNIFICANT
    JSR SETTIM
```

B-32. SETTMO

Funkce: nastavení Timeout-Flags pro IEEE-sběrnici
Startadresa: $FFA2(hex) 65442(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: žádné
Poznámka: Tato rutina se používá jen v souvislosti s IEEE-cartridge.
Popis: Užitím této rutiny dochází k zadání Timeout-Flag pro IEEE-sběrnici. Jestliže je tato značka položena, čeká počítač na hlášení od zařízení připojeného k IEEE-vstupu. Jestliže zařízení neodpoví na signál DAV C-64 ve vymezeném čase, pak pozná počítač chybu obsluhy a dojde k opuštění handshake-sekvence. Je-li při vyvolání rutiny v akumulátoru v bitu 7 načtena 0, pak jsou

Timeout aktivní. Naopak Timeout jsou neaktivní, jestliže načteme do bitu 7 hodnotu 1.

Postup:

Zadání Timeout-označení:

1. Bit 7 akumulátoru nastavit na hodnotu 0.
2. Vyvolat tuto rutinu.

Vrácení Timeout-označení:

1. Bit 7 akumulátoru nastavit na hodnotu 1.
2. Vyvolat tuto rutinu.

Příklad:

```
;DISABLE TIMEOUT
   JSR SETTMO
```

B-33. STOP

Funkce: dotaz na tlačítko "STOP"
Startadresa: $FFE1(hex) 65505(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: žádná
Použité registry: A, X
Popis: Jestliže došlo během volání UDTIM ke stisku klávesy STOP, pak dojde po vyvolání této rutiny k nastavení flagu Z. Pomocí toho dojde k nastavení všech kanálů na standardní hodnoty. Ostatní flagy zůstanou nezměněny. Jestliže ke stisku klávesy STOP nedošlo, obsahuje akumulátor byte, který představuje poslední prohledávanou řadu klávesnice. Tím může uživatel zjistit, zda byly stisknuty i jiné klávesy.

Postup:

0. Před vyvoláním této rutiny je třeba vyvolat UDTIM.
1. Vyvolat tuto rutinu.
2. Otestovat flag ZERO.

Příklad:

```
JSR UDTIM ;SCAN FOR STOP
JSR STOP
BNE*+5 ;KEY NOW DOWN
JMP READY ;= STOP
```

B-34. TALK

Funkce: příkaz TALK pro zařízení připojené na sériovou sběrnici
Startadresa: $FFB4(hex) 65460(dek)
Spojovací registr: A
Přípravná rutina: žádná
Zpětné hlášení chyb: viz READST
Spotřeba: 8
Použité registry: A
Popis: Pro práci s touto rutinou je třeba načíst do akumulátoru číslo zařízení (0-31). Po vyvolání rutiny dochází ke konverzi bit po bitu pomocí funkce OR a vyslání na sériový port.

Postup:

1. Do akumulátoru načíst číslo zařízení.
2. Vyvolat tuto rutinu.

Příklad:

```
;COMMAND DEVICE #4 TO TALK
   LDA #4
   JSR TALK
```

B-35. TKSA

Funkce: přenos sekundární adresy na zařízení, které obdrželo předtím příkaz TALK
Startadresa: $FF96(hex) 65430(dek)
Spojovací registr: A
Přípravná rutina: TALK

Zpětné hlášení chyb: viz READST
Spotřeba stacku: 8
Použité registry: A
Popis: Tato rutina přenáší sekundární adresu přes sekundární sběrnici k zařízení, jež obdrželo příkaz TALK. Při vyvolání této rutiny je třeba do akumulátoru načíst číslo 0-31, které je pak vysláno přes sériovou sběrnici jako sekundární adresa. Je bezpodmínečně nutné předtím vyvolat rutinu TALK. TKSA není po příkazu LISTEN účinná.
Postup:
0. Použít rutinu TALK.
1. Do akumulátoru uložit sekundární adresu.
2. Vyvolat tuto rutinu.

Příklad:

```
;TELL DEVICE #4 TO TALK WITH COMMAND #7
    LDA #4
    JSR TALK
    LDA #7
    JSR TKSA
```

B-36. UDTIM

Funkce: aktivace systémového času
Startadresa: $FFEA(hex) 65514(dec)
Spojovací registry: žádný
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A, X
Popis: Pomocí této rutiny dochází k aktivování systémových hodin. Běžně jsou vyvolány prostřednictvím KERNAL IRQ rutiny každou 1/60 sec. Jestliže uživatelský program využívá sám přerušení, pak je třeba ke korekci času vyvolat tuto rutinu. Jestliže má být i nadále účinné tlačítko STOP, pak se musí užít rutina STOP.

Postup:
1. Vyvolat tuto rutinu.

B-37. UNLSN

Funkce: přenos příkazu UNLISTEN
Startadresa: $FFAE(hex) 65454(dek)
Spojovací registry: žádné
Přípravná rutina: žádná
Zpětné hlášení chyb: viz READST
Spotřeba stacku: 8
Použité registry: A
Popis: Prostřednictvím této rutiny obdrží všechna zařízení připojená na sériovou sběrnici příkaz k ukončení příjmu dat. Pomocí této rutiny dochází k přenosu příkazu UNLISTEN přes sériovou sběrnici. Jsou ovlivněna jen ta zařízení, která předtím obdržela příkaz LISTEN. Obvykle se této rutiny užívá po ukončení přenosu dat k externímu zařízení. Po příkazu UNLISTEN nejsou již vnější zařízení připojena k sériové sběrnici a jsou připravena k jinému užití.
Postup:
1. Vyvolat tuto rutinu.

Příklad:

```
    JSR UNLSN
```

B-38. UNTLK

Funkce: přenos příkazu UNTALK
Startadresa: $FFAB(hex) 65451(dek)
Spojující registry: žádné
Přípravná rutina: žádná
Zpětné hlášení chyb: viz READST
Spotřeba stacku: 8
Použité registry: A

Popis: Pomocí této rutiny se přenáší příkaz UNTALK po sériové sběrnici. Všechna zařízení, která předtím dostala příkaz TALK, ukončují přenos dat.

Postup:

1. Vyvolat rutinu.

Příklad:

```
JSR UNTLK
```

B-39. VECTOR

Funkce: řízení vektoru RAM
Startadresa: $FF8D(hex) 65421(dek)
Spojovací registry: X, Y
Přípravná rutina: žádná
Zpětné hlášení chyb: žádné
Spotřeba stacku: 2
Použité registry: A, X, Y

Popis: Tato rutina spravuje všechny skokové vektory obsazené v RAM. Je-li při vyvolání rutiny nastaven flag přenosu, pak je současný stav vektoru v RAM uložen do seznamu, jehož adresa je dána obsahem registru X a Y. Je-li při vyvolání rutiny přenos vymazán, pak je obsah seznamu přenesen do RAM.

Poznámka: Při práci s touto rutinou je třeba si počínat opravdu pozorně. Nejdříve by měl být veškerý obsah vektoru načten do uživatelské oblasti a po změně vektoru opět přenesen zpět do systémových vektorů.

Postup:

Čtení systémových RAM vektorů:

1. Nastavit bit přenosu.
2. Nastavit X a Y registry na adresu, kam chceme přenést obsah vektorů.
3. Vyvolat tuto rutinu.

Načtení systémových RAM vektorů:

1. Vymazat přenosový bit.
2. Nastavit X a Y registry na adresu, odkud chceme přenést obsah vektoru.
3. Vyvolat tuto rutinu.

Příklad:

```
;CHANGE THE INPUT ROUTINES TO NEW SYSTEM
    LDX #-USER
    LDY #+USER
    SEC
    JSR VECTOR ;READ OLD VECTORS
    LDA #-MYINP ;CHANGE INPUT
    STA USER+10
    LDA #+MYINP
    STA USER+11
    LDX #-USER
    CLC
    JSR VECTOR ;ALTER SYSTEM
    ...
    USER*=*+26
```

Chybová hlášení

V následující tabulce je přehled chybových hlášení, která se mohou vyskytnout při práci s KERNALEM. Dojde-li k jedné z těchto chyb, pak se nastaví přenosový bit a číslo chyby se přenese do akumulátoru.

Poznámka: I/O rutiny KERNALU nepracují s těmito kódy chybových hlášení, ty jsou identifikovány prostřednictvím KERNAL rutiny READST.

ČÍSLO	VÝZNAM
0	rutina ukončena tlačítkem STOP
1	mnoho otevřených souborů

2	soubor již otevřen
3	soubor neotevřen
4	soubor nenalezen
5	zařízení není připojeno
6	soubor není vstupní
7	soubor není výstupní
8	chybí jméno souboru
9	nepřípustné číslo zařízení
240	konec paměti posunut (RS-232)