PICAXE – příručka programátora

Návěští (použitelné pro všechny typy PICAXE)

Slouží k označení míst v programu pro příkazy GOTO, GOSUB, BRANCH. Musí začínat písmenem nebo podtržítkem, končit dvojtečkou. Návěští nesmí být shodné s rezervovaným klíčovým slovem (příkazem) a nesmí obsahovat znaky s diakritikou.

Například:

```
loop:
     high 1              ' switch on output 1
     pause 5000          ' wait 5 seconds
     low 1               ' switch off output 1
     pause 5000          ' wait 5 seconds
     goto loop           ' loop back to start
```

Mezery (whitespace)

Mezery, tabulátory a oddělovače řádků (enter) mohou být použity kdekoliv v programu – například pro zvýšení čitelnosti.

Komentáře (použitelné pro všechny typy PICAXE)

Komentáře jsou uvozeny apostrofem ('), středníkem nebo klíčovým slovem REM a platí do konce řádku.

Například:

```
     high 0              ' make output 0 high
     high 0              ; make output 0 high
     high 0              REM make output 0 high
```

Konstanty (použitelné pro všechny typy PICAXE)

Desítkové (dekadické)	– číslo bez dalšího označení
Dvojkové (binární)	– s prefixem %, například %00101100
Šestnáctkové (hexadecimální)	– s prefixem $, například $F5
Rozsah 16 bitů	– 0 až 65535

Například:

```
     100                 ' 100 decimal
     $64                 ' 64 hex
     %01100100           ' 01100100 binary
     "A"                 ' "A" ascii (65)
     "Hello"             ' "Hello" - equivalent to "H","e","l","l","o"
     B1 = B0 ^ $AA       ' xor variable B0 with AA hex
```

Symboly (použitelné pro všechny typy PICAXE)

Konstantám a proměnným mohou být přiřazeny symbolická jména, což přispívá k čitelnosti programu. Tato jména nesmí obsahovat znaky s diakritikou.

Například:

```
     symbol RED_LED = 7          ' define a constant symbol
     symbol COUNTER = b0         ' define a variable symbol
     let COUNTER = 200           ' preload variable with value 200
loop:                            ' define a program address
                                 ' address symbol end with colons
     high RED_LED                ' switch on output 7
     pause COUNTER               ' wait 0,2 seconds
     low RED_LED                 ' switch off output 7
     pause COUNTER               ' wait 0,2 seconds
     goto loop                   ' loop back to start
```

Proměnné (použitelné pro všechny typy PICAXE)

Univerzální proměnné

K dispozici je 14 byte b0 až b13, které mohou být sdružovány po dvou v 16 bitová slova (word) w0 až w6. Jednotlivé bity b0 a b1 mohou být také adresovány jako bit0 až bit 15.

Všechny univerzální proměnné jsou bez znaménka (unsigned) a na začátku programu obsahují hodnotu nula.

word	***vyšší byte***	***nižší byte***
w0	b1	b0
w1	b3	b2
w2	b5	b4
w3	b7	b6
w4	b9	b8
w5	b11	b10
w6	b13	b12

byte	***MSB***							***LSB***
b0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
b1	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8

Speciální proměnné

pins – při čtení hodnoty (tj. napravo od rovnítka) odpovídá vstupům, při přiřazení hodnoty (tj. nalevo od rovnítka) odpovídá výstupům. Pro testování hodnoty jednotlivých vstupů v příkazu IF se dělí na jednotlivé bity. Bitové proměnné jsou přiřazené pouze existujícím vstupům.

byte	***MSB***						***LSB***
pins	pin7	pin6	–	–	–	pin2	pin1

infra – speciální proměnná pro použití v příkazu INFRAIN

keyvalue – speciální proměnná pro použití v příkazu KEYIN, překrývá se s proměnnou infra

Další paměť pro data

Obsah proměnných b0 až b13 je možné uložit do paměti RAM pomocí příkazů POKE a uložené hodnoty získat zpět příkazem PEEK. S takto uloženými daty není možné provádět žádné operace, dokud nejsou načtena zpět do proměnných b0 až b13 pomocí příkazu PEEK.

Operátory

Aritmetické

+	sčítání
–	odčítání
*****	násobení
******	při násobení 16 x 16 bitů předá vyšších 16 bitů výsledku
/	dělení
//	zbytek po dělení (modulo)
%	zbytek po dělení (modulo) – alternativní zápis
MAX	omezí výsledek shora – na maximální hodnotu
MIN	omezí výsledek zdola – na minimální hodnotu

Logické – fungují po jednotlivých bitech

|	AND	– logický součin
&	OR	– logický součet
^	XOR	– výlučně nebo
&/	AND NOT	– logický součin s druhým operandem negovaným, toto není NAND
|/	OR NOT	– logický součet s druhým operandem negovaným, toto není NOR
^/	XNOR	– logická ekvivalence

Výrazy

Přípustné jsou pouze jednoduché výrazy bez závorek a přednosti operátorů, vyhodnocuje se zleva doprava.

Například:
b0 = b1 + 12 / b3 což odpovídá obvyklejšímu zápisu b0 = (b1 + 12) / b3
w1 = 3 * w0 max 445

Příkazy seřazené podle kategorií

Přiřazení:
- LET

Programová smyčka:
- FOR .. NEXT

Skoky:
- GOTO
- BRANCH
- GOSUB
- RETURN
- INTERRUPT

Podmíněný skok:
- IF .. THEN

Motory:
- FORWARD
- BACKWARD
- HALT

Hodinová frekvence:
- SETFREQ
- CALIBFREQ

Výstupy:
- HIGH
- LOW
- TOGGLE
- SWITCH ON
- SWITCH OFF
- HIGH PORTC
- LOW PORTC

Ostatní vstupní a výstupní operace:
- PULSIN
- PULSOUT
- PWM
- PWMOUT
- SERVO

I^2C sběrnice:
- I2CSLAVE
- READI2C
- WRITEI2C

Infračervené dálkové ovládání:
- INFRAIN
- INFRAIN2
- INFRAOUT

Směr vývodů – vstup / výstup:
- INPUT
- OUTPUT
- REVERSE
- LET DIRS
- LET DIRSC

Komunikace s počítačovou klávesnicí:
- KEYIN
- KEYLED

Vyhledávání v tabulce hodnot:
- LOOKUP
- LOOKDOWN

Prodleva:
- NAP
- PAUSE
- WAIT

Ukončení programu:
- END
- STOP

Práce s RAM kontroleru:
- POKE
- PEEK

Práce s EEPROM kontroleru:
- DATA
- EEPROM
- READ
- WRITE
- READMEM
- WRITEMEM

Zvukový výstup:
- PLAY
- SOUND
- TUNE

One-wire sběrnice:
- READTEMP
- READTEMP12
- READOWCLK
- READOWSN
- RESETOWCLK

Analogově digitální převod:
- READADC
- READADC10

Sériová komunikace:
- SERIN
- SEROUT
- SERTXD

Ostatní:
- RANDOM
- DEBUG

Abecední seznam příkazů

backward (použitelné pro PICAXE: 18, 18A, 18X, 28A, 28X, 40X)

syntaxe:
BACKWARD motor
- motor je A nebo B

funkce:
Spustí otáčení motoru dozadu. Tento příkaz je ekvivalentní low 4, high 5 pro motor A, nebo low 6, high 7 pro motor B.
Bude fungovat správně na desce UMU rev. C (vyrobené v roce 2007). A je levý motor, B je pravý motor.

Například:

```
loop: forward A     ' motor a on forwards
      wait 5        ' wait 5 seconds
      backward A    ' motor a on backwards
      wait 5        ' wait 5 seconds
      halt A        ' motor A stop
      wait 5        ' wait 5 seconds
      goto loop     ' loop back to start
```

branch (použitelné pro všechny typy PICAXE)

syntaxe:
BRANCH offset,(address0,address1...addressN)
- offset je proměnná, určující na kterou adresu (0-N) se má skočit.
- adresy (address) jsou návěští, na která se větví program podle hodnoty proměnné offset.

funkce:
Tento příkaz umožňuje větvení programu podle proměnné offset. Pokud je její hodnota 0, skočí se na první návěští, pokud je hodnota 1, skočí se na druhé atd. Pokud je hodnota větší nežli odpovídá poslednímu uvedenému návěští, neprovede se žádný skok a program pokračuje na dalším řádku.

Například:

```
reset:      let b1 = 0
            low 0
            low 1
            low 2
            low 3
main: let b1 = b1 + 1
            if b1 > 3 then reset
            branch b1,(btn0,btn1, btn2, btn3)
btn0: high 0
            goto main
btn1: high 1
            goto main
btn2: high 2
            goto main
btn3: high 3
            goto main
```

button (použitelné pro všechny typy PICAXE)

syntaxe:

BUTTON pin,downstate,delay,rate,bytevariable,targetstate,address

funkce:

Příkaz ke čtení tlačítka, odstranění zákmitů a simulaci opakovaného stisku (autorepeat). Aby správně fungoval, musí být periodicky vykonáván.

Podrobněji je příkaz vysvětlen v anglickém manuálu.

calibfreq (použitelné pro PICAXE: 08M, 18A, 18X)

syntaxe:

CALIBFREQ {-} factor

- factor je konstanta nebo proměnná, nabývající hodnot -31 až 31

funkce:

Slouží k jemnému doladění interního oscilátoru. Po zapnutí napájení má konstanta hodnotu 0. Kladné hodnoty zvyšují kmitočet, záporné hodnoty snižují kmitočet oscilátoru.

Tento příkaz má smysl pouze u mikrokontrolérů s interním oscilátorem – 08M, 18A a 18X.

count (použitelné pro PICAXE: 08M, 18X, 28X, 40X)

syntaxe:

COUNT pin, period, variable

- pin je proměnná nebo konstanta, určující, na kterém vstupu se počítají impulsy.

- period je proměnná nebo konstanta, určující dobu měření (1-65535 ms při frekvenci oscilátoru 4 MHz).

- variable je proměnná, do které se zaznamená výsledek (přednostně proměnná word – s rozsahem 0-65535).

funkce:

Slouží k čítání impulsů na vstupu, počítá vzestupné hrany na určeném vstupu. Nejvyšší měřitelná frekvence vstupního signálu je 25 kHz (pokud je střída signálu 1:1) pro frekvenci oscilátoru 4MHz.

Například:

```
loop: count 1, 5000, w1          ' count pulses in 5 seconds
      debug w1                   ' display value
      goto loop                  ' else loop back to start
```

debug (použitelné pro všechny typy PICAXE)

syntaxe:

DEBUG {var}

- var je nepovinná proměnná, uvedená pouze z důvodu zpětné kompatibility, její hodnota nemá žádný vliv.

funkce:

Zobrazí hodnoty všech proměnných v ladícím okně na monitoru PC. Musí být připojen komunikační kabel. Vzhledem k množství přenášených dat tento příkaz značně zpomaluje běh programu. Rychlejší selektivní výpis ladících informací umožňuje příkaz SERTXD.

Například:

```
loop: let b1 = b1 + 1            ' increment value of b1
      readadc 2,b2               ' read an analogue value
      debug b1                   ' display values on computer screen
      pause 500                  ' wait 0.5 seconds
      goto loop                  ' loop back to start
```

data
eeprom (použitelné pro všechny typy PICAXE)

syntaxe:

DATA {location},(data,data...)

EEPROM {location},(data,data...)

- location je nepovinná konstanta (0-255), určující počáteční adresu paměti eeprom, kam se budou ukládat data. Pokud není počáteční adresa uvedena, ukládání pokračuje tam, kde předchozí příkaz skončil. Při prvním použití příkazu se začíná na adrese 0.

- data jsou konstanty (0-255), které budou uloženy v paměti eeprom.

funkce:

Příkazy DATA a EEPROM jsou synonyma, oba slouží k naplnění paměti eeprom konstantami při zavádění programu z PC. Tyto konstanty lze načíst v programu příkazem READ. Příkaz DATA či EEPROM neovlivňuje délku programu.

U PICAXE-08, 08M a 18 je EEPROM sdílena s pamětí programu. Pouze nevyužitá paměť může být zužitkována. Délku vlastního programu lze zjistit pomocí příkazu ‚Check Syntax' z menu PICAXE.

Dostupné adresy jsou pak:

PICAXE-08: 0 až (127 - délka programu)

PICAXE-08M: 0 až (255 - délka programu)

PICAXE-18: 0 až (127 - délka programu)

U následujících kontrolérů je paměť eeprom zcela oddělená, takže nemůže dojít ke konfliktu:

PICAXE-28, 28A: 0 až 63

PICAXE-28X, 40X: 0 až 127

PICAXE-18A, 18X: 0 až 255

Například:

```
            EEPROM 0,("Hello World")   ' save values in EEPROM
main:for b0 = 0 to 10                  ' start a loop
            read b0,b1                 ' read value from EEPROM
            serout 7,N2400,(b1)        ' transmit to serial LCD module
            next b0                    ' next character
```

end (použitelné pro všechny typy PICAXE)

syntaxe:

END

funkce:

Ukončí běh programu a uvede kontroler do režimu s minimální spotřebou. Obnova běhu programu je možná pouze vypnutím napájení, přivedením nízké úrovně na vstup MCLR (resetem) nebo zavedením nového programu z PC.

Příkaz END vypíná všechny časovače, po jeho provedení se ukončí činnost příkazů PWM a SERVO a kontrolér přejde do režimu s nízkým příkonem. Pokud je tento efekt nežádoucí, lze použít příkaz STOP**.**

Například:

```
loop:let b2 = 15               ' set b2 value
            pause 2000         ' wait for 2 seconds
            gosub flsh         ' call sub-procedure
            let b2 = 5         ' set b2 value
            pause 2000         ' wait for 2 seconds
            end                ' stop accidentally falling into sub
flsh:for b0 = 1 to b2          ' define loop for b2 times
            high 1             ' switch on output 1
            pause 500          ' wait 0.5 seconds
            low 1              ' switch off output 1
            pause 500          ' wait 0.5 seconds
            next b0            ' end of loop
            return             ' return from sub-procedure
```

for..next (použitelné pro všechny typy PICAXE)

syntaxe:

FOR variable = start TO end {STEP {-}increment}

..

příkazy programové smyčky

..

NEXT {variable}

- variable je proměnná, která je použita jako čítač cyklů

- start je počáteční hodnota čítače

- end je konečná hodnota čítače

- increment je nepovinná hodnota kroku čítače. Pokud není uvedena použije se hodnota +1. Pokud je uvedena záporna hodnota, předpokládá se, že Start je větší nežli End a čítá se směrem dolů.

funkce:

Slouží k opakovanému provádění kódu uvedeného mezi příkazy FOR a NEXT. Při použití proměnné byte je největší možný počet cyklů 255. Při každém provedení příkazu NEXT se hodnota čítače zvětší (nebo zmenší) o předepsaný krok a porovná se s konečnou hodnotou End. Pokud je čítač větší (nebo menší při záporném kroku) nežli End, smyčka se ukončí a program pokračuje dalším řádkem za NEXT. Příkaz FOR-NEXT může mít osm úrovní vnoření.

Například:

```
loop: for b0 = 1 to 20          ' define loop for 20 times
      high 1                    ' switch on output 1
      pause 500                 ' wait 0.5 seconds
      low 1                     ' switch off output 1
      pause 500                 ' wait 0.5 seconds
      next b0                   ' end of loop
      pause 2000                ' wait for 2 seconds
      goto loop                 ' loop back to start
```

forward (použitelné pro PICAXE: 18, 18A, 18X, 28A, 28X, 40X)

syntaxe:

FORWARD motor

- motor je A nebo B

funkce:

Spustí otáčení motoru dopředu. Tento příkaz je ekvivalentní high 4 low 5 pro motor A, nebo high 6 low 7 pro motor B.

Bude fungovat správně na desce UMU rev. C (vyrobené v roce 2007). A je levý motor, B je pravý motor.

Například:

```
forward A           ' motor a on forwards
wait 5              ' wait 5 seconds
backward A          ' motor a on backwards
wait 5              ' wait 5 seconds
halt A              ' motor A stop
wait 5              ' wait 5 seconds
goto loop           ' loop back to start
```

gosub (použitelné pro všechny typy PICAXE)

syntaxe:

GOSUB address

- address je návěští podprogramu, který příkaz GOSUB volá

funkce:

Předává řízení programu na udanou adresu, po vykonání příkazu RETURN se vrátí vykonávání programu na řádek následující po příkazu GOSUB. Příkaz GOSUB se liší od příkazu GOTO v tom, že uchovává návratovou adresu. Po každém příkazu GOSUB musí následovat vykonání příkazu RETURN, jinak by došlo k přeplnění zásobníku návratových adres. Příkazy GOSUB mohou mít čtyři úrovně vnoření.

Program kontrolérů 18X, 28X a 40X může obsahovat celkem 15 nebo 255 příkazů GOSUB, podle nastavení v menu Options. U ostatních kontrolérů je povoleno 15, případně 16 příkazů GOSUB v celém programu.

Například:

```
loop: let b2 = 15              ' set b2 value
            pause 2000         ' wait for 2 seconds
            gosub flsh         ' call sub-procedure
            let b2 = 5         ' set b2 value
            pause 2000         ' wait for 2 seconds
            gosub flsh         ' call sub-procedure
            end                ' stop accidentally falling into sub

flsh: for b0 = 1 to b2         ' define loop for b2 times
            high 1             ' switch on output 1
            pause 500          ' wait 0.5 seconds
            low 1              ' switch off output 1
            pause 500          ' wait 0.5 seconds
            next b0            ' end of loop
            return             ' return from sub-procedure
```

goto (použitelné pro všechny typy PICAXE)

syntaxe:

GOTO address

- address je návěští, na které se předá provádění programu – nepodmíněný skok na jiné místo v programu.

Například:

```
loop: high 1                   ' switch on output 1
            pause 5000         ' wait 5 seconds
            low 1              ' switch off output 1
            pause 5000         ' wait 5 seconds
            goto loop          ' loop back to start
```

halt (použitelné pro PICAXE: 18, 18A, 18X, 28A, 28X, 40X)

syntaxe:

HALT motor

- motor je A nebo B

funkce:

Zastaví otáčení motoru. Tento příkaz je ekvivalentní low 4 low 5 pro motor A, nebo low 6 low 7 pro motor B.

Bude fungovat správně na desce UMU rev. C (vyrobené v roce 2007). A je levý motor, B je pravý motor.

Například:

```
forward A           ' motor a on forwards
wait 5              ' wait 5 seconds
backward A          ' motor a on backwards
wait 5              ' wait 5 seconds
halt A              ' motor A stop
wait 5              ' wait 5 seconds
goto loop           ' loop back to start
```

high (použitelné pro všechny typy PICAXE)

syntaxe:

HIGH pin

- pin je proměnná nebo konstanta, označuje výstup, který se použije

funkce:

Nastaví vysokou výstupní úroveň. (U PICAXE-08 zároveň nastaví vývod jako výstupní.)

Například:

```
loop:       high 1              ' switch on output 1
            pause 5000          ' wait 5 seconds
            low 1               ' switch off output 1
            pause 5000          ' wait 5 seconds
            goto loop           ' loop back to start
```

high portc (použitelné pro PICAXE: 28X, 40X)

syntaxe:

HIGH PORTC pin

- pin je proměnná nebo konstanta, označuje výstup na portu C, který se použije

funkce:

Nastaví vysokou výstupní úroveň na výstupu portu C. (Pouze u kontrolérů 28X a 40X)

Například:

```
loop:       high portc 1        ' switch on output 1
            pause 5000          ' wait 5 seconds
            low portc 1         ' switch off output 1
            pause 5000          ' wait 5 seconds
            goto loop           ' loop back to start
```

i2cslave (použitelné pro PICAXE: 18X, 28X, 40X)

syntaxe:

I2CSLAVE slave, speed, address

- slave je adresa zařízení na I2C sběrnici

- speed je klíčové slovo i2cfast (400 kHz) nebo i2cslow (100 kHz) (krystal 4 MHz), určuje rychlost komunikace. Pokud je na sběrnici více zařízení, určuje se rychlost podle nejpomalejšího z nich – rychlejší periferie může pracovat s nižší rychlostí, naopak pracovat nelze.

- address je klíčové slovo i2cbyte nebo i2cword, určuje, zda je adresa 8-bitová nebo 16-bitová.

funkce:

Nastavuje parametry pro komunikaci po I2C sběrnici a konfiguruje vývody SCL a SDA jako vstupní. Komunikaci obstarávají příkazy READI2C, WRITEI2C.

Při používání sběrnice I2C nesmíme zapomenout na zdvihací (pull-up) rezistory na SCL a SDA. Obvyklá hodnota těchto rezistorů je 4k7, jsou použity pouze v jednom místě, nejčastěji na desce PICAXE.

Nastavení pro vybrané typy periferií:

Periferie	*Typ*	*Slave*	*Speed*	*Address*
24LC01B	EE 128bit	%1010xxxx	i2cfast	i2cbyte
24LC02B	EE 256bit	%1010xxxx	i2cfast	i2cbyte
24LC04B	EE 512bit	%1010xxbx	i2cfast	i2cbyte
24LC08B	EE 1kbit	%1010xbbx	i2cfast	i2cbyte
24LC16B	EE kbit	%1010bbbx	i2cfast	i2cbyte
24LC64	EE 8kbit	%1010dddx	i2cfast	i2cword
24LC256	EE 64kbit	%1010dddx	i2cfast	i2cword
DS1307	RTC	%1101000x	i2cslow	i2cbyte
MAX6953	5x7 LED	%101ddddx	i2cfast	i2cbyte
AD5245	Digit. pot	%010110dx	i2cfast	i2cbyte
SRF08	Ultrazvuk	$E0*	i2cfast	i2cbyte
AXE033	I2C LCD	$C6	i2cslow	i2cbyte
CMPS03	Kompas	$C0	i2cfast	i2cbyte
SPE030	Řečová syntéza	$C4	i2cfast	i2cbyte

x nemá vliv – libovolná hodnota
b výběr bloku – tyto bity jsou součástí adresy buňky paměti
d výběr zařízení – tyto bity musí souhlasit s adresovými vstupy na periferii. Pokud je připojeno pouze jedno zařízení svého druhu, je obvyklá praxe adresovací vstupy uzemnit a tyto bity jsou potom nulové.

* U SRF08 lze změnit adresu na libovolnou z rozsahu $E0 až $FE (16 možností). Podrobněji v dokumentaci senzoru.

if .. then
if .. and .. then
if .. or .. then (použitelné pro všechny typy PICAXE)

syntaxe:
IF variable ?? value {AND/OR variable ?? value ...} THEN address

- **variable** je proměnná. Bude porovnána s value
- **value** může být proměnná nebo konstanta
- **address** je návěští, na které se předá řízení programu. Pokud je podmínka splněna
- **??** může být jeden z následujících operátorů:
  - **=** rovná se
  - **is** rovná se (alternativní zápis)
  - **<>** nerovná se
  - **!=** nerovná se (alternativní zápis)
  - **>** větší než
  - **>=** větší nebo rovno
  - **<** menší než
  - **<=** menší nebo rovno

funkce:
Příkaz porovnává dvě proměnné nebo proměnnou s konstantou a skočí na určené místo, pokud je podmínka splněna. Pokud není podmínka splněna, pokračuje se na dalším řádku programu.

Například:

```
loop:if pin0 = 1 then flsh          ' jump to flsh if pin0 is high
           goto loop                ' else loop back to start

flsh:      high 1                   ' switch on output 1
           pause 5000               ' wait 5 seconds
           low 1                    ' switch off output 1
           goto loop                ' loop back to start
```

Vícenásobná podmínka může být vytvořena pomocí AND a OR:

2 násobné AND – musí být splněny všechny podmínky současně, aby se skočilo na label

```
if pin1 = 1 and pin2 = 1 then label
```

3 násobné AND – musí být splněny všechny podmínky současně, aby se skočilo na label

```
if pin0 = 1 and pin1 = 1 and b0 > 3 then label
```

2 násobné OR – musí být splněna alespoň jedna podmínka (nebo obě dvě), aby se skočilo na label

```
if pin1 = 1 or b3 = b2 then label
```

testování analogové hodnoty v daném intervalu, b1 musí být mezi 100 a 200, aby se skočilo na label

```
readadc 1,b1
if b1 >= 100 and b1 <= 200 then label
```

infrain (použitelné pro PICAXE: 18A, 18X, 28A, 28X, 40X)

syntaxe:

INFRAIN

funkce:

Čeká na příjem znaku z dálkového ovládání. Přijatý kód je v proměnné infrain. Vyžaduje připojení přijímače infračerveného signálu na vstup input0, používá protokol Sony.

Podrobnosti naleznete v originální anglické příručce.

infrain2 (použitelné pro PICAXE: 08M)

syntaxe:

INFRAIN2

funkce:

Obdoba předchozího příkazu pro PICAXE-08M. Vyžaduje připojení přijímače infračerveného signálu na vstup input3, používá protokol Sony.

Podrobnosti naleznete v originální anglické příručce.

infraout (použitelné pro PICAXE: 08M)

syntaxe:

INFRAOUT device,data

funkce:

Vyšle data protokolem Sony infračerveného dálkového ovládání, pouze na PICAXE-08M. Předpokládá připojení vysílací infračervené LED s omezovacím odporem na výstup 0.

Podrobnosti naleznete v originální anglické příručce.

input (použitelné pro PICAXE: 08, 08M)

syntaxe:

INPUT pin

- pin je proměnná nebo konstanta, označuje použitý vývod

funkce:

Nastaví vývod jako vstupní.

Po zapnutí napájení jsou všechny konfigurovatelné vývody nastaveny jako vstupní. Kromě příkazů k přímému nastavení (LET DIRS, INPUT, OUTPUT, REVERSE) se příslušné vývody nastavují také příkazy HIGH, LOW, TOGGLE, PULSOUT jako výstupní.

U kontrolérů 08 a 08M je pin0 vždy výstupní a pin3 vždy vstupní.

Například:

```
loop: input 1                  ' make pin input
      reverse 1                ' make pin output
      reverse 1                ' make pin input
      output 1                 ' make pin output
```

keyin (použitelné pro PICAXE: 18A, 18X, 28A, 28X, 40X)

syntaxe:
KEYIN

funkce:
Čeká na příjem znaku z počítačové klávesnice připojené přímo k mikrokontroléru. Přijatý kód je v proměnné keyin. Vyžaduje připojení klávesnice na vstupy input6 a input7 a zdvihací rezistory 4k7.

Například:

```
loop: keyin                                 'wait for new signal
            if keyvalue = $45 then swon1    'switch on 1
            if keyvalue = $16 then swon2    'switch on 2
            if keyvalue = $25 then swoff1   'switch off 1
            if keyvalue = $2E then swoff2   'switch off 2
            goto loop
swon1:      high 1
            goto loop
swon2:      high 2
            goto loop
swoff1:     low 1
            goto loop
swoff2:     low 2
            goto loop
```

keyled (použitelné pro PICAXE: 18A, 18X, 28A, 28X, 40X)

syntaxe:
KEYLED mask
- mask je proměnná nebo konstanta, určuje stav LED na klávesnici.

funkce:
Rozsvěcí nebo zhasíná LED na klávesnici PC. Vyžaduje připojení klávesnice na vstupy input6 a input7.

Například:

```
loop: keyled %10000111              ' all LEDs on
            pause 500               ' pause 0,5s
            keyled %10000000        ' all LEDs off
            pause 500               ' pause 0,5s
            goto loop               ' loop
```

let (použitelné pro všechny typy PICAXE)

syntaxe:
{LET} variable = {-} value ?? value...
- variable je proměnná, které bude přiřazen výsledek početní operace
- value jsou proměnné nebo konstanty spojené operátory
Klíčové slovo LET je nepovinné

funkce:
Provádí početní operace v celočíselné 16-bitové aritmetice (hodnoty 0 až 65535). Výrazy jsou vyhodnocovány zleva doprava bez upatnění přednosti operátorů. Všechna čísla jsou chápána jako kladná. Výsledek může být také 8-bitový nebo 1-bitový, v tom případě jsou vyšší bity ořízntuty.

Například:

```
loop: let b0 = b0 + 1                ' increment b0
      sound 7,(b0,50)                ' make a sound
      if b0 > 50 then rest           ' after 50 reset
      goto loop                      ' loop back to start

rest: let b0 = b0 max 10             ' limit b0 back to 10
                                     ' as 10 is the maximum value
      goto loop                      ' loop back to start
```

let dirs = (použitelné pro PICAXE: 08, 08M)
let dirsc = (použitelné pro PICAXE: 28X, 40X)

syntaxe:

{LET} dirs = value

{LET} dirsc = value

- value jsou proměnné nebo konstanty, jejichž hodnota je uložena do proměnné dirs (dirsc).

funkce:

Konfiguruje vývody kontroléru jako vstupní nebo výstupní (let dirs, na PICAXE-08/08M)

Konfiguruje vývody kontroléru na portu C jako vstupní nebo výstupní (let dirsc, na PICAXE-28X/40X). Hodnota 1 příslušného bitu značí vstup, hodnota 0 značí výstup.

Po zapnutí napájení jsou všechny konfigurovatelné vývody nastaveny jako vstupní. Kromě příkazů k přímému nastavení (LET DIRS, INPUT, OUTPUT, REVERSE) se příslušné vývody nastavují také příkazy HIGH, LOW, TOGGLE, PULSOUT jako výstupní.

U kontrolérů 08 a 08M je pin0 vždy výstupní a pin3 vždy vstupní.

Například:

```
let dirs = %00000011             ' switch pins 0 and 1 to outputs
let pins = %00000011             ' switch on outputs 0 and 1
```

let pins = (použitelné pro všechny typy PICAXE)
let pinsc = (použitelné pro PICAXE: 28X, 40X)

syntaxe:

{LET} pins = value

{LET} pinsc = value

- value je proměnná nebo konstanta, jejíž hodnota je uložena do proměnné pins (pinsc).

funkce:

Tento příkaz nastavuje současně všechny výstupy kontroléru individuálně na vysokou nebo nízkou úroveň, u kontrolérů PICAXE-28X/40X lze takto nastavit též všechny vývody portu C. Klíčové slovo LET je nepovinné.

K individuálnímu nastavení jednotlivých výstupů lze použít příkazy high a low. Příkaz LET PINS umožňuje hromadné nastavení všech osmi výstupů současně. S výhodou se konstanty uvádí v binárním tvaru, potom výstupu 7 odpovídá číslice zcela vlevo a výstupu 0 číslice na posledním místě. Uvedením hodnoty 0 se příslušný výstup nastaví na nízkou úroveň, hodnotou 1 se nastaví na vysokou úroveň.

Pozor na záměnu vstupů a výstupů, kromě PICAXE-08/08M jsou na rozdílných vývodech. Proměnná pins, pokud je čtena (vpravo od rovnítka), odpovídá vstupům. Při zápisu do proměnné pins odpovídá výstupům. Z tohoto důvodu není možné například uvést výstupy 0 až 3 do nízké úrovně příkazem LET PINS=PINS & %11110000.

Na kontrolérech s obousměrnými vývody (08/08M) se tento příkaz aplikuje pouze na vývody, které jsou nastaveny jako výstupní. Příslušné vývody lze aktivovat jako výstupy příkazem LET DIRS.

Například:

```
let pins = %11000011             ' switch outputs 7,6,0,1 on
pause 1000                       ' wait 1 second
let pins = %00000000             ' switch all outputs off
```

lookdown (použitelné pro všechny typy PICAXE)

syntaxe:

LOOKDOWN target,(value0,value1...valueN),variable

- target je proměnná nebo konstanta, která se porovnává s řadou hodnot v závorce.

- value0... jsou proměnné nebo konstanty

- variable obsahuje výsledek porovnávání.

funkce:

Porovnává target se seznamem hodnot v závorce, pokud najde stejnou hodnotu, uloží do proměnné za závorkou pořadové číslo shodné hodnoty. Číslování začíná nulou. Pokud se shoda nenajde, proměnná za závorkou zůstane beze změny.

Například:

```
lookdown b1,("abcde"),b2
```

lookup (použitelné pro všechny typy PICAXE)

syntaxe:

LOOKUP offset,(data0,data1...dataN),variable

- offset je proměnná nebo konstanta, určuje, která položka z data0 až dataN se uloží do proměnné variable.

- data jsou proměnné nebo konstanty

- variable předává výslednou hodnotu, nebo zůstává beze změny

funkce:

Vybírá z pole hodnot podle zadaného ofsetu (indexu). Pokud je ofset mimo rozsah uvedených hodnot, výstupní proměnná zůstává beze změny.

Například:

```
loop:let b0 = b0 + 1                ' increment b0
     lookup b0,("abcd"),b1          ' put ascii character into b1
     if b0 < 4 then loop            ' loop
     end
```

low (použitelné pro všechny typy PICAXE)

syntaxe:

LOW pin

- pin je proměnná nebo konstanta, označuje výstup, který se použije

funkce:

Nastaví nízkou výstupní úroveň. (U PICAXE-08 zároveň nastaví vývod jako výstupní.)

Například:

```
loop:high 1                 ' switch on output 1
     pause 5000             ' wait 5 seconds
     low 1                  ' switch off output 1
     pause 5000             ' wait 5 seconds
     goto loop              ' loop back to start
```

low portc (pro PICAXE: 28X, 40X)

syntaxe:

LOW PORTC pin

- pin je proměnná nebo konstanta, označuje výstup na portu C, který se použije

funkce:

Nastaví nízkou výstupní úroveň na výstupu portu C. (Pouze u kontrolérů 28X a 40X)

Například:

```
loop:       high portc 1        ' switch on output 1
            pause 5000          ' wait 5 seconds
            low portc 1         ' switch off output 1
            pause 5000          ' wait 5 seconds
            goto loop           ' loop back to start
```

nap (použitelné pro všechny typy PICAXE)

syntaxe:

NAP period

- period je proměnná nebo konstanta, určující dobu, na kterou přejde kontroler do režimu s nízkou spotřebou. Rozsah 0 až 7.

Doba zpoždění:

0	18 ms
1	32 ms
2	72 ms
3	144 ms
4	288 ms
5	576 ms
6	1,152 s
7	2,304 s

funkce:

Uvede kontroler do spánku na dobu 2^period*18 ms. Tento příkaz využívá watchdog timer s omezenou přesností časování. Delší prodlevy lze dosáhnout příkazem Sleep.

Například:

```
loop:       high 1          ' switch on output 1
            nap 4           ' nap for 288ms
            low 1           ' switch off output 1
            nap 7           ' nap for 2,3 s
            goto loop       ' loop back to start
```

output (použitelné pro PICAXE: 08, 08M)

syntaxe:

OUTPUT pin

- pin je proměnná nebo konstanta, označuje použitý vývod

funkce:

Nastaví vývod jako výstupní. Funguje pouze u kontrolérů 08 a 08M.

Například:

```
loop: input 1             ' make pin input
            reverse 1     ' make pin output
            reverse 1     ' make pin input
            output 1      ' make pin output
```

Po zapnutí napájení jsou všechny konfigurovatelné vývody nastaveny jako vstupní. Kromě příkazů k přímému nastavení (LET DIRS, INPUT, OUTPUT, REVERSE) se příslušné vývody nastavují také příkazy HIGH, LOW, TOGGLE, PULSOUT jako výstupní.

U kontrolérů 08 a 08M je pin0 vždy výstupní a pin3 vždy vstupní.

pause (použitelné pro všechny typy PICAXE)

syntaxe:

PAUSE milliseconds

- milliseconds je proměnná nebo konstanta v rozsahu 0 až 65535, určuje dobu v jednotkách milisekund, po kterou bude tento příkaz trvat. Toto platí pouze při nastavené hodinové frekvenci 4 MHz. Při nastavení hodinové frekvence 8 MHz se čas zkracuje na 0,5 ms a na 0,25 ms při nastavení hodinové frekvence na 16 MHz.

funkce:

Zastaví běh programu na určenou dobu. Přesnost je odvozena od hodinového kmitočtu kontroleru.

Například:

```
loop:       high 1        ' switch on output 1
            pause 5000    ' wait 5 seconds
            low 1         ' switch off output 1
            pause 5000    ' wait 5 seconds
            goto loop     ' loop back to start
```

peek (použitelné pro PICAXE: 08M, 18, 18A, 18X, 28A, 28X, 40X)

syntaxe:

PEEK location,variable

- location je proměnná nebo konstanta, určující adresu registru. Platné hodnoty jsou 0 až 255.

- variable je 8bitová proměnná, ve které je navrácen obsah registru na udané adrese.

funkce:

Čte data z registrů mikrokontroleru. Umožňuje obnovit data uschovaná příkazem POKE.

Například:

```
peek 80,b1          ' put value of register 80 into variable b1
```

play (použitelné pro PICAXE: 08M)

syntaxe:

PLAY tune,LED

- tune je proměnná nebo konstanta (0 - 3) určující, která skladba se zahraje:
0 - Happy Birthday
1 - Jingle Bells
2 - Silent Night
3 - Rudolf the Red Nosed Reindeer

- LED je proměnná nebo konstanta (0 -3) určující způsob blikání připojených LED během hraní:
0 - bez blikání
1 - výstup 0 se zapíná a vypíná
2 - výstup 4 se zapíná a vypíná
3 - výstupy 0 a 4 se střídavě zapínají a vypínají

funkce:

Přehrává skladbu, pouze na PICAXE-08M. Výstupní signál se objeví na výstupu 2.

Například:

```
play 3,1      ' rudolf red nosed reindeer with output 0 flashing
```

poke (použitelné pro PICAXE: 08M, 18, 18A, 18X, 28A, 28X, 40X)

syntaxe:

POKE location,data

- location je proměnná nebo konstanta určující adresu registru. Platné hodnoty jsou 0 až 255.

- data je proměnná nebo konstanta obsahující data, která budou zapsána na uvednou adresu.

funkce:

Zapisuje data do registrů kontroleru. Umožňuje uložit proměnné b0 až b13 do paměti a dále využít hardware prostřednictvím SFR (podle dokumentace v katalogovém listu příslušného kontroleru).

Například:

```
poke 80,b1    ' save value of b1 in register 80
```

pulsin (použitelné pro všechny typy PICAXE)

syntaxe:

PULSIN pin,state,variable

- pin je proměnná nebo konstanta (0-7), určující, který vývod bude použit.

- state je proměnná nebo konstanta (0 nebo 1), určující, která hrana se musí první objevit před začátkem měření.

- variable obsahuje výsledek měření (1 - 65535) v jednotkách 10 µs.

funkce:

Měří délku vstupního pulsu v jednotkách 10 µs. Jestliže se puls neobjeví do 0,65536 s, příkaz končí a výsledek je 0. Pokud je proměnná state = 1, měří se délka pulsu ve vysoké úrovni, měření začíná vzestupná hrana a končí sestupná hrana. Pokud je proměnná state = 0, měří se délka pulsu v nízké úrovni, měření začíná sestupná hrana a končí vzestupná hrana. Výstupní proměnná se obvykle používá 16bitová.

Vliv frekvence oscilátoru:

4 MHz – doba v jednotkách 10µs a ukončení příkazu po 0,65536 s

8 MHz – doba v jednotkách 5µs a ukončení příkazu po 0,32768 s

16 MHz – doba v jednotkách 5µs a ukončení příkazu po 0,16384 s

Například:

```
pulsin 3,1,w1       ' record the length of a pulse on pin 3 into b1
```

pulsout (použitelné pro všechny typy PICAXE)

syntaxe:

PULSOUT pin,time

- pin je proměnná nebo konstanta (0-7) určující, který vývod bude použit.

- time je proměnná nebo konstanta určující dobu trvání pulsu (0-65535) v jednotkách 10 µs.

funkce:

Vyšle impuls zadané délky. Polarita impulsu je určená počátečním stavem pinu, během pulsu se stav invertuje a po skončení se vrátí na původní úroveň

Vliv frekvence oscilátoru:

4 MHz doba (time) v jednotkách 10 µs

8 MHz doba (time) v jednotkách 5 µs

16 MHz doba (time) v jednotkách 2,5 µs

Například:

```
loop:pulsout 4,150          ' send a 1,50 ms pulse out of pin 4
     pause 20               ' pause 20 ms
     goto loop              ' loop back to start
```

pwm (použitelné pro PICAXE: 08, 08M)

syntaxe:

PWM pin,duty,cycles

- **pin** je proměnná nebo konstanta (0-7), určující, který vývod bude použit.
- **duty** je proměnná nebo konstanta (0-255), určující činitel plnění PWM (dobu, kterou výstup setrvá ve stavu 1)
- **cycles** je proměnná nebo konstanta (0-255), určující počet cyklů PWM, které na určeném vývodu proběhnout. Každý cyklus trvá asi 5 ms.

funkce:

Tento příkaz se používá zřídka, vhodnější je použít PWMOUT. Příkaz PWM ukončí svoji činnost po zadaném počtu cyklů, neprobíhá na pozadí jako příkaz PWMOUT. Ve spojení s RC filtrem může napodobit analogový výstup na kontroléru PICAXE-08. Příkaz musí být volán opakovaně.

Například:

```
loop:pwm 4,150,20              ' send 20 pwm bursts out of pin 4
           pause 20            ' pause 20 ms
           goto loop           ' loop back to start
```

pwmout (použitelné pro PICAXE: 08M, 18X, 28X, 40X)

syntaxe:

PWMOUT pin,period,duty cycles

- **pin** je proměnná nebo konstanta určující, který vývod bude použit.
  (pouze vývod 3 na 18X, pouze vývod 2 na 08M, vývod 1 nebo 2 na 28X a 40X)
- **period** je proměnná nebo konstanta (0-255) určující periodu pulsně šířkové modulace (PWM)
- **duty** je proměnná nebo konstanta (0-1023) určující činitel plnění PWM (dobu, kterou výstup setrvá ve stavu 1)

Generuje pulsně modulovaný výstupní signál na zvoleném vývodu s využitím interního hardware kontroleru. Tento signál zůstává aktivní i po ukončení příkazu PWMOUT. Pokud je třeba signál zrušit, použije se příkaz PWMOUT s periodou 0.

Perioda PWM	= (period + 1) * 4 * perioda oscilátoru
Činitel plnění PWM	= (duty) * perioda oscilátoru
	Perioda oscilátoru je 250 ns při 4MHz, 125 ns při 8MHz a 62,5 ns při 16MHz
Frekvence PWM	= 1 / perioda PWM

Tento příkaz používá hardware kontroleru, z čehož plynou následující omezení.

1) Příkaz funguje pouze na některých vývodech (28X/40X -1 a 2, 18X - 3, 08M -2).

2) Činitel plnění je 10 bitová hodnota, která by neměla být větší nežli čtyřnásobek periody PWM (je-li větší, PWM výstup zůstává stále na vysoké úrovni).

3) Na 28X/40X se používá stejný časovač pro oba vývody, takže perioda je vždy stejná.

4) Příkaz Servo nemůže být aktivní zároveň s příkazem PWMout, neboť využívá tentýž časovač.

5) PWM výstup se deaktivuje během příkazů nap, sleep, a po provedení příkazu end.

Například:

```
loop:pwmout 2,150,100          ' set pwm
           pause 1000          ' pause 1 s
           goto loop           ' loop back to start
```

random (použitelné pro všechny typy PICAXE)

syntaxe:

RANDOM wordvariable

- wordvariable slouží zároveň jako výsledek i jako pracovní proměnná (násada) pro příští použití příkazu. Musí být použita proměnná typu word a její hodnota se nesmí do dalšího použití příkazu změnit.
Vytváří sekvenci pseudonáhodných čísel mezi 0 a 65535.

Například:

```
loop: random w0                         ' put random value into w0
            if pin1 =1 then doit
            goto loop
doit: let pins = b1                     ' put random byte value on output pins
            pause 100                   ' wait 0.1s
            goto loop                   ' loop back to start
```

readadc (použitelné pro všechny typy PICAXE)

syntaxe:

READADC channel,variable

- channel je proměnná nebo konstanta, určující vstup (0-7)

- variable obsahuje výsledek A/D převodu

funkce:

Příkaz čte napětí na analogovém vstupu a převádí ho na osmibitové číslo. Pouze některé vstupy mohou sloužit jako analogové. Na některých kontrolerech jsou analogové a digitální vstupy sdílené, mohou plnit obě funkce.

Například:

```
loop: readadc 1,b1                      ' read value into b1
            if b1 > 50 then flsh        ' jump to flsh if b1 > 50
            goto loop                   ' else loop back to start
flsh: high 1                            ' switch on output 1
            pause 5000                  ' wait 5 seconds
            low 1                       ' switch off output 1
            goto loop                   ' loop back to start
```

readadc10 (použitelné pro PICAXE: 08M, 18X, 28X, 40X)

syntaxe:

READADC10 channel,wordvariable

- channel je proměnná nebo konstanta určující vstup (0-7)

- wordvariable obsahuje výsledek A/D převodu

funkce:

Příkaz čte napětí na analogovém vstupu a převádí ho na 10bitové číslo, proto se výsledek musí ukládat do 16bitové proměnné. Pouze některé vstupy mohou sloužit jako analogové. Na některých kontrolerech jsou analogové a digitální vstupy sdílené, mohou plnit obě funkce. Při použití příkazu DEBUG může komunikace s PC narušit výsledek A/D převodu. V tom případě se doporučuje doplnit obvod o Schottkyho diodu, která tento efekt potlačí.

Například:

```
loop: readadc 1,w1                      ' read value into b1
            debug w1                    ' transmit to computer
            pause 200                   ' short delay
            goto loop                   ' loop back to start
```

readi2c (použitelné pro PICAXE: 18X, 28X, 40X)

syntaxe:

READI2C location,(variable,...)

- location je proměnná nebo konstanta, určující adresu z které se bude číst.

- variable proměnné, které po provedení příkazu obsahují data přečtená z uvedené adresy.

funkce:

Čte data z I2C sběrnice a ukládá je do proměnné (proměnných).

Adresa zařízení na I2C sběrnici je určená příkazem I2CSLAVE, **location** udává adresu v rámci tohoto zařízení, například adresu dat v (externí) EEPROM nebo RTC obvodu apod.

Například:

```
            ' Example of how to use DS1307 Time Clock
            ' Note the data is sent/received in BCD format.
            ' set DS1307 slave address:
            ' i2cslave %11010000, i2cslow, i2cbyte
            ' read time and date and debug display
main: readi2c 0,(b0,b1,b2,b3,b4,b5,b6,b7)
            debug b1
            pause 2000
            goto main
```

read (použitelné pro všechny typy PICAXE)

syntaxe:

READ location,variable

- location je proměnná nebo konstanta, určující 8bitovou adresu v interní EEPROM (0-255).

- variable obsahuje přečtená data

funkce:

Příkaz READ načítá data z EEPROM. Obsah této paměti je zachován i po vypnutí napájení. Tato data jsou zapisována při každém zavedení nového programu do mikrokontroléru, podle definice v příkazu DATA/EEPROM. Za běhu programu mohou být tato data přepisována pomocí příkazu WRITE. Příkaz READ pracuje pouze s osmibitovými daty. Pokud je třeba načíst proměnnou word, dosáhne se toho použitím dvou příkazů READ s oběma osmibitovými proměnnými: (například w0 se načte tak, že se načte b0 a b1). Velikost, umístění a případná omezení použitelnosti eeprom u jednotlivých typů kontrolérů jsou uvedeny u příkazu DATA.

Například:

```
loop: for b0 = 0 to 63              ' start a loop
            read b0,b1              ' read value into b1
            serout 7,T2400,(b1)     ' transmit value to serial LCD
            next b0                 ' next loop
```

readtemp (použitelné pro PICAXE: 08M, 18A, 18X, 28A, 28X, 40X)

syntaxe:

READTEMP pin,variable

- pin je proměnná nebo konstanta, určující, který vývod bude použit.

- variable obsahuje přečtená data (byte).

funkce:

Příkaz přečte teplotu z digitálního čidla DS18B20 a uloží ji do proměnné. Převod může trvat až 750 ms. Teplota se předává v celých stupních celsia. Senzor pracuje v rozmezí teplot -55 až +125°C. Bit 7 představuje znaménko, je 0 pro teploty nad nulou a 1 pro teploty pod nulou, záporné teploty se vrací jako 128 + teplota bez znaménka.

Příkaz readtemp nefunguje se staršími čidly DS1820 nebo DS18S20, která mají odlišný formát dat.

Tento příkaz funguje pouze při frekvenci 4 MHz.

Například:

```
loop: readtemp 1,b1                    ' read value into b1
      if b1 > 127 then neg             ' test for negative
      serout 7,N2400,(#b1)             ' transmit value to serial LCD
      goto loop
neg:  let b1 = b1 - 128                ' adjust neg value
      serout 7,N2400,("-")             ' transmit negative symbol
      serout 7,N2400,(#b1)             ' transmit value to serial LCD
      goto loop
```

readtemp12 (použitelné pro PICAXE: 08M, 18X, 28X, 40X)

syntaxe:

READTEMP12 pin,wordvariable

- pin je proměnná nebo konstanta, určující, který vývod bude použit.

- wordvariable obsahuje přečtená data (12 bitů).

funkce:

Příkaz přečte teplotu v surovém 12 bitovém formátu z digitálního čidla DS18B20 a uloží ji do proměnné. Převod může trvat až 750ms. Podrobnosti o formátu dat lze nalézt v dokumentaci k teplotnímu čidlu.

Příkaz readtemp12 nefunguje se staršími čidly DS1820 nebo DS18S20, která mají odlišný formát dat.

Tento příkaz funguje pouze při frekvenci 4 MHz.

Například:

```
loop: readtemp12 1,w1                  ' read value into b1
      debug w1                         ' transmit to computer screen
      goto loop
```

readowsn (použitelné pro PICAXE: 08M, 18A, 18X, 28A, 28X, 40X)

READOWSN pin

syntaxe:

- pin je proměnná nebo konstanta (0-7), určující, který vývod bude použit.

funkce:

Přečte sériové číslo z obvodu připojeného na jednodrátovou sběrnici firmy Dallas. Může číst například z teplotního senzoru DS18B20, obvodu reálného času DS2415 nebo identifikačního obvodu DS1990A (iButton). U DS1990A je sériové číslo také vypálené laserem na pouzdru obvodu.

Příkaz ukládá přečtená data následovně - family code do proměnné b6, sériové číslo do proměnných b7 až b12 a kontrolní součet do b13.

Například:

```
main: resetowclk 2              ' reset the clock on pin2
loop: readowclk 2               ' read clock on input2
      debug b1                  ' display the elapsed time
      pause 10000               ' wait 10 seconds
      goto loop
```

return (použitelné pro všechny typy PICAXE)

syntaxe:

RETURN

funkce:

Návrat z podprogramu. Příkaz return smí být použit pouze po předchozím příkazu gosub. Příkaz navrací běh programu do místa, odkud byl podprogram vyvolán. Pokud by byl příkaz return použit bez předchozího gosub, program havaruje.

Například:

```
loop:
     let b2 = 15          ' set b2 value
     pause 2000           ' wait for 2 seconds
     gosub flsh           ' call sub-procedure
     let b2 = 5           ' set b2 value
     pause 2000           ' wait for 2 seconds
     gosub flsh           ' call sub-procedure
     end                  ' stop accidentally falling into sub
flsh:
     for b0 = 1 to b2     ' define loop for b2 times
     high 1               ' switch on output 1
     pause 500            ' wait 0.5 seconds
     low 1                ' switch off output 1
     pause 500            ' wait 0.5 seconds
     next b0              ' end of loop
     return               ' return from sub-procedure
```

reverse (použitelné pro PICAXE: 08, 08M)

syntaxe:

REVERSE pin

- pin je proměnná nebo konstanta, určující, který vývod bude použit.

funkce:

Změní směr signálu na vývodu – původně vstup nastaví jako výstup a původně výstup nastaví jako vstup. Funguje pouze u kontrolérů 08 a 08M.

Po zapnutí napájení jsou všechny konfigurovatelné vývody nastaveny jako vstupní. Kromě příkazů k přímému nastavení (LET DIRS, INPUT, OUTPUT, REVERSE) se příslušné vývody nastavují také příkazy HIGH, LOW, TOGGLE, PULSOUT jako výstupní.

U kontrolérů 08 a 08M je pin0 vždy výstupní a pin3 vždy vstupní.

Například:

```
loop: input 1             ' make pin input
      reverse 1           ' make pin output
      reverse 1           ' make pin input
      output 1            ' make pin output
```

serin (použitelné pro všechny typy PICAXE)

syntaxe:

SERIN pin,baudmode,(qualifier,qualifier...)

SERIN pin,baudmode,(qualifier,qualifier...),{#}variable,{#}variable...

SERIN pin,baudmode,{#}variable,{#}variable...

- pin je proměnná nebo konstanta (0-7) určující, který vývod bude použit.

- baudmode je proměnná nebo konstanta (0-7) která určuje přenosovou rychlost a polaritu signálu. Všechny přenosové rychlosti se vztahují k hodinovému kmitočtu 4 MHz:

T2400 normální polarita (True, klidová úroveň vysoká)
T1200 normální polarita
T600 normální polarita
T300/T4800 normální polarita
N2400 obrácená polarita (Negated, klidová úroveň nízká)
N1200 obrácená polarita
N600 obrácená polarita
N300/N4800 obrácená polarita

- qualifier je nepovinná proměnná nebo konstanta, která musí být přijata v určeném pořadí nežli se začnou zpracovávat data a ukládat do proměnné (proměnných).

- variable obdrží přijatý znak (0-255). Nepovinné znak # signalizuje, že se mají následující znaky interpretovat jako číslo v desítkové soustavě a uložit do proměnné tuto hodnotu, nikoliv kód znaku.

funkce:

Tento příkaz přijímá sériová data pouze ve formátu 8N1 (8 datových bitů, bez parity, 1 stop bit).

Příkaz serin přijímá sériová data na určeném vstupu. Nelze ho použít k příjmu na vstupu Sin, určeném pro zavedení programu.

Pin určuje na kterém vstupu se budou sériová data přijímat.

Baudmode určuje přenosovou rychlost a polaritu signálu. Pokud je signál úrovně RS232 přiveden pouze přes omezovací odpor, používá se polarita N (obrácená). Pokud je signál zpracováván obvodem typu MAX232, používá se polarita T (normální).

Přenosová rychlost 4800 je dostupná pouze na kontrolérech řady -X. Při této přenosové rychlosti se může stát, že kontrolér nebude schopen přijímat komplikované protokoly – zpracování dat bude příliš pomalé. Doporučuje se maximální přenosová rychlost 2400 při hodinovém kmitočtu 4MHz.

Qualifier značí specifický znak nebo sequenci znaků (řetězec), který musí být přijat nežli se začnou zpracovávat další znaky a plnit proměnné. Například příkaz:

```
serin 1,N2400,("ABC"),b1
```

čeká nejprve na řetězec "ABC" a když je přijat, uloží další znak do proměnné b1.

Příkaz bez uvedení qualifier

```
serin 1,N2400,b1
```

uloží do proměnné b1 první přijatý znak.

Během příkazu serin jsou veškeré další aktivity kontroleru zastaveny, dokud není přijat nějaký znak. Tento příkaz není také přerušen pomocí setint.

Následující příklad čeká, dokud není přijat řetězec "go".

```
serin 1,N2400,("go")
```

Při použití příkazu serin může být nutné resetovat kontrolér pomocí vstupu reset (MCLR), aby započalo zavedení nového programu.

Například:

```
loop:       for b0 = 0 to 63    ' start a loop
            serin 6,N2400,b1    ' receive serial value
            write b0,b1         ' write value into b1
            next b0             ' next loop
```

serout (použitelné pro všechny typy PICAXE)

syntaxe:

SEROUT pin,baudmode,({#}data,{#}data...)

- pin je proměnná nebo konstanta (0-7) určující, který vývod bude použit.

- baudmode je proměnná nebo konstanta (0-7) která určuje přenosovou rychlost a polaritu signálu.

T2400 normální polarita ([T]rue, klidová úroveň vysoká)
T1200 normální polarita
T600 normální polarita
T300 / T4800 normální polarita
N2400 obrácená polarita ([N]egated, klidová úroveň nízká)
N1200 obrácená polarita
N600 obrácená polarita
N300 / N4800 obrácená polarita

Přenosová rychlost 4800 je dostupná pouze na kontrolérech -X.

- data jsou proměnné nebo konstanty (0-255), jejichž hodnoty budou vyslány na určený výstup.

Nepovinný znak # signalizuje, že se následující hodnota má odeslat jako číslo v desítkové soustavě, nikoliv jako jeden znak. Textový řetězec může být uveden v uvozovkách („Hello“).

Vysílá sériová data ve formátu 8N1 (8 datových bitů, bez parity, 1 stop bit).

funkce:

Příkaz serout vysílá data asynchronním sériovým přenosem na určeném výstupu mikrokontroléru. Nelze ho použít s vývodem Sout, určeném ke komunikaci při zavádění nového programu. Na tomto vývodu pracuje příkaz sertxd.

Pin určuje na kterém vstupu se budou sériová data vysílat.

Baudmode určuje přenosovou rychlost a polaritu signálu. Pokud je signál úrovně RS232 přiveden pouze přes omezovací odpor, používá se polarita N (obrácená). Pokud je signál zpracováván obvodem typu MAX232, používá se polarita T (normální).

Přenosová rychlost 4800 je dostupná pouze na kontrolérech řady -X.

Symbol # značí konverzi na dekadický retězec. Například když b1 obsahuje hodnotu 126, potom #b1 způsobí vyslání znaků „1“ „2“ a „6“ namísto znaku s kódem 126.

Například:

```
loop:for b0 = 0 to 63                ' start a loop
            read b0,b1               ' read value into b1
            serout 7,N2400,(b1)      ' transmit value to serial LCD
            next b0                  ' next loop
```

sertxd (použitelné pro PICAXE: 08M, 18X, 28X, 40X)

syntaxe:

SERTXD ({#}data,{#}data...)

- data jsou proměnné nebo konstanty (0-255), jejichž hodnoty budou vyslány na výstup Sout.

funkce:

Asynchronní sériový výstup na vývodu Sout určeném ke komunikaci při zavádění nového programu. Přenosová rychlost je 4800Bd, 8N1. Tato data lze sledovat na PC v okně terminálu, které se otevře příkazem Terminal z menu Picaxe, nebo stiskem F8.

Například:

```
loop:for b1 = 0 to 63                ' start a loop
            sertxd("The value of b1 is ",#b1,13,10)
            pause 1000
            next b1                  ' next loop
```

servo (použitelné pro PICAXE: 08M, 18A, 18X, 28A, 28X, 40X)

syntaxe:

SERVO pin,pulse

- pin je proměnná nebo konstanta (0-7) určující, který vývod bude použit.

- pulse je proměnná nebo konstanta (75-225) určující polohu serva (šířku pulsu)

funkce:

Příkaz servo posílá na určený výstup řídící pulsy s opakovací periodou 20 ms, které mohou sloužit k ovládání modelářského servomechanismu (serva). Pulsy jsou vysílány trvale i po provedení příkazu.

Serva, často používaná modeláři, vyžadují ke svému řízení impulsy o šířce 1 až 2 ms opakované neustále každých 20 ms.

Některá serva fungují v širším rozsahu řídících pulsů až od 0,5 do 2,5 ms. Tento širší rozsah je třeba opatrně vyzkoušet, aby nedošlo ke zničení převodů, kdyby se servo snažilo překonat mechanický doraz.

Například:

```
loop:       servo 4,75          ' move servo to one end
            pause 2000          ' wait 2 seconds
            servo 4,150         ' move servo to centre
            pause 2000          ' wait 2 seconds
            servo 4,225         ' move servo to other end
            pause 2000          ' wait 2 seconds
            goto loop           ' loop back to start
```

setint (použitelné pro PICAXE: 08M, 18A, 18X, 28A, 28X, 40X)

syntaxe:

SETINT input,mask

- input je proměnná nebo konstanta (0-255), která udává hodnoty vstupů

- mask je proměnná nebo konstanta (0-255), která určuje masku – použité vstupy.

funkce:

Umožní přerušit běh programu při zadané kombinaci vstupů – zpracování asynchronních událostí. Podrobnosti naleznete v originální anglické příručce.

setfreq (použitelné pro PICAXE: 08M, 18A, 18X)

syntaxe:

SETFREQ freq

- freq je klíčové slovo m4 nebo m8.

funkce:

Nastaví hodinovou frekvenci u kontrolerů s interním oscilátorem na 4MHz nebo 8MHz. Obvyklá frekvence nastavená po zapnutí napájení je 4MHz. Vyšší frekvence umožňuje rychlejší běh programu, ale má vliv na řadu příkazů, které závisí nějakým způsobem na časování.

Například:

```
     setfreq m4              ' setfreq to 4 MHz
     readtemp 1,b1           ' do command at 4 MHz
     setfreq m8              ' set freq back to 8 MHz
```

sleep (použitelné pro všechny typy PICAXE)

syntaxe:

SLEEP period

- period je proměnná nebo konstanta (0-65535), která udává dobu uspání kontroleru v násobcích 2,3 s.

funkce:

Tento příkaz uspí kontroler na zadaný počet násobků 2,3 s. Příkaz využívá watchdog timer s omezenou přesností časování.

Například:

```
loop:high 1             ' switch on output 1
          sleep 10      ' sleep for 23 seconds
          low 1         ' switch off output 1
          sleep 100     ' sleep for 230 seconds
          goto loop     ' loop back to start
```

sound (použitelné pro všechny typy PICAXE)

syntaxe:

SOUND pin,(note,duration,note,duration...)

- pin je proměnná nebo konstanta (0-7), určující, který vývod bude použit.

- note jsou proměnné nebo konstanty (0-255), které určují typ a frekvenci tónu. 0 značí pauzu, hodnoty 1-127 jsou tóny stoupající frekvence, 128-255 je bílý šum stoupající frekvence.

- duration jsou proměnné nebo konstanty (0-255), které určují délku tónu v násobcích 10 ms.

funkce:

Tento příkaz vytváří pípnutí vhodné například k upozornění obsluhy, indikaci stisku tlačítka apod. Ke generování hudebního signálu jsou určeny příkazy play nebo tune. Parametry nota a duration musí být uvedeny vždy v páru.

(Vhodné obvody pro reprodukci jsou uvedeny u příkazu tune.)

Například:

```
loop: let b0 = b0 + 1           ' increment b0
      sound 7,(b0,50)           ' make a sound
      goto loop                 ' loop back to start
```

stop (použitelné pro všechny typy PICAXE)

syntaxe:

STOP

funkce:

Ukončí běh programu. K obnovení funkce mikrokontroléru dojde vypnutím a opětovným zapnutím napájení nebo zavedením nového programu z PC

Příkaz stop ukončí program. Na rozdíl od příkazu end se mikrokontrolér nenastaví do úsporného režimu, takže příkazy servo a pwmout pokračují v činnosti.

Například:

```
loop: pwmout 1,120,400
      stop
```

switch on/off (použitelné pro všechny typy PICAXE)

syntaxe:

SWITCH ON pin

SWITCHON pin

SWITCH OFF pin

SWITCHOFF pin

- pin je proměnná nebo konstanta, označující i/o pin

funkce:

Nastaví pin na high / low

Pro informaci:

Tento příkaz je vytvořen pro nejmladší uživatele PICAXE a je zcela ekvivalentní příkazům „high“ a „low“.

Například:

```
loop:       switch on 7   ' switch on output 7
            wait 5        ' wait 5 seconds
            switch off 7  ' switch off output 7
            wait 5        ' wait 5 seconds
            goto loop     ' loop back to start
```

toggle (použitelné pro všechny typy PICAXE)

syntaxe:

TOGGLE pin

- pin je proměnná nebo konstanta (0-7) určující, který vývod bude použit.

funkce:

Změní logickou úroveň na určeném výstupu.

Například:

```
loop: toggle 7           ' toggle output 7
      pause 1000         ' wait 1 second
      goto loop          ' loop back to start
```

tune (použitelné pro PICAXE: 08M)

syntaxe:

TUNE LED, speed, (note, note, note...)

- LED je proměnná nebo konstanta (0-3) určující způsob blikání připojených LED během hraní.
0 - bez blikání
1 - výstup 0 se zapíná a vypíná
2 - výstup 4 se zapíná a vypíná
3 - výstupy 0 a 4 se střídavě zapínají a vypínají

- speed je proměnná nebo konstanta (1-15) určující tempo přehrávání skladby.

- note jsou konstanty (data), generovaná pomocí programu Tune Wizard.

Podrobnosti naleznete v originální anglické příručce.

wait (použitelné pro všechny typy PICAXE)

syntaxe:

WAIT seconds

- seconds je konstanta, která určuje dobu čekání v sekundách

funkce:

Tento příkaz je ekvivalentní ‚pause * 1000', jedná se o jakési makro a nemůže být z tohoto důvodu použit s proměnnou.

Například:

```
loop: switch on 7        ' switch on output 7
      wait 5             ' wait 5 seconds
      switch off 7       ' switch off output 7
      wait 5             ' wait 5 seconds
      goto loop          ' loop back to start
```

write (použitelné pro všechny typy PICAXE)

syntaxe:

WRITE location,data

- location je proměnná nebo konstanta určující a adresu (0-255).

- data je proměnná nebo konstanta představující data určená k zápisu.

funkce:

Příkaz zapisuje data do interní eeprom mikrokontroleru. Obsah této paměti se uchová i po vypnutí napájení. Tato data jsou zapisována při každém zavedení nového programu do mikrokontroléru, podle definice v příkazu DATA/EEPROM. Za běhu programu mohou být tato data čtena pomocí příkazu read. Příkaz WRITE pracuje pouze s 8 bitovými daty. Pokud je třeba zapsat proměnnou word, dosáhne se toho použitím dvou příkazů WRITE s oběma 8bitovými proměnnými: (například w0 se zapíše tak, že se zapíše b0 a b1). Velikost, umístění a případná omezení použitelnosti eeprom u jednotlivých typů kontrolérů jsou uvedeny u příkazu DATA.

Například:

```
loop: for b0 = 0 to 63          ' start a loop
      serin 6,T2400,b1          ' receive serial value
      write b0,b1               ' write value into b1
      next b0                   ' next loop
```

writei2c (použitelné pro PICAXE: 18X, 28X, 40X)

syntaxe:

WRITEI2C location,(variable,...)

- location je proměnná nebo konstanta určující a osmibitovou nebo šestnáctibitovou adresu.

- variable obsahují data určená k zápisu.

funkce:

Zapíše obsah proměnných na určenou adresu v I2C zařízení. Adresa zařízení na I2C sběrnici je určená příkazem I2CSLAVE, Location udává adresu v rámci tohoto zařízení, například adresu dat v (externí) EEPROM nebo RTC obvodu a pod.

Podrobnější informace jsou v anglické příručce k I2C sběrnici.

Například:

```
     ; Example of how to use DS1307 Time Clock
     ; Note the data is sent/received in BCD format.
     ; Note that seconds, mins etc are variables that need
     ; defining e.g. symbol seconds = b0 etc.
     ; set DS1307 slave address i2cslave %11010000, i2cslow, i2cbyte
     ; write time and date e.g. to 11:59:00 on Thurs 25/12/03

start_clock:      let seconds = $00         ' 00 Note all BCD format
                  let mins = $59            ' 59 Note all BCD format
                  let hour = $11            ' 11 Note all BCD format
                  let day = $03             ' 03 Note all BCD format
                  let date = $25            ' 25 Note all BCD format
                  let month = $12           ' 12 Note all BCD format
                  let year = $03            ' 03 Note all BCD format
                  let control = %00010000   ' Enable output at 1Hz
                  writei2c 0,(seconds,mins,hour,day,date,month,year,control)
                  end
```