Mikropočítač

UCB/PIC

Popis jazyka

MITE Hradec Králové s.r.o., Veverkova 1343, 500 02 Hradec Králové
tel. 498 500 252, fax 498 500 260
e-mail: mite@mite.cz http://www.mite.cz

OBSAH

1 Úvod

1.1 Technické vybavení mikropočítače

Mikropočítač UCB/PIC je tvořen pouze několika součástkami. Základním prvkem zapojení je mikrořadič PIC16C56 - výrobek firmy Microchip Technology. Firma Parallax tento mikrořadič vybavila programovým zabezpečením, jehož klíčovou částí je interpretační překladač programovacího jazyka BASIC, přesněji řečeno, jeho vlastní účelová implementace nazvaná PBASIC. Překladač a pomocné programy jsou umístěny v interní paměti ROM mikrořadiče PIC16C56.

Generátor hodinových impulsů mikrořadiče je řízen piezokeramickým rezonátorem o kmitočtu 4MHz.

Pro uložení uživatelského (aplikačního) programu zapsaného v jazyku PBASIC je určena paměť EEPROM typu 93LC56 o kapacitě 256B. Paměť EEPROM umožňuje pohotové uložení programu bez manipulace s pouzdrem paměti, bez zvláštního zařízení (programátoru) a bez potřeby většího programovacího napětí. Jediným prostředkem potřebným pro zápis programu a jeho přemístění do paměti mikropočítače (a také pro jeho ladění) je běžný počítač PC.

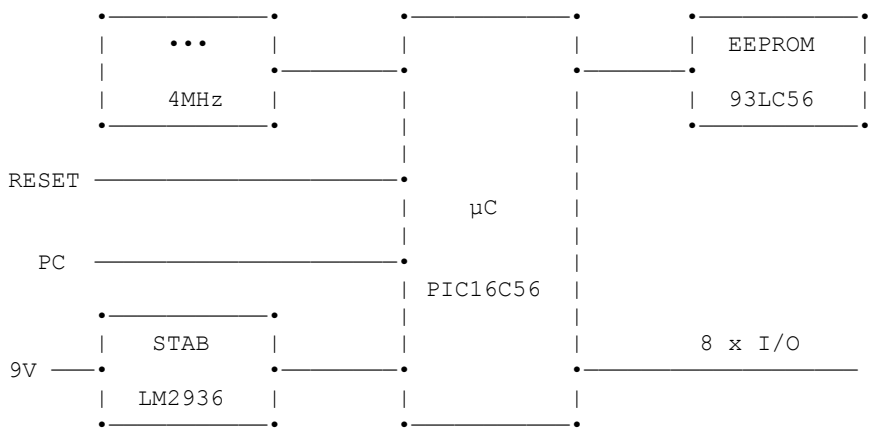
Propojovací kabel spojující mikropočítač UCB/PIC s osobním počítačem PC se na straně mikropočítače připojí na tříkolíkový konektor a na straně PC na jeho paralelní port.

Další konektor mikropočítače UCP/PIC ("systémový") slouží k připojení napájecího napětí, signálu reset a osmi aplikačních I/O linek.

Bohaté aplikační možnosti dává mikropočítači i široký rozsah napájecího napětí a malý příkon. Vestavěný stabilizátor přichází ke cti v případech, je-li napájecí napětí v rozsahu 5 až 15V. UCB/PIC může však být připojen i na napájecí napětí 3 až 5 V, protože v celém tomto intervalu jsou zaručeny funkce použitých integrovaných obvodů. Spotřeba mikropočítače je přibližně 2 mA (aplikační linky naprázdno), programově ji lze omezit na přibližně 20 uA uvedením do stavu SLEEP.

Každá z osmi jednobitových aplikačních linek může být nezávisle použita ve vstupním nebo výstupním módu. Nastaveny a rozpoznány mohou být dva (obvyklým způsobem přiřazené) stavy každé linky - stav L a stav H. Za pozornost stojí, že ve výstupním módu může být linka zapojena jako zdroj (proud jedné linky max. 20 mA, všech současně max. 40 mA) nebo spotřebič (proud jedné linky max. 25 mA, všech současně max. 50 mA).

1.2 Blokové schéma mikropočítače



1.3 Programové vybavení mikropočítače

Programovací jazyk PBASIC mikropočítače UCB/PIC je orientován především na práci s aplikačními linkami a na operace s celými čísly v rozsahu 0 až 65535. Konstanty mohou být deklarovány jako dekadické, hexadecimální (např.$64),binární (např.%11001010) a znakové (např."A","Ahoj").

Pro uložení proměnných má programovací jazyk PBASIC k dispozici 16 bajtů paměti RWM na čipu mikropočítače PIC16C56. Dva bajty jsou vyhrazeny pro konfiguraci a řízení aplikačních linek. Zbylý prostor může být využit jako obecné proměnné typu bajt (b0 až B13) nebo typu slovo (w0 až w6). Výjimkou je proměnná w6, v níž překladač uchovává návratovou adresu při volání podprogramu. Proměnné b0 a b1 (w0) mohou být navíc použity jako pole jednobitových proměnných.

Data lze přímo zapisovat i číst do/z paměti EEPROM.

Pro zápis výrazů s proměnnými a s konstantami je definováno čtrnáct operátorů především celočíselné aritmetiky a Booleovy algebry.

Konstantám, proměnným i aplikačním linkám mohou být v deklaračním úseku programu přiřazena symbolická jména (např. symbol TEPLOTA = b3), která přispívají k lepší čitelnosti programu. Příkazové řádky se nečíslují. Je-li třeba, příkaz může být opatřen návěštím.

PBASIC nerozlišuje, s výjimkou znakových konstant, velká a malá písmena.

Příkazy jazyka PBASIC jsou, jak již bylo dříve uvedeno, účelově zaměřeny. Výhodou je, že důsledkem jednoduchého volání některých z nich je ucelená činnost. Následující přehled obsahuje všechny příkazy jazyka.

BRANCH	větvení programu podle hodnoty parametru
BUTTON	snímání stavu tlačítka na vstupní lince, potlačení jeho zákmitů, auto-repeat, odpovídající větvení programu
EEPROM	zápis dat do paměti EEPROM před uložením programu
END	trvalé uvedení mikropočítače do stavu SLEEP
FOR..NEXT	programový cyklus s daným počtem opakování
GOSUB	volání podprogramu
GOTO	skok na určené návěští
HIGH	uvedení výstupní linky do stavu HI
IF..THEN	podmíněné větvení programu
INPUT	nastavení orientace linky směrem vstup
LET	přiřazení (není povinné)
LOOKDOWN	čtení indexu z převodní tabulky
LOOKUP	čtení hodnoty z převodní tabulky
LOW	nastavení výstupní linky do stavu LO
NAP	uvedení do stavu SLEEP na určenou dobu (18ms až 2.3s)
OUTPUT	nastavení orientace linky směrem výstup
PAUSE	přerušení vykonávání programu na určenou dobu
POT	čtení hodnoty proměnného rezistoru připojeného v sérii s kondenzátorem na vstupní linku
PULSIN	měření délky impulsu na vstupní lince
PULSOUT	generování impulsu určené délky na výstupní lince
PWM	generování šířkově modulovaných impulsů na výstupní lince
RANDOM	generování pseudonáhodného čísla
READ	čtení dat z paměti EEPROM
RETURN	návrat z podprogramu
REVERSE	změna orientace linky z jednoho směru do druhého
SERIN	ustavení asynchronního sériového vstupu na určené lince
SEROUT	ustavení asynchronního sériového výstupu na určené lince
SLEEP	uvedení do stavu SLEEP na určenou dobu (1s až asi 18 hodin)
SOUND	generování tónu určené výšky a délky
TOGGLE	změna stavu výstupní linky z LO na HI nebo z HI na LO
WRITE	zápis dat do paměti EEPROM

2 Vývojové prostředí

2.1 Ovládání vývojového prostředí

Vývojové prostředí poskytuje služby potřebné k tvorbě a ladění programů a jejich přenosu do paměti EEPROM mikropočítače UCB/PIC.
Po spuštění vývojového prostředí na počítači PC se na obrazovce monitoru zobrazí plocha editoru. V horní části se nachází řádek hlavní nabídky s pěti položkami (Help, Run, Load, Save, Quit). V názvu každé položky je zvýrazněno jedno písmeno. Stiskem klávesy s tímto názvem v kombinaci s klávesou ALT položku navolíme.

Help (Alt-H)

Tato volba otevře na obrazovce zvláštní okno s přehledným zobrazením ovládání vývojového prostředí.

Run (Alt-R)

Tento příkaz v sobě zahrnuje několik činností:
1) syntaktická kontrola programu (v případě chyby se činnost přeruší, na obrazovce se objeví chybové hlášení a místo chyby je označeno kurzorem)
2) překlad programu
3) kontrola spojení PC s mikropočítačem (není-li spojení navázáno je hlášena chyba)
4) přesun programu do paměti EEPROM mikropočítače (na obrazovce je graficky znázorněno obsazení paměti mikropočítače uživatelským programem)
5) spuštění uživatelského programu

Load (Alt-L)

Přesun programu z disku do editoru vývojového prostředí. Po navolení této položky se na obrazovce se otevře okno s názvy souborů uložených ve stejném adresáři jako vývojové prostředí. Kurzorovými klávesami lze vybrat požadovaný soubor. Stiskem klávesy Esc se otevře jiné okno, v němž můžeme zadat přístupovou cestu k souboru uloženém v jiném adresáři.

Save (Alt-S)

Uložení programu na disk. Tato volba otevře na obrazovce okno v němž je třeba zadat jméno, pod kterým bude program uložen na disk.

Quit (Alt-Q)

Opuštění vývojového prostředí a návrat do DOSu. Pokud dosud nebyl program z editoru uložen, je otevřeno okno s otázkou, zda se má před opuštěním vývojového prostředí uložit. Kladná odpověď otevře nové okno pro zadání jména, pod kterým bude program uložen.

2.2 Ovládání editoru

Přesuny kurzoru	
šipka vlevo	posun kurzoru o jednu pozici vlevo
šipka vpravo	posun kurzoru o jednu pozici vpravo
šipka nahoru	posun kurzoru o jeden řádek nahoru
šipka dolů	posun kurzoru o jeden řádek dolů
Ctrl-šipka vlevo	posun kursoru na následující slovo vlevo
Ctrl-šipka vpravo	posun kursoru na následující slovo vpravo
Home	přesun kurzoru na začátek řádku
End	přesun kurzoru na konec řádku
Page Up	přesun kurzoru o jednu obrazovku nahoru
Page Down	přesun kurzoru o jednu obrazovku dolů
Ctrl-Page Up	přesun kurzoru na začátek souboru
Ctrl-Page Down	přesun kurzoru na konec souboru

Mazání	
Backspace	mazání znaku na pozici před kurzorem
Delete	mazání znaku na pozici kurzoru
Shift-Backspace	mazání od začátku řádku po znak vlevo od
Shift-Delete	mazání od pozice kurzoru do konce řádku
Ctrl-Backspace	mazání řádku

Blokové operace	
Alt-X	přesun označeného textu do clipboardu
Alt-C	kopírování označeného textu do clipboardu
Alt-V	vložení textu z clipboardu na pozici kurzoru
Alt-F	hledání řetězce znaků
Alt-N	opakované hledání (další výskyt řetězce)

Označení textu	
Shift-šipka vlevo	jednoho znaku vlevo
Shift-šipka vpravo	jednoho znaku vpravo
Shift-šipka nahoru	předcházejícího řádku
Shift-šipka dolů	následujícího řádku
Shift-Ctrl-šipka vlevo	slova vlevo
Shift-Ctrl-šipka vpravo	slova vpravo
Shift-Home	do začátku řádku
Shift-End	do konce řádku
Shift-Page Up	jedné obrazovky nahoru
Shift-Page-Down	jedné obrazovky dolů
Shift-Ctrl-Page Up	do začátku souboru
Shift-Ctrl-Page Down	do konce souboru
Shift-Insert	slova na pozici kurzoru
ESC	zrušení označení textu

3 Programovací jazyk PBASIC

3.1 Zásady psaní programu

Konstanty

Konstanty mohou být vyjádřeny jako číslo desítkové, dvojkové, šestnáctkové nebo jako znak ASCII. Dvojkové číslo musí začínat znakem %, šestnáctkové znakem $. Znak ASCII musí být uzavřen v uvozovkách.

```
100              desítkové číslo
$64              šestnáctkové číslo
%10011000        dvojkové číslo
"A"              ASCII znak A (kód 65)
"Hello"          ASCII znaky  H,e,l,l,o
```

Návěští

Používá se k označení určitého místa v programu (na rozdíl od jiných verzí BASICu, které užívají číslování řádků programu).
Návěští tvoří libovolná kombinace písmen, číslic a znaku podtržítko.
První znak nesmí být číslice. Jako návěští nesmí být použito rezervované jméno (např.serin, toggle,goto...atd.).
Je-li návěští v programovém řádku použito jako první, musí být zakončeno dvojtečkou.

```
loop: toggle 0
        for b0 = 1 to 10
        toggle 1
        next
      goto loop
```

Jména proměnných a konstant

Proměnné a konstanty mohou být v programu označeny jménem. Tvoří se podle stejných zásad jako návěští, není však zakončeno dvojtečkou a musí být definováno direktivou **symbol**.

```
symbol start =  1
symbol end   = 10
symbol cou t = b0
       for cout = start to end
       toggle 1
       next
```

Komentář

Komentář začíná apostrofem a pokračuje až do konce řádku. Lze také použít obvyklé REM.

```
symbol relay  =  3      `pojmenování a definování konstanty
symbol length = b5      `pojmenování proměnné
```

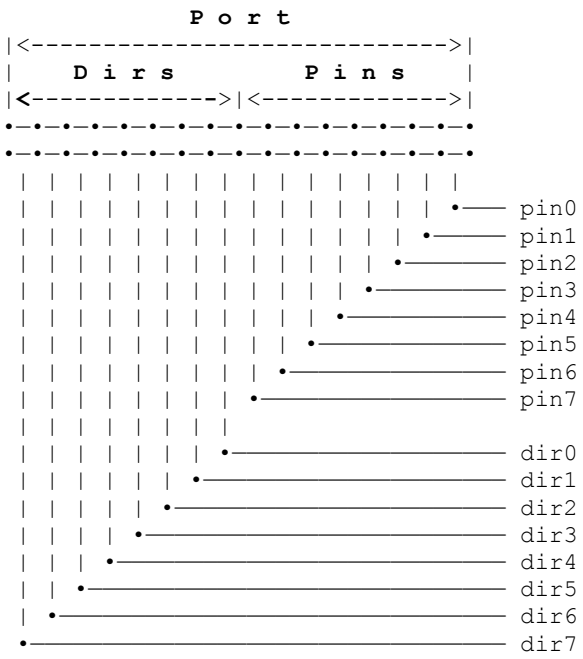
Poznámky

Editor nerozlišuje malá a velká písmena.
Jeden řádek programu může obsahovat více instrukcí oddělených dvojtečkou.

3.2 Proměnné jazyka PBASIC

Programovací jazyk PBASIC využívá 16 bajtů vnitřní paměti dat mikropočítače jako registry pro uložení proměnných.

Dva bajty jsou použity pro I/O port. Jeden bajt (Pins) jako výstupní vyrovnávací registr I/O portu, druhý bajt (Dirs) jako registr řídící směr přenosu dat I/O portem. Úroveň logické nuly v některém z jeho bitů způsobí, že odpovídající vývod portu se nastaví jako vstup. Úroveň logické jedničky nastaví odpovídající vývod jako výstup. Oba registry jsou bitově přístupné.

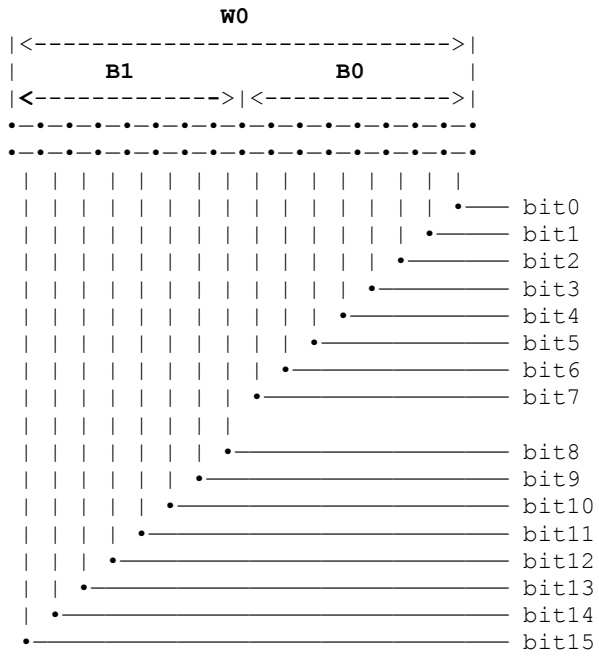


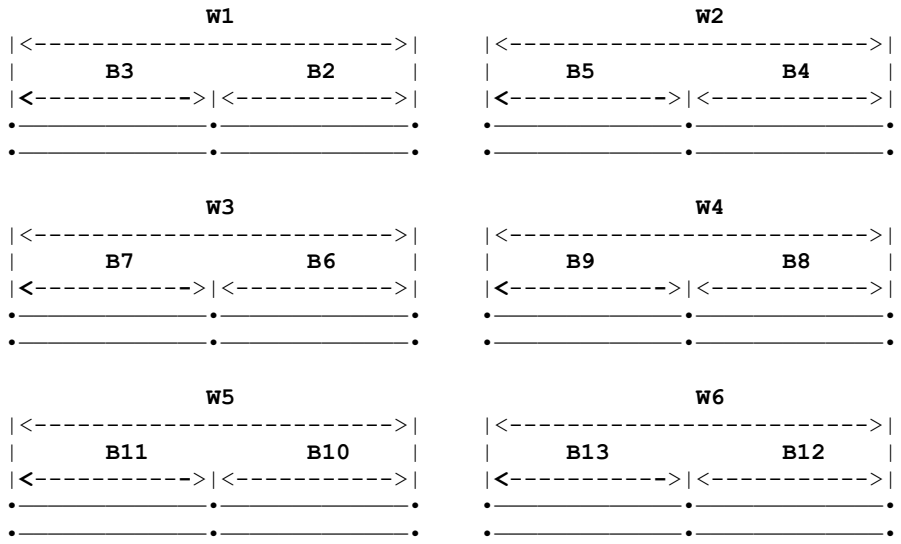
Například příkaz dirs = %00001111 nastaví dolní polovinu portu (piny číslo 0 až 3) jako výstup, horní polovinu portu (piny č.4 až 7) jako vstup.

Dalších 14 bajtů je použito pro data. Mohou být organizovány jako:

- proměnné o délce 16 bitů (proměnné W0 až W6),proměnná W0 je bitově přístupná
- proměnné o délce 8 bitů (proměnné B0 až B13), proměnné B0 a B1 jsou bitově přístupné

Pokud je v programu volán podprogram, je proměnná W6 použita pro uložení návratové adresy!


W0
B1
B0
bit0
bit1
bit2
bit3
bit4
bit5
bit6
bit7
bit8
bit9
bit10
bit11
bit12
bit13
bit14
bit15


W1
B3
B2
W2
B5
B4
W3
B7
B6
W4
B9
B8
W5
B11
B10
W6
B13
B12

3.3 Příkazy jazyka PBASIC

A) Příkazy větvení programu

IF ... THEN

IF *proměnnáA* ?? *hodnotaA* {AND|OR *proměnnáB* ?? *hodnotaB* ...} THEN *návěští*

Porovná velikost proměnné (proměnných) s hodnotou a provede skok při pozitivním výsledku porovnání. Při negativním výsledku porovnání pokračuje program následující instrukcí.

proměnnáA,proměnnáB ...- porovnávané proměnné
hodnotaA,hodnotaB ... - proměnné nebo konstanty jež se porovnávají s velikostí proměnných
návěští - určuje místo v programu, na které bude veden skok při pozitivním výsledku porovnání

?? - jeden z následujících operátorů: = , <> , > , < , >= , <=

!Za THEN může následovat jen návěští, nikoliv příkaz Basicu!

Příklad:

```
abc: serin 0,n2400,b2

     IF  B2=185 THEN  XYZ       'je-li bajt přijatý sériovým kanálem
                                'roven 185, pak skok na návěští xyz,
                                'jinak program pokračuje následujícím
                                'příkazem
     high 2

     goto abc

xyz: low 2

     goto abc
```

BRANCH

BRANCH *pořadí*,(*návěští0*,*návěští1*,...,*návěštíN*)

Skok na návěští určené parametrem pořadí.

pořadí - proměnná nebo konstanta (0 až N)
- určuje, na které návěští bude veden skok
návěští0,*návěští1*,.. - určují místa v programu, na která může být veden skok

Není-li parametr pořad v rozsahu 0 až N, žádný skok se neuskuteční a program bude pokračovat následujícím příkazem.

Příklad:

```
abc:    serin   0,n2400,("code"),b2     'čekání   na    přijetí   znaků
                                        '"c","o","d","e" sériovým kaná-
                                        'lem a po  jejich přijetí další
                                        'bajt uloží do proměnné b2

        BRANCH b2,(dej,ghi,jkl)         'pro b2=0 skok na návěští def
                                        'pro b2=1 skok na návěští ghi
                                        'pro b2=2 skok na návěští jkl

        goto  abc                       'není-li b2 v rozmezí 0 až 2,
                                        'skok na návěští abc

def:    .......

ghi:    .......

jkl:    .......
```

GOTO

GOTO *návěští*

Skok v programu do místa označeného parametrem návěští.

návěští - označuje místo v programu, na které bude veden skok

Příklad:

```
abc: pulsout 0,100

     GOTO abc           'skok na návěští abc
```

GOSUB

GOSUB *návěští*

Volání podprogramu.

návěští - označuje začátek volaného podprogramu

Příklad:

```
     for b4=0 to 10

     GOSUB abc          'úschova návratové adresy a pak přechod do
                        'podprogramu na návěští abc
     next

abc: pulsout 0,b4

     toggle 1

     return
```

Poznámka: Program může obsahovat až 16 volání podprogramu. Vnoření podprogramů je možné do čtvrté úrovně.

RETURN

RETURN

Návrat z podprogramu.

Program pokračuje příkazem za posledním voláním podprogramu příkazem GOSUB.

Příklad:

```
     for b4=0 to 10
     gosub abc
     next
abc: pulsout 0,b4
     toggle 1
     RETURN            'návrat do hlavního programu na příkaz next
```

B) Příkaz cyklu

FOR...NEXT

FOR *proměnná* = *hodnotaA* TO *hodnotaB* {STEP {-}*hodnotaC*}
 .
 .
 .
NEXT {*proměnná*}

- proměnné je přiřazena hodnotaA
- je vykonán program mezi FOR a NEXT
- proměnná je modifikována (zvýšena/snížena) o hodnotuC (není-li uvedena, je proměnná zvýšena o 1)
- program je opětovně vykonáván pokud je proměnná menší (při jejím zvyšování) nebo větší (při jejím snižování) než hodnotaB, jinak program pokračuje příkazem následujícím za příkazem NEXT
- bez ohledu na parametry hodnotaA a hodnotaB je program alespoň jednou vykonán

proměnná - proměnná o velikosti bitu, bajtu nebo slova (slouží jako vnitřní čítač průchodů cyklem)
hodnotaA - proměnná nebo konstanta, udává počáteční hodnotu proměnné
hodnotaB - proměnná nebo konstanta, udává koncovou hodnotu proměnné
hodnotaC - volitelná hodnota, o kterou je proměnná zvýšena nebo snížena (když je záporná)
- není-li uvedena, je proměnná zvýšena o 1

Poznámka: Vnoření smyček může být do osmé úrovně.

Příklad:

```
     FOR b2=0 TO 255     'nastavení rozsahu počítadla cyklů
     pins=b2
     NEXT                'opakování příkazů cyklu
```

C) Příkazy pro numerické operace

LET

{LET} *proměnná* = {-}*hodnotaA* ?? *hodnotaB* ...

Provede manipulaci s hodnotou proměnné.

proměnná - jméno proměnné, jejíž hodnotou příkaz mění

hodnotaA,*hodnotaB*,.. - proměnné nebo konstanty, které ovlivní hodnotu proměnné

?? - představuje některý z následujících operátorů

```
+   součet
-   rozdíl
*   násobení (nižší slovo výsledku)
**  násobení (vyšší slovo výsledku)
/   dělení (podíl)
//  dělení (zbytek po dělení)
MIN dolní limit proměnné
MAX horní limit proměnné
&   logický součin AND
|   logický součet OR
^   logický výběrový součet XOR
&/  negovaný logický součin NAND
|/  negovaný logický součet NOR
^/  negovaný logický výběrový součet NXOR
```

Příkaz LET je nepovinný.

Příklad:

```
abc: pot 0,100,b3

     LET b3=b3/2        'velikost proměnné zmenšena na polovinu

     b3=b3 max 100      'omezení velikosti proměnné na rozsah 0 až 100
                        'nepovinné klíčové slovo LET vynecháno
```

LOOKUP

LOOKUP *pořadí*,(*konstanta0*, *konstanta1*,...,*konstantaN*),*proměnná*

Vyjme z tabulky hodnotu odpovídající parametru pořadí a tuto hodnotu přiřadí proměnné.

pořadí - určuje hodnotu v tabulce, která bude přiřazena proměnné
konstanta0,konstanta1,... - tabulka hodnot

proměnná - proměnná, které je přiřazena velikost jedné z konstant v tabulce

Pokud velikost parametru pořadí není v rozsahu 0 až N, zůstane proměnná nezměněna.

Příklad:

```
     for b2=0 to 25

     LOOKUP b2,(65,66,67,....),b3     'převod obsahu proměnné (0 - 25)
                                      'na znak ASCII (A - Z)
     next
```

LOOKDOWN

LOOKDOWN *prvek*,(*konstanta0*, *konstanta1*,...,*konstantaN*),*proměnná*

Porovnává velikost parametru prvek s jednotlivými konstantami a v případě rovnosti uloží do parametru proměnná pořadové číslo konstanty.

prvek - proměnná nebo konstanta, která je porovnávána s konstantami v tabulce

konstanta0,*konstanta1*,... - tabulka hodnot, se kterými je porovnávána velikost prvku

proměnná - obsahuje pořadové číslo konstanty rovnající se prvku

Pokud se parametr prvek nerovná žádné z konstant, zůstane proměnná nezměněna.

Příklad:

```
     serin 0,n2400,b2

     LOOKDOWN b2,(65,88,93),b3        'pro b2=0 je b3=65
                                      'pro b2=1 je b3=88
                                      'pro b2=2 je b3=93
```

RANDOM

RANDOM *proměnná*

Příkaz generuje pseudonáhodné číslo v rozsahu 0 až 65535.

proměnná - proměnná (0 až 65535) je pracovní proměnnou příkazu a obsahuje také výsledek jeho činnosti

Tento příkaz negeneruje skutečně náhodná čísla. Pokud zadáme stejnou počáteční hodnotu proměnné, obdržíme stejný výsledek (v mnoha aplikacích to však vyhovuje).
Chceme-li skutečně náhodný výsledek, musíme do činnosti příkazu doplnit prvek neurčitosti, například počáteční hodnotu proměnné získáme čtením sÉrového portu nebo hodin reálného času.

Příklad:

```
abc: RANDOM W1          'generuje 16-ti bitové náhodné číslo uložené
                        'v proměnné w1
     sound 1,(b2,10)    'generuje náhodné tóny užitím dolního bajtu
                        'proměnné w1
     goto abc
```

D) Příkazy pro číslicový vstup/výstup

LOW

LOW *linka*

Nastaví vývod mikropočítače (určený parametrem linka) do režimu výstup s úrovní logické nuly.

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
abc: LOW 3              'nastaví linku číslo 3 jako výstup s úrovní
                        'logické nuly
     pause 1000

     high 3

     goto abc
```

HIGH

HIGH *linka*

Nastaví vývod mikropočítače (určený parametrem linka) do režimu výstup s úrovní logické jedničky.

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
ibm: HIGH 3             'nastaví linku číslo 3 jako výstup s úrovní
                        'logické jedničky
     low  2

     HIGH 2             'nastaví linku číslo 2 jako výstup s úrovní
                        'logické jedničky
     low  3

     goto ibm
```

TOGGLE

TOGGLE *linka*

Nastaví vývod mikropočítače (určený parametrem linka) do režimu výstup a změní jeho logickou úroveň na opačnou.

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
     for b2=1 to 25

     TOGGLE 5           'změní logickou úroveň linky číslo 5 na opačnou
                        '(linka je nastavena jako výstup)
     next
```

INPUT

INPUT *linka*

Vývod mikropočítače (určený parametrem linka) bude nastaven do režimu vstup.

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
     INPUT 5            'nastaví linku číslo 5 jako vstup
```

```
abc: if pin5=1 then xyz

     goto abc

xyz: serout 3,n300,(65)

nbc: if pin5=1 then nbc

     goto abc
```

OUTPUT

OUTPUT *linka*

Vývod mikropočítače (určený parametrem linka) bude nastaven do režimu výstup.

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
     bit0=0
     bit1=1

     OUTPUT 2          'nastaví linku číslo 2 jako výstup

abc: pin2=bit0
     pin2=bit1

     goto  abc
```

REVERSE

REVERSE *linka*

Nastaví směr přenosu dat vývodem mikropočítače na opačný (číslo vývodu je určeno parametrem linka).

linka - proměnná nebo konstanta (0 až 7)

Příklad:

```
     dir3=0            'nastaví linku číslo 3 jako vstup

     REVERSE 3         'nastaví opačnou orientaci linky číslo 3,
                       'tj. jako výstup
     REVERSE 3         'nastaví opačnou orientaci linky číslo 3,
                       'tj. jako vstup
```

PULSIN

PULSIN *linka*,*hrana*,*proměnná*

Měří délku impulsu na vývodu mikropočítače (určeného parametrem linka) v časových jednotkách 10 us.

linka - proměnná nebo konstanta (0 až 7)
hrana - proměnná nebo konstanta (0 nebo 1)
- určuje hranu, od které začne měření délky pulsu
0 - sestupná hrana 1 -> 0

1 - náběžná hrana 0 -> 1
proměnná - proměnná obsahující výsledek měření (1 až 65536) v časových jednotkách 10 us
- při výsledku větším než 0.65536s je obsah proměnné roven 0

Příklad:

```
     PULSIN 4,0,w2      'měří délku vstupního pulsu na lince číslo 4
                        'měření začíná od sestupné hrany a končí
                        'hranou vzestupnou

     serout 1,n300,(b5)
```

PULSOUT

PULSOUT *linka,délka*

Generuje puls na vývodu mikropočítače (určeného parametrem linka) tak, že na určitou dobu (danou parametrem délka) invertuje jeho stav.

linka - proměnná nebo konstanta (0 až 7)
délka - proměnná nebo konstanta (0 až 65535) určující délku pulsu v časových jednotkách 10 us

Příklad:

```
abc: PULSOUT 0,3        'na lince číslo 0 generuje puls délky tak, že
                        'na dobu 30 us invertuje její logickou úroveň
     pause 1

     goto abc
```

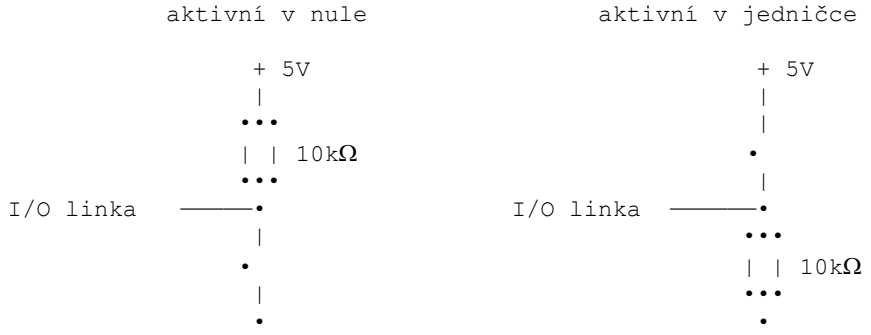
BUTTON

BUTTON *linka,stav1,zpoždění,perioda,proměnná,stav2,návěští*

Čtení stavu tlačítka, případně i s odstraněním jeho zákmitů, opakované čtení stisknutého tlačítka (auto-repeat) a skok, je-li tlačítko v cílové poloze.

linka - proměnná nebo konstanta (0 až 7)
- určuje číslo I/O linky mikropočítače, na které je připojeno tlačítko

stav1 - proměnná nebo konstanta (0 nebo 1)
- určuje logickou úroveň I/O linky při stisknutém tlačítku



zpoždění - proměnná nebo konstanta (0 až 255)
- určuje jak dlouho musí být tlačítko stisknuto než začne auto-repeat

perioda - proměnná nebo konstanta (0 až 255)
- určuje periodu auto-repeatu
proměnná - proměnná o velikosti bajtu
- pracovní proměnná příkazu
- před použitím příkazu se musí nulovat
stav2 - proměnná nebo konstanta (0 nebo 1)
- určuje stav tlačítka, pro který se má uskutečnit skok
0 - tlačítko nestisknuto
1 - tlačítko stisknuto
návěští - určuje místo v programu, na kterém bude program pokračovat, je-li tlačítko v požadovaném stavu

Poznámka:
a) Časový interval určený parametry zpoždění a perioda se vztahuje k době provedení jednoho cyklu příkazu BUTTON. Příslušnou dobu získáme vynásobením času potřebného na provedení jednoho cyklu příkazu BUTTON velikostí parametru.

b) Parametr zpoždění má dvě význačné hodnoty. Hodnotu 0, kdy se neprovede odstranění zákmitů tlačítka ani auto-repeat, a hodnotu 255, kdy se odstraní zákmity tlačítka, auto-repeat se však neprovádí.

Příklad:

```
     b2=0                        'nulování pracovní proměnné příkazu

loop:BUTTON 3,0,50,10,b2,0,abc   'tlačítko připojeno na linku číslo 3
                                 'při stisknutém tlačítku je na lince
                                 'úroveň logické nuly
                                 'po 50-ti průchodech smyčkou začne
                                 'auto-repeat
                                 'tlačítko je opakovaně čteno při
                                 'každém desátém průchodu smyčkou
                                 'b2 obsahuje počet průchodů smyčkou
                                 'není-li tlačítko stisknuto, provede
                                 'se skok na návěští abc
     pulsout 5,1000

abc: pause 10

     goto loop
```

E) Příkazy sériového vstupu/výstupu

SERIN

```
SERIN linka,mód,(hodnotaA,hodnotaB,...)
SERIN linka,mód,{#}proměnnáA,{#}proměnnáB,....
SERIN linka,mód,(hodnotaA,hodnotaB,...),
      {#}proměnnáA,{#}proměnnáB,..
```

linka - proměnná nebo konstanta (0-7)
- specifikuje použitou aplikační linku

mód - proměnná nebo konstanta (0-7), může být zadán jedním ze dvou způsobů
- číslem (s předcházejícím znakem #) nebo symbolem

#	symbol	přenosová rychlost(Bd)	vstup
0	T2400	2400	přímý
1	T1200	1200	přímý
2	T600	600	přímý
3	T300	300	přímý
4	N2400	2400	invertovaný
5	N1200	1200	invertovaný
6	N600	600	invertovaný
7	N300	300	invertovaný

hodnotaA,hodnotaB,..- proměnné nebo konstanty (0-255)
- představují volitelné podmínky, které musí být správně přijaty, aby mohl příkaz pokračoval ve své činnosti

proměnná - proměnná (0-255) pro uložení přijatých dat

Poznámka:
a) Před jméno proměnné můžeme zařadit znak #. Příkaz pak bude akceptovat data ve formátu ASCII. To je užitečné pro čtení dat ze zařízení s výstupem v ASCII kódu.

b) Přenos dat je vždy ve formátu - 8 datových bitů, bez paritního bitu, 1 stop bit.

Příklad:

```
     read 255,b2               'zjištění poslední adresy obsazené
                               'programem
loop:b2=b2-1                   'určení první volné adresy pro data

     SERIN 0,N300,("ABC"),b3   'linka číslo 0 jako sériový kanál
                               'čekání na příchod "ABC", následující
                               'bajt vložen do b3
     write b2,b3               'uložení bajtu přijatého sériovým
                               'kanálem do paměti EEPROM
     if b2 > 0 then loop       'opakování dokud je v paměti EEPROM
                               'volné místo
```

SEROUT

SEROUT *linka*,*mód*,({#}*hodnotaA*,{#}*hodnotaB*...)

linka - proměnná nebo konstanta (0-7)
- specifikuje použitou aplikační linku

mód - proměnná nebo konstanta (0-15), může být zadán jedním ze dvou způsobů
- číslem (s předcházejícím znakem #) nebo symbolem

#	symbol	přenosová rychlost(Bd)	výstup
0	T2400	2400	přímý
1	T1200	1200	přímý
2	T600	600	přímý
3	T300	300	přímý
4	N2400	2400	invertovaný
5	N1200	1200	invertovaný
6	N600	600	invertovaný
7	N300	300	invertovaný
8	OT2400	2400	přímý - otevřený výstup
9	OT1200	1200	přímý - otevřený výstup
10	OT600	600	přímý - otevřený výstup
11	OT300	300	přímý - otevřený výstup
12	ON2400	2400	invertovaný - otevřený výstup
13	ON1200	1200	invertovaný - otevřený výstup
14	ON600	600	invertovaný - otevřený výstup
15	ON300	300	invertovaný - otevřený výstup

proměnnáA,*proměnnáB*,...- proměnné nebo konstanty (0-255), které jsou vyslány sériovým kanálem

Poznámka:
a) Proměnné nebo konstanty mohou mít předřazen symbol #. Ten způsobí, že data budou vyslána v ASCII kódu. To je užitečné, při přenosu do zařízení, které akceptuje data v ASCII kódu.

b) Přenos dat je vždy ve formátu - 8 datových bitů, bez paritního bitu, 1 stop bit.

Příklad:

```
abc: pot 0,100,b2

     SEROUT 1,N300,(b2)  'vyslání hodnoty potenciometru sériovým
                         'kanálem na lince číslo 1
     goto abc
```

F) Příkazy pro analogový vstup/výstup

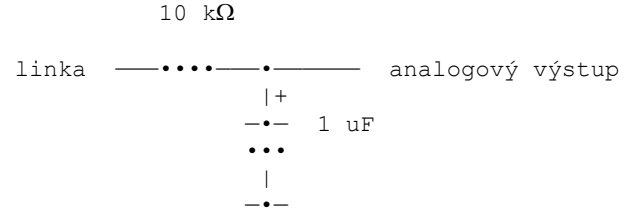
PWM

PWM *linka*,*plnění*, *počet*

Provede výstup pulsu se zadaným činitelem plnění na výstupní aplikační lince a pak linku převede do vstupního stavu.

linka - proměnná nebo konstanta (0-7)
- specifikuje použitou aplikační linku

plnění - proměnná nebo konstanta (0-255)
- určuje činitel plnění (tím i velikost analogového napětí v rozsahu 0-5V, je-li k lince připojen obvod dle schématu)



počet - proměnná nebo konstanta (0-255)
- určuje počet generovaných period (každá perioda trvá asi 5 ms)

Protože se kondenzátor samovolně vybíjí, je třeba příkaz PWM periodicky opakovat k doplnění náboje. Vybitý kondenzátor potřebuje několik cyklů k úplnému nabití. Potřebujeme-li odebírat větší proud, musíme analogový výstup doplnit zesilovačem (např. operačním).

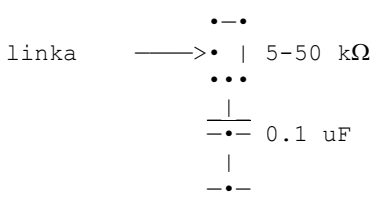
Příklad:

```
abc: serin 0,n300,b2    'příjem dat sériovým kanálem

     PWM 1,b2,20        'výstup analogového napětí na lince číslo 1
                        'velikost dána bajtem přijatým sériovým kanálem
     goto abc
```

POT

POT *linka*,*měřítko*,*proměnná*
Čte hodnotu potenciometru (termistoru,fotoodporu,nebo jiného proměnného odporu) připojeného na vstupní linku dle schématu měřením doby potřebné k nabití kondenzátoru.



linka - proměnná nebo konstanta (0-7)
- specifikuje použitou aplikační linku

měřítko - proměnná nebo konstanta (0-255); používá se k přepočtu vnitřního výsledku; přečtená 16-ti bitová hodnota je násobena koeficientem (*měřítko*/256) a tak je redukována na rozsah jednoho bajtu

proměnná - slouží k uložení konečného výsledku

Nalezení optimálního měřítka příkazu POT
1) Zadáme příkaz editoru Alt-P (mikropočítač musí být připojen k PC a rezistor, jehož hodnotu chceme příkazem POT zjišťovat, musí být připojen k mikropočítači).

2) V okně, jež se otevře na obrazovce, zadáme číslo aplikační linky s připojeným rezistorem

3) Editor přenese do mikropočítače krátký program (případný program v paměti mikropočítače bude tímto programem přepsán)

4) V okně, jež se otevře na obrazovce, se zobrazuje měřítko a hodnota odporu čtená příkazem POT. Jsou možné dvě funkce programu - zjišťování měřítka nebo čtení velikosti odporu (přepínají se klávesou MEZERA).

a) *Zjištění velikosti měřítka*
Měníme hodnotu rezistoru tak dlouho, dokud není zobrazena nejmenší velikost měřítka. Tuto hodnotu použijeme v příkazu POT.

b) *Čtení velikosti odporu*
Tato funkce se používá k ověření velikosti měřítka. Po stisknutí klávesy MEZERA se velikost měřítka zapamatuje a mikropočítač začne pro tuto velikost měřítka průběžně měřit velikost odporu. Měníme-li odpor měřeného rezistoru mezi krajními hodnotami, čtená hodnota se při optimální velikosti měřítka mění v rozsahu 0 až 255.

Podle potřeby můžeme celý postup opakovat.

Příklad:

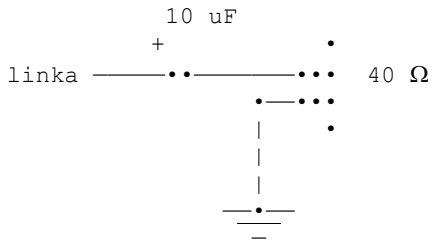
```
abc: POT 0,100,b2          'čtení hodnoty potenciometru připojeného na
                           'lince číslo 0
     serout 1,n300,(b2)    'vyslání přečtené hodnoty sériovým kanálem
                           'na lince číslo 1
     goto abc              'opakování programu
```

G) Příkaz pro zvukový výstup

SOUND

SOUND *linka*,(*tónA*,*délkaA*,*tónB*,*délkaB*,...)

Příkaz generuje tóny se zadanou délkou v reproduktoru připojeném k výstupní aplikační lince dle následujícího schématu (tvar tónů je obdélníkový).



linka - proměnná nebo konstanta (0-7)
- specifikuje použitou aplikační linku

tón - proměnná nebo konstanta (0-255) určující typ a kmitočet tónu
0 - ticho po zadanou dobu
1-127 - tóny s rostoucím kmitočtem
128-255 - bílý šum

délka - proměnná nebo konstanta (0-255) určující délku tónu

Příklad:

```
     for b2=0 to 255

     SOUND 1,(25,10,b2,10)      'generování dvojice tónů stejné délky
                                'kmitočet prvního tónu je stálý,
                                'kmitočet druhého se postupně zvyšuje
     next
```

H) Příkazy pro přístup do paměti EEPROM

READ

READ *umístění,proměnná*

Příkaz čte obsah adresy paměti EEPROM (určené parametrem umístění) do proměnné (určené parametrem proměnná).

umístění - proměnná nebo konstanta (0 až 255)
proměnná - obsahuje hodnotu (0 až 255) přečtenou z paměti

Poznámka:
Na adrese 255 je vždy uložena adresa posledního paměťového místa obsazeného programem.

Příklad:

```
     READ 255,b2          'zjištění adresy posledního paměťového místa
                          'obsazeného programem
loop:b2=b2-1

     serin 0,n300,b3

     write b2,b3

     if b2 > 0 then loop
```

WRITE

WRITE *umístění,data*

Příkaz zapisuje data (určená parametrem data) do paměti EEPROM na adresu určenou parametrem umístění.

umístění - proměnná nebo konstanta (0 až 255)
data - proměnná nebo konstanta (0 až 255)

Příklad:

```
     read 255,b2

loop:b2=b2-1

     serin 0,n300,b3

     WRITE b2,b3          'uložení obsahu proměnné b3 do paměti EEPROM
                          'na adresu uloženou v proměnné b2
     if b2 > 0 then loop
```

EEPROM

EEPROM {*umístění*,}(*hodnotaA*,*hodnotaB*,...)

Ukládá hodnoty do paměti EEPROM. Toto je užitečné pro vložení dat,která pak bude používat program. Příkaz se provede před uložením programu.Vlastní příkaz se nepřenáší do mikropočítače a nezabírá proto žádný prostor v jeho paměti programu.

umístění - proměnná nebo konstanta (0 až 255) určující počáteční adresu, od které budou data postupně ukládána do paměti EEPROM

hodnotaA,*hodnotaB*,... - proměnné nebo konstanty (0 až 255), které jsou postupně ukládány do paměti, první na pozici danou parametrem umístění

Příklad:

```
EEPROM 0,(5,23,17,158,2)    'ještě před zavedením programu je do
                            'paměti EEPROM uloženo 5 údajů
                            'adresa    0   1   2   3   4
                            '  data    5  23  17  158  2
```

I) Příkaz pro generování časového intervalu

PAUSE

PAUSE *délka*

Příkaz generuje časové zpoždění. Parametr délka je proměnná nebo konstanta v rozsahu 0 až 65535 určující dobu zpoždění v milisekundách.

Příklad:

```
abc: low 2

     PAUSE 100          'čekání 100 ms

     high 2

     PAUSE 100          'čekání 100 ms

     goto abc
```

J) Příkazy pro řízení napájení

NAP

NAP *interval*

Tento příkaz uvede mikropočítač do sleep módu (režim se sníženým příkonem - napájecí proud je snížen asi na 20 uA) na dobu 18ms až 2.3s.

interval - proměnná nebo konstanta (0 až 7) určující dobu trvání režimu se sníženým příkonem podle vztahu

$$2^{interval} * 18\ ms$$

Příklad:

```
     .....
     .....
     .....
     NAP 7              'uvedení mikropočítače do úsporného módu
                        'na dobu 2304 ms
```

SLEEP

SLEEP *interval*

Tento příkaz uvede mikropočítač do sleep módu (režim se sníženým příkonem - napájecí proud je snížen asi na 20 uA).

interval - proměnná nebo konstanta (0 až 65535) určující dobu trvání režimu se sníženým příkonem v sekundách

Příklad:

```
.....
.....
SLEEP 3600          'uvedení mikropočítače do úsporného módu na
                    'dobu jedné hodiny (3600 s)
goto xyz
```

END

END

Tento příkaz uvede trvale mikropočítač do sleep módu (režim se sníženým příkonem - napájecí proud je snížen asi na 20 uA).
Ukončení je možné jen odpojením a opětovným připojením napájení mikropočítače nebo kontaktem PC s mikropočítačem.

K) Příkaz pro ladění programu

DEBUG

Používá se pro vyslání textu nebo hodnoty proměnné z mikropočítače zpět do PC k zobrazení. Lze jej přirovnat k příkazu PRINT v jiných verzích jazyka BASIC. Nutnou podmínkou je zachování propojení mezi mikropočítačem a PC.

Text musí být uzavřen v uvozovkách. Hodnoty proměnných jsou zobrazovány v desítkové soustavě. Zobrazení v hexadecimální soustavě zajistíme znakem $, zobrazení v binární soustavě znakem % před jménem proměnné.
Znak # před jménem proměnné způsobí zobrazení pouze hodnoty proměnné, jméno proměnné zobrazeno nebude. Příkaz akceptuje také dva řídící příkazy - cls (clear screen) a cr (carriage return).

Několik příkladů:

```
b2=72

DEBUG b2            'zobrazení jména i hodnoty proměnné: b2=72

DEBUG #b2           'zobrazení jen hodnoty proměnné: 72

DEBUG #%b2          'zobrazení hodnoty proměnné binárně: %01001000

DEBUG "Napeti=",#b2,"V"                  'zobrazí se: Napeti=72 V

DEBUG "Prvni radek",cr,"Druhy radek"     'Zobrazí se: Prvni radek
                                                      Druhy radek
```