Jméno:

high

Syntaxe:

HIGH pin {,pin,pin...}

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Nastaví pin na vysokou úroveň = na pinu bude přivedeno napájecí napětí.  
Současně nastaví pin, jako výstupní.

Platí pro:

Vše

Příbuzné:

* [low](#low)
* [toggle](#toggle)
* [switch](http://www.picaxe.com/commands/switch)

Příklad:

Blikání LED diody s periodou 10 s.

main:

high B.1 ; nastaví pin B.1 na vysokou úroveň

pause 5000 ; čeká 5 s

low B.1 ; nastaví pin B.1 na nízkou úroveň

pause 5000 ; čeká 5 s

goto main ; skok na „main“

Jméno:

low

Syntaxe:

LOW pin {,pin,pin...}

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Nastaví pin na nízkou úroveň = na pinu bude nulové napětí.  
Současně nastaví pin jako výstupní.

Platí pro:

Vše

Příbuzné:

* [high](#high)
* [toggle](#toggle)
* [switch](http://www.picaxe.com/commands/switch)

Příklad:

Blikání LED diody s periodou 10 s.

main:

high B.1 ; nastaví pin B.1 na vysokou úroveň

pause 5000 ; čeká 5 s

low B.1 ; nastaví pin B.1 na nízkou úroveň

pause 5000 ; čeká 5 s

goto main ; skok na „main“

Jméno:

toggle

Syntaxe:

TOGGLE pin,pin,pin...

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Invertuje (mění) úroveň pinu = z nízké na vysokou a z vysoké na nízkou.  
Současně nastavuje pin, jako výstupní.

Platí pro:

Vše

Příbuzné:

* [high](#high)
* [low](#low)

Příklad:

Blikání LED s periodou 1 s.

main:

toggle B.7 ; změní stav na pinu B.7

pause 500 ; počká 500 ms

goto main ; skočí na „main“

Jméno:

backward

Syntaxe:

BACKWARD motor

*Motor* – je jméno motoru A nebo B.

Popis:

Jde o pseudopříkaz, používaný pro snadné programování typových 2 kolových robotů.  
Příkaz provede pro motor A - low B.4; high B.5 a pro motor B - low B.6; high B.7.

Platí pro:

Vše (kromě 08, 08M, 08M2)

Příbuzné:

* [forward](http://www.picaxe.com/commands/forward)
* [halt](http://www.picaxe.com/commands/halt)

Příklad:

Demonstrace příkazů pro motor.

main:

forward A ; motor A vpřed

wait 5 ; čekenj 5 s

backward A ; motor A zpět

wait 5 ; čekej 5 s

halt A ; motor A stop

wait 5 ; čekej 5 s

goto main ; skok na „main“

Jméno:

forward

Syntaxe:

FORWARD motor

*Motor* – je jméno motoru A, nebo B.

Popis:

Jde o pseudopříkaz, používaný pro snadné programování typových 2 kolových robotů.  
Příkaz provede pro motor A - high B.4; low B.5, pro motor B - 'high B.6; low B.7.

Platí pro:

Vše (kromě 08, 08M, 08M2)

Příbuzné:

* [backward](http://www.picaxe.com/commands/backward)
* [halt](http://www.picaxe.com/commands/halt)

Příklad:

Demonstrace příkazů pro motor.

main:

forward A ; motor A vpřed

wait 5 ; čekenj 5 s

backward A ; motor A zpět

wait 5 ; čekej 5 s

halt A ; motor A stop

wait 5 ; čekej 5 s

goto main ; skok na „main“

Jméno:

halt

Syntaxe:

HALT motor

*Motor -* je jméno motoru A, nebo B.

Popis:

Jde o pseudopříkaz, používaný pro snadné programování typových 2 kolových robotů.  
Příkaz provede pro motor A - low B.4; low B.5 a pro motor B - low B.6; low B.7.

Platí pro:

Vše (kromě 08, 08M, 08M2)

Příbuzné:

* [backward](http://www.picaxe.com/commands/backward)
* [forward](http://www.picaxe.com/commands/forward)

Příklad:

Demonstrace příkazů pro motor.

main:

forward A ; motor A vpřed

wait 5 ; čekenj 5 s

backward A ; motor A zpět

wait 5 ; čekej 5 s

halt A ; motor A stop

wait 5 ; čekej 5 s

goto main ; skok na „main“

Jméno:

sound

Syntaxe:

SOUND pin,(nota,délka,nota,délka...)

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Nota* – proměnná/konstanta udávající výšku tónu. Hodnoty < 128 jsou tóny, >127 jsou šumy.

*Délka* – proměnná/konstanta udávající délka tónu (0-255) udaná v násobcích 10 ms.

Popis:

Příkaz na pinu generuje definovaný tón pro piezoměnič. Pro přehrávání hudby  
jsou vhodnější příkazy PLAY a TUNE. Tón a délka musí být definované v páru.  
Výstup závisí na taktovací frekvenci - nastaveno pro 8 MHz.

Platí pro:

Vše

Příbuzné:

* [play](http://www.picaxe.com/commands/play)
* [tune](http://www.picaxe.com/commands/tune)

Play a sound on output B.7 then play the next higher sound frequency

Příklad:

Přehrání všech tónů na pinu B.7

main:

let b0 = b0 + 1 ; zvětšení hodnoty v proměnné „b0“

sound B.7,(b0,50) ; generování tónu po dobu 0,5 s

goto main ; skok na „main“

Jméno:

play

Syntaxe:

PLAY pin, tune (není určen pro 8 pinové typy)

PLAY pin, tune, LED\_mask (pouze pro typ M2)

PLAY tune, LED\_option (pro 8 pinové typy)

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení) U 8 pinových vývodů je pevně určen vývod C.2.

*Tune* – proměnná/konstanta udávající číslo melodie

|  |  |
| --- | --- |
| 0 | - Happy Birthday |
| 1 | - Jingle Bells |
| 2 | - Silent Night |
| 3 | - Rudolph the Red Nosed Reindeer |

*LED\_mask* (typ M2) - proměnná/konstanta která nastaví piny na daném portu, které budou blikat v rytmu. Maska se může nastavit např. dvojkovým zápisem %00000011; 1 - bliká, 0 - nebliká

*LED\_option* (typ 08M/08M2) – proměnná/konstanta v rozsahu (0 -3), která nastavuje rytmické blikání následovně:

|  |  |
| --- | --- |
| 0 | - žádné blikání |
| 1 | - blikání na C.0 |
| 2 | - blikání na C.4 |
| 3 | - blikání na C.0 a C.4 – střídavě |

Popis:

Příkaz na pinu přehraje předefinovanou skladbu, případně na dalších pinech generuje signál pro blikání LED diod. Pro přehrání vlastní skladby se používá příkaz TUNE. Během provádění příkazu je taktovací frekvence 4 MHz.

Platí pro:

Vše (kromě 08, 18, 18A, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [sound](http://www.picaxe.com/commands/sound)
* [tune](http://www.picaxe.com/commands/tune)

Příklad:

Přehraje „Rudolf red“ na typu 08M2, pinu C.2 hudba, na C.0 blikání

play 3,1 ; „Rudolf red“ + blikání

Jméno:

tune

Syntaxe:

TUNE pin, speed, (note, note, note...)

TUNE pin, speed, LED\_mask, (note, note, note...) (pouze pro typ M2)

TUNE LED\_option, speed, (note, note, note...) (pouze pro 8 pinové)

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení) U 8 pinových vývodů je pevně určen vývod C.2.

*speed* – proměnná/konstanta udávající rychlost v rozsahu (1-15).

*notes* – proměnná/konstanta udávající jednotlivé noty, nejsnáze se zadávají v programu Tune Wizard – součást Picaxe programovací editor.

*LED\_mask* (typ M2) - proměnná/konstanta která nastaví piny na daném portu, které budou blikat v rytmu. Maska se může nastavit např. dvojkovým zápisem %00000011; 1 - bliká, 0 - nebliká

*LED\_option* (typ 08M/08M2) – proměnná/konstanta v rozsahu (0 -3), která nastavuje rytmické blikání následovně:

|  |  |
| --- | --- |
| 0 | - žádné blikání |
| 1 | - blikání na C.0 |
| 2 | - blikání na C.4 |
| 3 | - blikání na C.0 a C.4 – střídavě |

Popis:

Příkaz na pinu přehraje zadané tóny, případně na dalších pinech generuje signál pro blikání LED diod. Změna stavu diody je po každé notě. Noty možno nejsnáze zadat pomocí Tune průvodce, načtením mono zvonění, nebo ručně po jednotlivých tónech. U 8 pinových typů je zvukový výstup na pinu C.2.

Platí pro:

Vše (kromě 08, 18, 18A, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [play](http://www.picaxe.com/commands/play)

Příklad:

tune C.2, 8, %00000000,($C0,$C2,$C4,$C5,$C7,$C9,$CB,$D0) ;stupnice Cdur

Jméno:

servo

Syntaxe:

SERVO pin,pulse

SERVO pin,OFF

SERVO [preload],pin,pulse (X2 only)

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Pulse* – proměnná/konstanta v rozsahu (75-225), určující polohu serva, v rozšířeném módu (0 – 255)

*Preload* – časová konstanta (pouze u X2), určená pro digitální serva.

Popis:

Příkaz na pinu generuje pulsy ovládající modelářské servo. Pulsy jsou generovány nezávisle na běhu programu, dokud není příkaz ukončen příkazem SERVO pin,OFF. Příkaz funguje správně při frekvenci 4 a 16 MHz.

Po inicializaci je vhodné používat příkaz [servopos](#sound), z důvodu omezení kolísaní pulsů.

Příkaz bude okamžitě zastaven při použití příkazu high, nebo low na daném pinu.  
Příkaz bude pozastaven po dobu provádění příkazů serin, serout, sertxd, debug, apod.

Při konstrukci by servo mělo být napájeno samostatným zdrojem 6 V.

Platí pro:

Vše (kromě 08, 18)

Příbuzné:

* [servopos](http://www.picaxe.com/commands/servopos)

Příklad:

Přesun serva mezi dvěma pozicemi.

init: servo 4,150 ; inicializace serva + středová poloha

main: servopos 4,100 ; přesun do jedné pozice

pause 2000 ; čekej 2 s

servopos 4,200 ; přesun do druhé pozice

pause 2000 ; čeken 2 s

goto main ; opakovací smyčka – skok na „main“

Jméno:

servopos

Syntaxe:

SERVOPOS pin,pulse

SERVOPOS pin,OFF

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Pulse* - proměnná/konstanta v rozsahu (75-225), určující polohu serva, v rozšířeném módu (0 – 255)

Popis:

Příkaz na pinu generuje pulsy ovládající modelářské servo. Pulsy jsou generovány nezávisle na běhu programu, dokud není příkaz ukončen příkazem   
SERVOPOS pin,OFF. Příkaz funguje správně při frekvenci 4 a 16 MHz.

Před použitím musí být pin inicializován příkazem [servo](#servo). Příkaz omezuje kolísání signálu a omezuje resetování procesoru.

Příkaz bude okamžitě zastaven při použití příkazu [high](#high), nebo [low](#low) na daném pinu.  
Příkaz bude pozastaven po dobu provádění příkazů [serin](#high), [serout](#high), [sertxd](#high), [debug](#high), apod.

Při konstrukci by servo mělo být napájeno samostatným zdrojem 6 V.

Platí pro:

Vše (kromě 08, 18)

Příbuzné:

[servo](http://www.picaxe.com/commands/servo)

Příklad:

Přesun serva mezi dvěma pozicemi.

init: servo 4,150 ; inicializace serva + středová poloha

main: servopos 4,100 ; přesun do jedné pozice

pause 2000 ; čekej 2 s

servopos 4,200 ; přesun do druhé pozice

pause 2000 ; čeken 2 s

goto main ; opakovací smyčka – skok na „main“

Jméno:

irin

Syntaxe:

IRIN pin, proměnná

IRIN [timeout], pin, proměnná

IRIN [timeout, address], pin, proměnná

*Timeout* - proměnná/konstanta udávající dobu čekání na signál v milisekundách

*Address* – adresa (návěští) skoku při překročení časového limitu

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Proměnná* – proměnná, do které budou uložena načtená data

Popis:

Příkaz čeká po definovanou dobu na přijetí IR signálu, potom skočí na definovanou adresu. Není-li doba definována, čeká do příchodu signálu. Po dobu čekání jsou pozastaveny ostatní příkazy.

Příkaz načte kód pomocí infračerveného přenosu, standard Sony, identifikace 1-TV. Pro příjem je třeba připojit IR přijímač typu SFH5110. Příkaz funguje správně při frekvenci 4 MHz.

Přenos je 7 bitový = možno přenést číslo v rozsahu 0 – 127.

Možná komunikace mezi dvěma PICAXE pomocí příkazu [infraout](#high).

Starší příkazy [infrain](http://www.picaxe.com/commands/infrain) / [infrain2](http://www.picaxe.com/commands/infrain2) lze nahradit následovně:

|  |  |
| --- | --- |
|  | symbol infra = b13 |
|  | irin C.0, infra |

+ provést korekci načtených hodnot.

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [irout](http://www.picaxe.com/commands/irout)
* [infrain](http://www.picaxe.com/commands/infrain)
* [infrain2](http://www.picaxe.com/commands/infrain2)
* [infraout](http://www.picaxe.com/commands/infraout)

Příklad:

Dálkové IR ovládání světla.

main: irin [1000,main],C.3,b0 ; čekání na signál

if b0 = 1 then

high B.1 ; zapnutí světla

endif

if b0 = 4 then

low B.1 ; vypnutí světla

endif

goto main ; skok na začátek - smyčka

Jméno:

irout

Syntaxe:

IROUT pin,device,data

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Device* – konstanta/proměnná identifikující zařízení (ID 1-31)

*Data* – konstanta/proměnná obsahující výstupní data v rozsahu (0-127)

Popis:

Pomocí IR LED na výstupním pinu, vyšle signál modulovaný na 38 kHz.  
Signál je v kódování Sony – SIRC. Signál přijímají zařízení Sony a lze jej přečíst příkazem [irin](#irin).  
Každý typ zařízení má své identifikační číslo, PICAXE a TV má ID = 1.

Přenos je 7 bitový = je možné vyslat číslo v rozsahu (0-127).  
Pro správnou funkci je třeba taktovací frekvence 4 MHz.

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [irin](http://www.picaxe.com/commands/irin)
* [infrain2](http://www.picaxe.com/commands/infrain2)
* [infraout](http://www.picaxe.com/commands/infraout)

Příklad:

Vyšle IR příkaz pro TV – stisk tlačítka „5“.

for b1 = 1 to 10 ; 10-krát pošle

irout B.1,1,5 ; kód„5“přes IR LED, připojenou na B.1, do TV

pause 45 ; čeká 45 ms

next b1

Jméno:

rfin

Syntaxe:

RFIN pin, prom0, prom1, prom2, prom3, prom4, prom5, prom6, prom7

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Prom 0* – 7 – 8 bytových proměnných pro uložení přijatých dat

Popis:

Příkaz vždy přijímá 8 bytů v kódování Manchester.  
Po dobu příjmu jsou pozastavený všechny ostatní procesy.  
Data může přijímat buď z dekodéru NKM2401, nebo z PIXAXE příkaz rfout.

Obvod NKM2401 lze použít jako nezávislý vysílač/přijímač a komunikovat s ním po sériové lince. Úplné informace o NKM2401 včetně použití RF modulu naleznete [AXE213 datasheet](http://www.picaxe.com/docs/axe213.pdf).

Příkaz běží správně při frekvenci 4 MHz.

Platí pro:

14M2, 20M2, 28X2, 40X2

Příbuzné:

* [rfout](http://www.picaxe.com/commands/rfout)

Příklad:

Přečtení paketu 8 x 1 byte z RF modulu, připojeného na C.0.

main: rfin C.0, b0,b1,b2,b3,b4,b5,b6,b7 ; načtení dat

debug ; přenesení do PC

goto main ; skok na „main“

Jméno:

rfout

Syntaxe:

RFOUT pin, (data, data, data, data, data, data, data, data)

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Data* - konstanta/proměnná obsahující odesílaná data

Popis:

Pošle 8 bytů pomocí Manchesterského protokolu přes RF modul, do NKM2401 přijímače nebo do PICAXE příkaz [rfin](#rfin).

Obvod NKM2401 lze použít jako nezávislý vysílač/přijímač a komunikovat s ním po sériové lince. Úplné informace o NKM2401 včetně použití RF modulu naleznete [AXE213 datasheet](http://www.picaxe.com/docs/axe213.pdf).

Příkaz běží správně při frekvenci 4 MHz.

Platí pro:

14M2, 20M2, 28X2, 40X2

Příbuzné:

* [rfin](http://www.picaxe.com/commands/rfin)

Příklad:

Vysílá aktuální teplotu. Načte aktuální teplotu z čidla na C.1 a pošle ji přes RF.

main: readtemp C.1, b7 ; načte teplotu do proměné b7

bintoascii b7,b8,b9,b10 ; převede hodnotu b7 do ASCII

high b.1 ; zapne vysílací modul (TXEN)

rfout b.0,(“Temp=”,b8,b9,b10) ; odešle data (TX)

low b.1 ; vypne vysílací modul (TXEN)

pause 2000 ; čeká 2 s

goto main ; skok na začátek

Jméno:

pwmout

Syntaxe:

PWMOUT pin, period, duty

PWMOUT PWMDIV4,pin, period, duty

PWMOUT PWMDIV16, pin, period, duty

PWMOUT PWMDIV64, pin, period, duty

PWMOUT pin, OFF

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Period* - proměnná/konstanta (0-255) určující periodu PWM

*Duty* - proměnná/konstanta (0-1023) vyjadřující dobu plnění PWM

PWMDIVčíslo – udává dělení taktovací frekvence procesoru a tím zpomalení PWM.

Popis:

Generuje kontinuální střídavý signál s proměnnou střídou – plněním. Je vhodný k regulaci výkonu.

Příkaz běží stále na pozadí, dokud není vypnut PWMOUT pin, OFF   
(= PWMOUT pin, 0, 0).

perioda PWM = (period + 1) x 4 x perioda procesoru  
plnění PWM = (duty) x perioda procesoru  
  
perioda procesoru = 1 / frekvence procesoru  
perioda procesoru pro frekvenci 4 MHz = 1/4000000 s = 0,25 us

Pro výpočet lze použít i PICAXE>Wizards>pwmout menu.

U starších typů může dojít ke kolizi při současném použití příkazu [servo](#pwmout), novější již obsahují více nezávislých časovačů.  
Například:

|  |  |
| --- | --- |
| 14M2 | PWM - B.2, B.4 + SERVO - C.0, C.2 |
| 18M2 | PWM - B.3, B.6 |
| 20M2 | PWM - B.1, C.2 + SERVO - C.3, C.5 |
| 28X2 | PWM - B.0, B.5 + SERVO - C.1, C.2 |

Pwmout je ukončen při použití příkazu: [nap](http://www.picaxe.com/commands/nap), [sleep](http://www.picaxe.com/commands/sleep), [end](http://www.picaxe.com/commands/end)   
Pwmout může být použit současně s [hpwm](http://www.picaxe.com/commands/hpwm)   
Pwmout je ovlivněn změnou frekvence procesoru, například při použití příkazu [readtemp](http://www.picaxe.com/commands/readtemp), apod.  
Pro rychlou změnu plnění je vhodné používat příkaz [pwmduty](http://www.picaxe.com/commands/pwmduty) .   
Při ovládání výkonu pomocí FET – tranzistoru, je pro správnou funkci doporučeno připojit na výstupní pin „pull-down“ rezistor = 10k k zemi.

Platí pro:

Vše (kromě 08, 18, 18A, 20M, 28, 28A)

Příbuzné:

* [pwmduty](http://www.picaxe.com/commands/pwmduty)
* [hpwm](http://www.picaxe.com/commands/hpwm)
* [hpwmduty](http://www.picaxe.com/commands/hpwmduty)
* [pwm](http://www.picaxe.com/commands/pwm)

Příklad:

Změna intenzity svícení v sekundových intervalech.

init: pwmout C.2,150,100 ; start pwm

main: pwmduty C.2,150 ; nastavení pwm plnění

pause 1000 ; pauza 1 s

pwmduty C.2,50 ; změna pwm plnění

pause 1000 ; pauza 1 s

goto main ; skok na „main“ – smyčka

Jméno:

pwmduty

Syntaxe:

PWMDUTY pin,duty

*Pin* - konstanta určující specifický pin v závislosti na typu - viz příloha

*Duty* - proměnná/konstanta (0-1023) vyjadřující dobu plnění PWM

Popis:

Upravuje dobu plnění PWM, bez resetování vnitřního časovače (na rozdíl od [pwmpout](#pwmout)).  
Příkaz může být použit až po aktivaci příkazem [pwmout](#pwmout).

Platí pro:

Vše (kromě 08, 18, 18A, 20M, 28, 28A)

Příbuzné:

* [pwmout](http://www.picaxe.com/commands/pwmout)
* [hpwm](http://www.picaxe.com/commands/hpwn)
* [hpwmduty](http://www.picaxe.com/commands/hpwmduty)

Příklad:

Změna intenzity svícení v sekundových intervalech.

init: pwmout C.2,150,100 ; start pwm

main: pwmduty C.2,150 ; nastavení pwm plnění

pause 1000 ; pauza 1 s

pwmduty C.2,50 ; změna pwm plnění

pause 1000 ; pauza 1 s

goto main ; skok na „main“ – smyčka

Jméno:

count

Syntaxe:

COUNT pin, period, proměnná

*Pin* - proměnná/konstanta specifikující vstupní pin (viz. příloha)

*Period* - proměnná/konstanta určující dobu počítání impulsů (1-65535ms při 4MHz).

*Proměnná* - typu word do níž bude uložen počet pulsů (0-65535).

Popis:

Počítá počet impulzů na vstupním pinu.

Příkaz počítá přechody 0->1 na vstupním pinu a počet uloží do proměnné typu word. Při frekvenci 4 MHz načet impuls delší než 20 us (tj. 25 kHz), za předpokladu 50% plnění.(0 a 1 trvají stejně dlouho)

Pozor na záchvěvy u mechanických spínačů, mohou zkreslit výsledek.

Rychlost procesoru se projevuje na funkci příkazu následovně:

|  |  |  |
| --- | --- | --- |
| **Frek.procesoru** | **Min. šířka sign.** | **Max. frekvence pulsů** |
| 4MHz | 40us | 25kHz |
| 8MHz | 20us | 50kHz |
| 16MHz | 10us | 100kHz |
| 32MHz | 5us | 200kHz |
| 64MHz | 2.5us | 400kHz |

Pro opakované měření, je nutno počítat i s dobou vyhodnocení, pak přibližně platí:

|  |  |
| --- | --- |
| **Frek.procesoru** | **Min.čas mezi dvěma pulzy** |
| 4MHz | 1ms (1000 us) |
| 8MHz | 500 us |
| 16MHz | 250 us |
| 32MHz | 125 us |
| 64MHz | 62.5 us |

Platí pro:

Vše

Příbuzné:

* [pulsin](http://www.picaxe.com/commands/pulsin)

Příklad:

Načítá v 5 sekundových intervalech pulzy na pinu C.1 a odesílá na PC.

main:

count C.1, 5000, w1 ; načítá pulzy po dobu 5secs (při 4 MHz)

debug ; odešle na PC

goto main ; skok na „main“ - opakuje

Jméno:

pulsin

Syntaxe:

PULSIN pin, state, proměnná

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*State* - proměnná/konstanta (0, 1): 0 - načítá sestupné hrany, 1 - načítá náběžné hrany. Signál musí trvat min. 10us (pro 4 MHz frekvenci procesoru).

*Proměnná* - typu word pro uložení délky pulsu (1-65535), v jednotkách závislých na frekvenci procesoru (0.00001 - 0.65535 s pro 4MHz). Při vypršení limitu bude výsledek 0.

Popis:

Změří délku pulsu.

Při state = 0 měří od sestupné hrany, při state = 1 od náběžné hrany.  
Délka je uložena v proměnné typu word, v jednotkách podle frekvence procesoru, viz. níže. Při absenci pulsu v měřeném úseku je výsledek 0.

Frekvence Časová jednotka Doba měření

|  |  |  |
| --- | --- | --- |
| 4MHz | 10us | 0.65536s timeout |
| 8MHz | 5us | 0.32768s timeout |
| 16MHz | 2.5us | 0.16384s timeout |
| 32MHz | 1.25us | 0.08192s timeout |
| 64MHz | 0.625us | 0.04096s timeout |

Platí pro:

Vše

Příbuzné:

* [pulsout](http://www.picaxe.com/commands/pulsout)
* [count](http://www.picaxe.com/commands/count)

Příklad:

Změří délku pulzu na C.3 a výsledek pošle do PC.

main:

pulsin C.3,1,w1 ; načtení délky pulzu z C.3 do w1

debug ; odeslání do PC

goto main ; opakování

Jméno:

pulsout

Syntaxe:

PULSOUT pin,time

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Time* - proměnná/konstanta určující délku pulsu v jednotkách závislých na taktovací frekvenci - jednotka 10us pro 4 MHz.

Popis:

Na zvoleném pinu generuje pulz o požadované délce (šířce).

Je-li na pinu 0 => pulz bude 1, je-li na pinu 1=> pulz bude 0. Před použitím příkazu je třeba pin nastavit do požadovaného stavu pomocí [high](http://www.picaxe.com/commands/high), [low](http://www.picaxe.com/commands/low) nebo [output](http://www.picaxe.com/commands/output).

Časové jednotky, pro parametr time:

|  |  |
| --- | --- |
| 4MHz => | 10us |
| 8MHz => | 5us |
| 16MHz => | 2.5us |
| 32MHz => | 1.25us |
| 64MHz => | 0.625us |

Platí pro:

Vše

Příbuzné:

* [pulsin](http://www.picaxe.com/commands/pulsin)

Příklad:

Generátor 1.5ms pulsu na pinu B.1, s intervalem mezi pulzy 20 ms.

main:

pulsout B.1,150 ; nastavení 1.50 ms pulzu na pin B.1

pause 20 ; pauza 20 ms

goto main ; opakovat

Jméno:

input

Syntaxe:

INPUT pin,pin,pin...

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Přepne programovatelné piny (viz. příloha - IN/OUT) na vstupní.   
Pevně nastavené piny nejsou ovlivněny. Jsou to:

|  |  |  |
| --- | --- | --- |
| 08, 08M, 08M2 | C.0 = pevný výstup | C.3 = pevný vstup |
| 14M2 | B.0 = pevný výstup | C.3 = pevný vstup |
| 18M2 | C.3 = pevný výstup | C.4, C.5 = pevný vstup |
| 20M2, 20X2 | A.0 = pevný výstup | C.6 = pevný vstup |
| 28X2, 40X2 | A.4 = pevný výstup |  |

Platí pro:

08, 08M, 08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [inputtype](http://www.picaxe.com/commands/inputtype)
* [pullup](http://www.picaxe.com/commands/pullup)
* [output](http://www.picaxe.com/commands/output)
* [reverse](http://www.picaxe.com/commands/reverse)

Příklad:

Přepínání pinu mezi vstupním a výstupním.

main:

input B.1 ; nastaví pin na vstupní

reverse B.1 ; nastaví pin na výstupní

reverse B.1 ; nastaví pin na vstupní

output B.1 ; nastaví pin na výstupní

Jméno:

output

Syntaxe:

OUTPUT pin,pin, pin...

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Přepne programovatelné piny (viz. příloha - IN/OUT) na výstupní.   
Pevně nastavené piny nejsou ovlivněny. Jsou to:

|  |  |  |
| --- | --- | --- |
| 08, 08M, 08M2 | C.0 = pevný výstup | C.3 = pevný vstup |
| 14M2 | B.0 = pevný výstup | C.3 = pevný vstup |
| 18M2 | C.3 = pevný výstup | C.4, C.5 = pevný vstup |
| 20M2, 20X2 | A.0 = pevný výstup | C.6 = pevný vstup |
| 28X2, 40X2 | A.4 = pevný výstup |  |

Platí pro:

08, 08M, 08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [high](http://www.picaxe.com/commands/high)
* [low](http://www.picaxe.com/commands/low)
* [toggle](http://www.picaxe.com/commands/toggle)
* [input](http://www.picaxe.com/commands/input)
* [reverse](http://www.picaxe.com/commands/reverse)

Příklad:

Přepínání pinu mezi vstupním a výstupním.

main:

input B.1 ; nastaví pin na vstupní

reverse B.1 ; nastaví pin na výstupní

reverse B.1 ; nastaví pin na vstupní

output B.1 ; nastaví pin na výstupní

Jméno:

reverse

Syntaxe:

REVERSE pin,pin,pin...

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Přepne programovatelné piny (viz. příloha - IN/OUT) ze vstupních na výstupní a z výstupních na vstupní.   
Pevně nastavené piny nejsou ovlivněny. Jsou to:

|  |  |  |
| --- | --- | --- |
| 08, 08M, 08M2 | C.0 = pevný výstup | C.3 = pevný vstup |
| 14M2 | B.0 = pevný výstup | C.3 = pevný vstup |
| 18M2 | C.3 = pevný výstup | C.4, C.5 = pevný vstup |
| 20M2, 20X2 | A.0 = pevný výstup | C.6 = pevný vstup |
| 28X2, 40X2 | A.4 = pevný výstup |  |

Platí pro:

08, 08M, 08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [input](http://www.picaxe.com/commands/input)
* [output](http://www.picaxe.com/commands/output)

Příklad:

Přepínání pinu mezi vstupním a výstupním.

main:

input B.1 ; nastaví pin na vstupní

reverse B.1 ; nastaví pin na výstupní

reverse B.1 ; nastaví pin na vstupní

output B.1 ; nastaví pin na výstupní

Jméno:

pullup

Syntaxe:

PULLUP mask

PULLUP OFF (= PULLUP 0)

PULLUP ON (= PULLUP 255)

*mask* - proměnná/konstanta nastavující bitovou masku cílového portu

Popis:

Připojí/odpojí vnitřní rezistory na vstupních pinech.

Po připojení vnitřního rezistoru má vstupní pin vysokou hodnotu (1) a je možné jej přímo propojovat se zemí (0), bez dalších součástek.  
Vnitřní rezistory jsou připojovány podle masky - 1 - připojit, 0 - odpojit.   
Maska může mít až 16bitů (14M2, 20M2).  
Při definování pinu jako výstupního, je vnitřní rezistor automaticky odpojen.  
Vnitřní rezistor je připojitelný pouze k některým pinům:

|  |  |  |
| --- | --- | --- |
| 08M2 | bit0-bit4 = C.0 to C.4 |  |
| 14M2 | bit0-bit7 = B.0 to B.7 | bit8-bit15 = C.0 to C.7 |
| 18M2 | bit0-bit7 = B.0 to B.7 |  |
| 20M2 | bit0-bit7 = B.0 to B.7 | bit8-bit15 = C.0 to C.7 |
| 20X2 | bit0-bit7 = C.0, C.6, C.7, B.0, B.1 B.5, B.6, B.7 |  |
| 28X2/40X2 | bit0-bit7 = B.0 to B.7 |  |
| 28X2-5V/40X2-5V | On = Vše PORTB |  |
| 28X2-3V/40X2-3V | bit0-bit7 = B.0 to B.7 |  |

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [input](http://www.picaxe.com/commands/input)
* [inputtype](http://www.picaxe.com/commands/inputtype)

Příklad:

Ukázka nastavení masky pro různě typy PICAXE.

pullup on ; nastavení všech pinů pro 28X2-5V

pullup %11110000 ; nastavení pinů na portB4-7 pro 28X2

pullup %00000111 ; nastavení pinů portC pro 20X2

pullup %0000001100000000 ; nastavení pinů C.0, C.1 pro 14M2

Jméno:

inputtype

Syntaxe:

INPUTTYPE mask

*Mask* - proměnná/konstanta - maska určující typ vstupu

Popis:

PICAXE umožňuje u některých typů (14M2, 20M2) nastavit dvě různé normy vstupů  
- TTL (hardware typ silicon) - v masce označen 0 - nastaveno defaultně  
- ST (Sxhmitt Trigger) - v masce označen 1

Maska znamená 0 => TTL ; 1 => ST.  
Bity0-7 nastavují B.0 - B.7 a bity 8-15 nastavují C.0 - C.7.  
Normy se liší v hraničních úrovních napětí, při digitálním vstupu.  
Norma TTL je považována za univerzálnější.

|  |  |  |
| --- | --- | --- |
| **Schmitt Trigger (ST)** | 5V | 3V |
| Úroveň '1'když U > 0.8 \* Unapájecí | >4V | >2.4V |
| Úroveň '0'když U < 0.2 \* Unapájecí | <1V | <0.6V |

|  |  |  |
| --- | --- | --- |
| **TTL (Unapájecí > 4.5V)** |  |  |
| Úroveň '1'když U > 2.0V | >2V |  |
| Úroveň '0'když U < 0.8V | <0.8V |  |

|  |  |  |
| --- | --- | --- |
| **TTL (Unapájecí < 4.5V)** |  |  |
| Úroveň '1'když U > 0.25 \* Unapájecí + 0.8V | >1.55V |  |
| Úroveň '0'když U < 0.15 \* Unapájecí | <0.45V |  |

Následující tabulka ukazuje, jakou normu používají jednotlivé typy PICAXE:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **08M2** | **08M** | **08** |
| **Serin** | TTL | TTL | TTL |
| **C.1** | TTL | TTL | TTL |
| **C.2** | ST | ST | ST |
| **C.3** | TTL | TTL | TTL |
| **C.4** | TTL | TTL | TTL |

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  | **14M2\*** | **14M** |
| **Serin** | TTL | TTL |
| **B.0** | TTL | n/a |
| **B.1** | TTL | n/a |
| **B.2** | TTL | n/a |
| **B.3** | TTL | n/a |
| **B.4** | TTL | n/a |
| **B.5** | TTL | n/a |
| **C.0** | TTL | TTL |
| **C.1** | TTL | TTL |
| **C.2** | TTL | TTL |
| **C.3** | TTL | TTL |
| **C.4** | TTL | TTL |

\* 14M2 možno konfigurovat příkazem inputtype

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **18M2** | **18X** | **18M** | **18A** | **18** |
| **Serin** | TTL | ST | ST | ST | ST |
| **B.0** | TTL | n/a | n/a | n/a | n/a |
| **B.1** | TTL | n/a | n/a | n/a | n/a |
| **B.2** | TTL | n/a | n/a | n/a | n/a |
| **B.3** | TTL | n/a | n/a | n/a | n/a |
| **B.4** | TTL | n/a | n/a | n/a | n/a |
| **B.5** | TTL | n/a | n/a | n/a | n/a |
| **B.6** | TTL | n/a | n/a | n/a | n/a |
| **B.7** | TTL | n/a | n/a | n/a | n/a |
| **C.0** | TTL | TTL | TTL | TTL | ST |
| **C.1** | TTL | TTL | TTL | TTL | ST |
| **C.2** | TTL | TTL | TTL | TTL | ST |
| **C.5** | TTL | n/a | n/a | n/a | n/a |
| **C.6** | TTL | ST | ST | ST | ST |
| **C.7** | TTL | ST | ST | ST | ST |
|  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **20X2** | **20M2\*** | **20M** |
| **Serin** | TTL | TTL | TTL |
| **B.0** | TTL | TTL | n/a |
| **B.1** | TTL | TTL | n/a |
| **B.2** | ST | TTL | n/a |
| **B.3** | ST | TTL | n/a |
| **B.4** | ST | TTL | n/a |
| **B.5** | TTL | TTL | n/a |
| **B.6** | TTL | TTL | n/a |
| **B.7** | TTL | TTL | n/a |
| **C.0** | TTL | TTL | TTL |
| **C.1** | ST | TTL | ST |
| **C.2** | ST | TTL | ST |
| **C.3** | ST | TTL | ST |
| **C.4** | ST | TTL | ST |
| **C.5** | ST | TTL | ST |
| **C.6** | TTL | TTL | TTL |
| **C.7** | TTL | TTL | TTL |
| **B.7** | TTL | n/a | n/a | n/a | n/a |
| **C.0** | TTL | TTL | TTL | TTL | ST |
| **C.1** | TTL | TTL | TTL | TTL | ST |
| **C.2** | TTL | TTL | TTL | TTL | ST |
| **C.5** | TTL | n/a | n/a | n/a | n/a |
| **C.6** | TTL | ST | ST | ST | ST |
| **C.7** | TTL | ST | ST | ST | ST |

\* 20M2 možno konfigurovat příkazem inputtype

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **28X2** | **28X2-5V** | **28X2-3V** | **28X1** | **28X** | **28A** | **28** |
| **Serin** | ST | ST | ST | ST | ST | ST | ST |
| **A.0** | TTL | TTL | TTL | TTL | TTL | ADC | ADC |
| **A.1** | TTL | TTL | TTL | TTL | TTL | ADC | ADC |
| **A.2** | TTL | TTL | TTL | TTL | TTL | ADC | ADC |
| **A.3** | TTL | TTL | TTL | TTL | TTL | ADC | ADC |
| **B.0** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.1** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.2** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.3** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.4** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.5** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.6** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **B.7** | TTL | TTL | TTL | n/a | n/a | n/a | n/a |
| **C.0** | TTL | ST | ST | ST | ST | ST | ST |
| **C.1** | TTL | ST | ST | ST | ST | ST | ST |
| **C.2** | TTL | ST | ST | ST | ST | ST | ST |
| **C.3** | TTL | ST | ST | ST | ST | ST | ST |
| **C.4** | TTL | ST | ST | ST | ST | ST | ST |
| **C.5** | TTL | ST | ST | ST | ST | ST | ST |
| **C.6** | TTL | ST | ST | ST | ST | ST | ST |
| **C.7** | TTL | ST | ST | ST | ST | ST | ST |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **40X2** | **40X2-5V** | **40X2-3V** | **40X1** | **40X** |
| **Serin** | ST | ST | ST | ST | ST |
| **A.0** | TTL | TTL | TTL | TTL | TTL |
| **A.1** | TTL | TTL | TTL | TTL | TTL |
| **A.2** | TTL | TTL | TTL | TTL | TTL |
| **A.3** | TTL | TTL | TTL | TTL | TTL |
| **A.5** | TTL | ST | ST | ADC | ADC |
| **A.6** | TTL | ST | ST | ADC | ADC |
| **A.7** | TTL | ST | ST | ADC | ADC |
| **B.0** | TTL | TTL | TTL | n/a | n/a |
| **B.1** | TTL | TTL | TTL | n/a | n/a |
| **B.2** | TTL | TTL | TTL | n/a | n/a |
| **B.3** | TTL | TTL | TTL | n/a | n/a |
| **B.4** | TTL | TTL | TTL | n/a | n/a |
| **B.5** | TTL | TTL | TTL | n/a | n/a |
| **B.6** | TTL | TTL | TTL | n/a | n/a |
| **B.7** | TTL | TTL | TTL | n/a | n/a |
| **C.0** | TTL | ST | ST | ST | ST |
| **C.1** | TTL | ST | ST | ST | ST |
| **C.2** | TTL | ST | ST | ST | ST |
| **C.3** | TTL | ST | ST | ST | ST |
| **C.4** | TTL | ST | ST | ST | ST |
| **C.5** | TTL | ST | ST | ST | ST |
| **C.6** | TTL | ST | ST | ST | ST |
| **C.7** | TTL | ST | ST | ST | ST |
| **D.0** | TTL | TTL | TTL | TTL | TTL |
| **D.1** | TTL | TTL | TTL | TTL | TTL |
| **D.2** | TTL | TTL | TTL | TTL | TTL |
| **D.3** | TTL | TTL | TTL | TTL | TTL |
| **D.4** | TTL | TTL | TTL | TTL | TTL |
| **D.5** | TTL | TTL | TTL | TTL | TTL |
| **D.6** | TTL | TTL | TTL | TTL | TTL |
| **D.7** | TTL | TTL | TTL | TTL | TTL |

Platí pro:

14M2, 20M2

Příbuzné:

* [input](http://www.picaxe.com/commands/input)

Příklad:

Demonstrace nastavení masky, pro nastavení normy vstupu.

main:

inputtype %0000000000001111 ; nastavení pinů B.0 - B.3 do ST

inputtype %0000111100000000 ; nastavení pinů C.0 - C.3 do ST

Jméno:

infrain

Syntaxe:

INFRAIN

Popis:

**Příkaz je zastaralý, je vhodné použít příkaz** [**irin**](http://www.picaxe.com/commands/irin)**.**

Příkaz zastaví veškerou činnost procesoru, dokud na pinu C.0 nepřijme signál.  
Načtená hodnota se umístí do interní proměnné „infra“.  
Pro obnovení činnosti v případě, že signál není přijat, je třeba použít tvrdý restart.

Příkaz správně pracuje pouze s frekvencí 4 MHz.

Další použití je obdobné jako u příkazu [irin](#irin).

Pro komunikaci s ovladačem Sony, platí následující tabulka:

|  |  |
| --- | --- |
| **Tlačítko** | **Hodnota** |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| P+ | 10 |
| 0 | 11 |
| V+ | 12 |
| P- | 13 |
| 10+ | 14 |
| V- | 15 |
| Mute | 16 |
| Power | 17 |

Platí pro:

18A, 18X, 28, 28A, 28X, 40X

Příbuzné:

* [irin](http://www.picaxe.com/commands/irin)
* [irout](http://www.picaxe.com/commands/irout)
* [infrain2](http://www.picaxe.com/commands/infrain2)
* [infraout](http://www.picaxe.com/commands/infraout)

Příklad:

Ovládání 3 LED dálkovým ovladačem.

main:

infrain ;čekání na IR signál

if infra = 1 then swon1 ;1 LED zapnout

if infra = 2 then swon2 ;2 LED zapnout

if infra = 3 then swon3 ;3 LED zapnout

if infra = 4 then swoff1 ;1 LED vypnout

if infra = 5 then swoff2 ;2 LED vypnout

if infra = 6 then swoff3 ;3 LED vypnout

goto main

swon1:

high 1

goto main

swon2:

high 2

goto main

swon3:

high 3

goto main

swoff1:

low 1

goto main

swoff2:

low 2

goto main

swoff3:

low 3

goto main

Jméno:

infrain2

Syntaxe:

INFRAIN2

Popis:

**Příkaz je zastaralý, je vhodné použít příkaz** [**irin**](http://www.picaxe.com/commands/irin)**.**

Příkaz zastaví veškerou činnost procesoru, dokud na pinu C.0 nepřijme signál.  
Načtená hodnota se umístí do proměnné „infra“, která je totožná s proměnnou „b13“.  
Pro obnovení činnosti v případě, že signál není přijat, je třeba použít tvrdý restart.

Příkaz správně pracuje pouze s frekvencí 4 MHz.

Další použití je obdobné jako u příkazu [irin](#irin).

Platí pro:

08M, 08M2, 14M, 14M2, 18M, 18M2, 18X, 20M, 20M2

Příbuzné:

* [irin](http://www.picaxe.com/commands/irin)
* [irout](http://www.picaxe.com/commands/irout)
* [infrain](http://www.picaxe.com/commands/infrain)
* [infraout](http://www.picaxe.com/commands/infraout)

Příklad:

Dálkové ovládání rozsvícení LED.

main:

infrain ;čekání na IR signál

if infra = 1 then swon1 ;1 LED zapnout

if infra = 4 then swoff1 ;1 LED vypnout

goto main

swon1:

high 1

goto main

swoff1:

low 1

goto main

goto main

Jméno:

infraout

Syntaxe:

INFRAOUT device,data

*device* - konstanta/proměnná, určující identifikační číslo přijímače ( ID 1-31)

*data* - 7 bitová konstanta/proměnná, která je vysílána (0-127)

Popis:

***Příkaz je zastaralý, je vhodné použít příkaz*** [***irout***](http://www.picaxe.com/commands/irout)***.***

Vyšle data na pin C.0. Signál je modulovaný 38 kHz, v protokolu Sony SIRC.  
Signál je možné vyslat pomocí IR LED.

|  |  |  |
| --- | --- | --- |
|  | device | 5 bit zařízení ID (0-31) |
|  | data | 7 bit data (0-127) |

Více informací na u příkazu [irout](#irout).  
Příkaz pracuje správně pouze při frekvenci procesoru 4 MHz.

ID kódy zařízení Sony

|  |  |  |  |
| --- | --- | --- | --- |
| TV | 1 | VTR3 | 11 |
| VTR1 | 2 | Surround Sound | 12 |
| Text | 3 | Audio | 16 |
| Widescreen | 4 | CD Player | 17 |
| MDP / Laserdisk | 6 | Pro-Logic | 18 |
| VTR2 | 7 | DVD | 26 |

Kódy tlačítek na ovladači Sony TV (ID 01)

|  |  |
| --- | --- |
| 000 | 1 button |
| 001 | 2 button |
| 002 | 3 button |
| 003 | 4 button |
| 004 | 5 button |
| 005 | 6 button |
| 006 | 7 button |
| 007 | 8 button |
| 008 | 9 button |
| 009 | 10 button/0 button |
| 011 | Enter |
| 016 | channel up |
| 017 | channel down |
| 018 | volume up |
| 019 | volume down |
| 020 | Mute |
| 021 | Power |
| 022 | Reset TV |
| 023 | Audio Mode:Mono/SAP/Stereo |
| 024 | Picture up |
| 025 | Picture down |
| 026 | Color up |
| 027 | Color down |
| 030 | Brightness up |
| 031 | Brightness down |
| 032 | Hue up |
| 033 | Hue down |
| 034 | Sharpness up |
| 035 | Sharpness down |
| 036 | Select TV tuner |
| 038 | Balance Left |
| 039 | Balance Right |
| 041 | Surround on/off |
| 042 | Aux/Ant |
| 047 | Power off |
| 048 | Time display |
| 054 | Sleep Timer |
| 058 | Channel Display |
| 059 | Channel jump |
| 064 | Select Input Video1 |
| 065 | Select Input Video2 |
| 066 | Select Input Video3 |
| 074 | Noise Reduction on/off |
| 078 | Cable/Broadcast |
| 079 | Notch Filter on/off |
| 088 | PIP channel up |
| 089 | PIP channel down |
| 091 | PIP on |
| 092 | Freeze screen |
| 094 | PIP position |
| 095 | PIP swap |
| 096 | Guide |
| 097 | Video setup |
| 098 | Audio setup |
| 099 | Exit setup |
| 107 | Auto Program |
| 112 | Treble up |
| 113 | Treble down |
| 114 | Bass up |
| 115 | Bass down |
| 116 | + key |
| 117 | - key |
| 120 | Add channel |
| 121 | Delete channel |
| 125 | Trinitone on/off |
| 127 | Displays a red RtestS on the screen |

Kódy tlačítek na ovladači Sony VCR (ID 02 nebo 07)

|  |  |
| --- | --- |
| 000 | 1 button |
| 001 | 2 button |
| 002 | 3 button |
| 003 | 4 button |
| 004 | 5 button |
| 005 | 6 button |
| 006 | 7 button |
| 007 | 8 button |
| 008 | 9 button |
| 009 | 10 button/0 button |
| 010 | 11 button |
| 011 | 12 button |
| 012 | 13 button |
| 013 | 14 button |
| 020 | X 2 play w/sound |
| 021 | power |
| 022 | eject |
| 023 | L-CH/R-CH/Stereo |
| 024 | stop |
| 025 | pause |
| 026 | play |
| 027 | rewind |
| 028 | FF |
| 029 | record |
| 032 | pause engage |
| 035 | X 1/5 play |
| 040 | reverse visual scan |
| 041 | forward visual scan |
| 042 | TV/VTR |
| 045 | VTR from TV |
| 047 | power off |
| 048 | single frame reverse/slow reverse play |
| 049 | single frame advance/slow forward play |
| 060 | aux |
| 070 | counter reset |
| 078 | TV/VTR |
| 083 | index (scan) |
| 106 | edit play |
| 107 | mark |

Platí pro:

08M, 08M2, 14M, 14M2, 18M, 18M2, 20M, 20M2

Příbuzné:

* [irin](http://www.picaxe.com/commands/irin)
* [irout](http://www.picaxe.com/commands/irout)
* [infrain](http://www.picaxe.com/commands/infrain)
* [infrain2](http://www.picaxe.com/infrain2)

Příklad:

Vyslání signálu „tlačítko 6“, pro TV Sony.

for b1 = 1 to 10 ;signál se vyšle 10 krát

infraout 1,5

pause 45

next b1

Jméno:

pwm

Syntaxe:

PWM pin,duty,cycles

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Duty* - proměnná/konstanta (0-255) nastavující plnění - v tomto případě odpovídá pseudo analogovému výstupu

*Cycles* - proměnná/konstanta nastavující počet cyklů (0-255), jeden cyklus je 5 ms, při frekvenci 4 MHz.

Popis:

Přepne pin na výstupní, vyšle na něj pwm signál a po skončení jej přepne zpět na vstupní.

Příkaz je zastaralý, je vhodnější použít [pwmout](http://www.picaxe.com/commands/pwmout). Používal se pro generování pseudo analogového výstupu, kdy nabíjel kondenzátor, vybíjený přes rezistor.

Platí pro:

08, 08M, 08M2, 14M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [pwmout](http://www.picaxe.com/commands/pwmout)
* [hpwm](http://www.picaxe.com/commands/hpwm)

Příklad:

Generování PWM signálu na pinu C.4 po dobu 100 ms s 20 ms přestávkou.

main:

pwm C.4,150,20 ; signál na C.4, s plněním 150 po dobu 20 cyklů

pause 20 ; čekej 20 ms

goto main ; skok na „main“

Jméno:

switch

Syntaxe:

SWITCH ON pin, pin, pin...

SWITCHON pin, pin, pin...

SWITCH OFF pin, pin, pin...

SWITCHOFF pin, pin, pin...

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

Popis:

Nastaví na vybraných pinech vysokou (on - 1) nebo nízkou (off - 0) úroveň.

Jde o pseudo příkaz nahrazující příkazy [high](http://www.picaxe.com/commands/high) a [low](http://www.picaxe.com/commands/low).

Platí pro:

Vše

Příbuzné:

* [high](http://www.picaxe.com/commands/high)
* [low](http://www.picaxe.com/commands/low)

Příklad:

Blikání LED připojené na C.7 s periodou 10 s.

main:

switch on C.7 ; vysoká úroveň na C.7

wait 5 ; čekej 5 s

switch off 7 ; nízká úroveň na C.7

wait 5 ; čekej 5 s

goto main ; skok na „main“ - opakuj

Jméno:

button

Syntaxe:

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

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz. příloha - schéma zapojení)

*Downstate* - proměnná/konstanta (0/1) nastavuje která hodnota (0/1) bude považována jako stisknuté tlačítko.  
0 - přes 10k k 0V; 1 - přes 10k k +V

*Delay* - proměnná/konstanta (1-254, 0 nebo 255), prodleva před prvním opakováním tlačítka. Hodnota 255 zakáže opakování, hodnota 0 zakáže opakování i prodlevu.

*Rate* - proměnná/konstanta (0-255) určující rychlost následného opakování

*Proměnná* - proměnná typu byte, je pomocná proměnná příkazu, před použitím příkazu musí být vynulována

*Targetstate* - proměnná/konstanta (0/1) určuje, kdy se má skočit na adresu „address“.   
0 - skok když tlačítko není stisknuto, 1 - skok když je tlačítko stisknuto.

*Address* - adresa kam bude proveden skok

Popis:

Detekuje a reaguje na stisk tlačítka. Nastavitelným zpožděním eliminuje zákmity u mechanických spínačů. Umožňuje nastavení opakování klávesy, jako u PC klávesnice.

Po stisku/uvolnění provede skok na definovanou adresu.

Platí pro:

Vše

Příbuzné:

* [inputtype](http://www.picaxe.com/commands/inputtype)

Příklad:

Při stisku tlačítka na C.0, se rozsvítí LED na výstupu B.7 a pošle se zpráva na PC.

init:

b2 = 0 ; vynulování pomocné proměnné

; vstup na C.0, stisk +V, když je tlačítko stisknuto skok na „pushed“

myloop:

button C.0,1,200,100,b2,1,pushed ; skok, když je C.0 = 1

low B.7 ; vypnutí LED

pause 10 ; čekání

goto myloop ; opakování

pushed:

high B.7 ; zapnutí LED

sertxd (“PUSH”) ; odeslání zprávy

goto myloop ; skok na „myloop“

Jméno:

high portc

Syntaxe:

HIGH PORTC pin {,pin,pin...}

*Pin* - proměnná/konstanta (0-7) specifikuje, který pin portu C bude nastaven na „1“

Popis:

Nastaví požadovaný pin portu C na „1“.

Tento příkaz je určen pro starší modely 14M a 28X/28X1. Ostatní modely řady M2 a X2 používají jméno PORT.PIN => [high](http://www.picaxe.com/commands/high) C.2.

Platí pro:

14M, 28X, 28X1

Příbuzné:

* [high](http://www.picaxe.com/commands/high)

Příklad:

Blikání LED na pinu C.1 s periodou 10 s.

main:

high portc 1 ; na C.1 nastav 1

pause 5000 ; čekej 5 s

low portc 1 ; na C.1 nastav 0

pause 5000 ; čekej 5 s

goto main ; skok na „main“

Jméno:

low portc

Syntaxe:

LOW PORTC pin {,pin,pin...}

*Pin* - proměnná/konstanta (0-7) specifikuje, který pin portu C bude nastaven na „0“

Popis:

Nastaví požadovaný pin portu C na „0“.

Tento příkaz je určen pro starší modely 14M a 28X/28X1. Ostatní modely řady M2 a X2 používají jméno PORT.PIN => low C.2.

Platí pro:

14M, 28X, 28X1

Příbuzné:

* [low](http://www.picaxe.com/commands/high)

Příklad:

Blikání LED na pinu C.1 s periodou 10 s.

main:

high portc 1 ; na C.1 nastav 1

pause 5000 ; čekej 5 s

low portc 1 ; na C.1 nastav 0

pause 5000 ; čekej 5 s

goto main ; skok na „main“

Jméno:

readoutputs

Syntaxe:

READOUTPUTS proměnná

*proměnná* - proměnná typu byte

Popis:

Přečte hodnotu pouze výstupních pinů a uloží do proměnné.

Tím se liší od 'let var = pinsX', které načte stav všech vstupních pinů na portu.  
Příkaz je zastaralý u řad M2, X1, X2, se používá příkaz 'let var = outpinsX'

Platí pro:

Vše

Příbuzné:

* [readportc](http://www.picaxe.com/commands/readportc)

Příklad:

Načte hodnotu výstupních pinů do b1.

main:  
 readoutputs b1 ; načte hodnotu výstupních pinů do b1

Jméno:

readportc

Syntaxe:

READPORTC proměnná

*proměnná* - proměnná typu byte

Popis:

Načte hodnotu vstupních pinů na portu C a uloží do proměnné.

Příkaz určen pro načtení aktuálního stavu portu C u modelů 40X1, 28X1.  
U ostatních modelů se používá příkaz 'let var = pinsC'.

Platí pro:

28X1, 40X1

Příbuzné:

* [readoutputs](http://www.picaxe.com/commands/readoutputs)

Příklad:

Načte hodnotu vstupních pinů na portu C a uloží do proměnné b1.

main:

readportc b1 ; načtení portu C do b1

debug ; odeslání na PC

goto main ; opakování

Jméno:

touch

Syntaxe:

TOUCH pin, proměnná

Pin - proměnná/konstanta specifikující ADC pin - jméno pinu

Proměnná - typu byte, do které bude uložena načtená hodnota

Popis:

Načte hodnotu z kapacitního dotykového snímače na ADC pinu a uloží ji do proměnné.

POZOR, snímač musí být překrytý nevodivým materiálem, při přímém kontaktu by mohlo dojít k poškození procesoru.

Příkaz nastaví pin jako vstupní a „touch“. Jde o pseudo příkaz který použije příkaz [touch16](http://www.picaxe.com/commands/touch16) a výsledek upraví na bytovou (0-255) velkost, to umožní jednoduché vyhodnocení.  
Pro vyšší přesnost je vhodné použít příkaz [touch16](http://www.picaxe.com/commands/touch16), který ukládá výsledek do proměnné typu word.

Při kalibraci dotykových senzorů je vhodné nepoužívat sériový kabel, nebo použít speciální sériový kabel [AXE027 USB cable](http://www.picaxe.com/products/axe027). Každý pin se musí kalibrovat zvlášť.

Výsledek příkazu závisí na taktovací frekvenci, proto je třeba provést kalibraci při pracovní frekvenci.

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X2, 40X2

Příbuzné:

* [touch16](http://www.picaxe.com/commands/touch16)

Příklad:

Ovládání LED na B. 2 dotykovým senzorem na C.1.

main:

touch C.1,b0 ; načte hodnotu ze senzoru do b0

if b0 > 100 then ; vyhodnocení načtené hodnoty

high b.2 ; B.2 = 1

else

low b.2 ; B.2 = 0

endif

goto main ; skok na začátek

Jméno:

touch16

Syntaxe:

TOUCH16 pin, proměnná

TOUCH16 [config], pin, proměnná

Pin - proměnná/konstanta specifikuje pin s ADC a funkcí „touch“

proměnná - proměnná typu word pro uložení načtené hodnoty (10 bit pro X2)

Config - konfigurace obvodu „touch“ - nepovinná

Popis:

Načte hodnotu z kapacitního dotykového snímače na ADC pinu a uloží ji do proměnné typu word.

POZOR, snímač musí být překrytý nevodivým materiálem, při přímém kontaktu by mohlo dojít k poškození procesoru.

Příkaz nastaví pin jako vstupní a „touch“ (dotykový spínač) a načte hodnotu do proměnné typu word.

Kapacitní senzor pracuje na principu otevřeného oscilátoru, kdy přiblížením prstu dojde ke zvýšení kapacity a tím ke snížení frekvence.

Při kalibraci dotykových senzorů je vhodné nepoužívat sériový kabel, nebo použít speciální sériový kabel [AXE027 USB cable](http://www.picaxe.com/products/axe027). Každý pin se musí kalibrovat zvlášť.

Výsledek příkazu závisí na taktovací frekvenci, proto je třeba provést kalibraci při pracovní frekvenci.

Konfigurační byte pro M2 je konfigurován následovně:

|  |  |
| --- | --- |
| bit7,6,5 | = nastavení oscilátoru |
|  | = 000 počet kmitů = 256 |
|  | = 010 počet kmitů = 192 |
|  | = 100 počet kmitů = 128 |
|  | = 110 počet kmitů = 64 |
|  | = 111 počet kmitů = 32 |
| bit4,3 | = nastavení citlivosti |
|  | = 00 senzor vypnut |
|  | = 01 nízká citlivost (0.1uA) |
|  | = 10 střední citlivost (1.2uA) |
|  | = 11 vysoká citlivost (18uA) |
| bit2,1,0 | = dělič frekvence (2 - 256) |
|  | = 001 dělí 4 -mi |

Defaultní nastavení pro M2 je %000 01 001.

Konfigurační byte pro X2 je konfigurován následovně:

|  |  |
| --- | --- |
| bit7, 6 | = nepoužito |
| bit5,4 | = 00 senzor vypnut |
|  | = 01 nízká citlivost |
|  | = 10 střední citlivost |
|  | = 11 vysoká citlivost |
| bit 3,2,1,0 | = perioda oscilátoru v násobcích 2us . (1-15) |

Defaultní nastavení pro X2 je %0011 0010.

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X2, 40X2

Příbuzné:

* [touch](http://www.picaxe.com/commands/touch)

Příklad:

Ovládání LED na B. 2 dotykovým senzorem na C.1.

main:

touch16 C.1,w0 ; načte hodnotu z C.1 do w0

if w0 > 3000 then

high B.2 ; b.2 = 1

else

low B.2 ; b.2 = 0

endif

goto main ; skok na začátek

Jméno:

readadc

Syntaxe:

READADC pin,proměnná

Pin - proměnná/konstanta specifikující pin s ADC

Proměnná - typu byte, do které bude uložen výsledek

Popis:

Načte hodnotu z ADC (analogově digitálního převodníku) do 8 bitové proměnné.

Příkaz měří napětí na vybraném ADC pinu. Ne všechny piny obsahují ADC převodník, viz. příloha.

Platí pro:

Vše

Příbuzné:

* [readadc10](http://www.picaxe.com/commands/readadc10)

Příklad:

Změří napětí na C.1 a pokud je větší než 50, tak začne blikat LED na B.2.

main:

readadc C.1,b1 ; načte hodnotu z C.1 do proměnné b1

if b1 > 50 then flsh ; pokud je b1>50 skočí na „flsh“

goto main ; jinak skočí na „main“

flsh:

high B.2 ; rozsvítí LED na B.2

pause 1000 ; čekej 1 s

low B.2 ; zhasne LED na B.2

pause 1000 ; čekej 1 s

goto main ; skok na začátek

Jméno:

readadc10

Syntaxe:

READADC10 pin,proměnná

Pin - proměnná/konstanta specifikující pin s ADC

Proměnná - typu word, do které bude uložen 10-ti bitový výsledek

Popis:

Načte 10-ti bitovou hodnotu z ADC (analogově digitálního převodníku) do 16 bitové proměnné.

Příkaz měří napětí na vybraném ADC pinu. Ne všechny piny obsahují ADC převodník, viz. příloha.

Při přesném měření může docházet k ovlivnění sériovým kabelem.

Platí pro:

Vše (kromě 08, 18, 18A, 28, 28A, 28X, 40X)

Příklad:

Změří napětí na vývodu C.1 a výsledek pošle do PC.

main:

readadc10 C.1,w1 ; načte hodnotu z C.1 a uloží ji do w1

debug ; odešle hodnotu do PC

pause 200 ; čeká 0,2 s

goto main ; skok na začátek

Jméno:

readtemp

Syntaxe:

READTEMP pin,proměnná

Pin - vstupní pin.

Proměnná - typu byte pro uložení dat

Popis:

Přečte teplotu z digitálního teplotního čidla [DS18B20](http://www.picaxe.com/creates/ds18b20) a uloží ji do proměnné typu byte.

Převod trvá až 750ms, teplota je načtena v 12 bitovém přesnosti a následně zaokrouhlena na celé stupně, aby bylo možné výsledek uložit do proměnné typu byte.  
Teplota se měří v rozsahu -55 - +125°C, znaménko je vyjádřeno 7.bitem:   
0 - kladná, 1 - záporná.

Pro přesnější měření je třeba použít příkaz [readtemp12](http://www.picaxe.com/commands/readtemp12).

U senzoru DS18B20 musí být vždy připojen kontakt 5V.

Následující tabulka ukazuje, které piny není možno použít.

|  |  |
| --- | --- |
| 08M, 08M2 | C.3, C.0 |
| 14M, 14M2 | C.3, B.0 |
| 18M2 | C.3, C.4, C.5 |
| 20M, 20M2, 20X2 | C.6, A.0 |

Příkaz pracuje korektně při frekvenci 4 MHz.

Platí pro:

Vše (kromě 08, 18, 28)

Příbuzné:

* [readtemp12](http://www.picaxe.com/commands/readtemp12)
* [readinternaltemp](http://www.picaxe.com/commands/readinternaltemp)

Příklad:

Načte hodnotu z teplotního čidla DS18B20 na C.1 a pošle ji po sériové lince do LCD.

main:

readtemp C.1,b1 ; načte teplotu a uloží ji to b1

if b1 > 127 then neg ; zjistí zda je teplota záporná, pak skok

serout B.7,N2400,(#b1) ; pošle hodnotu do LCD

goto loop

neg:

let b1 = b1 - 128 ; zjistí zápornou teplotu

serout B.7,N2400,(“-”) ; pošle znak „-„

serout B.7,N2400,(#b1) ; odešle teplotu na LCD

goto main

Jméno:

readtemp12

Syntaxe:

READTEMP12 pin,proměnná

Pin - vstupní pin.

Proměnná - typu word pro uložení dat

Popis:

Přečte teplotu z digitálního teplotního čidla [DS18B20](http://www.picaxe.com/creates/ds18b20) a uloží ji do proměnné typu word.

Převod trvá až 750ms, teplota je načtena v 12 bitovém přesnosti a následně uložena do proměnné typu word, v násobcích 0,0625°C.  
Teplotu je nutno přepočítat dle dokumentace k [DS18B20](http://www.picaxe.com/creates/ds18b20).  
U senzoru DS18B20 musí být vždy připojen kontakt 5V.

Následující tabulka ukazuje, které piny není možno použít.

|  |  |
| --- | --- |
| 08M, 08M2 | C.3, C.0 |
| 14M, 14M2 | C.3, B.0 |
| 18M2 | C.3, C.4, C.5 |
| 20M, 20M2, 20X2 | C.6, A.0 |

Příkaz pracuje korektně při frekvenci 4 MHz.

Platí pro:

Vše (kromě 08, 18, 28)

Příbuzné:

* [readtemp](http://www.picaxe.com/commands/readtemp)
* [readinternaltemp](http://www.picaxe.com/commands/readinternaltemp)

Příklad:

Načte teplotu v 12-ti bitové přesnosti a výsledek pošle do PC.

main:

readtemp12 B.1,w1 ; načte teplotu z B.1 do w1

debug ; odešle do PC

goto main

Jméno:

daclevel

Syntaxe:

DACLEVEL level

Level - proměnná/konstanta (0-31), nastaví výstupní napětí na DAC

Popis:

Nastaví napětí na DAC (digitálně analogovém převodníku) ve 32 krocích (0-31).

Každý krok je 1/32 maximálního napětí nastaveného příkazem DACSETUP.  
Hodnotu napětí na pinu je možné přečíst příkazem [readdac](http://www.picaxe.com/commands/readdac).

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [dacsetup](http://www.picaxe.com/commands/dacsetup)
* [readdac](http://www.picaxe.com/commands/readdac)
* [readdac10](http://www.picaxe.com/commands/readdac10)

Set up the DAC and cycle through the output levels

Příklad:

Generuje pilový signál na pinu DAC.

init:

dacsetup %10100000 ; inicializace DAC

main:

for b1 = 0 to 31

daclevel b1 ; nastavení signálu

pause 1000

next b1

goto main ; skok na začátek

Jméno:

dacsetup

Syntaxe:

DACSETUP config

Config - proměnná/konstanta specifikující nastavení DAC

Popis:

Nastaví - nakonfiguruje DAC (digitálně analogový převodník), jeho referenční napětí.

Výstup z DAC dodává jen velmi malý proud, proto je určen pouze jako referenční zdroj a nehodí se pro přímé napájení spotřebičů včetně LED.  
Je vhodný pro použití s OZ (operačním zesilovačem), případně tranzistorem.

Funkce jednotlivých bitů, konfiguračního bytu:

|  |  |
| --- | --- |
| bit7 | = 0 DAC zakázáno |
|  | = 1 DAC povoleno |
| bit6 | = 0 nepoužívá se |
| bit5 | = 0 DAC pouze interní |
|  | = 1 DAC interní současně vyvedený na vnější DAC pin |
| bit4 | = 0 nepoužívá se |
| bit3-2 | = 00 DAC max. = napájecí napětí Vs |
|  | = 01 DAC max. = Vref (více příkaz adcconfig) |
|  | = 10 DAC max. = FVR napětí (více příkaz fvrsetup) |
|  | = 11 nepoužívá se |
| bit1 | = 0 nepoužívá se |
| bit0 | = 0 DAC min. = 0 V napájecí |
|  | = 1 DAC min. = Vref = 0 (více příkaz adcconfig) |

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [daclevel](http://www.picaxe.com/commands/daclevel)
* [readdac](http://www.picaxe.com/commands/readdac)
* [readdac10](http://www.picaxe.com/commands/readdac10)

Příklad:

Generátor pilového signálu.

init:

low DAC\_PIN ; nastavení DAC pinu do výstupní úrovně

dacsetup %10100000 ; nakonfigurování DAC

main:

for b1 = 0 to 31

daclevel b1 ; generování pilového signálu

pause 1000

next b1

goto main ; skok na „main“

Jméno:

readdac

Syntaxe:

READDAC proměnná

Proměnná - typu byte, do níž je uložena hodnota napětí na DAC

Popis:

Přečte hodnotu napětí na pin DAC. Musí následovat až po příkazech dacsetup a daclevel.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [readdac10](http://www.picaxe.com/commands/readdac10)
* [daclevel](http://www.picaxe.com/commands/daclevel)
* [dacsetup](http://www.picaxe.com/commands/dacsetup)

Příklad:

Načte velikost napětí na DAC pinu a pošle jej do PC.

main:

readdac b1 ; načtení napětí z DAC do b1

debug ; odeslání do PC

goto main ; skok na začátek

Jméno:

readdac10

Syntaxe:

READDAC10 proměnná

Proměnná - proměnná typu word, do níž je uložena hodnota napětí na DAC

Popis:

Načte napětí na pinu DAC v 10 bitové přesnosti a hodnotu uloží do proměnné typu word. Příkazu musí předcházet příkazy: [dacsetup](http://www.picaxe.com/commands/dacsetup) a [daclevel](http://www.picaxe.com/commands/daclevel).

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [readdac](http://www.picaxe.com/commands/readdac)
* [daclevel](http://www.picaxe.com/commands/daclevel)
* [dacsetup](http://www.picaxe.com/commands/dacsetup)

Příklad:

Načte hodnotu napětí na DAC v 10bitovém rozlišení a pošle na PC.

main:

readdac10 w1 ; načte velikost napětí z DAC a uloží do w1

debug ; odešle do PC

goto main ; skok na začátek

Jméno:

pause

Syntaxe:

PAUSE milliseconds

Milliseconds - proměnná/konstanta určující časový úsek v milisekundách (0-65535) při frekvenci 8 MHz u X2 a 4 MHz u všech ostatních

Popis:

Čekání - pozastaví činnost na požadovanou dobu.

Doba čekání je odvozena od frekvence procesoru, proto při zvýšení frekvence dojde ke zkrácení čekání.  
V době čekání lze na vstupy reagovat pouze prostřednictvím přerušení - [setint](http://www.picaxe.com/commands/setint).

U modelů M2, umožňujících paralelní zpracovávání více procesů, může být časová nestabilita až 20 ms.

Platí pro:

Vše

Příbuzné:

* [pauseus](http://www.picaxe.com/commands/pauseus)
* [wait](http://www.picaxe.com/commands/wait)

Příklad:

Blikání LED s periodou 10 s.

main:

high B.1 ; rozsvícení LED na B.1

pause 5000 ; čekání 5 s

low B.1 ; zhasnutí LED na B.1

pause 5000 ; čekání 5 s

goto main ; skok na začátek

Jméno:

pauseus

Syntaxe:

PAUSEUS microseconds

Microseconds - proměnná/konstanta určující časový úsek v násobcích 10 us  
(0-65535) při frekvenci 8 MHz u X2 a 4 MHz u všech ostatních

Popis:

Čekání - pozastaví činnost na požadovanou dobu.

Doba čekání je odvozena od frekvence procesoru, proto při zvýšení frekvence dojde ke zkrácení čekání. Parametr udává počet násobků 10 us.  
V době čekání lze na vstupy reagovat pouze prostřednictvím přerušení - [setint](http://www.picaxe.com/commands/setint).

U velmi krátkých časů může dojít k časové nestabilitě z důvodu „režie zpracování“.

Platí pro:

Všechny řady M2, X1, X2

Příbuzné:

* [pause](http://www.picaxe.com/commands/pause)

Příklad:

Blikání LED s periodou 100 ms.

main:

high B.1 ; rozsvícení LED na B.1

pauseus 5000 ; čekání 5000.10 us = 50000 us = 50 ms

low B.1 ; zhasnutí LED na B.1

pauseus 5000 ; čekání 5000.10 us = 50000 us = 50 ms

goto main ; skok na začátek

Jméno:

nap

Syntaxe:

NAP period

Period - proměnná/konstanta obvykle 0-7, u řady M2 0 - 14, určující časový údaj, podle následující tabulky:

|  |  |
| --- | --- |
| **Period** | **Time Delay** |
| 0 | 18ms |
| 1 | 32ms |
| 2 | 72ms |
| 3 | 144ms |
| 4 | 288ms |
| 5 | 576ms |
| 6 | 1.1s |
| 7 | 2.3s |
| 8 | 4s |
| 9 | 8s |
| 10 | 16s |
| 11 | 32s |
| 12 | 64s (1 min) |
| 13 | 128s (2 mins) |
| 14 | 256s (4 mins) |

Popis:

Nap - převede na krátkou (požadovanou) dobu procesor do stavu snížené spotřeby energie.

Při snížení spotřeby dojde k vypnutí všech časovačů (pwmout, servo…), ty přestanou pracovat. Časová tolerance je závislá na vnějších podmínkách i procesoru samotné a činí maximálně -50% - +100%.

Příkaz je nezávislý na taktovací frekvenci, využívá samostatný oscilátor watchdog.  
Pro delší přechod do úsporného režimu se používá příkaz [sleep](http://www.picaxe.com/commands/sleep).

Platí pro:

Vše kromě 20X2, 28X2, 40X2

Příbuzné:

* [sleep](http://www.picaxe.com/commands/sleep)
* [doze](http://www.picaxe.com/commands/doze)
* [hibernate](http://www.picaxe.com/commands/hibernate)
* [pause](http://www.picaxe.com/commands/pause)

Příklad:

Blikání LED v úsporném režimu.

main:

high B.1 ; rozsvícení LED na B.1

nap 4 ; přechod do úsporného režimu na dobu 288ms

low B.1 ; zhasnutí LED na B.1

nap 7 ; přechod do úsporného režimu na dobu 2.3 s

goto main ; skok na začátek

Jméno:

sleep

Syntaxe:

SLEEP period

Period - proměnná/konstanta určující časový údaj v násobcích 2.3 s (1-65535).

Popis:

Převede procesor do režimu spánku na požadovanou dobu, určenou násobky přibližně 2.3 s. (u X1/X2 násobky 2.1 s).

Příkaz sleep přepne procesor na určitou dobu do režimu nízké spotřeby.  
Současně dojde k ukončení činnost časovačů - pwmout, servo.  
Časová tolerance je závislá na vnějších podmínkách i procesoru samotné a činí maximálně -50% - +100%. Čas je udávám v násobcích přibližně 2.3 s. (X1/X2 - 2.1 s)  
Příkaz je nezávislý na taktovací frekvenci, využívá samostatný oscilátor watchdog.  
  
Kratší dobu „spánku“ je možné vyvolat příkazem nap.  
Ještě větší úspory energie je možné dosáhnout použitím příkazů [disablebod](http://www.picaxe.com/commands/disablebod)/[enablebod](http://www.picaxe.com/commands/enablebod).  
Kromě X2 je příkaz 'sleep 0' ignorován. U X2 je procesor trvale uveden do spánku, činnost je obnovitelná jen hardwarovým přerušením.

Platí pro:

Vše

Příbuzné:

* [nap](http://www.picaxe.com/commands/nap)
* [doze](http://www.picaxe.com/commands/doze)
* [hibernate](http://www.picaxe.com/commands/hibernate)
* [pause](http://www.picaxe.com/commands/pause)

Příklad:

Blikání LED ve spánkovém režimu.

main:

high B.1 ; rozsvícení LED na B.1

sleep 10 ; přechod do spánkového režimu na 23 s

low 1 ; zhasnutí LED na B.1

sleep 100 ; přechod do spánkového režimu na 230 s

goto main ; skok na začátek

Jméno:

doze

Syntaxe:

DOZE period

Period - proměnná/konstanta určující čas v násobcích přibližně 2.1 s

Popis:

Doze - dřímot, převede procesor do stavu nízké spotřeby, přičemž funkce časovačů je zachována, tím i činnost příkazů pwmout, servo.

Doba, po kterou má být procesor ve stavu nížené spotřeby je stanovena v násobcích 2.1 s. Časová tolerance je závislá na vnějších podmínkách i procesoru samotné a činí maximálně -50% - +100%.  
Při zadání 'doze 0' přejde procesor do trvalé „dřímoty“, činnost je možné obnovit hardwarovým přerušením.

Příkaz je nezávislý na frekvenci procesoru

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [nap](http://www.picaxe.com/commands/nap)
* [sleep](http://www.picaxe.com/commands/sleep)
* [hibernate](http://www.picaxe.com/commands/hibernate)
* [pause](http://www.picaxe.com/commands/pause)

Příklad:

Blikání LED s přechodem do úsporného režimu - „dřímotu“.

main:

high B.1 ; rozsviť LED na B.1

doze 1 ; „dřímot“ po dobu 2.1 s

low B.1 ; zhasni LED na B.1

doze 1 ; „dřímot“ po dobu 2.1 s

goto main ; loop back to start

Jméno:

wait

Syntaxe:

WAIT seconds

Seconds - konstanta (1-65) určující čas v sekundách

Popis:

Čeká po stanovenou dobu.

Jedná se o pseudo příkaz, je ekvivalentní '[pause](http://www.picaxe.com/commands/pause) \* 1000'.  
Parametrem může být pouze konstanta.  
Doporučuje se používání příkazu [pause](http://www.picaxe.com/commands/pause).

Platí pro:

Vše

Příbuzné:

* [pause](http://www.picaxe.com/commands/pause)
* [sleep](http://www.picaxe.com/commands/sleep)

Příklad:

main:

switch on B.7 ; rozsviť LED na B.7

wait 5 ; čekej 5 s

switch off B.7 ; zhasni LED na B.7

wait 5 ; čekej 5 s

goto main ; skok na začátek

Jméno:

setint

Syntaxe:

|  |  |
| --- | --- |
| SETINT OFF |  |
| SETINT input,mask | (AND se stávajícím stavem) |
| SETINT AND input,mask | (AND se stávajícím stavem) |

Další možnosti pro M2, X1, X2

|  |  |
| --- | --- |
| SETINT OR input,mask | (OR se stávajícím stavem) |
| SETINT NOT input,mask | (NAND se stávajícím stavem) |
| SETINT input,mask,port | (pouze X2) |
| SETINT NOT input,mask,port | (pouze X2) |

Input - proměnná/konstanta (0-255), určuje jaká má být hodnota na konkrétním vstupním pinu, při které dojde k přerušení

Mask - proměnná/konstanta (0-255), určuje masku = hodnota, která určuje, které piny mají být vyhodnocovány

Port - pro X2 určuje port (A,B,C,D)  

Popis:

Nastavuje vnější přerušení.

Při požadované hodnotě na vstupních pinech portu C, přeruší běh programu a skočí do podprogramu (gosub) na návěští „interrupt“. Po ošetření přerušení se vrátí zpět do hlavního programu. Pozor při skoku se neukládají žádné proměnné z hlavního programu.   
U řady X2 je možné vybrat sledovaný port, který vyvolá přerušení - port.

Defaultně musí být splněny všechny podmínky na všech pinech (AND), u M2 je možno použit funkci OR = stačí, aby byla splněna podmínka na jednom z hlídaných pinu a funkci NOT z AND = aspoň jedna podmínka nesmí být splněna.

Nastavení přerušení je nejrychlejší způsob, jak reagovat na požadované kombinace na vstupních pinech. Dotaz na piny se provádí po každém příkazu, taktu melodie a stále během příkazu wait.

Příklady nastavení proměnné input a mask:  
  
- vyvolání přerušení, pokud je C.1 = 1: setint %00000010,%00000010  
  
- vyvolání přerušení, pokud je C.1 = 0: setint %00000000,%00000010  
  
- vyvolání přerušení, pokud je C.0 = 1 AND C.1 = 1 AND C.2 = 0:  
 (AND = a zároveň => přerušení se vyvolá, jen když jsou splněny všechny podmínky) setint %00000011,%00000111  
  
Pro M2 je možno použít i následující možnosti:   
  
- vyvolání přerušení, pokud je C.0 = 1 OR C.1 = 1 OR C.2 = 0:  
 (OR = nebo => přerušení se vyvolá, pokud je splněna aspoň jedna podmínka) setint OR %00000011,%00000111  
  
- vyvolání přerušení, pokud je NOT(C.0 = 1 AND C.1 = 1 AND C.2 = 0):  
 (NOT z AND = není aspoň jeden => přerušení se vyvolá, pokud aspoň jedna podmínka není splněna) setint NOT %00000011,%00000111  
  
K ukončení přerušení slouží příkaz SETINT OFF.

Omezení

U některých modelů jsou omezení ve využitelných pinech. Defaultní port je port C.

|  |  |
| --- | --- |
| 14M/14M2 | pouze vstupy 0,1,2 |
| 20M | pouze vstupy 1-5 |
| 20M2/20X2 | pouze vstupy C.1 - C.5 |
| 40X2 | při použití portu mohou být použity pouze piny A.0 - A.3 |

Poznámky k obsluze přerušení

1) Přerušení vyvolané příkazem SETINT musí mít odpovídající obsluhu - proceduru, která přerušení vyhodnotí (obslouží) a je ukončena příkazem [return](http://www.picaxe.com/commands/return).

2) Pokud má být přerušení prováděnou opakovaně, je třeba ho v rámci obsluhy přerušení znovu povolit - nastavit.

3) Nedojde-li k vynulování přerušení - zrušení vstupního stavu, a současně k opětovnému povolení přerušení, může dojít k opakovanému vyvolání hned po provedení příkazu [return](http://www.picaxe.com/commands/return).

4) Po návratu z přerušení pokračuje program dalším řádkem = může dojít ke zkrácení příkazů: [pause](http://www.picaxe.com/commands/pause), [wait](http://www.picaxe.com/commands/wait), [play](http://www.picaxe.com/commands/play) nebo [tune](http://www.picaxe.com/commands/tune).

Platí pro:

Vše (kromě 08, 18, 28)

Příbuzné:

* [hintsetup](http://www.picaxe.com/commands/hintsetup)
* [setintflags](http://www.picaxe.com/commands/setintflags)
* [pullup](http://www.picaxe.com/commands/pullup)

Příklad:

Ukázka správné obsluhy přerušení, po dobu kdy C.7 = 1, bude svítit LED na B.1.

setint %10000000,%10000000,C ; nastavení vstupních pinů a masky na C.7=1

main: ; hlavní program

low B.1 ; vypne LED na B.1

pause 2000 ; čeká 2 s

goto main ; skok na začátek hlavního programu

interrupt: ; obsluha přerušení

high B.1 ; rozsvítí LED na B.1

if pinC.7 = 1 then interrupt ; opakuje, dokud důvod přerušení   
; nepomine

pause 2000 ; čeká 2 s

setint %10000000,%10000000.C ; znovu nastaví přerušení

return ; ukončí ošetření přerušení a vrátí  
; řízení hlavnímu programu

Jméno:

setintflags

Syntaxe:

SETINTFLAGS OFF

SETINTFLAGS flags,mask

SETINTFLAGS AND flags,mask

SETINTFLAGS OR flags,mask

SETINTFLAGS NOT flags,mask

Flags - proměnná/konstanta (0-255), určuje jaká má být hodnota na konkrétních bitech stavového bytu, při které dojde k přerušení

Mask - proměnná/konstanta (0-255), určuje masku = hodnota, která určuje, které piny mají být vyhodnocovány

Popis:

Nastavuje vnitřní přerušení.

Při požadované hodnotě na stavovém bytu - flagu, dojde k přerušení běhu programu   
a skoku do podprogramu (gosub) na návěští „interrupt“. Po ošetření přerušení se vrátí zpět do hlavního programu. Pozor při skoku se neukládají žádné proměnné z hlavního programu.   
U řady X2 je možné vybrat sledovaný port, který vyvolá přerušení - port.

POZOR - na PICAXE může být aktivní přerušení jen jedno, buď vnější „setint“, nebo vnitřní „setnitflags“  
  
Defaultně musí být splněny všechny podmínky na všech pinech (AND), u M2 je možno použit funkci OR = stačí, aby byla splněna podmínka na jednom z hlídaných pinu a funkci NOT z AND = aspoň jedna podmínka nesmí být splněna.

Nastavení přerušení je nejrychlejší způsob, jak reagovat na požadované kombinace na stavovém bytu. Dotaz na piny se provádí po každém příkazu, taktu melodie a stále během příkazu wait.

Příklady nastavení proměnné flags a mask:  
  
- vyvolání přerušení, pokud je C.1 = 1: setintflags %00000010,%00000010  
  
- vyvolání přerušení, pokud je C.1 = 0: setintflags %00000000,%00000010

- vyvolání přerušení, pokud je C.0 = 1 AND C.1 = 1 AND C.2 = 0:  
 (AND = a zároveň => přerušení se vyvolá, jen když jsou splněny všechny podmínky) setintflags %00000011,%00000111  
  
  
- vyvolání přerušení, pokud je C.0 = 1 OR C.1 = 1 OR C.2 = 0:  
 (OR = nebo => přerušení se vyvolá, pokud je splněna aspoň jedna podmínka) setintflags OR %00000011,%00000111  
  
- vyvolání přerušení, pokud je NOT(C.0 = 1 AND C.1 = 1 AND C.2 = 0):  
 (NOT z AND = není aspoň jeden => přerušení se vyvolá, pokud aspoň jedna podmínka není splněna) setintflags NOT %00000011,%00000111  
  
K ukončení přerušení slouží příkaz SETINTFLAGS OFF.

Stavový byte - flag je složen z následujících bitů:

|  |  |  |
| --- | --- | --- |
| **Jméno bitu** | **Jméno funkce** | **Funkce** |
| flag0 | hint0flag | pro X2 - přerušení na INT0 |
| flag1 | hint1flag | pro X2 - přerušení na INT1 |
| flag2 | hint2flag | pro X2 - přerušení na INT2 |
| flag3 | hintflag | pro X2 - přerušení na INT0 OR INT1 OR INT2 |
| flag4 | compflag | pro X2 - přetečení komparátoru |
| flag5 | hserflag | hserial - žádost o přijetí dat po sériové lince - hsersetup |
| flag6 | hi2cflag | žádost o zápis po hi2c ve slave módu  - hi2csetup |
| flag7 | toflag | timer - přetečení časovače - settimer |

Ukázka generování vnitřního přerušení:

|  |  |
| --- | --- |
| setintflags: %10000000,%10000000 | při přetečení časovače |
| setintflags %01000000,%01000000 | při hi2c zápisu (slave mód) |
| setintflags %00100000,%00100000 | při sériovém příjmu |

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [setint](http://www.picaxe.com/commands/setint)
* [hintsetup](http://www.picaxe.com/commands/hintsetup)

Příklad:

Nastavení přerušení při přetečení časovače.

setintflags %10000000,%10000000 ; nastavení přerušení

Jméno:

restart

Syntaxe:

RESTART task

Task - proměnná/konstanta určující číslo procesu

Popis:

Příkaz restartuje proces s číslem „task“.

Řada M2 umožňuje zpracovávat paralelně více procesů značených: start0, start1 … Při použití příkazu dojde k resetu zvoleného resetu = skok na 1. řádek zvoleného procesu. Nedochází k uložení, nebo vynulování proměnných. Pozastavené procesy budou obnoveny, ostatní procesy běží beze změny. Pro obnovení celkové činnosti procesoru se používá příkaz reset.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [resume](http://www.picaxe.com/commands/resume)
* [suspend](http://www.picaxe.com/commands/suspend)

Příklad:

Restartování procesu „0“ každých 10 s, procesem „1“.

start0: ; proces „0“

b3 = 0 ; nastavení proměné b3

loop0:

high B.0 ; B.0 = 1

pause 1000 ; čekej 1 s

low B.0 ; B.0 = 0

pause 1000 ; čekej 1 s

inc b3 ; zvětši o 1 proměnnou b3

goto loop0 ; skok na začátek procesu „0“

start1: ; proces „1“

inc b4 ; zvětši o 1 proměnnou b.4

if b4 > 10 then ; když je b4 > 10 pak

restart 0 ; restart procesu „0“ a vynulování b4

b4 = 0

end if

debug ; odeslání výpisu proměnných na PC

pause 1000

goto start1 ; skok na začátek procesu „1“

Jméno:

resume

Syntaxe:

RESUME task

Task - proměnná/konstanta označující číslo procesu

Popis:

Příkaz obnoví činnost zvoleného procesu.

Řada M2 umožňuje zpracovávat paralelně více procesů značených: start0, start1 … Při použití příkazu dojde k obnovení činnosti procesu, který byl dříve pozastaven příkazem suspend. Ostatní procesy běží beze změny. Pokud už proces běží, je příkaz ignorován.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [restart](http://www.picaxe.com/commands/restart)
* [suspend](http://www.picaxe.com/commands/suspend)

Příklad:

Proces „1“ pozastavuje a obnovuje činnost procesu „0“

start0: ; začátek procesu „0“

high B.0 ; rozsvícení LED na B.0

pause 100 ; čekání 0.1 s

low B.0 ; zhasnutí LED na B.0

pause 100 ; čekání 0.1 s

goto start0 ; skok na začátek procesu „0“

start1: ; začátek procesu „1“

pause 5000 ; čekání 5 s

suspend 0 ; pozastavení činnosti procesu „0“

pause 5000 ; čekání 5 s

resume 0 ; obnovení činnosti procesu „0“

goto start1 ; skok na začátek procesu „1“

Jméno:

suspend

Syntaxe:

SUSPEND task

Task - proměnná/konstanta určující číslo procesu

Popis:

Pozastaví zvolený proces.

Řada M2 umožňuje zpracovávat paralelně více procesů značených: start0, start1 …   
Při použití příkazu dojde k pozastavení činnosti procesu, ten může následně pokračovat použitím příkazu [resume](http://www.picaxe.com/commands/resume).   
Pokud je už proces pozastaven, je příkaz ignorován.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [restart](http://www.picaxe.com/commands/restart)
* [resume](http://www.picaxe.com/commands/resume)

Příklad:

Proces „1“ pozastavuje a obnovuje činnost procesu „0“

start0: ; začátek procesu „0“

high B.0 ; rozsvícení LED na B.0

pause 100 ; čekání 0.1 s

low B.0 ; zhasnutí LED na B.0

pause 100 ; čekání 0.1 s

goto start0 ; skok na začátek procesu „0“

start1: ; začátek procesu „1“

pause 5000 ; čekání 5 s

suspend 0 ; pozastavení činnosti procesu „0“

pause 5000 ; čekání 5 s

resume 0 ; obnovení činnosti procesu „0“

goto start1 ; skok na začátek procesu „1“

Jméno:

symbol

Syntaxe:

SYMBOL symbolname = value

SYMBOL symbolname = value ?? konstanta

Symbolname - textový řetězec začínající písmenem nebo podtržítkem

Value - konstanta nebo proměnná, která je nahrazena textovým řetězcem

?? - může být matematická operace, např. +, - , \*, /…

Popis:

Přiradí hodnotu symbolu, nebo vytvoří přesměrování na proměnnou.

Matematické operátory je možné použít jen na konstanty.  
Symboly zpřehledňují strukturu programu a usnadňují jeho změnu.   
Symboly nemají vliv na délku programu.  
Symboly mohou ve jméně obsahovat číselné znaky, ale nesmějí jimi začínat. Samozřejmě nemohou být stejná, jako vyhrazená slova - příkazy. Pro vstupy a výstupy nepoužívat jméno pin0, …

Platí pro:

Vše

Příbuzné:

* [let](http://www.picaxe.com/commands/let)
* [#define](http://www.picaxe.com/commands/hash-define)

Příklad:

Použití symbolu pro výstupní pin, vstupní pin a proměnnou.

symbol RED\_LED = B.7 ; definice výstupního pinu

symbol PUSH\_SW = pinC.1 ; definice vstupního pinu

symbol DELAY = b0 ; definice proměnné

let DELAY = 200 ; přiřadí do DELAY (b0) hodnotu 200

main:

high RED\_LED ; rozsvítí RED\_LED - B.7 = 1

pause DELAY ; čeká 0.2 s

low RED\_LED ; zhasne RED\_LED - B.7 = 0

pause DELAY ; čeká 0.2 s

goto main ; skok na začátek programu

Jméno:

let

Syntaxe:

{LET} proměnná = {-} value ?? value ...

Proměnná - proměnná do které bude přiřazováno (pro M2: b0 - b27, w0 - w14)

Value(s) - proměnná/konstanta, která má být uložena

{LET} dirs = value

{LET} dirsA = value

{LET} dirsB = value

{LET} dirsC = value

{LET} dirsD = value

Value(s) - proměnná/konstanta, která konfiguruje piny konkrétního portu jako:  
 1 - vstupní, 0 - výstupní

**{LET} pins = value**

**{LET} pinsA = value**

**{LET} pinsB = value**

**{LET} pinsC = value**

**{LET} pinsD = value**

Value(s) proměnné/konstanty, nastavující hodnotu jednotlivých pinů na konkrétním portu

Popis:

Přiřadí do proměnné „proměnná“ výsledek operace za rovnítkem, nebo přiřadí portu hodnotu určenou hodnotou „value“. Vždy musí být zachována datová šířka na obou stranách rovnítka (byte = byte; word = word).   
Klíčové slovo „let“ je nepovinné.   
  
POZOR - matematické operace se provádějí zásadně zleva - doprava.  
Výsledek operace je vždy celočíselný v rozsahu word, nebo byte.

Podporované matematické operátory a funkce

|  |  |
| --- | --- |
| + | sčítání |
| - | odčítání |
| \* | násobení (nižší byty výsledku) |
| \*\* | násobení (vyšší byty výsledku) |
| / | celočíselné dělení |
| // (or %) | modulo = zbytek po celočíselném dělení |
| MAX | omezí výsledek shora na maximální hodnotu |
| MIN | omezí výsledek zdola na minimální hodnotu |
| AND (or &) | bitový operátor AND, provede log. součin mezi jednotlivými bity |
| OR (or |) | bitový operátor OR, provede log. součet mezi jednotlivými bity |
| XOR | bitový operátor XOR, provede exkluzivní disjunkci mezi jedn. bity |
| NAND | bitový operátor NAND, provede negovaný součin mezi jedn. bity |
| NOR | bitový operátor NOR, provede negovaný součet mezi jedn. bity |
| ANDNOT (or &/) | bitový operátor AND NOT, provede logický součin s druhým operátorem negovaným (nerovná se NAND) |
| ORNOT (or |/) | bitový operátor OR NOT, provede logický součet s druhým operátorem negovaným (nerovná se NOR) |
| XNOR (or ^/) | bitový operátor XOR NOT, provede exkluzivní disjunkci s druhým operátorem negovaným (nerovná se XOR) |

Řada X1 a X2 také podporuje:

|  |  |  |
| --- | --- | --- |
| << | bitové posunutí doleva | |
| >> | bitové posunutí doprava | |
| \*/ | dělení (vrací prostřední část výsledku) | |
| SIN | | sinus, argument ve stupních (0 to 65535), vrací výsledek\*100 | |
| COS | | cosinus, argument ve stupních (0 to 65535), vrací výsledek\*100 | |
| SQR | | druhá odmocnina | |
| INV | | invertování | |
| NCD | | NCD n, je totožné jako - celá část z druhé odmocniny n | |
| DCD | | DCD n, je totožné jako - 2n | |
| BINTOBCD | | převod z binárního čísla do BCD kódu | |
| BCDTOBIN | | převod z BCD kódu na binární číslo | |
| REV | | upraví šířku dat | |
| DIG | | vrátí BCD číslice | |

LET dirs / dirsA / dirsB / dirsC / dirsD = value

Pro práci s vnějšími piny je třeba nastavit zda jde o pin vstupní, či výstupní. Je mnoho způsobu (input, outpur, high, low…) nebo je možné použít příkaz LET dirs.   
Platí, že při nastavení je 1 = vstupní pin a 0 = výstupní pin.

***LET pins / pinsA / pinsB / pinsC / pinsD = value***

Při práci s vnějšími výstupními piny je třeba nastavit jejich výstup. Je možné je nastavovat pomocí high, low, apod. nebo je možné použít příkaz LET pins. Před jeho použitím je třeba nakonfigurovat pin jako výstupní.   
Platí, pokud bychom pracovali s portem C (LET pinsC), že:  
0.bit = C.0, 1.bit = C.1, 2.bit = C.2,…, dále platí, že high = 1, low = 0.

Platí pro:

Vše

Příbuzné:

* [symbol](http://www.picaxe.com/commands/symbol)
* [random](http://www.picaxe.com/commands/random)
* [swap](http://www.picaxe.com/commands/swap)

Příklad:

Zvyšující se signál, s demonstrací matematických operací.

main:

inc b0 ; zvýšení b0 o +1

sound B.7,(b0,50) ; tón na výstup B.7

if b0 > 50 then rest ; když je b0 větší než 50 odskok na „rest“

goto main ; opakování hlavního programu

rest:

let b0 = b0 max 10 ; omezení horní hranice obsahu proměnné b.0

goto main ; návrat do hlavního programu

Jméno:

inc

Syntaxe:

INC var

Var - proměnná

Popis:

Provede inkrementaci = přičte k proměnné +1.   
Operace je rychlejší než 'let var = var + 1'

Platí pro:

Vše

Příbuzné:

* [dec](http://www.picaxe.com/commands/dec)
* [let](http://www.picaxe.com/commands/let)

Příklad:

Ukázka použití příkazu inc.

b2 = 0   
for b1 = 1 to 10 step 2

inc b2 ; 5 krát se zvětší proměnná b2

next b1

Jméno:

dec

Syntaxe:

DEC var

Var - proměnná

Popis:

Provede dekrementaci = odečte od proměnné -1  
Operace je rychlejší než 'let var = var - 1'

Platí pro:

Vše

Příbuzné:

* [inc](http://www.picaxe.com/commands/inc)
* [let](http://www.picaxe.com/commands/let)

Příklad:

Ukázka použití příkazu dec.

b2 = 10   
for b1 = 1 to 10 step 2

dec b2 ; 5 krát se zmenší proměnná b2

next b1

Jméno:

swap

Syntaxe:

SWAP proměnná1, proměnná2

Popis:

Vzájemně vymění obsah proměnných.

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [let](http://www.picaxe.com/commands/let)

Příklad:

Opakovaně vyměňuje obsah proměnných b1 a b2, výsledek odesílá na PC.

b1 = 5 ; počáteční inicializace

b2 = 10

main:

swap b1,b2 ; výměna obsahu proměnných

debug ; odeslání na PC

pause 1000 ; čekání 1 s

goto main ; opakování programu

Jméno:

bcdtoascii

Syntaxe:

BCDTOASCII proměnná, tens, units

BCDTOASCII wordproměnná, thousands, hundreds, tens, units

Proměnná - proměnná typu byte s hodnotou 0-99, v BCD kódu

Wordproměnná - proměnná typu word s hodnotou 0-9999, v BCD kódu

Thousands - proměnná do které bude uložen ASCII kód číslice v řádu tisíců

Hundreds - proměnná do které bude uložen ASCII kód číslice v řádu stovek

Tens - proměnná do které bude uložen ASCII kód číslice v řádu desítek

Units - proměnná do které bude uložen ASCII kód číslice v řádu jednotek

Popis:

Převede číslo z BCD kódu do ASCII kódu, každý řád je v jednom bytu.

Platí pro:

Vše

Příbuzné:

* [bintoascii](http://www.picaxe.com/commands/bintoascii)

Příklad:

Převod čísla z BCD do ASCII a výstup na PC.

main:

inc b1 ; zvýší hodnotu b1

bcdtoascii b1,b2,b3 ; převede z BCD na ASCII

debug ; odešle na PC

goto main ; skočí na začátek

Pozn. V příkladu není ošetřeno překročení rozsahu BCD => po 9 jsou generovány následující znaky z ASCII tabulky.

Jméno:

bintoascii

Syntaxe:

BINTOASCII proměnná, hundreds, tens, units

BINTOASCII wordproměnná, tenthousands, thousands, hundreds, tens, units

Proměnná - proměnná typu byte (0-255)

Wordproměnná - proměnná typu word (0-65535)

TenThousands - proměnná do které bude uložen ASCII kód číslice v ř. desetitisíců

Thousands - proměnná do které bude uložen ASCII kód číslice v řádu tisíců

Hundreds - proměnná do které bude uložen ASCII kód číslice v řádu stovek

Tens - proměnná do které bude uložen ASCII kód číslice v řádu desítek

Units - proměnná do které bude uložen ASCII kód číslice v řádu jednotek

Popis:

Převod binární hodnoty do samostatných ASCII bytů. V každé proměnné je uložena číslice jednoho řádu.

Platí pro:

Vše

Příbuzné:

* [bcdtoascii](http://www.picaxe.com/commands/bcdtoascii)

Příklad:

Převod čísla do ASCII a výstup na PC.

main:

inc b1 ; zvýší hodnotu b1

bintoascii b1,b2,b3 ; převede číslo na ASCII

debug ; odešle na PC

goto main ; skočí na začátek

Jméno:

lookdown

Syntaxe:

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

Target - proměnná/konstanta která bude vyhledávána

Values - proměnné/konstanty mezi kterými bude vyhledáváno, může být zadán i řetězec v uvozovkách - každé písmeno je považováno za konstantu typu byte

Proměnná - proměnná, do níž je uložena pozice hledané hodnoty, není-li nalezena, hodnota se nezmění

Popis:

Uloží pořadové číslo hledané hodnoty, pokud hodnota byla nalezena. Počítá se od 0-té pozice => (0-N).

Platí pro:

Vše

Příbuzné:

* [lookup](http://www.picaxe.com/commands/lookup)

Příklad:

Pokud se b1 = „d“, pak b2 se bude rovnat 3.   
Pokud se b1 = „e“, pak b2 se bude rovnat 4.

lookdown b1,("abcde"),b2

Jméno:

lookup

Syntaxe:

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

Offset - proměnná/konstanta určující pořadové číslo hodnoty ***Data***, která bude vybrána a uložena do Proměnná

Data - konstanty/proměnné, ze kterých bude vybrána výsledná hodnota

Proměnná - proměnná, do níž bude uložena vybraná hodnota

Popis:

Příkaz překopíruje hodnotu proměnné s pořadovým číslem „offset“ do „proměnná“. Pokud by došlo k překročení rozsahu, „proměnná“ se nezmění.

Vyhledávání je omezeno na 0-255 položek, přičemž každá může být různého typu.

Platí pro:

Vše

Příbuzné:

* [lookdown](http://www.picaxe.com/commands/lookdown)

Příklad:

Proměnná b1 bude postupně nabývat hodnoty „a“, „b“…“e“.

main:

lookup b0,("abcde"),b1 ; uložení kódu ASCII znaku do b1

inc b0 ; zvýšení b0

if b0 < 4 then main ; je-li b0 < 4 skok na začátek

end

Jméno:

random

Syntaxe:

RANDOM wordproměnná

Wordproměnná - proměnná typu word, jejíž počáteční hodnota ovlivňuje výsledné pseudonáhodné číslo, které je opět do proměnné uloženo.

Popis:

Příkaz vygeneruje pseudonáhodné číslo v rozsahu 0 - 65535.

Jde o matematický výpočet, takže při stejných počátečních podmínkách, generována stále stejná čísla. To lze částečně odstranit využitím čítačů a změnou počáteční hodnoty. Například takto:

|  |  |  |
| --- | --- | --- |
|  | let w0 = timer | ; do w0 se uloží aktuální čas |
|  | random w0 | ; a následně se vygeneruje náhodné číslo w0 |

Další možností je opakované volaní náhodného, například při čekání na stlačení tlačítka.

Pokud je potřeba náhodné číslo v rozsahu byte, vygenerujeme číslo do w0, které je složeno z b1 a b0. Následně použijeme hodnotu typu byte.

Platí pro:

Vše

Příbuzné:

* [let](http://www.picaxe.com/commands/let)

Příklad:

Část programu, kdy čekání na stisk tlačítka je využito i pro generování náhodných čísel. Tím dojde ke zvýšení „náhodnosti“ při dalším použití příkazu random.  
V případě stisknutí tlačítka dojde k náhodnému blikání LED na pinech portu B.

main:

random w0 ; generování náhodného čísla

if pinC.1 =1 then ; zjištění stavu tlačítka

let pinsB = b1 ; zaslání náhodné hodnot na port B

pause 100 ; čekání 0.1s

end if

goto main ; skok na začátek

Jméno:

clearbit

Syntaxe:

CLEARBIT var, bit

Var - proměnná.

Bit - konstanta/proměnná, 0-7 pro proměnnou typu byte, 0-15 pro proměnnou typu word.

Popis:

Vymaže (nastaví na 0) konkrétní bit v proměnné.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [setbit](http://www.picaxe.com/commands/setbit)
* [togglebit](http://www.picaxe.com/commands/togglebit)
* [if bit](http://www.picaxe.com/commands/if-bit)

Příklad:

Vymaže 0. bit v proměnné b6.

clearbit b6, 0

Jméno:

setbit

Syntaxe:

SETBIT var, bit

Var - proměnná.

Bit - konstanta/proměnná, 0-7 pro proměnnou typu byte, 0-15 pro proměnnou typu word.

Popis:

Nastaví (přiřadí 1) konkrétní bit v proměnné.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [clearbit](http://www.picaxe.com/commands/clearbit)
* [togglebit](http://www.picaxe.com/commands/togglebit)
* [if bit](http://www.picaxe.com/commands/if-bit)

Příklad:

Nastaví 0. bit v proměnné b6 a 15.bit v proměnné w4.

setbit b6, 0

setbit w4, 15

Jméno:

togglebit

Syntaxe:

TOGGLEBIT var, bit

Var - proměnná.

Bit - konstanta/proměnná, 0-7 pro proměnnou typu byte, 0-15 pro proměnnou typu word.

Popis:

Invertuje - neguje (0=>1, 1=>0), konkrétní bit v proměnné.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [clearbit](http://www.picaxe.com/commands/clearbit)
* [setbit](http://www.picaxe.com/commands/setbit)
* [if bit](http://www.picaxe.com/commands/if-bit)

Příklad:

Zneguje 0.bit v proměnné b6 a 15.bit v proměnné w4

togglebit b6, 0

togglebit w4, 15

Jméno:

peek

Syntaxe:

**PEEK location,proměnná,proměnná,WORD wordproměnná...**

Location - proměnná/konstanta specifikující adresu požadované hodnoty

Proměnná - proměnná typu byte, do níž se hodnota uloží, pokud je požadována hodnota wort, je třeba použít slovo WORD a pak proměnnou

Popis:

Přečte data uložená na adrese „location“ v paměti RAM.  
Dostupný prostor se překrývá s úložištěm proměnných b0 - bxx, a některými funkčními registry. Proto je použitelný prostor omezen.

Pro verze kromě M2/X2, je využitelný prostor následující:

|  |  |
| --- | --- |
| Adresy $50 to $7E | obecně využitelné registry - paměťová místa |
| Adresy $C0 to $EF | dále využitelné u PICAXE-18X |
| Adresy $C0 to $FF | dále využitelné u PICAXE-28X, 40X |
| Adresy $C0 to $EF | dále využitelné u PICAXE-28X1, 40X1 |

Verze M2

Paměťový prostor je rozšířen na 512 B RAM, pracují vedle sebe 2 paměťové banky, takže je možné použít celých 256 B RAM. Výjimkou je 18M2, kdy byty 128-255 jsou vyhrazeny pro multitasking. Speciální registry jsou dostupné přes [peeksfr](http://www.picaxe.com/commands/peeksfr) a [pokesfr](http://www.picaxe.com/commands/pokesfr).

Platí pro:

Vše (kromě 08)

Příbuzné:

* [poke](http://www.picaxe.com/commands/poke)
* [peeksfr](http://www.picaxe.com/commands/peeksfr)
* [pokesfr](http://www.picaxe.com/commands/poksefr)

Příklad:

Načte hodnotu z adresy 80 do b1, následně načte hodnoty z adres 80 + 81 do w1.

peek 80,b1

peek 80, word w1

Jméno:

poke

Syntaxe:

POKE location,data,data,WORD wordproměnná...

Location - proměnná/konstanta specifikující adresu paměťového místa

Data - proměnná/konstanta typu byte, z níž se hodnota uloží, pokud je požadována hodnota wort, je třeba použít slovo WORD a pak proměnnou

Popis:

Uloží data na adrese „location“ v paměti RAM.  
Dostupný prostor se překrývá s úložištěm proměnných b0 - bxx, a některými funkčními registry. Proto je použitelný prostor omezen.

Pro verze kromě M2/X2, je využitelný prostor následující:

|  |  |
| --- | --- |
| Adresy $50 to $7E | obecně využitelné registry - paměťová místa |
| Adresy $C0 to $EF | dále využitelné u PICAXE-18X |
| Adresy $C0 to $FF | dále využitelné u PICAXE-28X, 40X |
| Adresy $C0 to $EF | dále využitelné u PICAXE-28X1, 40X1 |

Verze M2

Paměťový prostor je rozšířen na 512 B RAM, pracují vedle sebe 2 paměťové banky, takže je možné použít celých 256 B RAM. Výjimkou je 18M2, kdy byty 128-255 jsou vyhrazeny pro multitasking. Speciální registry jsou dostupné přes [peeksfr](http://www.picaxe.com/commands/peeksfr) a [pokesfr](http://www.picaxe.com/commands/pokesfr).

Platí pro:

Vše (kromě 08)

Příbuzné:

* [peek](http://www.picaxe.com/commands/peek)
* [peeksfr](http://www.picaxe.com/commands/peeksfr)
* [pokesfr](http://www.picaxe.com/commands/pokesfr)

Příklad:

Uloží hodnotu na adresu 80 z b1, následně uloží hodnotu na adresy 80 + 81 z w1.

poke 80,b1

poke 80, word w1

Jméno:

get

Syntaxe:

GET location,proměnná,proměnná,WORD wordproměnná...

Location - proměnná/konstanta určující adresu v pomocné paměti, adresní prostor je následující:

|  |  |
| --- | --- |
| 0 to 127 | pro X1 |
| 0 to 127 | pro 20X2 |
| 0 to 1023 | pro ostatní X2 |

Proměnná - proměnná typu byte, do níž se hodnota uloží, pokud je požadována hodnota wort, je třeba použít slovo WORD a pak proměnnou

Popis:

Funkce načte hodnotu z pomocné paměti RAM.  
Funkce neovlivňuje hodnotu ukazatelů.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [put](http://www.picaxe.com/commands/put)

Příklad:

Načte hodnotu z adresy 1 v pomocné paměti a uloží ji do b1.  
 Načte hodnotu z adresy 1 a 2 v pomocné paměti a uloží ji do w1.

get 1,b1

get 1, word w1

Jméno:

put

Syntaxe:

PUT location,data,data,WORD wordproměnná...

Location - proměnná/konstanta určující adresu v pomocné paměti, adresní prostor je následující:

|  |  |
| --- | --- |
| 0 to 127 | pro X1 |
| 0 to 127 | pro 20X2 |
| 0 to 1023 | pro ostatní X2 |

Data - proměnná/konstanta typu byte, z níž se hodnota uloží, pokud je požadována hodnota wort, je třeba použít slovo WORD a pak proměnnou

Popis:

Funkce uloží hodnotu do pomocné paměti RAM.  
Funkce neovlivňuje hodnotu ukazatelů.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [get](http://www.picaxe.com/commands/get)

Příklad:

Uloží proměnou b1 do pomocné paměti na adresu 1.  
Uloží proměnnou w1 do pomocné paměti na adresu 1 a 2.

put 1,b1

put 1, word w1

Jméno:

read

Syntaxe:

READ location,proměnná,proměnná, WORD wordproměnná

Location - proměnná/konstanta typu byte, určující adresu (0-255)

Proměnná - proměnná typu byte, do níž bude načtená hodnota uložena. V případě načtení hodnoty do proměnné typu word, je nutné použít slovo WORD.

Popis:

Čte z paměti EEPROM data a ukládá je to proměnné.

Paměť EEPROM uchovává data i po vypnutí zdroje. Data jsou přepsána při nahrávání nového programu. Data je možné uložit příkazem [write](http://www.picaxe.com/commands/write).

Paměť EEPROM je společná pro data a program u modelů: PICAXE-08, 08M, 08M2, 14M, 18, 18M, 18M2. Podrobná mapa paměti je u příkazu [EEPROM](http://www.picaxe.com/commands/eeprom).

Platí pro:

Vše

Příbuzné:

* [write](http://www.picaxe.com/commands/write)
* [eeprom](http://www.picaxe.com/commands/eeprom)
* [readtable](http://www.picaxe.com/commands/readtable)
* [table](http://www.picaxe.com/commands/table)

Příklad:

Výpis prvních 64 bytů paměti EPROM na LCD displej.

main:

for b0 = 0 to 63 ; cyklus pro b0 = 0 až 63

read b0,b1 ; načtení hodnoty z EEPROM a uložení do b1

serout B.7,N2400,(b1) ; odeslání po sériové lince na LCD

next b0 ; další cyklus

Jméno:

write

Syntaxe:

WRITE location,data ,data, WORD wordproměnná...

Location - proměnná/konstanta typu byte, určující adresu (0-255)

Data - proměnná/konstanta typu byte, z níž bude hodnota uložena. V případě uložení hodnoty z proměnné typu word, je nutné použít slovo WORD.

Popis:

Zapisuje data z proměnné/konstanty do paměti EEPROM.

Paměť EEPROM uchovává data i po vypnutí zdroje. Data jsou přepsána při nahrávání nového programu.

Paměť EEPROM je společná pro data a program u modelů: PICAXE-08, 08M, 08M2, 14M, 18, 18M, 18M2. Podrobná mapa paměti je u příkazu [EEPROM](http://www.picaxe.com/commands/eeprom).

Platí pro:

Vše

Příbuzné:

* [read](http://www.picaxe.com/commands/read)
* [eeprom](http://www.picaxe.com/commands/eeprom)

Příklad:

Zápis 64 bytů přijatých po sériové lince do paměti EEPROM.

main:

for b0 = 0 to 63 ; cyklus b0 = 0 až 63

serin C.6,N2400,b1 ; příjem po sériové lince

write b0,b1 ; zápis do EEPROM paměti

next b0 ; opakování cyklu

Jméno:

eeprom

Syntaxe:

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

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

Location - konstanta - adresa paměti EEPROM (0-255), odkud se mají data ukládat.  
Pokud není zadána, pokračuje se dál od posledního konce. Pokud ještě nebyla vůbec specifikována, začne se ukládat od 0.

Data - konstanty typu byte, které budou do paměti uloženy.

Popis:

Příkaz specifikuje data, která mají být uložena do paměti EEPROM, při zavádění programu.

Pokud není příkaz použit uloží se na použitelná místa nuly, pokud tomu chceme zabránit, použije se direktiva #no\_data.

Slovo DATA a EEPROM jsou ekvivalentní a nemají vliv na délku programu.   
S uloženými daty se dá pracovat pomocí příkazů [read](http://www.picaxe.com/commands/read) a [write](http://www.picaxe.com/commands/write).

Všechny současné typy PICAXE mají k dispozici 256 bytes (adresy 0-255) paměti EEPROM. Pouze následující starší typy mají paměti méně:

|  |  |
| --- | --- |
| PICAXE-28, 28A | 0 to 63 |
| PICAXE-08, 18, 28X, 40X | 0 to 127 |

Sdílená paměť  
= stejná paměť je využívána pro ukládání dat i pro uložení programu.  
Proto je možné použít pouze část paměti, která neobsahuje program. Délku programu lze zjistit v editoru PICAXE. Následně pro uložení dat platí:

|  |  |  |
| --- | --- | --- |
| PICAXE-08 / 18 | 0 to (127 - délka programu) |  |
| PICAXE-08M | 0 to (255 - délka programu) |  |
| PICAXE-14M / 20M | 0 to (255 - délka programu) |  |
| PICAXE-18M | 0 to (255 - délka programu) |  |
| PICAXE- 08M2 / 18M2 (not 18M2+) | 0 to (255 - délka programu + 1792) |  |

Platí pro:

Vše

Příbuzné:

* [read](http://www.picaxe.com/commands/read)
* [write](http://www.picaxe.com/commands/write)
* [table](http://www.picaxe.com/commands/table)

Příklad:

Vypsání uvítací zprávy na sériový LCD modul.

eeprom 0,(“Hello World”) ; uložení textu do EEPROM

main:

for b0 = 0 to 10 ; cyklus

read b0,b1 ; načtení 1 bytu z EEPROM

serout B.7,N2400,(b1) ; odeslání 1 bytu na sériové LCD

next b0 ; další cyklus

Jméno:

readtable

Syntaxe:

**READTABLE location,proměnná**

Location - proměnná/konstanta určující adresu tabulky

Proměnná - proměnná typu byte, do níž je uložena načtená hodnota

Popis:

Načte hodnotu z vyhledávací tabulky.

Tabulka může být vložena v rámci nahrávání programu a následně data mohou být čtena, nebo blokově kopírována do paměti RAM.

Platí pro:

M2, X1, X2

Příbuzné:

* [table](http://www.picaxe.com/commands/table)
* [read](http://www.picaxe.com/commands/read)
* [eeprom](http://www.picaxe.com/commands/eeprom)
* [tablecopy](http://www.picaxe.com/commands/tablecopy)

Příklad:

Vypsání uvítací zprávy na sériový LCD modul.

table 0,(“Hello World”) ; uložení textu do vyhledáv. tabulky

main:

for b0 = 0 to 10 ; cyklus

readtable b0,b1 ; načtení 1 bytu z tabulky

serout B.7,N2400,(b1) ; odeslání 1 bytu na sériové LCD

next b0 ; další cyklus

Jméno:

table

Syntaxe:

**TABLE {location},(data,data...)**

Location - konstanta - adresa paměti tabulky, odkud se mají data ukládat.  
Pokud není zadána, pokračuje se dál od posledního konce. Pokud ještě nebyla vůbec specifikována, začne se ukládat od 0.

Data - konstanty typu byte (0-255), které budou uloženy v tabulce

Popis:

Zadání hodnot, které se uloží do vyhledávací tabulky, současně s uložením programu.

Řada M2 má samostatných 512 bytů (0-511), které nijak neovlivňují paměť programu.

Řada X1/X2 může využít 256 bytů, přičemž při použití dochází k zmenšení prostoru pro program.

Platí pro:

M2, X1 and X2 parts

Příbuzné:

* [readtable](http://www.picaxe.com/commands/readtable)
* [tablecopy](http://www.picaxe.com/commands/tablecopy)

Příklad:

Vypsání uvítací zprávy na sériový LCD modul.

table 0,(“Hello World”) ; uložení textu do vyhledáv. tabulky

main:

for b0 = 0 to 10 ; cyklus

readtable b0,b1 ; načtení 1 bytu z tabulky

serout B.7,N2400,(b1) ; odeslání 1 bytu na sériové LCD

next b0 ; další cyklus

Jméno:

tablecopy

Syntaxe:

TABLECOPY start\_location,block\_size

Start\_location - startovací adresa, od které se budou data kopírovat (0-511)

Block\_size - velikost datového bloku, který se bude kopírovat do   
paměti RAM (1-512)

Popis:

Zkopíruje vyhledávací tabulkou do stejného paměťového prostoru paměti RAM.

Jedná se o velmi rychlé kopírování bloku dat z tabulky do paměti RAM.  
V paměti RAM už je možné používat příkazy [peek](http://www.picaxe.com/commands/peek) or @bptr.

Kopírování končí, pokud je překročena maximální adresa tabulky (511)

Platí pro:

14M2, 20M2, 18M2+

Příbuzné:

* [table](http://www.picaxe.com/commands/table)
* [readtable](http://www.picaxe.com/commands/readtable)

Příklad:

Překopírování dat z tabulky do paměti RAM a následný výstup na PC.

table 0,("Hello World") ; uložení dat do tabulky

main:

tablecopy 0,5 ; překopírování tabulky do RAM

debug ; odeslání na PC

goto main ; skok na začátek

Jméno:

goto

Syntaxe:

GOTO address

*Address* - adresa (návěští) určující, kam se má provést skok.

Popis:

Přejde (skočí) na místo určené adresou *Address*. Při definici musí být za Address uvedena „:“. Skok se provede vždy.

Platí pro:

Vše

Příbuzné:

* [branch](http://www.picaxe.com/commands/branch)
* [on goto](http://www.picaxe.com/commands/on-goto)
* [gosub](http://www.picaxe.com/commands/gosub)

Příklad:

Blikání LED diody v nekonečném cyklu.

main: ; definice skokové adresy - návěští

high B.1 ; vysoká úroveň B.1 - rozsvícení LED

pause 5000 ; 5 s čekání

low B.1 ; nízká úroveň B.1 - zhasnutí LED

pause 5000 ; 5 s čekání

goto main ; skok na „main“

Jméno:

branch

Syntaxe:

BRANCH offset,(address0,address1...addressN)

*Offset* - proměnná/konstanta 0-N, určující pozici skokové adresy - address0-addressN.

*Address0-addressN* - jsou skokové adresy - návěští.

Popis:

Skok na adresu (*address0-addressN*), s pořadovým číslem specifikovaným v parametru *offset*.

Příkaz umožnuje přejít na jinou část programu v závislosti na hodnotě *offset*. Pro hodnotu *offset* = 0, skočí na adresu *address0*, pro hodnotu 1 na *address1*…  
Pokud je *offset* mimo rozsah, příkaz je ignorován a pokračuje se dalším příkazem.  
Příkaz **branch** je totožný s příkazem **on goto**.

Platí pro:

Vše

Příbuzné:

* [on goto](http://www.picaxe.com/commands/on-goto)
* [goto](http://www.picaxe.com/commands/goto)

Příklad:

Rozsvěcení LED v závislosti na hodnotě v proměnné b1.

reset1: ; návěští pro inicializaci proměnné a portu

let b1 = 0 ; vynulování proměnné b1

low B.0 ; nastavení nízké úrovně na pony B.0 - B.3

low B.1

low B.2

low B.3

main:

inc b1 ; zvýšení obsahu proměnné b1 o +1

branch b1,(btn0,btn1, btn2, btn3, btn4);skok v závislosti na b1

goto reset1 ; skok na inicializaci při překročení rozsahu

btn0:

high B.0 ; nastavení vysoké úrovně na B.0 a skok na „main“

goto main

btn1:

high B.1 ; nastavení vysoké úrovně na B.1 a skok na „main“

goto main

btn2:

high B.2 ; nastavení vysoké úrovně na B.2 a skok na „main“

goto main

btn3:

high B.3 ; nastavení vysoké úrovně na B.3 a skok na „main“

goto main

btn4:

high B.4 ; nastavení vysoké úrovně na B.4 a skok na „main“

goto main

Jméno:

for

Syntaxe:

**FOR proměnná = start TO end**  {code}  
**NEXT {proměnná}**

**FOR proměnná = start TO end STEP {-}increment**  {code}  
**NEXT {proměnná}**

*Proměnná* - proměnná cyklu, ukládá se do ní aktuální hodnota cyklu

*Start* - počáteční hodnota cyklu, která se na začátku uloží do proměnné cyklu

*End* - konečná hodnota cyklu, při jejím překročení dojde k ukončení cyklu

*Increment* - hodnota která se přičte/odečte k proměnné cyklu. V případě, že je záporná, musí být *start* > *end.* Není-li uvedena, je nastaveno +1 (defaultní nastavení).

Popis:

For...next je cyklus s předem známým počtem opakování, pro opakované provedení části programu.  
Na počátku je do proměnné cyklu *proměnná* uložena hodnota *start*. Při dosažení příkazu next, je *proměnná* upravena přičtením/odečtením hodnoty *increment*.  
Cyklus se ukončí při překročení hranice intervalu <*start;end>*.  
Cyklus for…next, se vždy provede minimálně jednou.

V PICAXE může být vnořeno 8 cyklů for...next, přičemž každý musí mít jinou proměnnou cyklu. Cyklus může být předčasně ukončen příkazem [exit](http://www.picaxe.com/commands/exit) a program bude pokračovat následujícím příkazem za cyklem.

Je-li proměnná cyklu typu byte, pak je možno použít rozsah hodnot 0..255, při použití proměnné typu word je možné použít rozsah 0..65534, při větším rozsahu dojde k přetečení čítače a následně nekonečné smyčce.

Cyklus for…next, je kompaktnější podobou cyklu do…loop:

|  |  |
| --- | --- |
|  | LET proměnná = start DO    {code}    LET proměnná = proměnná +/- increment LOOP WHILE proměnná >= start AND proměnná <= end |

Platí pro:

Vše

Příbuzné:

* [next](http://www.picaxe.com/commands/next)
* [exit](http://www.picaxe.com/commands/exit)

Příklad:

Zrychlující se série bliknutí LED a 2s pauzou a opakováním.

main:

for b0 = 250 to 0 step -50 ; cyklus for…next, b0 nabyde hodnot

; 250, 200, 150, 100, 50, 0

if pinC.1 = 1 then exit ; je-li na pinu C.1 napětí, pak

; ukončit cyklus for…next

high B.1 ; nastavit B.1 na vysokou úroveň

pause b0 ; čekat b0 milisekund

low B.1 ; nastavit b.1 na nízkou úroveň

pause b0 ; čekat b0 milisekund

next b0 ; konec cyklu

pause 2000 ; čekání 2 s

goto main ; skok na začátek programu

Jméno:

next

Syntaxe:

**FOR proměnná = start TO end**  {code}  
**NEXT {proměnná}**

**FOR proměnná = start TO end STEP {-}increment**  {code}  
**NEXT {proměnná}**

*Proměnná* - proměnná cyklu, ukládá se do ní aktuální hodnota cyklu

*Start* - počáteční hodnota cyklu, která se na začátku uloží do proměnné cyklu

*End* - konečná hodnota cyklu, při jejím překročení dojde k ukončení cyklu

*Increment* - hodnota která se přičte/odečte k proměnné cyklu. V případě, že je záporná, musí být *start* > *end.* Není-li uvedena, je nastaveno +1 (defaultní nastavení).

Popis:

For...next je cyklus s předem známým počtem opakování, pro opakované provedení části programu.  
Na počátku je do proměnné cyklu *proměnná* uložena hodnota *start*. Při dosažení příkazu next, je *proměnná* upravena přičtením/odečtením hodnoty *increment*.  
Cyklus se ukončí při překročení hranice intervalu <*start;end>*.  
Cyklus for…next, se vždy provede minimálně jednou.

V PICAXE může být vnořeno 8 cyklů for...next, přičemž každý musí mít jinou proměnnou cyklu. Cyklus může být předčasně ukončen příkazem [exit](http://www.picaxe.com/commands/exit) a program bude pokračovat následujícím příkazem za cyklem.

Je-li proměnná cyklu typu byte, pak je možno použít rozsah hodnot 0..255, při použití proměnné typu word je možné použít rozsah 0..65534, při větším rozsahu dojde k přetečení čítače a následně nekonečné smyčce.

Cyklus for…next, je kompaktnější podobou cyklu do…loop:

|  |  |
| --- | --- |
|  | LET proměnná = start DO    {code}    LET proměnná = proměnná +/- increment LOOP WHILE proměnná >= start AND proměnná <= end |

Platí pro:

Vše

Příbuzné:

* [next](http://www.picaxe.com/commands/next)
* [exit](http://www.picaxe.com/commands/exit)

Příklad:

Zrychlující se série bliknutí LED a 2s pauzou a opakováním.

main:

for b0 = 250 to 0 step -50 ; cyklus for…next, b0 nabyde hodnot

; 250, 200, 150, 100, 50, 0

if pinC.1 = 1 then exit ; je-li na pinu C.1 napětí, pak

; ukončit cyklus for…next

high B.1 ; nastavit B.1 na vysokou úroveň

pause b0 ; čekat b0 milisekund

low B.1 ; nastavit b.1 na nízkou úroveň

pause b0 ; čekat b0 milisekund

next b0 ; konec cyklu

pause 2000 ; čekání 2 s

goto main ; skok na začátek programu

Jméno:

do

Syntaxe:

**DO**  
  {code}  
**LOOP**

**DO WHILE/UNTIL condition**  
  {code}  
**LOOP**

**DO**  
  {code}  
**LOOP WHILE/UNTIL condition**

*Condition* - logický výraz - podmínka cyklu, který určuje, zda bude cyklus proveden nebo ne.

*Code* - příkazy, které budou prováděny v rámci cyklu.

Popis:

Do…loop ohraničuje část programového kódu, který má být prováděn opakovaně, dokud je/není splněna podmínka cyklu - *condition*.

|  |
| --- |
| Příklady použití podmínky: - podle stavu pinů - je-li splněno, že pin C.1 se rovná 0, pak prováděj cyklus   (překlápěj B.7)   DO WHILE pinC.1 = 0   TOGGLE B.7 LOOP |
|  |
| - podle hodnot proměnných - je-li splněno, že b0 je menší než b1, pak prováděj cyklus   (překlápěj B.7 a zvětšuj b0) |
| DO WHILE b0 < b1   TOGGLE B.7   b0 = b0 + 1 LOOP |

Operátory, které mohou být použity pro vytvoření podmínky cyklu *condition:*

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Vícenásobná podmínka cyklu (složená podmínka)  
Pro vytvoření složitější podmínky cyklu se používají logické operátory:  
logický součin - AND (a zároveň) - musí platit obě podmínky a platila i výsledná   
logický součet - OR (nebo) - platí-li jedna z podmínek, pak platí celá podmínka.

|  |  |
| --- | --- |
|  | condition **AND**condition |
|  | condition **OR**condition |
|  | DO WHILE pin2 = 1 AND pin4 = 1 ; proveď, pokud je pin2=1 a zároveň i pin4=1 |
|  | LOOP UNTIL b0 = 2 OR b3 > b4 ; opakuj, pokud je b0=2 nebo b3>b4 |

 Ukázky použití cyklu do…loop:

**DO**  
  {code}  
**LOOP**

Nekonečná smyčka, možnost ukončení cyklu pouze příkazem [exit](http://www.picaxe.com/commands/exit), pak bude program pokračovat následujícím příkazem.

|  |  |  |
| --- | --- | --- |
| **DO WHILE condition** {code} **LOOP** |  | **DO UNTIL condition**   {code} **LOOP** |

Cyklus se bude provádět tak dlouho, dokud je(while)/není(until) podmínka splněna. Cyklus nemusí proběhnout ani jednou. Možnost předčasného ukončení cyklu pomocí příkazu [exit](http://www.picaxe.com/commands/exit).

|  |  |  |
| --- | --- | --- |
| **DO** {code} **LOOP WHILE condition** |  | **DO**   {code} **LOOP UNTIL condition** |

Cyklus se bude provádět tak dlouho, dokud je(while)/není(until) podmínka splněna. Cyklus vždy aspoň jednou proběhne. Možnost předčasného ukončení cyklu pomocí příkazu [exit](http://www.picaxe.com/commands/exit).

Platí pro:

Vše

Příbuzné:

* [loop](http://www.picaxe.com/commands/loop)
* [exit](http://www.picaxe.com/commands/exit)

Příklad:

LED 5-krát blikne, možno předčasně ukončit přivedením log. 1 na C.1.

let b2 = 1 ; nastavení počáteční hodnoty do b2

do ; začátek do…loop cyklu

high B.1 ; nastavení vysoké úrovně na B.1

pause 1000 ; čekání 1s

low B.1 ; nastavení nízké úrovně na B.1

pause 1000 ; čekání 1s

inc b2 ; zvýšení hodnoty v b2 o +1

if pinC.1 = 1 then exit ; zjištění stavu pinu C.1, případně

; ukončení cyklu

loop while b2 < 5 ; opakování cyklu do…loop

Jméno:

loop

Syntaxe:

**DO**  
  {code}  
**LOOP**

**DO WHILE/UNTIL condition**  
  {code}  
**LOOP**

**DO**  
  {code}  
**LOOP WHILE/UNTIL condition**

*Condition* - logický výraz - podmínka cyklu, který určuje, zda bude cyklus proveden nebo ne.

*Code* - příkazy, které budou prováděny v rámci cyklu.

Popis:

Do…loop ohraničuje část programového kódu, který má být prováděn opakovaně, dokud je/není splněna podmínka cyklu - *condition*.

|  |
| --- |
| Příklady použití podmínky: - podle stavu pinů - je-li splněno, že pin C.1 se rovná 0, pak prováděj cyklus   (překlápěj B.7)   DO WHILE pinC.1 = 0   TOGGLE B.7 LOOP |
|  |
| - podle hodnot proměnných - je-li splněno, že b0 je menší než b1, pak prováděj cyklus   (překlápěj B.7 a zvětšuj b0) |
| DO WHILE b0 < b1   TOGGLE B.7   b0 = b0 + 1 LOOP |

Operátory, které mohou být použity pro vytvoření podmínky cyklu *condition:*

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Vícenásobná podmínka cyklu (složená podmínka)  
Pro vytvoření složitější podmínky cyklu se používají logické operátory:  
logický součin - AND (a zároveň) - musí platit obě podmínky a platila i výsledná   
logický součet - OR (nebo) - platí-li jedna z podmínek, pak platí celá podmínka.

|  |  |
| --- | --- |
|  | condition **AND**condition |
|  | condition **OR**condition |
|  | DO WHILE pin2 = 1 AND pin4 = 1 ; proveď, pokud je pin2=1 a zároveň i pin4=1 |
|  | LOOP UNTIL b0 = 2 OR b3 > b4 ; opakuj, pokud je b0=2 nebo b3>b4 |

 Ukázky použití cyklu do…loop:

**DO**  
  {code}  
**LOOP**

Nekonečná smyčka, možnost ukončení cyklu pouze příkazem [exit](http://www.picaxe.com/commands/exit), pak bude program pokračovat následujícím příkazem.

|  |  |  |
| --- | --- | --- |
| **DO WHILE condition** {code} **LOOP** |  | **DO UNTIL condition**   {code} **LOOP** |

Cyklus se bude provádět tak dlouho, dokud je(while)/není(until) podmínka splněna. Cyklus nemusí proběhnout ani jednou. Možnost předčasného ukončení cyklu pomocí příkazu [exit](http://www.picaxe.com/commands/exit).

|  |  |  |
| --- | --- | --- |
| **DO** {code} **LOOP WHILE condition** |  | **DO**   {code} **LOOP UNTIL condition** |

Cyklus se bude provádět tak dlouho, dokud je(while)/není(until) podmínka splněna. Cyklus vždy aspoň jednou proběhne. Možnost předčasného ukončení cyklu pomocí příkazu [exit](http://www.picaxe.com/commands/exit).

Platí pro:

Vše

Příbuzné:

* [loop](http://www.picaxe.com/commands/loop)
* [exit](http://www.picaxe.com/commands/exit)

Příklad:

LED 5-krát blikne, možno předčasně ukončit přivedením log. 1 na C.1.

let b2 = 1 ; nastavení počáteční hodnoty do b2

do ; začátek do…loop cyklu

high B.1 ; nastavení vysoké úrovně na B.1

pause 1000 ; čekání 1s

low B.1 ; nastavení nízké úrovně na B.1

pause 1000 ; čekání 1s

inc b2 ; zvýšení hodnoty v b2 o +1

if pinC.1 = 1 then exit ; zjištění stavu pinu C.1, případně

; ukončení cyklu

loop while b2 < 5 ; opakování cyklu do…loop

Jméno:

if

Syntaxe:

*Při použití jednoho řádku:*

**IF condition THEN label**

**IF condition THEN GOTO label**

**IF condition THEN GOSUB label**

**IF condition THEN EXIT**

*Při použití více řádků:*

**IF condition THEN** {code} **ELSEIF condition THEN** {code} **ELSE** {code} **ENDIF**

*Condition* - logický výraz - podmínka, při splnění podmínky dojde k provedení příkazů umístěných za **then**. Při nesplnění podmínky, budou provedeny příkazy za **else** (je-li uvedeno).

*Code* - příkazy které budou provedeny v závislosti na vyhodnocení podmínky - *condition*. Mohou obsahovat výkonné příkazy, nebo další podmíněné příkazy **if**.   
(Tzv. vnořená podmínka)

Popis:

Provede část programového kódu v závislosti na pravdivosti logického výrazu - podmínky - *condition.*   
  
Pokud je podmínka pravdivá, provede příkazy umístěné za příkazem **then**, v případě, že podmínka není pravdivá a je přítomen příkaz **else** provede příkazy umístěné za tímto příkazem. Není-li příkaz **else** přítomen, přejde na následující příkaz. Příkaz **if** musí být ukončen příkazem **endif**.

Podmínka může obsahovat nejen proměnné, ale také může testovat stav pinů portů.  
  
Podmínku testující vnější piny je vhodné vložit do těla nekonečné smyčky, pro její ukončení.

Podmínky - conditions  
Pro porovnání obsahu proměnné s konkrétní hodnotou:

|  |
| --- |
| IF b1 >= 10 THEN ... |
|  |
| Pro porovnání dvou proměnných: |
| IF b2 < b3 THEN ... |

Použitelné relační operátory:

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Pro zjištění stavu vstupního pinu:

|  |  |
| --- | --- |
|  | IF pin2 = 1 THEN GOSUB buttonpushed |
|  | IF pinC.0 = 0 THEN GOSUB buttonnotpushed |

Vícenásobná podmínka cyklu (složená podmínka)  
  
Pro vytvoření složitější podmínky cyklu se používají logické operátory:  
logický součin - AND (a zároveň) - musí platit obě podmínky a platila i výsledná   
logický součet - OR (nebo) - platí-li jedna z podmínek, pak platí celá podmínka.

|  |  |  |
| --- | --- | --- |
|  | condition **AND**condition | |
|  | condition **OR**condition | |
|  | IF b0 > 10 AND b0 < 20 THEN GOTO between   ; je-li b0>10 a zároveň b0<20, pak skoč na „between“  ; (musí platit obě dvě podmínky současně) |
|  | IF pinC.0 = 1 OR b1 = 1 OR b2 = 3 THEN EXIT   ; je-li C.0=1 nebo b1=1 nebo b2=3, pak ukonči tuto část  ; programu (stačí, aby platila aspoň jedna z podmínek) |
|  |  |

**IF condition THEN label**  
  
Je-li podmínka splněna, program přejde (skočí) na místo specifikované *label* (návěští). Příkaz je zkráceným zápisem příkazu **IF condition THEN GOTO label**  
  
  
**IF condition THEN GOTO label**  
Je-li podmínka splněna, program přejde (skočí) na místo specifikované *label* (návěští).  
  
  
**IF condition THEN GOSUB label**  
  
Je-li podmínka splněna, program spustí podprogram (odskočí si do podprogramu) na místo specifikované *label* (návěští). Pro návrat z podprogramu se použije příkaz [return](http://www.picaxe.com/commands/return), po návratu pokračuje program následujícím příkazem.   
  
  
**IF condition THEN EXIT**  
  
Je-li podmínka splněna, program ukončí aktuální cyklus ([do...loop](http://www.picaxe.com/commands/do) nebo [for...next](http://www.picaxe.com/commands/for))   
a přejde na následující příkaz za cyklem.

**IF condition THEN**  {code} **ENDIF**

Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **endif**.

**IF condition THEN**  {code} **ELSE**  {code} **ENDIF**  
  
Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **else**. Jinak (= podmínka není splněna) provede příkazy mezi slovy **else** a **endif**.

**IF condition THEN**  {code} **ELSEIF condition THEN**  {code} **ENDIF**  
  
Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **else**. Jinak (= podmínka není splněna) vyhodnotí 2. podmínku uvedenou za **elseif** a pokud bude splněna, provede příkazy mezi slovy **then** (2.then) a **endif**. Pokud ani 2. podmínka není splněna, přejde na další příkaz za **endif**.

**IF condition THEN**  {code} **ELSEIF condition THEN**  {code} **ELSEIF condition THEN**  {code} **ENDIF**  
  
Použití stejné jako v přecházejícím případě, pouze testováno více podmínek. Pokud je to možné, je vhodné raději použít příkaz [select case](http://www.picaxe.com/commands/select).

**IF condition THEN**  {code} **ELSEIF condition THEN**  {code} **ELSE**  {code} **ENDIF**  
  
Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **else**. Jinak (= podmínka není splněna) vyhodnotí 2. podmínku uvedenou za **elseif** a pokud bude splněna, provede příkazy mezi slovy **then** (2.then) a **else**. Pokud ani 2. podmínka není splněna, provede příkazy mezi slovy **else** a **endif**.

Platí pro:

Vše

Příbuzné:

* [if pin](http://www.picaxe.com/commands/if-pin)
* [else](http://www.picaxe.com/commands/else)
* [endif](http://www.picaxe.com/commands/endif)

Příklad:

Rozsvícení LED na B.1, po dobu 5s, po stisku tlačítka na C.0.

main:

if pinC.0 = 1 then ; hlídání stisku tlačítka na C.0

goto flsh ; v případě stisku (C.0=1) skok na „flsh“

endif

goto main ; skok na začátek (nekonečná smyčka)

flsh:

high B.1 ; rozsvícení LED na B.1

pause 5000 ; čekání 5 s

low B.1 ; zhasnutí LED na B.1

goto main ; skok na začátek programu

Jméno:

else

Syntaxe:

**IF condition THEN** {code} **ELSEIF condition THEN** {code} **ELSE** {code} **ENDIF**

**SELECT CASE proměnná**  
**CASE condition**  
     {code}  
**ELSE**  
     {code}  
**ENDSELECT**

*Condition* - logický výraz - podmínka, na základě jeho pravdivosti jsou prováděny následující příkazy.

*Code* - příkazy, které se provedou na základě vyhodnocení podmínky - *condition*.

Popis:

Příkaz, který se používá jak součást dalších příkazu - *if...then \ else \ endif* , nebo  
*seelct case \ case \ else \ endselect.*

**IF condition THEN**  {code} **ELSE**  {code} **ENDIF**  
  
Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **else**. Jinak (= podmínka není splněna) provede příkazy mezi slovy **else** a **endif**.  
Více informací u příkazu [if](http://www.picaxe.com/commands/if).

**IF condition THEN**  {code} **ELSEIF condition THEN**  {code} **ELSE**  {code} **ENDIF**  
  
Je-li podmínka splněna, program provede příkazy, které se nacházejí mezi slovy **then** a **else**. Jinak (= podmínka není splněna) vyhodnotí 2. podmínku uvedenou za **elseif** a pokud bude splněna, provede příkazy mezi slovy **then** (2.then) a **else**. Pokud ani 2. podmínka není splněna, provede příkazy mezi slovy **else** a **endif**.

**SELECT CASE proměnná**  
**CASE condition**  
     {code}  
**ELSE**  
     {code}  
**ENDSELECT**

Pokud nebude žádná z „case“ podmínek splněna, potom se provedou příkazy mezi slovem **else** a **endselect**.  
Více informací u [select case](http://www.picaxe.com/commands/select).

Platí pro:

Vše

Příbuzné:

* [if](http://www.picaxe.com/commands/if)
* [if pin](http://www.picaxe.com/commands/if-pin)
* [endif](http://www.picaxe.com/commands/endif)

Příklad:

Rozsvícení LED na B.1, po dobu 5s, po stisku tlačítka na C.0.

main:

if pinC.0 = 1 then ; hlídání stisku tlačítka na C.0

goto flsh ; v případě stisku (C.0=1) skok na „flsh“

endif

goto main ; skok na začátek (nekonečná smyčka)

flsh:

high B.1 ; rozsvícení LED na B.1

pause 5000 ; čekání 5 s

low B.1 ; zhasnutí LED na B.1

goto main ; skok na začátek programu

Jméno:

endif

Syntaxe:

**IF condition THEN** {code} **ELSEIF condition THEN** {code} **ELSE** {code} **ENDIF**

*Condition* - logický výraz - podmínka, při splnění podmínky dojde k provedení příkazů umístěných za **then**. Při nesplnění podmínky, budou provedeny příkazy za **else** (je-li uvedeno).

*Code* - příkazy které budou provedeny v závislosti na vyhodnocení podmínky - *condition*. Mohou obsahovat výkonné příkazy, nebo další podmíněné příkazy **if**.

Popis:

Příkaz se používá jako součást příkazu **if…then**, pro ohraničení jeho konce. Lze jej použít jako jedno slovo „endif“, nebo dvě slova „end if“.   
Více informací u příkazu **if**.

Platí pro:

Vše

Příbuzné:

* [if](http://www.picaxe.com/commands/if)
* [if pin](http://www.picaxe.com/commands/if-pin)
* [else](http://www.picaxe.com/commands/else)

Příklad:

Rozsvícení LED na B.1, po dobu 5s, po stisku tlačítka na C.0.

main:

if pinC.0 = 1 then ; hlídání stisku tlačítka na C.0

goto flsh ; v případě stisku (C.0=1) skok na „flsh“

endif

goto main ; skok na začátek (nekonečná smyčka)

flsh:

high B.1 ; rozsvícení LED na B.1

pause 5000 ; čekání 5 s

low B.1 ; zhasnutí LED na B.1

goto main ; skok na začátek programu

Jméno:

select

Syntaxe:

**SELECT CASE proměnná**  
**CASE value** ; proměnná se rovná hodnotě  
     {code}  
**CASE value, value...** ; proměnná se rovná některé z hodnot  
     {code}  
**CASE value TO value** ; hodnota proměnné se nachází v intervalu od - do  
     {code}  
**CASE comparison value** ; proměnná je: >,<,>=… vůči hodnotě  
     {code} **ELSE** ; pro všechny ostatní případy (nic z předešlého neplatilo)  
     {code}  
**ENDSELECT**

*Proměnná* - proměnná, jejíž hodnota je testována.

*Value* - proměnná, nebo konstanta s níž *proměnná* testována.

*Comparison* - porovnávací pravidlo

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Popis:

Provede patřičnou část kódu, podle nastavené podmínky. Používá se pro vícenásobné testování jedné proměnné. Pokud je některá podmínka splněna, provedou se příkazy umístěné hned za ní, a ostatní podmínky se již netestují.  
V případě, že nebyla splněna žádná podmínka, provedou se příkazy za slovem **else**.  
Po provedení příkazu **select case**, program pokračuje dalším příkazem za slovem **endselect**.

Příkaz select case je výhodné použít místo vícenásobné vnořené podmínky, pokud je testována jedna proměnná.

**SELECT CASE proměnná**

Určuje proměnnou, která bude testována.

**CASE value**

Pokud je proměnná rovna hodnotě *value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE value, value...**

Pokud je proměnná rovna některé z hodnot *value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE value TO value**

Pokud je hodnota proměnné v intervalu ohraničeném *value TO value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE comparison value**

Pokud proměnná odpovídá výrazu (výraz je pravdivý), pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

*Comparison* - porovnávací pravidla

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

**ELSE**

Pokud nebyla splněna žádná podmínka, provedou se příkazy za slovem **else**.   
Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**ENDSELECT**

Ukončení příkaz **select case**. Příkaz je možné zadat jako jediné slovo „endselect“, nebo jako dvě slova „end select“.

Platí pro:

Vše

Příbuzné:

* [case](http://www.picaxe.com/commands/case)
* [else](http://www.picaxe.com/commands/else)
* [endselect](http://www.picaxe.com/commands/endselect)

Příklad:

Rozsvěcení LED na C.1 a C.2, v závislosti na hodnotě proměnné b1.

select case b1

case 1 ; když b1=1, nastav C.1 na vysokou úroveň

high C.1

case 2,3 ; když b1=2 nebo 3, nastav C.1 na nízkou úroveň

low C.1

case 4 to 6 ; když je b1 v intervalu <4;6> (včetně), nastav C.2   
; na vysokou úroveň

high C.2

else ; pokud nebyla splněna žádná podmínka, nastav C.2   
; na nízkou úroveň

low C.2

endselect

Jméno:

case

Syntaxe:

**SELECT CASE proměnná**  
**CASE value**  
     {code}  
**CASE value, value...**  
     {code}  
**CASE value TO value**  
     {code}  
**CASE comparison value**  
     {code} **ELSE**  
     {code}  
**ENDSELECT**

*Proměnná* - proměnná, jejíž hodnota je testována.

*Value* - proměnná, nebo konstanta s níž *proměnná* testována.

*Comparison* - porovnávací pravidlo

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Popis:

Je součástí příkazu [select case](http://www.picaxe.com/commands/select), uvozuje podmínky pro spuštění konkrétní části programu. Provede patřičnou část kódu, podle nastavené podmínky. Používá se pro vícenásobné testování jedné proměnné. Pokud je některá podmínka splněna, provedou se příkazy umístěné hned za ní, a ostatní podmínky se již netestují.  
V případě, že nebyla splněna žádná podmínka, provedou se příkazy za slovem **else**.  
Po provedení příkazu **select case**, program pokračuje dalším příkazem za slovem **endselect**.

Příkaz select case je výhodné použít místo vícenásobné vnořené podmínky, pokud je testována jedna proměnná.

**CASE value**

Pokud je proměnná rovna hodnotě *value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE value, value...**

Pokud je proměnná rovna některé z hodnot *value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE value TO value**

Pokud je hodnota proměnné v intervalu ohraničeném *value TO value*, pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

**CASE comparison value**

Pokud proměnná odpovídá výrazu (výraz je pravdivý), pak provede následující příkazy. Po provedení pokračuje za slovem [endselect](http://www.picaxe.com/commands/endselect).

*Comparison* - porovnávací pravidla

|  |  |  |
| --- | --- | --- |
|  | = | rovná se |
|  | is | rovná se |
|  | <> | nerovná se |
|  | != | nerovná se |
|  | > | větší (EN - klávesnice) |
|  | < | menší (EN - klávesnice) |
|  | >= | větší nebo rovno |
|  | <= | menší nebo rovno |

Platí pro:

Vše

Příbuzné:

* [select](http://www.picaxe.com/commands/select)
* [endselect](http://www.picaxe.com/commands/endselect)

Příklad:

Rozsvěcení LED na C.1 a C.2, v závislosti na hodnotě proměnné b1.

select case b1

case 1 ; když b1=1, nastav C.1 na vysokou úroveň

high C.1

case 2,3 ; když b1=2 nebo 3, nastav C.1 na nízkou úroveň

low C.1

case 4 to 6 ; když je b1 v intervalu <4;6> (včetně), nastav C.2   
; na vysokou úroveň

high C.2

else ; pokud nebyla splněna žádná podmínka, nastav C.2   
; na nízkou úroveň

low C.2

endselect

Jméno:

endselect

Syntaxe:

**SELECT CASE proměnná**  
**CASE condition**  
     {code}  
**ELSE**  
     {code}  
**ENDSELECT**

*Condition* - logický výraz - podmínka, při splnění podmínky jsou provedeny následující příkazy *code*.

*Code* - příkazy, které budou provedeny při splnění podmínky *condition*.

Popis:

Zakončuje příkaz [select case](http://www.picaxe.com/commands/select). Lze psát jako jedno slovo „endselect“, nebo dvě slova „end select“.

Platí pro:

Vše

Příbuzné:

* [select](http://www.picaxe.com/commands/select)
* [case](http://www.picaxe.com/commands/case)

Příklad:

Rozsvěcení LED na C.1 a C.2, v závislosti na hodnotě proměnné b1.

select case b1

case 1 ; když b1=1, nastav C.1 na vysokou úroveň

high C.1

case 2,3 ; když b1=2 nebo 3, nastav C.1 na nízkou úroveň

low C.1

case 4 to 6 ; když je b1 v intervalu <4;6> (včetně), nastav C.2   
; na vysokou úroveň

high C.2

else ; pokud nebyla splněna žádná podmínka, nastav C.2   
; na nízkou úroveň

low C.2

endselect

Jméno:

gosub

Syntaxe:

**GOSUB address**

**CALL address**

*Address* - místo - návěští, kam se provede odskočení.

Popis:

Volání (odskočení) podprogramu na adrese *address* z hlavního programu, návrat z podprogramu zajistí příkaz [return](http://www.picaxe.com/commands/return).  
Je možné volat podprogram z různých míst hlavního programu (i jiného podprogram), přičemž si program vždy zapamatuje místo ze kterého byl podprogram zavolán a po použití příkazu [return](http://www.picaxe.com/commands/return) se vrátí zpět na příkaz, který následuje za **gosub,** které volání způsobilo. Ke každému volání **gosub** musí existovat návratový příkaz **return**.  
Rozdíl **goto** - skok na jinou část programu, **gosub…return** - volání (odskočení do) podprogramu, ze kterého se vrátí do hlavního programu.  
Vhodné používat, pokud se některá část programového kódu opakuje na více místech. Pak tuto část použít jako podprogram a z více míst ji pouze volat.  
Použije-li se příkaz **gosub** v podprogramu, který byl již volám jiným (případně i stejným) příkazem **gosub**, mluvíme o vnořeném volání podprogramu.

Tabulka s maximálními počty použití příkazu **gosub** a maximální hloubky vnoření (= hloubka zásobníku mikrokontroleru).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **gosubs** | **interrupt** | **stack depth - hloubka zásobníku** |
| **Všechny 'M2'** | 255 | 1 | 8 - pro každý paralelní proces |
| **Všechny 'X2'** | 255 | 1 | 8 |
| **Všechny 'X1'** | 255 | 1 | 8 |
| **Všechny 'X' (zastaralé)** | 255 | 1 | 4 |
| **Všechny 'M' (zastaralé)** | 15 | 1 | 4 |
| **Všechny 'A' (zastaralé)** | 16 | 0 | 4 |

Platí pro:

Vše

Příbuzné:

* [return](http://www.picaxe.com/commands/return)
* [on gosub](http://www.picaxe.com/commands/on-gosub)
* [#gosubs](http://www.picaxe.com/commands/hash-gosubs)

Příklad:

Blikání LED, blikání jako podprogram, počet bliknutí je určen proměnou b2.

main:

let b2 = 15 ; do b2 přiřaď 15

gosub flsh ; volej podprogram „flsh“ pro blikání

let b2 = 5 ; do b2 přiřaď 5

gosub flsh ; volej podprogram „flsh“ pro blikání

end ; konec hlavního programu

flsh: ; podprogram pro blikání na adrese „flsh“

for b0 = 1 to b2 ; cyklus for…next, pro b0 od 1 do b2

high B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

low B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

next b0 ; konec cyklu for…next

return ; návrat z podprogramu

Jméno:

return

Syntaxe:

**RETURN**

Popis:

Návrat z podprogramu.

Návratový příkaz **return** se používá s odpovídajícím příkazem **gosub** a provede návrat z podprogramu zpět do hlavního (nadřazeného) programu. Při použití příkazu **return** bez příkazu **gosub** dojde ke zhroucení programu. Také pro správný běh programu je důležité, aby každé volání **gosub** bylo ukončeno příkazem **return**.  
(počet volání **gosub** = počet návratů **return**)

Platí pro:

Vše

Příbuzné:

* [gosub](http://www.picaxe.com/commands/gosub)
* [on gosub](http://www.picaxe.com/commands/on-gosub)

Příklad:

Blikání LED, blikání jako podprogram, počet bliknutí je určen proměnou b2.

main:

let b2 = 15 ; do b2 přiřaď 15

gosub flsh ; volej podprogram „flsh“ pro blikání

let b2 = 5 ; do b2 přiřaď 5

gosub flsh ; volej podprogram „flsh“ pro blikání

end ; konec hlavního programu

flsh: ; podprogram pro blikání na adrese „flsh“

for b0 = 1 to b2 ; cyklus for…next, pro b0 od 1 do b2

high B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

low B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

next b0 ; konec cyklu for…next

return ; návrat z podprogramu

Jméno:

end

Syntaxe:

**END**

Popis:

Neobnovitelný přechod do spícího režimu, používá se na konci programu do pro přechod do režimu minimální spotřeby a pro oddělení - ukončení hlavního programu, aby nedošlo k nechtěnému provedení následujícího podprogramu. Obnovení je možné pouze restartováním, nebo připojení PC a znovu nahrátím programu.

Pro dosažení minimální spotřeby jsou vypnuty i vnitřní čítače a časovače, takže nefungují ani příkazy [servo](http://www.picaxe.com/commands/servo) a [pwmout](http://www.picaxe.com/commands/pwmout). Pokud nechceme vypnout vnitřní časovače, používá se příkaz [stop](http://www.picaxe.com/commands/stop).

Platí pro:

Vše

Příbuzné:

* [stop](http://www.picaxe.com/commands/stop)
* [reset](http://www.picaxe.com/commands/reset)
* [#no\_end](http://www.picaxe.com/commands/hash-no-end)

Příklad:

Blikání LED, blikání jako podprogram, počet bliknutí je určen proměnou b2.

main:

let b2 = 15 ; do b2 přiřaď 15

gosub flsh ; volej podprogram „flsh“ pro blikání

let b2 = 5 ; do b2 přiřaď 5

gosub flsh ; volej podprogram „flsh“ pro blikání

end ; konec hlavního programu

flsh: ; podprogram pro blikání na adrese „flsh“

for b0 = 1 to b2 ; cyklus for…next, pro b0 od 1 do b2

high B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

low B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

next b0 ; konec cyklu for…next

return ; návrat z podprogramu

Jméno:

reset

Syntaxe:

**RESET**

Popis:

Softwarový restart celého mikrokontroleru (uP). Stav je stejný, jako při vypnutí a následném zapnutí uP do elektrického zdroje = ztráta obsahu všech proměnných. Po restartu je program spuštěn od prvního příkazu.

Platí pro:

Vše

Příbuzné:

* [end](http://www.picaxe.com/commands/end)
* [stop](http://www.picaxe.com/commands/stop)

Příklad:

Ukázka funkce restartování uP, místo skoku na začátek, se provede restart uP.

main:

let b2 = 15 ; do b2 přiřaď 15

gosub flsh ; volej podprogram „flsh“ pro blikání

let b2 = 5 ; do b2 přiřaď 5

gosub flsh ; volej podprogram „flsh“ pro blikání

reset ; restart uP - mikrokontroleru

flsh: ; podprogram pro blikání na adrese „flsh“

for b0 = 1 to b2 ; cyklus for…next, pro b0 od 1 do b2

high B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

low B.1 ; rozsviť LED na B.1

pause 500 ; čekej 0.5s

next b0 ; konec cyklu for…next

return ; návrat z podprogramu

Jméno:

stop

Syntaxe:

**STOP**

Popis:

Trvalé zastavení chodu programu (nekonečná smyčka odkaz sám na sebe). Obnovení provozu hardwarovým restartem, nebo připojením PC a znovu nahrátím programu. Nedochází k přechodu do nízké spotřeby. Časovače a čítače zůstávají v provozu, takže příkazy [servo](http://www.picaxe.com/commands/servo) a [pwmout](http://www.picaxe.com/commands/pwmout) stále pracují.

Platí pro:

Vše

Příbuzné:

* [end](http://www.picaxe.com/commands/end)
* [reset](http://www.picaxe.com/commands/reset)

Příklad:

Nekonečné generování PWM signálu.

main:

pwmout C.1,120,400 ; generování PWM signálu na C.1

stop ; zastavení programu, ale generování PWM   
; pokračuje stále

Jméno:

on goto

Syntaxe:

**ON offset GOTO address0,address1...addressN**

*Offset* - proměnná/konstanta 0-N, určující pozici skokové adresy - address0-addressN.

*Address0-addressN* - jsou skokové adresy - návěští.

Popis:

Skok na adresu (*address0-addressN*), s pořadovým číslem specifikovaným v hodnotě *offset*.

Příkaz umožnuje přejít na jinou část programu v závislosti na hodnotě *offset*. Pro hodnotu *offset* = 0, skočí na adresu *address0*, pro hodnotu 1 na *address1*…  
Pokud je *offset* mimo rozsah, příkaz je ignorován a pokračuje se dalším příkazem.  
Příkaz **on goto** je totožný s příkazem **branch**. Vhodnější je používat příkaz **on goto.**

Platí pro:

Vše

Příbuzné:

* [branch](http://www.picaxe.com/commands/branch)
* [on gosub](http://www.picaxe.com/commands/on-gosub)

Příklad:

Rozsvěcení LED v závislosti na hodnotě v proměnné b1.

reset1: ; návěští pro inicializaci proměnné a portu

let b1 = 0 ; vynulování proměnné b1

low B.0 ; nastavení nízké úrovně na pony B.0 - B.3

low B.1

low B.2

low B.3

main:

inc b1 ; zvýšení obsahu proměnné b1 o +1

on b1 goto btn0, btn1, btn2, btn3, btn4 ;skok v závislosti na b1

goto reset1 ; skok na inicializaci při překročení rozsahu

btn0:

high B.0 ; nastavení vysoké úrovně na B.0 a skok na „main“

goto main

btn1:

high B.1 ; nastavení vysoké úrovně na B.1 a skok na „main“

goto main

btn2:

high B.2 ; nastavení vysoké úrovně na B.2 a skok na „main“

goto main

btn3:

high B.3 ; nastavení vysoké úrovně na B.3 a skok na „main“

goto main

btn4:

high B.4 ; nastavení vysoké úrovně na B.4 a skok na „main“

goto main

Jméno:

on gosub

Syntaxe:

**ON offset GOSUB address0, address1, ...addressN**

*Offset* - proměnná/konstanta 0-N, určující pozici volané adresy - address0-addressN.

*Address0-addressN* - jsou adresy - návěští, pro volání podprogramů.

Popis:

Volání podprogramu na adrese (*address0-addressN*), s pořadovým číslem specifikovaným v hodnotě *offset*.

Příkaz umožnuje volat podprogram v závislosti na hodnotě *offset*. Pro hodnotu *offset* = 0, zavolá podprogram na adrese *address0*, pro hodnotu 1 na *address1*…  
Pokud je *offset* mimo rozsah, příkaz je ignorován a pokračuje se dalším příkazem.  
Návrat z podprogramu musí být zajištěn příkazem **return**. Příkaz **on gosub** je kompilátorem započítán jako jeden příkaz [gosub](http://www.picaxe.com/commands/gosub).

Platí pro:

Vše

Příbuzné:

* [on goto](http://www.picaxe.com/commands/on-goto)
* [gosub](http://www.picaxe.com/commands/gosub)
* [return](http://www.picaxe.com/commands/return)

Příklad:

Rozsvěcení LED v závislosti na hodnotě v proměnné b1 - použití ON GOSUB.

reset1: ; návěští pro inicializaci proměnné a portu

let b1 = 0 ; vynulování proměnné b1

low B.0 ; nastavení nízké úrovně na pony B.0 - B.3

low B.1

low B.2

low B.3

main:

if b1 > 3 then reset1 ; skok na inicializaci při překročení rozsahu

on b1 gosub btn0, btn1, btn2, btn3 ;skok v závislosti na b1

inc b1 ; zvýšení obsahu proměnné b1 o +1

goto main

btn0:

high B.0 ; nastavení vysoké úrovně na B.0 a skok na „main“

return

btn1:

high B.1 ; nastavení vysoké úrovně na B.1 a skok na „main“

return

btn2:

high B.2 ; nastavení vysoké úrovně na B.2 a skok na „main“

return

btn3:

high B.3 ; nastavení vysoké úrovně na B.3 a skok na „main“

return

Jméno:

exit

Syntaxe:

**EXIT**

Popis:

Slouží k okamžitému ukončení cyklu [do...loop](http://www.picaxe.com/commands/do) a [for...next](http://www.picaxe.com/commands/for).

Platí pro:

Vše

Příbuzné:

* [do](http://www.picaxe.com/commands/do)
* [loop](http://www.picaxe.com/commands/loop)
* [for](http://www.picaxe.com/commands/for)
* [next](http://www.picaxe.com/commands/next)

Příklad:

Ukončení nekonečného cyklu do…loop, pokud je b1=100.

main:

b1 = 0 ; inicializace proměnné b1

do ; start nekonečného cyklu do…loop

inc b1 ; zvětšení hodnoty v proměnné b1 o +1

if b1 = 100 then ; pokud je b1 = 100 pak

exit ; ukončení nekonečného cyklu příkazem exit

endif

loop ; konec nekonečného cyklu

b2 = 200 ; příkaz přiřazení po ukončení cyklu

Jméno:

debug

Syntaxe:

**DEBUG {var}**

*Var* - název proměnné (např. b3), hodnota není důležitá, slouží pouze pro zpětnou kompatibilitu se staršími programy.

Popis:

Zobrazí aktuální obsah všech proměnných v ladícím okně programového editoru, při reálném běhu programu na mikrokontroleru.

Bytové informace se zobrazí v desítkové, dvojkové (binární), šestnáctkové (hexadecimální) soustavě a v ASCII kódu. Wordovské informace se zobrazí v desítkové a šestnáctkové soustavě.  
Odešlou se informace o všech proměnných po programovacím kabelu, využívá se pro ladění programu na mikrokontroleru.   
Protože se přenáší velké množství dat, výrazně se zpomalují cykly, ve kterých se příkaz **debug** umístěn.   
Pro posílání jen uživatelem požadovaných informací (např. jen jedné proměnné) je vhodné použít příkaz [sertxd](http://www.picaxe.com/commands/sertxd).

U mikrokontrolerů řady 08 a 14 je pin pro programovací kabel společný s pinem B.0 (output 0). Proto je vhodné před použitím příkazu **debug** sekvenci:

|  |  |
| --- | --- |
| low B.0 | ; nastavení B.0 do nízké úrovně |
| pause 500 | ; počkání 500 ms |
| debug | ; odeslání informací do PC |

Platí pro:

Vše

Příbuzné:

* [serout](http://www.picaxe.com/commands/serout)
* [sertxd](http://www.picaxe.com/commands/sertxd)
* [hserout](http://www.picaxe.com/commands/hserout)
* [sertxd](http://www.picaxe.com/commands/sertxd)

Příklad:

Vypsání aktuálního obsahu všech proměnných na PC.

main:

inc b1 ; přičtení +1 k proměnné b1

readadc A.2,b2 ; načtení napětí z pinu A.2 do proměnné b2

debug ; výpis všech hodnot na PC

pause 500 ; čekání 0,5s

goto main ; skok na začátek programu

Jméno:

sertxd

Syntaxe:

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

*Data* - proměnné nebo konstanty typu byte, která budou vyslána na PC přes programový kabel.

Popis:

Sériový výstup po programovacím kabelu s definovanými parametry přenosu - komunikace probíhá po výstupním programovém pinu s nastavením (4800 baud, 8 data, no parity, 1 stop).

**Sertxd** je podobný [serout](http://www.picaxe.com/commands/serout), ale neumožňuje měnit komunikační pin a parametry přenosu. Data jsou zobrazována v programovém editoru: PICAXE>Terminal window.  
Nastavení je možné v části „Možnosti“ - 4800,n,8,1 (pro řadu X2: 9600,n,8,1)

Symbol „#“ převede hodnoty typu byte i word do ASCII podoby. Je-li b1 = 126, pak zápis #b1, odešle znakovou posloupnost "1" "2" "6".

*Efekt zvýšení taktovací rychlosti*Při zvýšení taktovací rychlosti, dojde ke zvýšení přenosové rychlosti dat (viz. tabulka) a je třeba přenastavit terminál v PC. To se netýká řady X2.

|  |  |
| --- | --- |
| 4MHz | 4800 |
| 8MHz | 9600 |
| 16MHz | 19200 |
| 32MHz | 38400 |
| 64MHz | 76800 |

Platí pro:

Vše (kromě 08, 18, 28, 28A)

Příbuzné:

* [serout](http://www.picaxe.com/commands/serout)
* [hserout](http://www.picaxe.com/commands/hserout)
* [serrxd](http://www.picaxe.com/commands/serrxd)
* [debug](http://www.picaxe.com/commands/debug)

Příklad:

Pošle obsah proměnné b1 do PC po programovacím kabelu.

main:

for b1 = 0 to 63 ; start cyklu

sertxd("The value of b1 is ",#b1,13,10)

; odeslání textu + proměnné b1 + odřádkování   
(13,10)

pause 1000 ; čekání 1s

next b1 ; konec cyklu

Jméno:

serrxd

Syntaxe:

**SERRXD (qualifier,qualifier...)**

**SERRXD (qualifier,qualifier...),{#}proměnná,{#}proměnná...**

**SERRXD {#}proměnná,{#}proměnná...**

*Nastavení časového limitu pro řady M2, X1 a X2*

**SERRXD [timeout], (qualifier...)**

**SERRXD [timeout], (qualifier...),{#}proměnná,{#}proměnná**

**SERRXD [timeout], {#}proměnná,{#}proměnná**

**SERRXD [timeout,address], (qualifier...)**

**SERRXD [timeout,address], (qualifier...),{#}proměnná,{#}proměnná**

**SERRXD [timeout,address], {#}proměnná,{#}proměnná**

*Qualifiers* - proměnné nebo konstanty (typu byte), které musí být přijaty, než začnou být plněny jednotlivé proměnné. Jedná se o rozpoznání, že zpráva je adresována danému procesoru.

*Proměnná* - proměnná typu byte. Lze použít symbol „#“ pro vstup v ASCII desítkové soustavě.

*Timeout* - volitelný parametr nastavující maximální dobu čekání na data v milisekundách. (jen u řady M2, X1 a X2).

*Address* - volitelný parametr určující adresu skoku, pokud vyprší časový limit.

Popis:

Sériový vstup přes vstupní programovací pin s nastavenými parametry - 4800 baud (X2 - 9600 baud), 8 data, no parity, 1 stop.

**Serrxd** je podobné jako příkaz [serin](http://www.picaxe.com/commands/serin), ale umožňuje komunikaci pouze po programovacím kabelu.  
PICAXE stále monitoruje stav vstupního programovacího pinu, pokud uvidí PC, umožní nahrání nového programu. To může být při použití příkazu **serrxd** problém, proto se doporučuje použit příkaz [disconnect](http://www.picaxe.com/commands/disconnect) (odpojit) a po ukončení sériové komunikace obnovit spojení některým z následujících způsobů:

1) příkazem [reconnect](http://www.picaxe.com/commands/reconnect) - znovupřipojení

2) příkazem [reset](http://www.picaxe.com/commands/reset) - softwarový restart mikrokontroleru

3) [hardwarový reset](http://www.picaxe.com/hard-reset) - pomocí hardwarového přerušení

4) vždy fungující [hard-reset](http://www.picaxe.com/hard-reset) - krátkodobé odpojení od zdroje

*Čtení číselné hodnoty*

Pokud je použit formát *#proměnná,* data budou přijata a převedena na číslo.  
Všechny nečíselné znaky budou ignorovány, dokut se neobjeví první číslice. Načítání čísla do proměnné bude ukončeno prvním nečíselným znakem.

*Efekt zvýšení taktovací rychlosti*  
Při zvýšení taktovací rychlosti, dojde ke zvýšení přenosové rychlosti dat (viz. tabulka) a je třeba přenastavit terminál v PC. To se netýká řady X2.

|  |  |
| --- | --- |
| 4MHz | 4800 |
| 8MHz | 9600 |
| 16MHz | 19200 |
| 32MHz | 38400 |
| 64MHz | 76800 |

Platí pro:

Vše (kromě 08, 08M, 18, 18A, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [disconnect](http://www.picaxe.com/commands/disconnect)
* [reconnect](http://www.picaxe.com/commands/reconnect)
* [serin](http://www.picaxe.com/commands/serin)
* [hserin](http://www.picaxe.com/commands/hserin)
* [sertxd](http://www.picaxe.com/commands/sertxd)

Příklad:

Ukládání dat přijatých po programovacím kabelu do paměti mikrokontroleru.

disconnect ; ukončení sledování vstupního programovacího pinu

serrxd [1000, timeout],@ptrinc,@ptrinc,@ptr

; uložení 3 hodnot do paměti na pozice: ptr, ptr+1,

; ptr+2

reconnect ; obnovení sledování vstupního programovacího pinu

Jméno:

serin

Syntaxe:

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

**SERIN pin,baudmode,(qualifier,qualifier...),{#}proměnná,{#}proměnná...**

**SERIN pin,baudmode,{#}proměnná,{#}proměnná...**

*Nastavení časového limitu pro řady M2, X1 a X2*

**SERIN [timeout], pin,baudmode,(qualifier...)**

**SERIN [timeout], pin,baudmode,(qualifier...),{#}proměnná,{#}proměnná**

**SERIN [timeout], pin,baudmode,{#}proměnná,{#}proměnná**

**SERIN [timeout,address], pin,baudmode,(qualifier...)**

**SERIN [timeout,address], pin,baudmode,(qualifier...),{#}proměnná,{#}proměnná**

**SERIN [timeout,address], pin,baudmode,{#}proměnná,{#}proměnná**

*Pin* - konstanta označující I/O kontakt pro komunikaci (B.3, C.1… viz schéma zapojení)

*Baudmode* - proměnná, nebo konstanta (0-7) určující režim přenosu:

Txxx - přijímá skutečný signál (když je vysoká úroveň, tak nečinný).

Nxxx - přijímá invertovaný signál (když je nízká úroveň, tak nečinný).

*Pro starší modely řady 08 / 08M / 18 / 18A / 28 / 28A*

|  |  |  |
| --- | --- | --- |
| 4MHz | 8MHz | 16MHz |
| T300\_4 | T600\_8 | T1200\_16 |
| T600\_4 | T1200\_8 | T2400\_16 |
| T1200\_4 | T2400\_8 | T4800\_16 |
| T2400\_4 | T4800\_8 | T9600\_16 |
| N300\_4 | N600\_8 | N1200\_16 |
| N600\_4 | N1200\_8 | N2400\_16 |
| N1200\_4 | N2400\_8 | N4800\_16 |
| N2400\_4 | N4800\_8 | N9600\_16 |

*Pro všechny ostatní řady (např. všechny řady X1, X2, M2)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4MHz | 8MHz | 16MHz | 32MHz | 64MHz |
| T600\_4 | T1200\_4 | T2400\_16 | T4800\_32 | T9600\_64 |
| T1200\_4 | T2400\_8 | T4800\_16 | T9600\_32 | T19200\_64 |
| T2400\_4 | T4800\_8 | T9600\_16 | T19200\_32 | T38400\_64 |
| T4800\_4 | T9600\_8 | T19200\_16 | T38400\_32 | T76800\_64 |
| N600\_4 | N1200\_8 | N2400\_16 | N4800\_32 | N9600\_64 |
| N1200\_4 | N2400\_8 | N4800\_16 | N9600\_32 | N19200\_64 |
| N2400\_4 | N4800\_8 | N9600\_16 | N19200\_32 | N38400\_64 |
| N4800\_4 | N9600\_8 | N19200\_16 | N38400\_32 | N76800\_64 |

*Qualifiers* - proměnné nebo konstanty (typu byte), které musí být přijaty, než začnou být plněny jednotlivé proměnné. Jedná se o rozpoznání, že zpráva je adresována danému procesoru.

*Proměnná* - proměnná typu byte. Lze použít symbol „#“ pro vstup v ASCII desítkové soustavě.

*Timeout* - volitelný parametr nastavující maximální dobu čekání na data v milisekundách. (jen u řady M2, X1 a X2).

*Address* - volitelný parametr určující adresu skoku, pokud vyprší časový limit.

Popis:

Sériový vstup s volitelným nastavením. (implicitně - 8 bit data, no parity, 1 stop bit).

Pro sériový vstup dat lze použít libovolný vstupní pin, kromě pinu, který používá příkaz [serrxd](http://www.picaxe.com/commands/serrxd).

*Pin* určuje, který vstupní pin bude použit, *Baudmode* definuje rychlost a polaritu datového signálu. Pro příjem RS232 pomocí odporového rozhraní se používá N (invertovaný) signál. Pro rozhraní s integrovaným obvodem MAX232,   
se používá typ T (přímý) signál. Protokol je pevně stanoven - N,8,1 (bez parity, 8 bitů data, 1 stop bit).

Rychlost 4800 baud může být pro pomalejší řady M, X, X1 a X2, vysoká. V tom případě se doporučuje vložit při vysílání dat mezi jednotlivé byty krátkou pauzu (2ms).

*Qualifiers* je specifická posloupnost dat, která musí být načtena, než dojde k ukládání hodnot do proměnných. Funguje jako klíč k odemčení zámku - načítání hodnot. Na jedné sériové lince může „poslouchat“ více procesorů.  
Např. serin *1,N2400,("ABC"),b1* - vyžaduje řetězec "ABC" a následující byte uloží do b1. Pokud *qualifiers* není uveden, do b1 se uloží první přijatý znak.

Příkaz **serin** nemůže být přerušen ani přerušeními nastavenými příkazem [setint](http://www.picaxe.com/commands/setint).

Příklad ukazuje, jak mikrokontroler začne pracovat, až po obdržení „příkazu go“:  
*serin 1,N2400,("go")*

Pozor! Při použití proměnné jako *qualifiers* nepoužívat závorky:  
Špatně - *serin 1,N2400,(b1)*. Správně - *serin 1,N2400, b1*

Řady M2, X1 a X2 mohou mít definovaný maximální čas, po který se má čekat na signál (*timeout* v milisekundách), při překročení časového limitu může přejít na definovanou adresu - *address*.   
V případě, že doba čekání není definována a signál se neobjeví, je nutný [hard reset](http://www.picaxe.com/hard-reset) (dočasné odpojení od zdroje).

Maximální přenosová rychlost 4800 baud, při frekvenci procesoru 8MHz, je vhodná i pro složitější sériové přenosy. Pro vyšší rychlosti je vhodnější použít externí taktování.  
Přesto pro většinu aplikací stačí použít vnitřní taktování s využitím příkazu [calibfreq](http://www.picaxe.com/commands/calibfreq).

*Čtení číselné hodnoty*

Pokud je použit formát *#proměnná,* data budou přijata a převedena na číslo.  
Všechny nečíselné znaky budou ignorovány, dokut se neobjeví první číslice. Načítání čísla do proměnné bude ukončeno prvním nečíselným znakem.

*Pro 8 a 14 pinové obvody*

Je nutné pro využití pinu input3 (C.3) připojit diodu 1N4148, tak aby „proužek“ (katoda) byl připojen na V+. Všechny ostatní piny mají interní diody.

*Všechny 20 pinové obvody*

Je nutné pro využití pinu input6 (C.6) připojit diodu 1N4148, tak aby „proužek“ (katoda) byl připojen na V+. Všechny ostatní piny mají interní diody.

Platí pro:

Vše

Příbuzné:

* [serrxd](http://www.picaxe.com/commands/serrxd)
* [hserin](http://www.picaxe.com/commands/hserin)
* [serout](http://www.picaxe.com/commands/serout)

Příklad:

Načtení sériových dat a jejich uložení do EEPROM na pozice 0 to 63

main:

for b0 = 0 to 63 ; start cyklu for…next

serin 6,N2400,b1 ; načtení sériových dat - 1 byte z pinu C.6

write b0,b1 ; zapsání hodnoty b1 na pozici b0 v EEPROM

next b0 ; konec cyklu for…next

Jméno:

serout

Syntaxe:

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

*Pin* - konstanta označující I/O kontakt (B.3, C.1 ... viz schéma zapojení)

*Baudmode* - proměnná, nebo konstanta (0-7) určující režim přenosu:

Txxx - přijímá skutečný signál (když je vysoká úroveň, tak nečinný).

Nxxx - přijímá invertovaný signál (když je nízká úroveň, tak nečinný).

*Pro starší modely řady 08 / 08M / 18 / 18A / 28 / 28A*

|  |  |  |
| --- | --- | --- |
| 4MHz | 8MHz | 16MHz |
| T300\_4 | T600\_8 | T1200\_16 |
| T600\_4 | T1200\_8 | T2400\_16 |
| T1200\_4 | T2400\_8 | T4800\_16 |
| T2400\_4 | T4800\_8 | T9600\_16 |
| N300\_4 | N600\_8 | N1200\_16 |
| N600\_4 | N1200\_8 | N2400\_16 |
| N1200\_4 | N2400\_8 | N4800\_16 |
| N2400\_4 | N4800\_8 | N9600\_16 |

*Pro všechny ostatní řady (např. všechny řady X1, X2, M2)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4MHz | 8MHz | 16MHz | 32MHz | 64MHz |
| T600\_4 | T1200\_4 | T2400\_16 | T4800\_32 | T9600\_64 |
| T1200\_4 | T2400\_8 | T4800\_16 | T9600\_32 | T19200\_64 |
| T2400\_4 | T4800\_8 | T9600\_16 | T19200\_32 | T38400\_64 |
| T4800\_4 | T9600\_8 | T19200\_16 | T38400\_32 | T76800\_64 |
| N600\_4 | N1200\_8 | N2400\_16 | N4800\_32 | N9600\_64 |
| N1200\_4 | N2400\_8 | N4800\_16 | N9600\_32 | N19200\_64 |
| N2400\_4 | N4800\_8 | N9600\_16 | N19200\_32 | N38400\_64 |
| N4800\_4 | N9600\_8 | N19200\_16 | N38400\_32 | N76800\_64 |

*Data* - proměnná nebo konstanta typu byte (0-255), která bude odeslána.  
Je možné použít znak „#“ pro zápis čísel ve formátu ASCII, pro zápis textu se používají uvozovky "Hello".

Popis:

Odešle sériová data přes definovaný výstupní pin, s nastavením - 8 data bits, no parity, 1 stop bit.

Pro sériový výstup dat lze použít libovolný vstupní pin, kromě pinu, který používá příkaz [sertxd](http://www.picaxe.com/commands/sertxd).

*Pin* určuje, který výstupní pin bude použit, *Baudmode* definuje rychlost a polaritu datového signálu. Pro vyslání na RS232 pomocí odporového rozhraní se používá N (invertovaný) signál. Pro rozhraní s integrovaným obvodem MAX232,   
se používá typ T (přímý) signál. Protokol je pevně stanoven - N,8,1 (bez parity, 8 bitů data, 1 stop bit).

Při „T“ přenosu může dojít k poškození prvního bytu. Tomu lze zabránit, pokud výstupní pin nastavíme na vysokou úroveň ([high](http://www.picaxe.com/commands/high)) a počkáme 5 ms, před odesláním prvního bytu.

Symbol „#“ převede hodnoty typu byte i word do ASCII podoby. Je-li b1 = 126, pak zápis #b1, odešle znakovou posloupnost "1" "2" "6".

Vliv změny taktovací frekvence procesoru a přenosovou rychlostí sériových dat, je stejný jako u příkazu [serin](http://www.picaxe.com/commands/serin).

Platí pro:

Vše

Příbuzné:

* [sertxd](http://www.picaxe.com/commands/sertxd)
* [hserout](http://www.picaxe.com/commands/hserout)
* [serin](http://www.picaxe.com/commands/serin)

Příklad:

Odeslání dat z EEPROM pozice 0 až 63, po sériové lince do sériového LCD displeje.

main:

for b0 = 0 to 63 ; začátek for…next cyklu

read b0,b1 ; načtení hodnoty z pozice b0 do proměnné b1

serout 7,N2400,(b1) ; odeslání hodnoty b1 do sériového LCD

next b0 ; konec for…next cyklu

Jméno:

hserin

Syntaxe:

*Pro řadu X2*

**HSERIN spaddress, count {,(qualifier)}**

**HSERIN [timeout, address], spaddress, count {,(qualifier)}**

*Qualifier* - jedna proměnná/konstanta typu byte (0-255), která musí být přijata, před ukládáním hodnot do paměti.

*Spaddress* - první adresa paměti, kam budou ukládána data

*Count* - počet bytů, které mají být načteny

*Timeout* - proměnná/konstanta definující čas čekání na signál v milisekundách

*Address* - adresa kam dojde ke skoku v případě překročení časového limitu

*Pro řadu M2*

**HSERIN var**

*Var* - proměnná typu byte, do níž bude přijat jeden byte

Popis:

Sériový vstup, přes hardwarový sériový vstupní pin (formát: 8 data bit, no parity, 1 stop bit).

Příkaz **hserin** slouží pro sériový vstup dat, přes pin spojený s integrovaným hardwarovým sériovým převodníkem. (viz schéma) Není možné ho použít na ostatních pinech.

Nastavení přenosových vlastností se provádí před prvním použitím **hserin**, příkazem [hsersetup](http://www.picaxe.com/BASIC-Commands/Serial-RS232-Interfacing/hsersetup).  
Přenosová rychlost je mnohem vyšší, než u serin. Proto není možné použití pomocných funkcí, např. „#“, ale „surová“ data jsou ukládána rovnou do paměti. Po kompletním přenosu je musí obsloužit uživatelský program.

U řady X2 je příjem dat prováděn na pozadí přímo do paměti.

U řady M2 je využito dvoj bytového zásobníku FIFO, do nějž jsou na pozadí stále ukládána přijatá data. Při použití příkazu **hserin** je vyšší hodnota uložena do proměnné. Pokud nejsou nová data, hodnota se nezmění, pokud jsou uloženy 2 hodnoty, je třeba příkaz **hserin** volat opakovaně. V případě „přetečení“ FIFO zásobníku jsou starší data ztracena.

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [serin](http://www.picaxe.com/commands/serin)
* [serrxd](http://www.picaxe.com/commands/serrxd)
* [hsersetup](http://www.picaxe.com/commands/hsersetup)
* [hserout](http://www.picaxe.com/commands/hserout)

Příklad:

Pro řadu X2 - odeslání/přijetí 4 bytů po sériové lince rychlostí 19200 baud

hsersetup B19200\_16, %00 ;nastavení přenosu 19200 baud / 16MHz

main:

hserin [1000,main],0,4 ; příjem 4 bytů do paměti sp

ptr = 0 ; vynulování ukazatele pozice v sp

hserout 0,(@ptrinc,@ptrinc,@ptrinc,@ptr) ; odeslání 4 bytů z sp

goto main ; skok na začátek programu

Jméno:

hserout

Syntaxe:

**HSEROUT break, ({#}data,{#}data...)**

*Break* - proměnná/konstanta (0 nebo 1) určuje, zda před odesláním dat bude nebo nebude odeslán signál „break“ (wake-up).

*Data* - proměnná nebo konstanta typu byte (0-255), která bude odeslána.  
Je možné použít znak „#“ pro zápis čísel ve formátu ASCII, pro zápis textu se používají uvozovky "Hello".

Popis:

Sériové odeslání dat přes pin spojený s interním hardwarovým převodníkem. Data jsou přenášena ve formátu (8 data bitů, no parity, 1 stop bit).

Příkaz **hserout** slouží pro sériový výstup dat, přes pin spojený s integrovaným hardwarovým sériovým převodníkem. (viz schéma) Není možné ho použít na ostatních pinech.

Nastavení přenosových vlastností se provádí před prvním použitím **hserout**, příkazem [hsersetup](http://www.picaxe.com/BASIC-Commands/Serial-RS232-Interfacing/hsersetup).  
  
Symbol „#“ převede hodnoty typu byte i word do ASCII podoby. Je-li b1 = 126, pak zápis #b1, odešle znakovou posloupnost "1" "2" "6".

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [serout](http://www.picaxe.com/commands/serout)
* [sertxd](http://www.picaxe.com/commands/sertxd)
* [hsersetup](http://www.picaxe.com/commands/hsersetup)
* [hserin](http://www.picaxe.com/commands/hserin)

Příklad:

Odeslání dat z paměti EEPROM umístěných na pozici 0 až 63, do sériového LCD.

hsersetup B2400\_4, %10 ; nastavení hw sériového přenosu 2400baud,N

main:

for b0 = 0 to 63 ; začátek for…next cyklu

read b0,b1 ; načtení hodnoty z pozice b0 do proměnné b1

hserout 0,(b1) ; odeslání bytu b1 do sériového LCD

next b0 ; konec for…next cyklu

Jméno:

hsersetup

Syntaxe:

**HSERSETUP OFF**

**HSERSETUP baud\_setup, mode**

*Baud\_setup* - proměnná/konstanta určující přenosovou rychlost:

|  |  |  |
| --- | --- | --- |
|  | B300\_X | kde X = |
|  | B600\_X | 4 pro 4MHz |
|  | B1200\_X | 8 pro 8MHZ |
|  | B2400\_X | 16 pro 16MHz |
|  | B4800\_X | 20 pro 20MHZ |
|  | B9600\_X | 32 pro 32MHz |
|  | B19200\_X | 40 pro 40MHz |
|  | B31250\_X | 64 pro 64MHz |
|  | B38400\_X |  |
|  | B57600\_X |  |
|  | B115200\_X |  |

*Mode* - proměnná/konstanta nastavující zvláštní funkce (ne všechny funkce jsou podporovány ve všech typech PICAXE):

bit0 - přenos dat do paměti na pozadí (ne pro řadu M2)  
bit1 - invertovat sériový datový výstup (0 = ‘T’, 1 = “N”)  
bit 2 - invertovat sériový datový vstup (0 = ‘T’, 1 = “N”)  
bit 3 - zakázání hserout (je-li 1 pak hserout vypnut a pin se chová jako běžný i/o pin)  
bit 4 - zakázání hserin (je-li 1 pak hserin vypnut a pin se chová jako běžný i/o pin)

Popis:

Konfigurace interního hardwarového sériového portu.  
Nastavení se provádí přes 3+4 bit parametru *mode*. Přenosová rychlost se nastavuje parametrem *baud\_setup,* kde jsou i předefinovány nejpoužívanější varianty přenosu (např. B9600\_4 pro rychlost 9600,n,8,1 při 4MHz taktování).

Hardwarový sériový vstup lze nakonfigurovat dvěma způsoby:

1) vyvolání hserin pouze příkazem (*mode* bit0 = 0).

2) automaticky na pozadí (*mode* bit0 = 1) (ne řada M2).

Při automatickém čtení sériových dat na pozadí, jsou data ukládána do „zápisníku“ cyklická vyrovnávací paměti, na pozici určenou *hserptr*. Po uložení dojde k nastavení *hserinflag* a zvýšení *fserptr*+1. Flag musí být nulován uživatelským programem, adresa posledního uloženého bytu je *fserptr*-1. Vyrovnávací paměť může přetéct bez varování. (více také u [setintflags](http://www.picaxe.com/BASIC-Commands/commands/setintflags) )

*Polarita*

Pokud bit1 = 0, pak polarita je typu „True“, stejně jak „Txxx“ pro příkaz [serout](http://www.picaxe.com/BASIC-Commands/commands/serout).  
Používá se s převodníkem MAX232, např. pro komunikaci s PC.

Pokud bit1 = 1, pak je polarita invertovaná, stejně jak „'Nxxx“ pro příkaz [serout](http://www.picaxe.com/BASIC-Commands/commands/serout).   
Používá se s odporovým převodníkem, nejčastěji pro komunikaci s ostatními sériovými periferiemi (např. AXE033 sériový LCD).

Na některých typech není možno měnit 2.bit (možno měnit jen na řadě X2). Proto je nutné použít převodník MAX232.

*Další technické informace*

Je možné nadefinovat vlastní přenosovou rychlost *baud\_setup*, podle vzorce:

přenosová\_rychlost = frekvence\_oscilátoru / (4 (*baud\_setup* + 1) )

pak

*baud\_setup* = (( frekvence\_oscilátoru / přenosová\_rychlost ) / 4 ) - 1

*Příklad:*

Pro protokol ODB2 je potřebná nestandardní rychlost 10400 baud, při frekvenci oscilátoru 4MHz. Vypočtěte hodnotu *baud\_setup.*

*baud\_setup* = (( frekvence\_oscilátoru / přenosová\_rychlost ) / 4 ) - 1

*baud\_setup* = ((4 000 000 / 10400) / 4 ) - 1 = 95 (zaokrouhleno)

Zpětný výpočet přesné přenosové rychlosti pro *baud\_setup = 95, 4MHz*:

přenosová\_rychlost = frekvence\_oscilátoru / (4 (*baud\_setup* + 1) )

přenosová\_rychlost = 4000 000 / (4 (95+1)) = 10416 (což je dostačující přesnost)

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [hserin](http://www.picaxe.com/commands/hserin)
* [hserout](http://www.picaxe.com/commands/hserout)

Příklad:

Odeslání dat z EEPROM pozice 0 až 63, po sériové lince do sériového LCD displeje.

hsersetup B9600\_4, %10 ; nastavení přenosu (9600 baud,N)

main:

for b0 = 0 to 63 ; začátek for…next cyklu

read b0,b1 ; načtení hodnoty z pozice b0 do proměnné b1

hserout 0,(b1) ; odeslání b1 po hserout do sériového LCD

next b0 ; konec for…next cyklu

Jméno:

hi2cin

Syntaxe:

**HI2CIN (proměnná,...)**

**HI2CIN location,(proměnná,...)**

**HI2CIN [newslave],(proměnná,...)** (pouze řada X2)

**HI2CIN [newslave],location,(proměnná,...)** (pouze řada X2)

Location - proměnná/konstanta typu byte/word specifikující umístění dat v i2c zařízení.

Proměnná(s) - proměnná pro příjem dat typu byte.

Newslave - nastavení nové slave adresy pro tento a následující příkazy I2C (pouze v řadě X2).

Popis:

Načtení hodnoty do *proměnná* přes rozhraní i2c.

Komunikace a zapojení i2c je popsáno v samostatném dokumentu „i2c Tutorial“.

Načte jeden nebo více bytů dat, do proměnných, od pozice specifikované v parametru *location*. Nastavení parametrů přenosu se provádí příkazem [hi2csetup](http://www.picaxe.com/commands/hi2csetup) a je třeba jej provést před prvním čtení/zápisem i2c dat.

Pro opakované změny slave adresy se využívá parametr *newslave*. Při špatném nakonfigurování i2c komunikace se uloží do všech proměnný hodnota 255 ($FF).

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [hi2cout](http://www.picaxe.com/commands/hi2cout)
* [hi2csetup](http://www.picaxe.com/commands/hi2csetup)

Příklad:

Komunikace i2c s časovým modulem DS1307, data jsou přenášena v BCD formátu.

hi2csetup i2cmaster, %11010000, i2cslow, i2cbyte

;nastavení i2c komunikace: PICAXE = master; slave adr.DS1307

main:

hi2cin 0,(b0,b1,b2,b3,b4,b5,b6,b7) ; načtení dat přes i2c

debug b1 ; výpis dat na PC

pause 2000 ; čekání 2s

goto main ; skok na začátek programu

Jméno:

hi2cout

Syntaxe:

**HI2COUT location,(proměnná,...)**

**HI2COUT (proměnná,...)**

**HI2COUT [newslave],location,(proměnná,...)** (pouze řada X2)

**HI2COUT [newslave],(proměnná,...)** (pouze řada X2)

Location - proměnná/konstanta typu byte/word specifikující umístění dat v i2c zařízení.

Proměnná(s) - proměnná pro příjem dat typu byte.

Newslave - nastavení nové slave adresy pro tento a následující příkazy I2C (pouze v řadě X2).

Popis:

Odeslání proměnné typu byte po sběrnici i2c, pokud je PICAXE ve stavu master.

Komunikace a zapojení i2c je popsáno v samostatném dokumentu „i2c Tutorial“.

Používá se pro odeslání jednoho nebo více bytů dat přes i2c na zařízení typu slave. Hodnota *location* se musí shodovat se specifikací i2c zařízení. [Hi2csetup](http://www.picaxe.com/commands/hi2csetup) je třeba provést před prvním čtení/zápisem i2c dat. Pro opakované změny slave adresy se využívá parametr *newslave*.

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [hi2cin](http://www.picaxe.com/commands/hi2cin)
* [hi2csetup](http://www.picaxe.com/commands/hi2csetup)

Příklad:

Nastavení „aktuálního“ času do časového modulu DS1307, komunikace probíhá v BCD kódu.

symbol seconds = b0

symbol mins = b1

symbol hour = b2

symbol day = b3

symbol date = b4

symbol month = b5

symbol year = b6

symbol control = b7

; nastavení PICAXE = master a nastavení slave adresy DS1307

hi2csetup i2cmaster, %11010000, i2cslow, i2cbyte

; zapsání času a data, např. 11:59:00 on Thurs 25/12/03

; hodnoty budou zapsány v BCD kódu = čísla budou uvozena znakem „$“

start\_clock:

let seconds = $00

let mins = $59

let hour = $11

let day = $03

let date = $25

let month = $12

let year = $03

; nastavení výstupu na 1Hz = každou 1s bude odeslán aktuální čas

let control = %00010000

; odeslání dat na časový modul

hi2cout 0,(seconds,mins,hour,day,date,month,year,control)

end

Jméno:

hi2csetup

Syntaxe:

**HI2CSETUP OFF**

**HI2CSETUP I2CSLAVE, slaveaddress**

**HI2CSETUP I2CMASTER, slaveaddress, mode, addresslen**

Master režim je, když PICAXE řídí i2c sběrnici. Tím ovládá jiná zařízení připojená na sběrnici např. EEPROM. Slave režim je když PICAXE je řízen jiným zařízením a nemůže řídit ostatní zařízení na sběrnici.

SlaveAddress je i2c slave adresa PICAXE

Mode, klíčová slova i2cfast (400kHz), nebo i2cslow (100kHz) pro 4MHz taktování. Pro vyšší taktování se používá i2cfast\_8, i2cslow\_8 pro 8MHz, atd.

Addresslen - klíčová slova i2cbyte, nebo i2cword. Označují délku adresy konkrétního zařízení. Např. některé paměti používají adresování typu word. NEJDE o délku přenášených dat, ta je vždy typu byte.

Popis:

Používá se k nastavení parametrů i2c komunikace a rozdělení rolí master - slave.

**Slave mód** na PICAXE je podporován pouze v řadě X1 a X2.

Slave Address

Je adresa, kterou je „označen“ PICAXE, je-li v režimu slave.   
Pro adresu je možné použít hodnot 1 až 127, ale zapsaných do bitů 7 až 1 (ne jak normálně 6 až 0), např. %1010000x. Bit 0 je využit na rozlišení čtení/zápisu.   
Pokud si nejste jistí správností adresy, je možno použit univerzální adresu pro i2c EEPROM = %10100000. Naopak nepoužívejte speciální i2c adresy (%00000000, %1111xxx, %0000xxxx).

Popis

Když je PICAXE v režimu slave může k němu master procesor přistupovat jako k 128B (X1, 20X2) nebo 256B (X2) sériové EEPROM typu 24LCxx.  
Chod programu v PICAXE nebude nijak ovlivněn, ke kolizím muže dojít pouze v případě použití příkazů ovlivňujících hardwarové přerušení (např. serout).

Uživatelský program může využívat uložená data. O jejich příchodu je informován nastavením **hi2cflag**, mazání musí provést uživatel, poslední hodnota je uložena na adrese **hi2clast**.

**Master mód - PICAXE řídí komunikaci na I2C sběrnici**

PICAXE řídí komunikaci a komunikuje s ostatními i2c slave zařízeními. Přístup k zařízením je řízen slave adresou zařízení.

Pokud PICAXE komunikuje jen s jedním slave zařízením, stačí použít **hi2csetup** jen jednou. Pokud komunikuje s více zařízeními, je třeba vždy nastavit slave adresu zařízení, se kterým chce komunikovat. Nastavení se provádí buď příkazem **hi2csetup,** nebo nastavením hodnoty newslave u příkazů [hi2cin](http://www.picaxe.com/commands/hi2cin) a [hi2cout](http://www.picaxe.com/commands/hi2cout).

Slave adresy

Slave adresy se liší pro různé typy i2c zařízení, viz tabulka. Pro populární řadu sériových pamětí EEPROM je adresa %1010xxxx. Některá zařízení např. 24LC16B, začleňuje od bitů 1-3 adresy paměťový blok který bude využit. Jiná zařízení mohou svoji adresu měnit pomocí vnějších přepínačů. Vždy je proto nutné zkontrolovat, zda jsou adresy na masteru i slavu stejně nastavené.

Bit 0 určuje zda slave zařízení má číst data, nebo může data odeslat. Tento bit se nastavuje při použití příkazů [hi2cin](http://www.picaxe.com/commands/hi2cin) a [hi2cout](http://www.picaxe.com/commands/hi2cout) ([readi2c](http://www.picaxe.com/commands/readi2c) a [writei2c](http://www.picaxe.com/commands/writei2c)).

Většina i2c zařízení používá 8 nebo 16 bitovou adresu, jsou ale výjimky které používají pouze 7 bitovou adresu, zde je nutné provést bitový posun doleva.

Rychlost

Rychlost komunikace se mění parametrem mode, klíčová slova i2cfast (400kHz), nebo i2cslow (100kHz) pro 4MHz taktování. Pro vyšší taktování se používá i2cfast\_8, i2cslow\_8 pro 8MHz, atd.  
Rychlost na sběrnici i2c se má řídit nejpomalejší připojenou periferií. (např. 100kHz pro DS1307)

Délka adres

Nastavuje se parametrem Addresslen, běžně i2c periférie používají jedno bytovou adresu (i2cbyte), nebo dvoj bytovou adresu (i2cword).   
Je nutné mít správně nastavený hi2csetup, [hi2cin](http://www.picaxe.com/commands/hi2cin) a [hi2cout](http://www.picaxe.com/commands/hi2cout).

Nastavení některých často používaných i2c periferií

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ zařízení** | **Slave adresa** | **Rychlost** | **Délka adresy** |
| 24LC01B EE 128 | %1010xxxx | i2cfast | i2cbyte |
| 24LC02B EE 256 | %1010xxxx | i2cfast | i2cbyte |
| 24LC04B EE 512 | %1010xxbx | i2cfast | i2cbyte |
| 24LC08B EE 1kb | %1010xbbx | i2cfast | i2cbyte |
| 24LC16B EE 2kb | %1010bbbx | i2cfast | i2cbyte |
| 24LC64 EE 8kb | %1010dddx | i2cfast | i2cword |
| 24LC128 EE 16kb | %1010dddx | i2cfast | i2cword |
| 24LC256 EE32kb | %1010dddx | i2cfast | i2cword |
| 24LC512 EE64kb | %1010dddx | i2cfast | i2cword |
| DS1307 RTC | %1101000x | i2cslow | i2cbyte |
| MAX6953 5x7 LED | %101ddddx | i2cfast | i2cbyte |
| AD5245 Digital Pot | %010110dx | i2cfast | i2cbyte |
| SRF08 Sonar | %1110000x | i2cfast | i2cbyte |
| AXE033 I2C LCD | %1100011x | i2cslow | i2cbyte |
| CMPS03 Compass | %1100000x | i2cfast | i2cbyte |
| SPE030 Speech | %1100010x | i2cfast | i2cbyte |

x = bit se ignoruje

b = používá se pro definování bloku paměti (volba vnitřní paměťové stránky)

d = podle nastavení zařízení (adresa podle polarity na adresních pinech)

**hi2csetup off**

Vypne i2c komunikaci a vrátí pin do „běžného“ I/O režimu.  
V praxi se příliš nepoužívá, protože je málo pravděpodobné, že v průběhu programu dojde k odpojení i2c periferií.

Vliv zvýšení kmitočtu na i2c

Ujistěte se, že při změně kmitočtu, došlo také ke změně parametru *mode*.   
Pro 8MHz - i2cfast\_8, i2cslow\_8, pro 16MHz - i2cfast\_16, i2cslow\_16, atd.

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [hi2cin](http://www.picaxe.com/commands/hi2cin)
* [hi2cout](http://www.picaxe.com/commands/hi2cout)

Příklad:

Odeslání dat do i2c EEPROM paměti.

init:

; nastavení PICAXE na master, adresa zařízení %10100000, komunikační ; rychlost pomalá, adresa délky byte

hi2csetup i2cmaster, %10100000, i2cslow, i2cbyte

main:

inc b1 ; zvýšení b1 o +1

hi2cout 0,(b1) ; odeslání b1 se specifikací 0 po i2c

pause 500 ; čekej 0,5s

goto main ; skok na začátek programu

Jméno:

kbin

Syntaxe:

**KBIN proměnná**

**KBIN [timeout], proměnná**

**KBIN [timeout, address],proměnná**

**KBIN #proměnná**

**KBIN [timeout], #proměnná**

**KBIN [timeout, address], #proměnná**

Proměnná - hodnota stisknuté klávesy

Timeout - proměnná/konstanta určující dobu čekání v milisekundách

Address - adresa návěští kam dojde ke skoku při překročení časového limitu.

Popis:

Čekej, dokud není stisknuta klávesa a přijat její kód.   
Podobný příkaz jako starší příkaz [keyin](http://www.picaxe.com/commands/keyin), ale umožňuje omezit dobu čekání.

Příkaz čeká na nový stisk klávesy z PS2 klávesnice připojené přímo k PICAXE.  
(Popis připojení je v anglické verzi u příkazu keyled - [manual](http://www.picaxe.com/manuals).) Po dobu čekání jsou pozastaveny všechny procesy. Hodnota klávesy se uloží do *proměnné*.

Poznámka: Klávesnice neposílá ASCII kódy, interní kódy se musí převést na ASCII pomocí převodní tabulky. Při stisku některých kláves jsou odeslány dva byty - první $E0 je ignorován (ztracen) a druhý uložen do proměnné.  
Proto není možné spolehlivě detekovat klávesy PAUSE, PRINT SCREEN a problémy mohou také nastat při použití NUM LOCK.  
Pro zjednodušení práce s klávesnicí, byl u řady M2 přidán parametr *#proměnná*  
, kdy to proměnné je uložen rovnou ASCII kód a pro nepodporované znaky „?“ (např. CTRL).   
Pro zápis kódů se používá hexadecimálního tvaru = používá se $xx.

Po dobu vykonávání příkazu je automaticky nastaveno taktování na 4MHz.

Platí pro:

14M2, 18M2+, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [kbled](http://www.picaxe.com/commands/kbled)
* [keyin](http://www.picaxe.com/commands/keyin)
* [keyled](http://www.picaxe.com/commands/keyled)

Příklad:

Rozsvícení LED na b.1 při stisku klávesy na PS2 klávesnici, s kódem $25 a zhasnutí  
při kódu $45.

main:

kbin [1000,main],b1 ; čekání na stisk klávesy na PS2 klávesnici,  
 ; příkaz opakuje každou 1s

if b1 = $45 then ; pokud stisknuta klávesa $45, zhasnutí LED

low b.1

end if

if b1= $25 then ; pokud stisknuta klávesa $25, rozsvícení LED

high b.1

end if

goto main

Jméno:

kbled

Syntaxe:

**KBLED mask**

Mask - proměnná/konstanta určující, které LED budou na PS2 klávesnici rozsvíceny.

Popis:

Rozsvítí/zhasne LED na PS2 klávesnici.

Příkaz řídí rozsvěcení LED kontrolek na PS2 klávesnici, připojené přímo k PICAXE. Svit LED je řízen hodnotou masky:

|  |  |
| --- | --- |
| Bit 0 | Scroll Lock (1=on, 0=off) |
| Bit 1 | Num Lock (1=on, 0=off) |
| Bit 2 | Caps Lock (1=on, 0=off) |
| Bit 3-6 | není použito |
| Bit 7 | Disable Flash (1=no flash, 0=flash) |

Obnovení masky se provede jejím vynulováním. Tím je nastavena i „flash“ funkce = zablikání LED při stisku klávesy. Po zrušení „flash“ funkce (bit 7 = 1) lze rozsvěcení LED ovládat jen přes bity 0-2 masky.

Příkaz funguje jen při taktovací frekvenci 4MHz.

Platí pro:

14M2, 18M2+, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [kbin](http://www.picaxe.com/commands/kbin)
* [keyin](http://www.picaxe.com/commands/keyin)
* [keyled](http://www.picaxe.com/commands/keyled)

Jméno:

owin

Syntaxe:

**OWIN pin,mode,(proměnná, proměnná...)**

Pin - proměnná/konstanta určující vstupní komunikační pin.

Mode - proměnná/konstanta určující režim komunikace.

Funkce jednotlivých bitů parametru *mode*:

|  |  |
| --- | --- |
| Bit 0 | * reset před odesláním dat |
| Bit 1 | * reset po odeslání dat |
| Bit 2 | * bitový režim ( 1 - přenáší se 1 bit, 0 - přenáší se 8 bitů (1 byte)) |
| Bit 3 | * nastavení „pullup“ režimu (poslechového režimu) po odeslání dat |

Jsou předdefinovány následující hodnoty:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | ownoreset | 4 | ownoreset\_bit |
| 1 | owresetbefore | 5 | owresetbefore\_bit |
| 2 | owresetafter | 6 | owresetafter\_bit |
| 3 | owresetboth | 7 | owresetboth\_bit |

Proměnná - proměnná/é do nichž se uloží načtená data.

Popis:

Umožňuje načítat data z periferií po jednom kabelu.

Čtení dat (bit/byte) z jednoho zařízení, připojeného ke vstupnímu pinu, s nastavitelnými „reset“ (nulovacími) impulzy před a po čtení.   
Příkaz nelze použít na pinu C.6 u PICAXE 20X2.

Podrobný popis datové komunikace po jednom kabelu je uvedena v originální dokumentaci „One-Wire Tutorial“.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [owout](http://www.picaxe.com/commands/owout)
* [readowsn](http://www.picaxe.com/commands/readowsn)
* [readtemp](http://www.picaxe.com/commands/readtemp)
* [readtemp12](http://www.picaxe.com/commands/readtemp12)

Příklad:

Komunikace po jednom kabelu - načtení teploty z čidla DS18B20.

main:

owout C.1,%1001,($CC,$44) ; nastavení zařízení pro komunikaci

pause 750 ; čekání 750ms se silným „pullup“

owout C.1,%0001,($CC,$BE) ; příkaz pro zaslání teploty

owin C.1,%0000,(b0,b1) ; načtení teploty do 2x byte proměnné

sertxd (#w0,CR,LF) ; odeslání teploty do PC

goto main

Jméno:

owout

Syntaxe:

**OWOUT pin,mode,(proměnná,proměnná...)**

Pin - proměnná/konstanta určující výstupní komunikační pin.

Mode - proměnná/konstanta určující režim komunikace.

Funkce jednotlivých bitů parametru *mode*:

|  |  |
| --- | --- |
| Bit 0 | * reset před odesláním dat |
| Bit 1 | * reset po odeslání dat |
| Bit 2 | * bitový režim ( 1 - přenáší se 1 bit, 0 - přenáší se 8 bitů (1 byte)) |
| Bit 3 | * nastavení „pullup“ režimu (poslechového režimu) po odeslání dat |

Jsou předdefinovány následující hodnoty:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | ownoreset | 4 | ownoreset\_bit |
| 1 | owresetbefore | 5 | owresetbefore\_bit |
| 2 | owresetafter | 6 | owresetafter\_bit |
| 3 | owresetboth | 7 | owresetboth\_bit |

Proměnná - proměnná/konstanta obsahující data, která budou odeslána.

Popis:

Umožňuje posílat data do periferií po jednom kabelu.

Zápis dat (bit/byte) do jednoho zařízení, připojeného k výstupnímu pinu, s nastavitelnými „reset“ (nulovacími) impulzy před a po čtení.   
Příkaz nelze použít na pinu C.6 u PICAXE 20X2.

Podrobný popis datové komunikace po jednom kabelu je uvedena v originální dokumentaci „One-Wire Tutorial“.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [owin](http://www.picaxe.com/commands/owin)
* [readowsn](http://www.picaxe.com/commands/readowsn)
* [readtemp](http://www.picaxe.com/commands/readtemp)
* [readtemp12](http://www.picaxe.com/commands/readtemp12)

Příklad:

Komunikace po jednom kabelu - načtení teploty z čidla DS18B20.

main:

owout C.1,%1001,($CC,$44) ; nastavení zařízení pro komunikaci

pause 750 ; čekání 750ms se silným „pullup“

owout C.1,%0001,($CC,$BE) ; příkaz pro zaslání teploty

owin C.1,%0000,(b0,b1) ; načtení teploty do 2x byte proměnné

sertxd (#w0,CR,LF) ; odeslání teploty do PC

goto main

Jméno:

readowsn

Syntaxe:

**READOWSN pin**

Pin - proměnná/konstanta určující, který vstupní pin se bude používat.

Popis:

Načtení sériového čísla z libovolného Dallas/Maxim „jednodrátového“ zařízení.

Příkaz načte sériové číslo z Dallas, „jednodrátového“ zařízení (1-wire device), např. DS18B20 - teplotní čidlo, DS2415 - časový modul, DS1990A iButton - identifikačkní čip… (u DS1990A je sériové číslo také vypáleno laserem na povrchu čipu)

Při provádění příkazu se načte „rodinný - typový“ kód do b6, sériové výrobní číslo   
do b7 až b12 a kontrolní součet do b13.  
Během načítání hodnoty by se proměnné b6 až b13, neměly používat.

Příkaz nelze použít na pinech:

|  |  |
| --- | --- |
| 08, 08M, 08M2 | C.3 |
| 14M, 14M2 | C.3 |
| 18M2 | C.4, C.5 |
| 20M, 20M2, 20X2 | C.6 |

Platí pro:

Vše (kromě 08, 18, 28, 28A)

Příbuzné:

* [owin](http://www.picaxe.com/commands/owin)
* [owout](http://www.picaxe.com/commands/owout)
* [readtemp](http://www.picaxe.com/commands/readtemp)
* [readtemp12](http://www.picaxe.com/commands/readtemp12)

Příklad:

Načtení identifikačního - sériového čísla z Dallas součástky na pinu C.2.

main:

; vynulování typového (family) kódu v b6

let b6 = 0

; čekání na načtení sériového čísla ze vstupu C.2

loop1:

readowsn C.2

if b6 = 0 then loop1

; jednoduchá kontrola načtení

; je málo pravděpodobné, že b12 bude rovno $FF,

; proto, je-li b12 = $FF je pravděpodobné,

; že nedošlo k úplnému načtení kódu

if b12 = $FF then main

; pokud je načtený kód v pořádku, pošly hodnoty do PC a opakuj

debug

pause 1000

goto main

Jméno:

srlatch

Syntaxe:

**SRLATCH config1, config2**

Config1 - proměnná/konstanta nastavující vlastnosti klopného obvodu

|  |  |
| --- | --- |
| Bit 7 | = 1 SR klopný obvod je aktivní |
|  | = 0 SR klopný obvod je pasivní |
| Bit 6-4 | SR časovač - nastaví časování klopného obvodu (počet bitů pro děličku): |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | bity | dělička | 16MHz | 8MHz | 4MHz | | = 000 | 1/4 | 0.25us | 0.5us | 1us | | = 001 | 1/8 | 0.5 | 1 | 2 | | = 010 | 1/16 | 1 | 2 | 4 | | = 011 | 1/32 | 2 | 4 | 8 | | = 100 | 1/64 | 4 | 8 | 16 | | = 101 | 1/128 | 8 | 16 | 32 | | = 110 | 1/256 | 16 | 32 | 64 | | = 111 | 1/512 | 32 | 64 | 128 | |
| Bit 3 | = 1 výstup Q se přenáší na výstupní pin SRQ (je-li přítomen) |
|  | = 0 výstup Q se nepřenáší na pin SRQ |
| Bit 2 | = 1 výstup NOT\_Q se přenáší na výstupní pin SRNQ (je-li přítomen) |
|  | = 0 výstup NOT\_Q se nepřenáší na pin SRNQ |
| Bit 1 | = 0 nepoužívá se, ponechat hodnotu 0 |
| Bit 0 | = 0 nepoužívá se, ponechat hodnotu 0 |

Poznámka: Ne u všech typů PICAXE jsou vyvedeny piny SRQ a SRNQ.  
U modelu 28X2/40X2 bude omezena funkce [debug](http://www.picaxe.com/commands/debug) a [sertxd](http://www.picaxe.com/commands/sertxd) při použití výstupu SRNQ.

Config2 - proměnná/konstanta specifikující nastavení SR klopného obvodu, funkce pinů se liší podle typu PICAXE. Při nastavení pinu na 0 nemá žádný efekt.

Pro řadu 20X2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bit 7 = 1 | při HW přerušení INT1 nastaví Q | ([hintsetup](http://www.picaxe.com/commands/hintsetup)) |
|  | Bit 6 = 1 | časovač nastaví Q | (viz. výše) |
|  | Bit 5 = 1 | C2 komparátor nastaví Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 4 = 1 | C1 komparátor nastaví Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 3 = 1 | při HW přerušení INT1 vynuluje Q | ([hintsetup](http://www.picaxe.com/commands/hintsetup)) |
|  | Bit 2 = 1 | časovač vynuluje Q | (viz. výše) |
|  | Bit 1 = 1 | C2 komparátor vynuluje Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 0 = 1 | C1 komparátor vynuluje Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |

Pro řadu 28X2/40X2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bit 7 = 1 | SRI nastaví Q |  |
|  | Bit 6 = 1 | časovač nastaví Q | (viz. výše) |
|  | Bit 5 = 1 | C2 komparátor nastaví Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 4 = 1 | C1 komparátor nastaví Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 3 = 1 | SRI vynuluje Q |  |
|  | Bit 2 = 1 | časovač vynuluje Q | (viz. výše) |
|  | Bit 1 = 1 | C2 komparátor vynuluje Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |
|  | Bit 0 = 1 | C1 komparátor vynuluje Q | ([compsetup](http://www.picaxe.com/commands/compsetup)) |

U 28X2/40X2 může pin SRI sloužit pro nastavení i vynulování klopného obvodu,   
bit 3 a 7. Nenastavujte oba bity současně!

Pro řadu M2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bit 7 = 1 | SRI nastaví Q |  |
|  | Bit 6 = 1 | časovač nastaví Q | (viz výše) |
|  | Bit 5 = 0 | není použit, nechte 0 |  |
|  | Bit 4 = 0 | není použit, nechte 0 |  |
|  | Bit 3 = 1 | SRI vynuluje Q |  |
|  | Bit 2 = 1 | časovač vynuluje Q | (viz výše) |
|  | Bit 1 = 0 | není použit, nechte 0 |  |
|  | Bit 0 = 0 | není použit, nechte 0 |  |

U řady M2, může pin SRI sloužit pro nastavení i vynulování klopného obvodu,   
bit 3 a 7. Nenastavujte oba bity současně!

Popis:

Nastaví vnitřní hardwarový SR klopný obvod.

Klopný obvod je možné nastavit (Q = 1) příkazem [SRSET](http://www.picaxe.com/commands/srset), nebo některou periferií (časovač…), stejně jako vynulovat klopný obvod (Q = 0) příkazem [SRRESET](http://www.picaxe.com/commands/srreset), nebo některou periferií.  
Činnost klopného obvodu může být nezávislá na programu - nastavení klopného obvodu vstupními piny, či přerušením a výstup na výstupní Q pin.

SR klopný obvod může být řízen vnitřními hodinami, pak lze Q výstup nakonfigurovat podobně, jako časovač typu „555“.

Výstup Q/NOT\_Q, lze přivést na piny SRQ/SRNQ (jsou-li přítomné). Toto nastavení však není defaultní, proto je na začátku vždy třeba provést konfiguraci příkazem **srlatch**.

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [srset](http://www.picaxe.com/commands/srset)
* [srreset](http://www.picaxe.com/commands/srreset)

Příklad:

Programové nastavení a ovládání SR klopného obvodu.

ini: ; inicializace použitých pinů

low B.1

high C.4

srlatch %10001100, %00000000 ; nastavení konfigurace SR obvodu

main:

srset ; nastavení SR klopného obvodu

pause 5000 ; čekání 5s

srreset ; vynulování SR klopného obvodu

pause 5000 ; čekání 5s

goto main ; skok na „main“

Jméno:

srreset

Syntaxe:

**SRRESET**

Popis:

Vynuluje hardwarový SR klopný obvod.

Příkazy **srset**/**srreset** umožňují programové nastavení/vynulování hardwarového SR klopného obvodu. Toto nastavení/vynulování lze také provést pomocí vnější a vnitřních periferií, více u příkazu [srlatch](http://www.picaxe.com/commands/srlatch).

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [srlatch](http://www.picaxe.com/commands/srlatch)
* [srset](http://www.picaxe.com/commands/srset)

Příklad:

Programové nastavení a ovládání SR klopného obvodu.

ini: ; inicializace použitých pinů

low B.1

high C.4

srlatch %10001100, %00000000 ; nastavení konfigurace SR obvodu

main:

srset ; nastavení SR klopného obvodu

pause 5000 ; čekání 5s

srreset ; vynulování SR klopného obvodu

pause 5000 ; čekání 5s

goto main ; skok na „main“

Jméno:

srset

Syntaxe:

**SRSET**

Popis:

Nastaví hardwarový SR klopný obvod.

Příkazy **srset**/**srreset** umožňují programové nastavení/vynulování hardwarového SR klopného obvodu. Toto nastavení/vynulování lze také provést pomocí vnější a vnitřních periferií, více u příkazu [srlatch](http://www.picaxe.com/commands/srlatch).

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [srlatch](http://www.picaxe.com/commands/srlatch)
* [srreset](http://www.picaxe.com/commands/srreset)

Příklad:

Programové nastavení a ovládání SR klopného obvodu.

ini: ; inicializace použitých pinů

low B.1

high C.4

srlatch %10001100, %00000000 ; nastavení konfigurace SR obvodu

main:

srset ; nastavení SR klopného obvodu

pause 5000 ; čekání 5s

srreset ; vynulování SR klopného obvodu

pause 5000 ; čekání 5s

goto main ; skok na „main“

Jméno:

uniin

Syntaxe:

**UNIIN pin, device, command, (var, var...)**

**UNIIN pin, device, command, address, address, (var, var...)**

Pin - proměnná/konstanta označující I/O pin (B.3, C.1 ... viz schéma zapojení)

Device - je specifikace typu UNI/O zařízení, %10100000 pro EEPROM zařízení

Command - nastavuje typ čtení:

|  |  |
| --- | --- |
| UNI\_READ | čte ze specifikované adresy - *address* |
| UNI\_CRRD | čte z aktuální adresy |
| UNI\_RDSR | čte stavový byte |

Address - proměnná/konstanta typu 2x byte, používá se pouze pro UNI\_READ.

Proměnná - proměnná typu byte pro přijímaná data.

Popis:

Načte data z UNI/O rozhraní do PICAXE a uloží je do *proměnné*.

Příkaz umožnuje číst data z vnějšího UNI/O zařízení, např. 11LCxxx sériová EEPROM. UNI/O potřebuje pro sériovou komunikaci pouze jeden vstupně-výstupní pin, který se propojí s vnějším zařízením. Je vhodné připojit rezistor 4k7, jako pullup rezistor. (mezi vstupní pin a kladnému napětí)

Poznámka: Při zapnutí je vnější zařízení v úsporném stavu, proto je potřeba jej „probudit“ přivedením náběžné hrany na vstupní pin. Následně je možné používat příkazy pro UNI/O komunikaci.

Příkaz nelze použít na pinu C.6 u řady 20X2.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [uniout](http://www.picaxe.com/commands/uniout)

Příklad:

Zápis a čtení dat do/z EEPROM, přes rozhraní UNI/O.

reset\_uni:

pulsout C.3, 1 ; pulz pro probuzení vnějšího UNI/O zařízení

main:

inc b1 ; zvětši b1 o +1

uniout C.3, %10100000, UNI\_WRSR, (0) ; vymazání stavového bytu

uniout C.3, %10100000, UNI\_WREN ; povolení zápisu

uniout C.3, %10100000, UNI\_WRITE, 0, 1, (b1) ; zápis b1 na adr. 0,1

pause 10 ; čekání pro provedení zápisu

uniout C.3, %10100000, UNI\_WRDI ; zakázání zápisu

pause 1000 ; čekání 1s

uniin C.3, %10100000, UNI\_READ, 0, 1, (b2) ; načtení b2 z adr.0,1

debug ; odeslání do PC

goto main ; opakování - skok na „main“

Jméno:

uniout

Syntaxe:

**UNIOUT pin, device, command**

**UNIOUT pin, device, command, (data)**

**UNIOUT pin, device, command, address, address, (data, data...)**

Pin - proměnná/konstanta označující I/O pin (B.3, C.1 ... viz schéma zapojení)

Device - je specifikace typu UNI/O zařízení, %10100000 pro EEPROM zařízení

Command - nastavuje typ zápisu:

|  |  |
| --- | --- |
| UNI\_WRITE | zápis dat |
| UNI\_WREN | zápis povolen |
| UNI\_WRDI | zápis zakázán |
| UNI\_WRSR | zápis stavového bytu |
| UNI\_ERAL | vymazání všeho - 0 |
| UNI\_SETAL | nastavení všeho - 1 |

Address proměnná/konstanta typu 2x byte, používá se pouze pro UNI\_WRITE.

Data - proměnná/konstanta typu byte pro odesílaná data.

Popis:

Zápis dat do UNI/O zařízení.

Poznámka: UNI/O zařízení používají 16 bytové stránkování a jeden zápis nesmí přesáhnout stránku. Uložení 10 hodnot od adresy 5 je možné, ale uložení 10 hodnot od adresy 10 není možné, došlo by k překročení hranice stránky.

Příkaz umožnuje číst data z vnějšího UNI/O zařízení, např. 11LCxxx sériová EEPROM. UNI/O potřebuje pro sériovou komunikaci pouze jeden vstupně-výstupní pin, který se propojí s vnějším zařízením. Je vhodné připojit rezistor 4k7, jako pullup rezistor. (mezi vstupní pin a kladnému napětí)

Poznámka: Při zapnutí je vnější zařízení v úsporném stavu, proto je potřeba jej „probudit“ přivedením náběžné hrany na vstupní pin. Následně je možné používat příkazy pro UNI/O komunikaci.

Příkaz nelze použít na pinu C.6 u řady 20X2.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [uniin](http://www.picaxe.com/commands/uniin)

Příklad:

Zápis a čtení dat do/z EEPROM, přes rozhraní UNI/O.

reset\_uni:

pulsout C.3, 1 ; pulz pro probuzení vnějšího UNI/O zařízení

main:

inc b1 ; zvětši b1 o +1

uniout C.3, %10100000, UNI\_WRSR, (0) ; vymazání stavového bytu

uniout C.3, %10100000, UNI\_WREN ; povolení zápisu

uniout C.3, %10100000, UNI\_WRITE, 0, 1, (b1) ; zápis b1 na adr. 0,1

pause 10 ; čekání pro provedení zápisu

uniout C.3, %10100000, UNI\_WRDI ; zakázání zápisu

pause 1000 ; čekání 1s

uniin C.3, %10100000, UNI\_READ, 0, 1, (b2) ; načtení b2 z adr.0,1

debug ; odeslání do PC

goto main ; opakování - skok na „main“

Jméno:

hpwm

Syntaxe:

**HPWM mode, polarity, setting, period, duty**

**HPWM PWMDIV4, mode, polarity, setting, period, duty**

**HPWM PWMDIV16, mode, polarity, setting, period, duty**

**HPWM PWMDIV64, mode, polarity, setting, period, duty**

**HPWM OFF**

Mode - proměnná/konstanta nastavující režim pwm hardwaru (pwm mode):

|  |  |  |
| --- | --- | --- |
|  | pwmsingle | 0 |
|  | pwmhalf | 1 |
|  | pwmfull\_f | 2 |
|  | pwmfull\_r | 3 |

Polarity - proměnná/konstanta určující aktivní polaritu (DCBA):

|  |  |  |
| --- | --- | --- |
|  | pwmHHHH | 0 |
|  | pwmLHLH | 1 |
|  | pwmHLHL | 2 |
|  | pwmLLLL | 3 |

Setting - proměnná/konstanta nastavující zvolený režim *mode*:

|  |  |  |
| --- | --- | --- |
|  | single mode | bitová maska %0000 - %1111, pro on/off DCBA |
|  | half mode | zpoždění (hodnota 0-127) |
|  | full mode | nepoužívá se, použijte 0 |

Period - proměnná/konstanta (0-255) určující dobu provádění PWM v počtu celých provedených period.

Duty - proměnná/konstanta (0-1023) nastavuje střídu PWM (dobu plnění).

**PWMDIV** - umožňuje zpomalit PWM frekvenci a to dělením základní frekvence 4, 16 nebo 64. Hodnota 64 není podporování všemi PICAXE.

Pro výpočet požadované hodnoty *duty* je možné použít „'PWMout Wizard“ v „PICAXE>Wizards menu“ v Programovacím editoru. Více informací u příkazu [pwmout](http://www.picaxe.com/commands/pwmout).

Před použitím hpwm je třeba prostudovat specifikaci čipu, platí především pro 28X1, 28X2, 40X2 a 40X1.

Popis:

Hardwarové PWM je moderní metoda regulace PWM. Počet výstupů a režimů se liší podle typu PICAXE.

**Hpwm** nelze použít současně s příkazem [pwmout](http://www.picaxe.com/commands/pwmout) 2 (28/40 pinové verze). Avšak [pwmout](http://www.picaxe.com/commands/pwmout) 1 současně použít možné je.

**Hpwm** umožňuje přístup k řadiči PIC mikrokontroleru. Umožňuje až 4 výstupy A-D, které jsou při použití příkazu automaticky nastaveny na výstupní.

|  |  |
| --- | --- |
| **Na 20 pinových obvodech:** | **Na 14 pinových obvodech:** |
| A - input (C.5) | A - input (C.5) |
| B - input (C.4) | B - input (C.4) |
| C - input (C.3) | C - input (C.3) |
| D - output (B.4) | D - output (C.2) |

|  |  |
| --- | --- |
| **Na 28 pinových obvodech:** | **Na 40 pinových obvodech:** |
| A - input (C.2) | A - input (C.2) |
| B - output (B.2) | B - input (D.5) |
| C - output (B.1) | C - input (D.6) |
| D - output (B.4) | D - input (D.7) |

Ne všechny piny se používají ve všech režimech, proto nevyužité piny zůstávají jako normální i/o piny.

|  |  |  |
| --- | --- | --- |
|  | single | - A a/nebo B a/nebo C a/nebo D (každý bit je volitelný) |
|  | half | - pouze A, B |
|  | full | - A, B, C, D |

Aktivní polaritu každé dvojice pinů lze nastavit nastavením bitů:

|  |  |  |  |
| --- | --- | --- | --- |
|  | pwm\_HHHH | A a C aktivní vysoká | B a D aktivní vysoká |
|  | pwm\_LHLH | A a C aktivní vysoká | B a D aktivní nízká |
|  | pwm\_HLHL | A a C aktivní nízká | B a D aktivní vysoká |
|  | pwm\_LLLL | A a C aktivní nízká | B a D aktivní nízká |

Je důležité použít při nastavení - aktivní při vysoké úrovni, pull-down rezistor   
(4k7 mezi pin a 0V) a při nastavení - aktivní při nízké úrovni, pull-up rezistor   
(4k7 mezi pin a 5V).  
Je to zejména důležité z důvodu definování stavu vstupu FET tranzistoru, při zapnutí PICACE.

Single Mode

|  |  |
| --- | --- |
| Podporováno: | 20X2, 28X1, 28X2, 28X2-3V, 40X2, 40X2-3V |
| Nepodporováno: | 14M, 14M2, 20M2, 28X2-5V, 40X1, 40X2-5V |

Každý pin PWM - A, B, C, D - je řízen nezávisle, podobně jako příkaz [pwmout](http://www.picaxe.com/commands/pwmout). Ovšem řízení je současně a více pinech, proto se využívá především:

1) Jako ekvivalent příkazu [pwmout](http://www.picaxe.com/commands/pwmout) na více pinech.

2) Signál PWM je na více (až 4) pinech ve stejnou dobu. To je výhodné při ovládání výkonu na více zařízeních současně.

Nastavení výstupních pinů se provede parametrem *nastavení*. Bity %1111 odpovídají pinům D-C-B-A.

Poloviční režim (podporují všechny typy)

V polovičním režimu je ovládána jen polovina výstupů - A a B. Výstupy C a D se nepoužívají. PWM signál je vyslán na pin A, doplňkový PWM signál je vyslán   
na pin B. Zde je velmi důležité nastavení prodlevy v parametru *nastavení*, při nesprávném nastavení může dojít k zničení PWM modulu, případně externího řídícího modulu (současné sepnutí více FET do zkratu). Parametr prodlevy je v rozsahu (0-127) a zajišťuje zpoždění - (prodleva x perioda oscilátoru) / 4.(pozn. perioda oscilátoru = 1 / frekvence oscilátoru; pro 4MHz: 0,000 000 25 [s] = 1 / 4 000 000[Hz])

Tato hodnota musí odpovídat on/off charakteristice použitého FET tranzistoru.

Plný režim (podporují všechny modely)

Je řízen celý PWM modul a výstup je na všech pinech A, B, C a D.

V přímém režimu (pwmfull\_f) je A nastaven do aktivního stavu a D je modulován. B a C jsou neaktivní. V reverzním režimu (pwmfull\_r) je C nastaven do aktivního stavu a B je modulován. A a D jsou neaktivní. Definování zpoždění nebývá vyžadováno s výjimkou 100% plnění, zde by mohlo dojít ke kolizi.   
V tomto případě se doporučuje:  
1) vypnout PWM před změnou směru,  
2) použít speciální FET ovladač, který se zapne rychleji, než se vypne.

Platí pro:

14M, 14M2, 20M2, 20X2, 28X1, 28X2, 40X2

Příbuzné:

* [hpwmduty](http://www.picaxe.com/commands/hpwmduty)
* [pwm](http://www.picaxe.com/commands/pwm)
* [pwmout](http://www.picaxe.com/commands/pwmout)

Příklad:

Povolení PWM signálu a změna plnění - střídy.

init:

hpwm 0,0,%1111,150,100 ; start pwm

main:

hpwmduty 150 ; nastavení pwm plnění

pause 1000 ; čekání 1 s

hpwmduty 50 ; nastavení nového pwm plnění

pause 1000 ; čekání 1 s

goto main ; skok na „main“

Jméno:

hpwmduty

Syntaxe:

**HPWMDUTY duty**

Duty - proměnná/konstanta (0-1023) která nastaví plnění - střídu PWM signálu.

Popis:

Nastaví plnění - střídu PWM signálu na novou hodnotu *duty*.

Příkaz **hpwmduty** lze použít k upravení PWM cyklu, bez obnovení vnitřního časovače (na rozdíl od příkazu [hpwm](http://www.picaxe.com/commands/hpwm)). Příkaz **hpwmduty** lze použít až po příkazu [hpwm](http://www.picaxe.com/commands/hpwm).

Platí pro:

14M2, 20M2, 20X2, 28X1, 28X2, 40X2

Příbuzné:

* [hpwm](http://www.picaxe.com/commands/hpwm)
* [pwmout](http://www.picaxe.com/commands/pwmout)
* [pwmduty](http://www.picaxe.com/commands/pwmduty)

Příklad:

Povolení PWM signálu a změna plnění - střídy.

init:

hpwm 0,0,%1111,150,100 ; start pwm

main:

hpwmduty 150 ; nastavení pwm plnění

pause 1000 ; čekání 1 s

hpwmduty 50 ; nastavení nového pwm plnění

pause 1000 ; čekání 1 s

goto main ; skok na „main“

Jméno:

keyin

Syntaxe:

**KEYIN**

Popis:

**Tento příkaz je zastaralý, zvažte použití příkazu** [kbin](http://www.picaxe.com/commands/kbin)**.**

Počká, dokud není přijata nová klávesa - stisk nové klávesy.

Příkaz čeká na nový stisk klávesy z PS2 klávesnice připojené přímo k PICAXE.  
(Popis připojení je v anglické verzi u příkazu keyled - [manual](http://www.picaxe.com/manuals).) Po dobu čekání jsou pozastaveny všechny procesy. Hodnota klávesy se uloží do předdefinované proměnné „keyvalue“.  
Poznámka: Klávesnice neposílá ASCII kódy, interní kódy se musí převést na ASCII pomocí převodní tabulky. Při stisku některých kláves jsou odeslány dva byty - první $E0 je ignorován (ztracen) a druhý uložen do proměnné.  
Proto není možné spolehlivě detekovat klávesy PAUSE, PRINT SCREEN a problémy mohou také nastat při použití NUM LOCK.  
  
Pro zápis kódů se používá hexadecimálního tvaru = používá se $xx.

Příkaz funguje pouze při taktovací frekvenci 4MHz.

Platí pro:

18A, 18M, 18X, 28X, 40X

Příbuzné:

* [kbin](http://www.picaxe.com/commands/kbin)
* [kbled](http://www.picaxe.com/commands/kbled)
* [keyled](http://www.picaxe.com/commands/keyled)

Příklad:

Rozsvícení LED na b.1 při stisku klávesy na PS2 klávesnici, s kódem $25 a zhasnutí  
při kódu $45.

main:

keyin ; čekání na stisk klávesy na PS2 klávesnici,

if keyvalue = $45 then ; pokud stisknuta klávesa $45, zhasnutí LED

low b.1

end if

if keyvalue = $25 then ; pokud stisknuta klávesa $25, rozsvícení LED

high b.1

end if

goto main

Jméno:

keyled

Syntaxe:

**KEYLED mask**

Mask - proměnná/konstanta určující které LED budou použity.

Popis:

**Tento příkaz je zastaralý, zvažte použití příkazu** [kbled](http://www.picaxe.com/commands/kbled)**.**

Rozsvítí/zhasne LED na PS2 klávesnici.

Příkaz řídí rozsvěcení LED kontrolek na PS2 klávesnici, připojené přímo k PICAXE. Svit LED je řízen hodnotou masky:

|  |  |
| --- | --- |
| Bit 0 | Scroll Lock (1=on, 0=off) |
| Bit 1 | Num Lock (1=on, 0=off) |
| Bit 2 | Caps Lock (1=on, 0=off) |
| Bit 3-6 | není použito |
| Bit 7 | Disable Flash (1=no flash, 0=flash) |

Obnovení masky se provede jejím vynulováním. Tím je nastavena i „flash“ funkce = zablikání LED při stisku klávesy. Po zrušení „flash“ funkce (bit 7 = 1) lze rozsvěcení LED ovládat jen přes bity 0-2 masky.

Příkaz funguje jen při taktovací frekvenci 4MHz.

Platí pro:

18A, 18M, 18X, 20X2, 28X, 28X1, 28X2, 40X, 40X1, 40X2

Příbuzné:

* [kbin](http://www.picaxe.com/commands/kbin)
* [kbled](http://www.picaxe.com/commands/kbled)
* [keyin](http://www.picaxe.com/commands/keyin)

Příklad:

Blikání všech LED kontrolek na PS2 klávesnici.

main:

keyled %10000111 ; rozsvítí všechny LED kontrolky

pause 500 ; čekání 0.5s

keyled %10000000 ; zhasne všechny LED kontrolky

pause 500 ; čekání0.5s

goto main ; skok na „main“

Jméno:

i2cslave

Syntaxe:

**I2CSLAVE slaveaddress, mode, addresslen**

**SLAVEI2C slaveaddress, mode, addresslen**

SlaveAddress je adresa i2c slave zařízení

Mode, klíčová slova i2cfast (400kHz), nebo i2cslow (100kHz) pro 4MHz taktování. Pro vyšší taktování se používá i2cfast\_8, i2cslow\_8 pro 8MHz, atd.

Addresslen - klíčová slova i2cbyte, nebo i2cword. Označují délku adresy konkrétního zařízení. Např. některé paměti používají adresování typu word. NEJDE o délku přenášených dat, ta je vždy typu byte.

Popis:

**Tento příkaz je zastaralý, zvažte použití příkazu** [hi2csetup](http://www.picaxe.com/commands/hi2csetup)**.**

Příkaz i2cslave (akceptováno i slavei2c) nastaví piny PICAXE pro komunikaci i2c (v režimu MASTER) a definuje typ i2c zařízení.

Více detailů o i2c komunikaci naleznete v originální dokumentaci - „i2c Tutorial“

Pokud PICAXE komunikuje jen s jedním i2c zařízením, vystačí program s jedním nastavením **i2cslave**, v opačném případě je potřeba provádět aktualizaci před komunikací s dalším zařízením.  
Při použití PICAXE-18X je vhodné pro úsporu energie na začátku programu nastavit piny SDA a SCL jako vstupní.

Po nastavení **i2cslave** je možné použít příkazy [readi2c](http://www.picaxe.com/commands/readi2c) a [writei2c](http://www.picaxe.com/commands/writei2c) pro i2c komunikaci.

Slave adresy

Slave adresy se liší pro různé typy i2c zařízení, viz tabulka. Pro populární řadu sériových pamětí EEPROM je adresa %1010xxxx. Některá zařízení např. 24LC16B, začleňuje od bitů 1-3 adresy paměťový blok který bude využit. Jiná zařízení mohou svoji adresu měnit pomocí vnějších přepínačů. Vždy je proto nutné zkontrolovat, zda jsou adresy na masteru i slavu stejně nastavené.

Bit 0 určuje zda slave zařízení má číst data, nebo může data odeslat. Tento bit se nastavuje při použití příkazů [readi2c](http://www.picaxe.com/commands/readi2c) a [writei2c](http://www.picaxe.com/commands/writei2c).

Většina i2c zařízení používá 8 nebo 16 bitovou adresu, jsou ale výjimky, které používají pouze 7 bitovou adresu, zde je nutné provést bitový posun doleva.

Rychlost

Rychlost komunikace se mění parametrem mode, klíčová slova i2cfast (400kHz), nebo i2cslow (100kHz) pro 4MHz taktování. Pro vyšší taktování se používá i2cfast\_8, i2cslow\_8 pro 8MHz, atd.  
Rychlost na sběrnici i2c se má řídit nejpomalejší připojenou periferií. (např. 100kHz pro DS1307)

Délka adres

Nastavuje se parametrem Addresslen, běžně i2c periférie používají jedno bytovou adresu (i2cbyte), nebo dvoj bytovou adresu (i2cword) - přenáší se jako 2x byte.   
Je nutné mít správně nastavený **i2cslave**, **readi2c** a **writei2c**.

Nastavení některých často používaných i2c periferií

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ zařízení** | **Slave adresa** | **Rychlost** | **Délka adresy** |
| 24LC01B EE 128 | %1010xxxx | i2cfast | i2cbyte |
| 24LC02B EE 256 | %1010xxxx | i2cfast | i2cbyte |
| 24LC04B EE 512 | %1010xxbx | i2cfast | i2cbyte |
| 24LC08B EE 1kb | %1010xbbx | i2cfast | i2cbyte |
| 24LC16B EE 2kb | %1010bbbx | i2cfast | i2cbyte |
| 24LC64 EE 8kb | %1010dddx | i2cfast | i2cword |
| 24LC128 EE 16kb | %1010dddx | i2cfast | i2cword |
| 24LC256 EE32kb | %1010dddx | i2cfast | i2cword |
| 24LC512 EE64kb | %1010dddx | i2cfast | i2cword |
| DS1307 RTC | %1101000x | i2cslow | i2cbyte |
| MAX6953 5x7 LED | %101ddddx | i2cfast | i2cbyte |
| AD5245 Digital Pot | %010110dx | i2cfast | i2cbyte |
| SRF08 Sonar | %1110000x | i2cfast | i2cbyte |
| AXE033 I2C LCD | %1100011x | i2cslow | i2cbyte |
| CMPS03 Compass | %1100000x | i2cfast | i2cbyte |
| SPE030 Speech | %1100010x | i2cfast | i2cbyte |

x = bit se ignoruje

b = používá se pro definování bloku paměti (volba vnitřní paměťové stránky)

d = podle nastavení zařízení (adresa podle polarity na adresních pinech)

Vliv zvýšení kmitočtu na i2c

Ujistěte se, že při změně kmitočtu, došlo také ke změně parametru *mode*.   
Pro 8MHz - i2cfast\_8, i2cslow\_8, pro 16MHz - i2cfast\_16, i2cslow\_16, atd.

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [i2cread](http://www.picaxe.com/commands/i2cread)
* [i2cwrite](http://www.picaxe.com/commands/i2cwrite)
* [hi2csetup](http://www.picaxe.com/commands/hi2csetup)
* [hi2cin](http://www.picaxe.com/commands/hi2cin)
* [hi2cout](http://www.picaxe.com/commands/hi2cout)

Příklad:

Nastavení „aktuálního“ času do časového modulu DS1307, komunikace probíhá v BCD kódu.

symbol seconds = b0

symbol mins = b1

symbol hour = b2

symbol day = b3

symbol date = b4

symbol month = b5

symbol year = b6

symbol control = b7

; nastavení PICAXE = master a nastavení slave adresy DS1307

i2cslave %11010000, i2cslow, i2cbyte

; zapsání času a data, např. 11:59:00 on Thurs 25/12/03

; hodnoty budou zapsány v BCD kódu = čísla budou uvozena znakem „$“

start\_clock:

let seconds = $00

let mins = $59

let hour = $11

let day = $03

let date = $25

let month = $12

let year = $03

; nastavení výstupu na 1Hz = každou 1s bude odeslán aktuální čas

let control = %00010000

; odeslání dat na časový modul

writei2c 0,(seconds,mins,hour,day,date,month,year,control)

end

Jméno:

readi2c

Syntaxe:

**READI2C (proměnná,...)**

**READI2C location,(proměnná,...)**

Location - proměnná/konstanta typu byte/word specifikující umístění dat v i2c zařízení.

Proměnná(s) - proměnná pro příjem dat typu byte.

Popis:

**Tento příkaz je zastaralý, zvažte použití příkazu** [hi2cin](http://www.picaxe.com/commands/hi2cin)**.**

Příkaz **readi2c** (akceptováno i **i2cread**) načte hodnotu z i2c a uloží ji do proměnné *proměnná*.

Komunikace a zapojení i2c je popsáno v samostatném dokumentu „i2c Tutorial“.

Načte jeden nebo více bytů dat, do proměnných, od pozice specifikované v parametru *location*. Nastavení parametrů přenosu se provádí příkazem [i2cslave](http://www.picaxe.com/commands/i2cslave) a je třeba jej provést před prvním čtením/zápisem i2c dat.   
Při špatném nakonfigurování i2c komunikace se uloží do všech proměnný hodnota 255 ($FF).

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [writei2c](http://www.picaxe.com/commands/writei2c)
* [hi2cin](http://www.picaxe.com/commands/hi2cin)
* [hi2cout](http://www.picaxe.com/commands/hi2cout)

Příklad:

Komunikace i2c s časovým modulem DS1307, data jsou přenášena v BCD formátu.

i2cslave %11010000, i2cslow, i2cbyte

;nastavení i2c komunikace: PICAXE = master; slave adr.DS1307

main:

readi2c 0,(b0,b1,b2,b3,b4,b5,b6,b7) ; načtení dat přes i2c

debug b1 ; výpis dat na PC

pause 2000 ; čekání 2s

goto main ; skok na začátek programu

Jméno:

writei2c

Syntaxe:

**WRITEI2C location,(proměnná,...)**

**WRITEI2C (proměnná,...)**

Location - proměnná/konstanta typu byte/word specifikující umístění dat v i2c zařízení.

Proměnná(s) - proměnná pro příjem dat typu byte.

Popis:

**Tento příkaz je zastaralý, zvažte použití příkazu** [hi2cout](http://www.picaxe.com/commands/hi2cout)**.**

Příkaz **writei2c** (akceptováno i **i2cwrite**) zapíše data typu byte na externí i2c zařízení.

Komunikace a zapojení i2c je popsáno v samostatném dokumentu „i2c Tutorial“.

Používá se pro odeslání jednoho nebo více bytů dat přes i2c na zařízení typu slave. Hodnota *location* se musí shodovat se specifikací i2c zařízení. [i2cslave](http://www.picaxe.com/commands/hi2csetup) je třeba provést před prvním čtení/zápisem i2c dat.

Platí pro:

08M2, 14M2, 18M2, 18X, 20M2, 20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [readi2c](http://www.picaxe.com/commands/readi2c)
* [hi2cin](http://www.picaxe.com/commands/hi2cin)
* [hi2cout](http://www.picaxe.com/commands/hi2cout)
* [hi2csetup](http://www.picaxe.com/commands/hi2csetup)

Příklad:

Nastavení „aktuálního“ času do časového modulu DS1307, komunikace probíhá v BCD kódu.

symbol seconds = b0

symbol mins = b1

symbol hour = b2

symbol day = b3

symbol date = b4

symbol month = b5

symbol year = b6

symbol control = b7

; nastavení PICAXE = master a nastavení slave adresy DS1307

i2cslave %11010000, i2cslow, i2cbyte

; zapsání času a data, např. 11:59:00 on Thurs 25/12/03

; hodnoty budou zapsány v BCD kódu = čísla budou uvozena znakem „$“

start\_clock:

let seconds = $00

let mins = $59

let hour = $11

let day = $03

let date = $25

let month = $12

let year = $03

; nastavení výstupu na 1Hz = každou 1s bude odeslán aktuální čas

let control = %00010000

; odeslání dat na časový modul

writei2c 0,(seconds,mins,hour,day,date,month,year,control)

end

Jméno:

ultra

Syntaxe:

**ULTRA pin, proměnná**

Pin - proměnná/konstanta označující I/O pin (B.3, C.1 ... viz schéma zapojení)

Proměnná - proměnná typu byte, obdrží výsledek (0 - žádný objekt nebyl detekován; 1-255 vzdálenost k objektu v centimetrech).

Popis:

Příkaz **ultra** vypočte vzdálenost překážky pomocí vzdálenostního čidla [SRF005](http://www.picaxe.com/products/srf005) .

[SRF005](http://www.picaxe.com/products/srf005) modul umožňuje pracovat ve dvou módech. Mód 2 se nastaví uzemněním pinu 4 na SRF005 modulu. Potom je možná komunikace přes 1 pin PICAXE.   
Podrobné zapojení je popsáno v originálním manuálu [SRF005 datasheet](http://www.picaxe.com/docs/srf005.pdf).

Použitý PICAXE pin musí být vstupně/výstupní, umožňující odeslat puls a načíst zpětnou odezvu - přijmout naměřenou hodnotu. Hodnota 0 znamená, že nebyl detekován žádný objekt, 1 - 254 je vzdálenost objektu v centimetrech, 255 - objekt ve vzdálenosti větší než 254 cm. (max. dosah +- 3m)

Příkaz funguje správně pouze při taktovací frekvenci 4MHz (u řady X2 při 8MHz).

**Ekvivalent příkazu „ultra pin, proměnná“:**

- pro řadu M2

    pulsout pin, 2  
    pulsin pin, 1, wordproměnná  
   proměnná = wordproměnná max 1479 \* 10 / 58

- pro řadu X2

    pulsout pin, 2  
    pulsin pin, 1, wordproměnná  
   proměnná = wordproměnná max 2958 \* 5 / 58

Platí pro:

Celou řadu M2 a X2

Příbuzné:

* [pulsout](http://www.picaxe.com/commands/pulsout)
* [pulsin](http://www.picaxe.com/commands/pulsin)

Příklad:

Rozsvícení LED na B.2, je-li překážce blíže jak 10cm, měření pomocí   
čidla SRF005 na C.1.

main:

ultra C.1, b1 ; zahájení měření na C.1, výsledek do b1

if b1 < 10 then ; je-li vzdálenost menší než 10 cm

high B.2 ; pak rozsviť LED na B.2

else ; jinak

low B.2 ; zhasni LED na B.2

end if

pause 200 ; čekej 200 ms

goto main ; opakuj měření - skok na „main“

Jméno:

peeksfr

Syntaxe:

**PEEKSFR location,proměnná,proměnná,...**

Location - proměnná/konstanta (0 - 255) určující adresu v registru procesoru

Proměnná - proměnná pro načtení dat.

Popis:

Přečte data ze speciálních funkčních registrů (SFR) mikrokontroleru (uP).

Příkaz umožňuje studovat vnitřní strukturu uP, je určen pro zkušené programátory. Je implementován u řady M2 a X2.

SFR umožňuje přistupovat pouze k interním periferiím, např. ADC, časovač…  
Čtení, nebo zápis do SFR provázaného s PICAXE programem (např. FSR, EEPROM nebo TABLE registrů) způsobí okamžitý reset čipu.

Řada X2

Hodnoty mohou být pouze 0 - 255, proto se vynechává počáteční „F“.  
Pak např. BAUDCON FB8h se stává $B8

Řada M2

Hodnoty mohou mít pouze 0 - 255, pak adresace se provádí následovně:

|  |  |  |
| --- | --- | --- |
| Bit 7-5 | Paměťová banka | $00-$07 |
| Bit 4-0 | Adresa | $0C to $1F v nastavené bance |
|  |  | ($00-$0B jsou neplatné a vyvolají restart) |

např. BAUDCON, adresa 01Fh v bance 3, odpovídá hodnotě %011 11111

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [pokesfr](http://www.picaxe.com/commands/pokesfr)
* [peek](http://www.picaxe.com/commands/peek)
* [poke](http://www.picaxe.com/commands/poke)

Příklad:

Načtení systémové hodnoty ladění oscilátoru.

peeksfr $9B,b1 ; načtení systémové proměnné OSCTUNE do b1

Jméno:

pokesfr

Syntaxe:

**POKESFR location,data,data,...**

Location - proměnná/konstanta (0 - 255) určující adresu v registru procesoru

Data - proměnná/konstanta pro uložení dat.

Popis:

Uloží data do speciálních funkčních registrů (SFR) mikrokontroleru (uP).

Příkaz umožňuje studovat a upravovat vnitřní strukturu a nastavení uP, je určen pro zkušené programátory. Je implementován u řady M2 a X2.

SFR umožňuje přistupovat pouze k interním periferiím, např. ADC, časovač…  
Čtení, nebo zápis do SFR provázaného s PICAXE programem (např. FSR, EEPROM nebo TABLE registrů) způsobí okamžitý reset čipu.

Řada X2

Hodnoty mohou být pouze 0 - 255, proto se vynechává počáteční „F“.  
Pak např. BAUDCON FB8h se stává $B8

Řada M2

Hodnoty mohou mít pouze 0 - 255, pak adresace se provádí následovně:

|  |  |  |
| --- | --- | --- |
| Bit 7-5 | Paměťová banka | $00-$07 |
| Bit 4-0 | Adresa | $0C to $1F v nastavené bance |
|  |  | ($00-$0B jsou neplatné a vyvolají restart) |

např. BAUDCON, adresa 01Fh v bance 3, odpovídá hodnotě %011 11111

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [peeksfr](http://www.picaxe.com/commands/peeksfr)
* [peek](http://www.picaxe.com/commands/peek)
* [poke](http://www.picaxe.com/commands/poke)

Příklad:

Uložení systémové hodnoty pro ladění oscilátoru.

pokesfr $9B,b1 ; uložení b1 do systémové proměnné OSCTUNE

Jméno:

calibfreq

Syntaxe:

**CALIBFREQ {-} factor**

factor - konstanta/proměnná v rozsahu (-31; + 31)

Popis:

Kalibrace vnitřního oscilátoru procesoru. Pro firemní nastavení je factor = 0.

PICAXE čipy mají vnitřní oscilátor, který lze nastavit na různé rychlosti pomocí příkazu [setfreq](http://www.picaxe.com/commands/setfreq). Tyto frekvence je možné ještě „doladit“ příkazem calibfreq, avšak pro většinu uživatelů je tovární nastavení dostatečně přesné.

Obecně se používá pro jemné doladění frekvence při problémech při komunikace po sériové lince se zařízeními jiných výrobců. Při změně se doporučuje prvně vyzkoušet hodnoty v rozsahu (-4;+ 4) a až potom hodnoty dle potřeby zvyšovat.   
Kladná hodnota zvyšuje rychlost, záporná snižuje rychlost oscilátoru.

Příkaz použijte s opatrností, aby nedošlo ke ztrátě sériové komunikace. Pokud k ní dojde, proveďte hardwarový reset (krátkodobé přerušení napájení).  
**Calibfreq** je pseudopříkaz nahrazující příkaz **pokesfr OSCTUNE, factor**.

Pro *factor* = (0; 31), pak

|  |  |
| --- | --- |
|  | pokesfr OSCTUNE, factor |
|  | pause 2 |

Pro *factor* = (- 31;-1), pak

|  |  |
| --- | --- |
|  | let b12 = 64 - factor |
|  | pokesfr OSCTUNE, b12 |
|  | pause 2 |

Pozor. Příkaz používá proměnnou b12.

Platí pro:

Vše (kromě 08, 18, 18A, 28, 28A, 28X, 40X)

Příbuzné:

* [setfreq](http://www.picaxe.com/commands/setfreq)

Příklad:

Mírně zvýšit rychlost vnitřního oscilátoru.

calibfreq 2

Jméno:

setfreq

Syntaxe:

**SETFREQ freq**

Freq - proměnná/konstanta typu word, nastavující odpovídající frekvenci.

|  |  |  |
| --- | --- | --- |
| **řada M2** | vnitřní oscilátor k31, k250, k500, m1, m2, m4, m8, m16, m32 |  |
| **20X2** | vnitřní oscilátor k31, k250, k500, m1, m2, m4, m8, m16, m32 ,m64 |  |
| **28X2, 40X2** | vnitřní oscilátor k31, k250, k500, m1, m2, m4, m8, m16 | externí oscilátor em16, em32, em40, em64 |

Některé starší řady:

|  |  |  |
| --- | --- | --- |
| **08M, 14M, 20M** | vnitřní oscilátor m4, m8 |  |
| **18A, 18M, 18X** | vnitřní oscilátor m4, m8 |  |
| **28X1, 40X1** | vnitřní oscilátor  k31, k125, k250, k500, m1, m2, m4, m8 | externí oscilátor  em4, em8, em10, em16, em20 |
| **28X2-5V, 40X2-5V** | vnitřní oscilátor  k31, k250, k500, m1, m2, m4, m8 | externí oscilátor  em16, em32, em40 |
| **28X2-3V, 40X2-3V** | vnitřní oscilátor  k31, k250, k500, m1, m2, m4, m8, m16 | externí oscilátor  em16, em32, em40, em64 |

|  |  |  |
| --- | --- | --- |
| kde | k31 | = 31 kHz vnitřní oscilátor |
|  | m4 | = 4 MHz vnitřní oscilátor atd. |
|  | em16 | = 16 MHz externí oscilátor atd. |

Popis:

Nastavení vnitřní taktovací frekvence.

Výchozí hodnota pro řadu X2 je 8 MHz, pro všechny ostatní 4 MHz.

Změna frekvence ovlivňuje řadu příkazů. Např. při přechodu z 4MHz na 8MHz, dojde u příkazu **pause** ke zkrácení doby čekání na polovinu.  
Ke změně frekvence dochází okamžitě. Má vliv také na příkazy prováděné na pozadí (pwmout, hpwm…)

U řady X2 se frekvence externího oscilátoru dělí čtyřmi. Pak em40 odpovídá 10 MHz.  
Při využití multitaskingu v řadě M2 nelze použít **setfreq**.

Při nahrávání programu z editoru dochází k automatickému nastavení výchozí frekvence.

Platí pro:

Vše (kromě 08, 18, 18A, 28, 28A, 28X, 40X)

Příbuzné:

* [calibfreq](http://www.picaxe.com/commands/calibfreq)
* [#freq](http://www.picaxe.com/commands/hash-freq)

Příklad:

Nastavení vnitřní frekvence pro korektní načtení teploty.

setfreq em32 ; nastavení externího oscilátoru na 32MHz (8MHz)

pause 4000 ; čekání - reálně 2s

setfreq m4 ; nastavení vnitřního oscilátoru na 4MHz

readtemp 1,b1 ; načtení teploty s korektní frekvencí 4MHz

setfreq em32 ; přechod zpět na vysokou frekvenci

Jméno:

disablebod

Syntaxe:

**DISABLEBOD**

Popis:

Zakázaní funkce kontroly napájecího napětí.

Některé PICAXE mají implementovánu funkci kontroly napětí. Při poklesu napětí pod stanovenou hranici dojde k resetu procesoru. Tato kontrola je defaultně povolena.

Mohou být ovšem situaci, kdy to není vhodné, například v režimu [sleep](http://www.picaxe.com/commands/sleep) se zákazem sledování napětí sníží spotřeba energie.

Hraniční napětí jsou pro jednotlivé modely v tabulce:

|  |  |
| --- | --- |
| 1.8V | 28X2-3V, 40X2-3V |
| 1.9V | 20X2, 14M2, 18M2, 20M2, 28X2, 40X2 |
| 2.1V | 08, 08M, 14M, 20M, 28X1, 40X1 |
| 2.4V | 08M2 |
| 3.2V | 28X2-5V, 40X2-5V |
| nepodporuje | 18, 18A, 18M, 18X, 28A, 28X, 40X |

Kontrola napětí se obnoví po příkazu [reset](http://www.picaxe.com/commands/reset) a [enablebod](http://www.picaxe.com/commands/enablebod).

Platí pro:

Vše (kromě 08, 18, 18A, 18M, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [enablebod](http://www.picaxe.com/commands/enablebod)

Příklad:

Minimalizace spotřeby procesoru.

main:

disablebod ; zákaz sledování napětí

sleep 10 ; nastavení režimu sleep na 23s (2.3x10)

enablebod ; povolení sledování napětí

goto main ; skok na začátek programu

Jméno:

enablebod

Syntaxe:

**ENABLEBOD**

Popis:

Povolení funkce kontroly napájecího napětí.

Některé PICAXE mají implementovánu funkci kontroly napětí. Při poklesu napětí pod stanovenou hranici dojde k resetu procesoru. Tato kontrola je defaultně povolena a je ji možné zakázat příkazem **disablebod**.

Mohou být ovšem situaci, kdy to není vhodné, například v režimu [sleep](http://www.picaxe.com/commands/sleep) se zákazem sledování napětí sníží spotřeba energie.   
Kontrola napětí se obnoví po příkazu [reset](http://www.picaxe.com/commands/reset) a [enablebod](http://www.picaxe.com/commands/enablebod).

Hraniční napětí jsou pro jednotlivé modely v tabulce:

|  |  |
| --- | --- |
| 1.8V | 28X2-3V, 40X2-3V |
| 1.9V | 20X2, 14M2, 18M2, 20M2, 28X2, 40X2 |
| 2.1V | 08, 08M, 14M, 20M, 28X1, 40X1 |
| 2.4V | 08M2 |
| 3.2V | 28X2-5V, 40X2-5V |
| nepodporuje | 18, 18A, 18M, 18X, 28A, 28X, 40X |

Platí pro:

Vše (kromě 08, 18, 18A, 18M, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [disablebod](http://www.picaxe.com/commands/disablebod)

Příklad:

Minimalizace spotřeby procesoru.

main:

disablebod ; zákaz sledování napětí

sleep 10 ; nastavení režimu sleep na 23s (2.3x10)

enablebod ; povolení sledování napětí

goto main ; skok na začátek programu

Jméno:

readinternaltemp

Syntaxe:

**READINTERNALTEMP voltage, offset, proměnná**

**READINTERNALTEMP voltage, - offset, proměnná**

Voltage - konstanta specifikující použité napájecí napětí.  
Jsou následující možnosti:

|  |  |
| --- | --- |
| IT\_5V0 | 5V |
| IT\_4V5 | 4.5V |
| IT\_4V0 | 4V |
| IT\_3V5 | 3.5V |
| IT\_3V3 | 3.3V |
| IT\_3V0 | 3V |
| IT\_RAW\_H | načte neupravená data typu word (pro napájení vyšší než 4V) |
| IT\_RAW\_L | načte neupravená data typu word (pro napájení nižší než 4V) |

Offset - volitelný opravný faktor, defaultně 0

Proměnná - proměnná typu word, do níž je uložena načtená teplota.

Popis:

Příkaz přečte pokles napětí na interních diodách. Slouží k orientačnímu měření teploty čipu, pro jeho bezpečnost, ne k přesnému měření. Pro přesné měření lze použít senzor DS18B20 a příkaz [readtemp](http://www.picaxe.com/commands/readtemp).

Měření je prováděno ADC na 2, nebo 4 interních diodách připojených ke zdroji. Měření je orientační a závisí na okamžitém napětí baterie. Při použití parametru IT\_RAW\_H, nebo IT\_RAW\_L nejsou data matematicky upravována, proto by měl být parametr *offset* = 0. V ostatních případech je výstup matematicky upraven a výsledek udává přibližnou teplotu v °C.

Upřesňující informace

Matematický vzorec převádějící originální hodnotu na teplotu:

|  |  |  |
| --- | --- | --- |
|  | 5V0 | RAW\_H +/- K -508 \* 14 / 13 + 5 |
|  | 4V5 | RAW\_H +/- K -450 \* 14 / 15 + 5 |
|  | 4V0 | RAW\_H +/- K -378 \* 14 / 18 + 5 |
|  | 3V5 | RAW\_L +/- K -668 \* 14 / 10 + 5 |
|  | 3V3 | RAW\_L +/- K -647 \* 14 / 10 + 5 |
|  | 3V0 | RAW\_L +/- K -609 \* 14 / 10 + 5 |

kde K = offset

Platí pro:

08M2, 14M2, 20M2, 18M2+

Příbuzné:

* [readtemp](http://www.picaxe.com/commands/readtemp)
* [readtemp12](http://www.picaxe.com/commands/readtemp12)

Příklad:

Načtení vnitřní teploty procesoru do proměnné b1.

main:

readinternaltemp IT\_5V0,0,b1 ; načtení vnitřní teploty do b1

debug ; odeslání na PC

pause 500 ; čekání 0,5s

goto main ; skok na začátek

Jméno:

readfirmware

Syntaxe:

**READFIRMWARE proměnná**

Proměnná - proměnná typu byte, do níž je uloženo číslo verze firmwaru.

Popis:

Načte číslo firmawaru zaváděcího programu nainstalovaného v PICAXE a uloží do proměnné.

Neplést s číslem verze uživatelského programu.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [readsilicon](http://www.picaxe.com/commands/readsilicon)
* [#picaxe](http://www.picaxe.com/commands/hash-picaxe)

Příklad:

Načtení verze firmwaru.

main: readfirmware b1 ; načte verzi firmwaru do proměnné b1

Jméno:

readrevision

Syntaxe:

**READREVISION proměnná**

Proměnná - proměnná typu byte, do níž je uloženo číslo verze uživatelského programu.

Popis:

Načte číslo revize uživatelského programu, nastaveného direktivou [#revision](http://www.picaxe.com/commands/hash-revision)   
a uloží do proměnné.

Neplést s číslem verze firmwaru.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [booti2c](http://www.picaxe.com/commands/booti2c)
* [#revision](http://www.picaxe.com/commands/hash-revision)
* [#slot](http://www.picaxe.com/commands/hash-slot)

Příklad:

main: readrevision b1 ; načte číslo revize uživatelského

; programu do b1

Jméno:

readsilicon

Syntaxe:

**READSILICON proměnná**

Proměnná - proměnná typu byte.

Popis:

Načte číslo Silicon čipu, na kterém běží PICAXE. Platí pro řadu X2.

|  |  |  |
| --- | --- | --- |
| Bity 7-5 | PICAXE Type |  |
|  | 000 vyhrazeno pro budoucí použití |  |
|  | 001 PICAXE-20X2 | (PIC18F14K22) |
|  | 010 PICAXE-28X2-5V | (PIC18F2520) |
|  | 011 PICAXE-40X2-5V | (PIC18F4520) |
|  | 100 PICAXE-28X2 | (PIC18F25K22) |
|  | 101 PICAXE-40X2 | (PIC18F45K22) |
|  | 110 PICAXE-28X2-3V | (PIC18F25K20) |
|  | 111 PICAXE-40X2-3V | (PIC18F45K20) |
| Bity 4-0 | verze Microchip Silicon |  |

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [readfirmware](http://www.picaxe.com/commands/readfirmware)
* [#picaxe](http://www.picaxe.com/commands/hash-picaxe)

Příklad:

Načtení čísla Silicon čipu.

main:

readsilicon b1 ; načti Silicon číslo do b1

Jméno:

adcconfig

Syntaxe:

**ADCCONFIG config**

Config - konstanta/proměnná nastavující konfiguraci ADC.

Popis:

Nastaví ADC referenční napětí.

V základním nastavení je (Vref+) pro ADC roven (V+) a (Vref-) je roven (0V).  
To je ale možné změnit pomocí parametru *config*.

Pro řadu X2 je obsazení pinů parametru config:

|  |  |
| --- | --- |
| Bit3-2 | = 11 nepoužívá se |
|  | = 10 VRef+ = FVR (viz. [FVRsetup](http://www.picaxe.com/commands/FVRsetup) ) |
|  | = 01 VRef+ = externí pin |
|  | = 00 VRef+ = V+ (napájecí napětí) |
| Bit1-0 | = 11 nepoužívá se |
|  | = 10 nepoužívá se |
|  | = 01 VRef- = externí pin |
|  | = 00 VRef- = 0V |

Pro řadu M2 je obsazení pinů parametru config:

|  |  |
| --- | --- |
| Bit2 | = 1 VRef- = externí pin |
|  | = 0 VRef- = 0V |
| Bit1-0 | = 11 VRef+ = FVR (viz. [FVRsetup](http://www.picaxe.com/commands/FVRsetup) ) |
|  | = 10 VRef+ = externí pin |
|  | = 01 nepoužívá se |
|  | = 00 VRef+ = V+ (napájecí napětí) |

Vnější referenční (Vref+/Vref-) piny jsou následující:

|  |  |  |
| --- | --- | --- |
| **PICAXE** | **Externí Vref+ pin** | **Externí Vref- Pin** |
| **08M2** | C.1 | není |
| **14M2** | B.1 | není |
| **18M2** | není | C.2 |
| **20M2** | B.0 | není |
| **28X2** | A.3 | A.2 |
| **40X2** | A.3 | A.2 |

Platí pro:

08M2, 14M2, 18M2, 20M2, 28X2, 40X2

Příbuzné:

* [readadc](http://www.picaxe.com/commands/readadc)
* [readadc12](http://www.picaxe.com/commands/readadc12)

Příklad:

Nastavení referenčního napětí na Vref+ = 2.048V pro PICAXE-18M2.

fvrsetup FVR2048 ; nastavení FVR = 2.048V

adcconfig %011 ; nastavení Vref+ = FVR, Vref- = 0

Jméno:

adcsetup

Syntaxe:

**{LET} adcsetup = channels**

Channels - proměnná/konstanta vytvářející masku nastavující piny pro použití ADC.

Popis:

U řady X2 je třeba nakonfigurovat piny používané pro ADC (analogově digitální převodník) před použitím [readadc](http://www.picaxe.com/commands/readadc)/[readadc10](http://www.picaxe.com/commands/readadc10). Vše ostatní se nakonfiguruje automaticky.   
U řady M2 se piny do režimu ADC nastavují automaticky při použití příkazů [readadc](http://www.picaxe.com/commands/readadc)/[readadc10](http://www.picaxe.com/commands/readadc10)/[touch](http://www.picaxe.com/commands/touch). Takže jediné praktické využití je pro vrácení pinů zpět do digitálního režimu.

**Adcsetup** není příkaz, jde o systémovou proměnnou typu word, která odpojuje vnitřní digitální vstupy, čímž zpřesňuje naměřenou analogovou hodnotu příkazem [readadc](http://www.picaxe.com/commands/readadc).

Poznámka: Zde bude popsáno pouze použití u řady M2, popis pro řadu X2 je v originální příručce - [adcsetup](http://www.picaxe.com/commands/adcsetup).

**Řada M2**

Individuální nastavení masky pinů

Nastavením bitu masky dojde k odpojení digitálního vstupu na patřičném pinu.

U řady M2 k tomuto nastavení dochází automaticky při použití příkazů [readadc](http://www.picaxe.com/commands/readadc) / [readadc10](http://www.picaxe.com/commands/readadc10) / [touch](http://www.picaxe.com/commands/touch) / [touch16](http://www.picaxe.com/commands/touch16). Proto praktické využití je využití masky pro návrat pinu do digitálního režimu.

**08M2**

|  |  |
| --- | --- |
| Bit 1 | - ADC on C.1 |
| Bit 2 | - ADC on C.2 |
| Bit 4 | - ADC on C.4 |

**14M2, 18M2, 20M2**

|  |  |  |  |
| --- | --- | --- | --- |
| Bit 0 | - ADC on B.0 | Bit 8 | - ADC on C.0 |
| Bit 1 | - ADC on B.1 | Bit 9 | - ADC on C.1 |
| Bit 2 | - ADC on B.2 | Bit 10 | - ADC on C.2 |
| Bit 3 | - ADC on B.3 | Bit 11 | - ADC on C.3 |
| Bit 4 | - ADC on B.4 | Bit 12 | - ADC on C.4 |
| Bit 5 | - ADC on B.5 | Bit 13 | -ADC on C.5 |
| Bit 6 | - ADC on B.6 | Bit 14 | -ADC on C.6 |
| Bit 7 | - ADC on B.7 | Bit 15 | -ADC on C.7 |

Referenční napětí

Výchozí nastavení referenčního napětí je (Vref+) = (V+) a (Vref-) = (0V), takže rozsah měřeného napětí je stejný jako napájecí napětí.  
(Vref+) je však možné změnit pomocí příkazu [adcconfig](http://www.picaxe.com/commands/adcconfig).

Platí pro:

08M2, 14M2, 18M2, 20M2, 20X2, 28X2, 40X2

Příbuzné:

* [readadc](http://www.picaxe.com/commands/readadc)
* [readadc12](http://www.picaxe.com/commands/readadc12)

Jméno:

calibadc

Syntaxe:

**CALIBADC proměnná**

**CALIBADC10 wordproměnná**

Proměnná - proměnná typu byte/word, pro načtení hodnoty ADC

Popis:

Pro kalibraci ADC používá mikrokontroler vnitřní referenční napětí:

|  |  |
| --- | --- |
| 0.6V | 20M, 28X1, 40X1 |
| 1.2V | 28X2-3V, 28X2-3V |
| 1.024V | všechny ostatní typy, podporující tento příkaz |

Příkaz není k dispozici na 28X2-5V a 40X2-5V.

Příkazy [readadc](http://www.picaxe.com/commands/readadc) a [readadc10](http://www.picaxe.com/commands/readadc10) používají za referenční napětí nejčastěji napájecí napětí, které se může v čase měnit (vybíjení baterie). Proto se používá příkaz **calibadc**, který provede měření známého referenčního napětí v daném čase (při aktuálním stavu baterií). Pomocí získané hodnoty lze matematicky korigovat naměřená napětí příkazem **readadc** a zlepšit přesnost měření.

**Matematická korekce**

Calibadc dá výsledek (Nref), který závisí na velkosti vnitřního referenčního napětí (Vref) a napájecím napětí PICAXE (Vpsu), pak:  
  
     Nref = Vref \* 255 / Vpsu  
=>  
     Vpsu = Vref \* 255 / Nref  
  
Pro Vref = 1.024 V :  
  
     Vpsu = 1.024 \* 255 / Nref  
  
     Vpsu = 261.12 / Nref  
  
zaokrouhleno  
   
     Vpsu = 261 / Nref

Platí pro:

Vše (kromě 08, 08M, 14M, 18, 18A, 18M, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [calibadc10](http://www.picaxe.com/commands/calibadc10)
* [fvrsetup](http://www.picaxe.com/commands/fvrsetup)

Příklad:

Načtení kalibrační hodnoty.

main:

calibadc b1 ; načtení kalibrační hodnoty

debug ; odeslání na PC

pause 500 ; čekání 0,5s

goto main ; opakování, skok na „main“

Jméno:

calibadc10

Syntaxe:

**CALIBADC10 wordproměnná**

**CALIBADC proměnná**

Proměnná - proměnná typu byte/word, pro načtení hodnoty ADC

Popis:

Pro kalibraci ADC používá mikrokontroler vnitřní referenční napětí:

|  |  |
| --- | --- |
| 0.6V | 20M, 28X1, 40X1 |
| 1.2V | 28X2-3V, 28X2-3V |
| 1.024V | všechny ostatní typy, podporující tento příkaz |

Příkaz není k dispozici na 28X2-5V a 40X2-5V.

Příkazy [readadc](http://www.picaxe.com/commands/readadc) a [readadc10](http://www.picaxe.com/commands/readadc10) používají za referenční napětí nejčastěji napájecí napětí, které se může v čase měnit (vybíjení baterie). Proto se používá příkaz **calibadc**, který provede měření známého referenčního napětí v daném čase (při aktuálním stavu baterií). Pomocí získané hodnoty lze matematicky korigovat naměřená napětí příkazem **readadc** a zlepšit přesnost měření.

**Matematická korekce**

Calibadc10 dá výsledek (Nref), který závisí na velkosti vnitřního referenčního napětí (Vref) a napájecím napětí PICAXE (Vpsu), pak:  
  
     Nref = Vref \* 1023 / Vpsu  
=>  
     Vpsu = Vref \* 1023 / Nref  
  
Pro Vref = 1.024 V :  
  
     Vpsu = 1.024 \* 1023 / Nref  
  
     Vpsu = 1047.552 / Nref  
  
zaokrouhleno  
   
     Vpsu = 1048 / Nref

Platí pro:

Vše (kromě 08, 08M, 14M, 18, 18A, 18M, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [calibadc](http://www.picaxe.com/commands/calibadc)
* [fvrsetup](http://www.picaxe.com/commands/fvrsetup)

Příklad:

Načtení kalibrační hodnoty.

main:

calibadc10 w1 ; načtení kalibrační hodnoty

debug ; odeslání na PC

pause 500 ; čekání 0,5s

goto main ; opakování, skok na „main“

Jméno:

fvrsetup

Syntaxe:

**FVRSETUP config**

**FVRSETUP OFF**

Config - konstanta/proměnná nastavující pevné referenční napětí FVR.

Popis:

Nastavuje vnitřní referenční napětí FVR pro ADC a DAC převodníky.

Některé PICAXE mají nastavitelné pevné referenční napětí, které je možné vypnout, nebo nastavit na jedno ze tří napětí:

|  |  |
| --- | --- |
| FVR1024 | 1.024V |
| FVR2048 | 2.048V |
| FVR4096 | 4.096V \* |

\* Poznámka: Napětí FVR nesmí přesáhnout napájecí napětí. Tzn., že referenční napětí 4.096 V je možné použít pouze při napájecím napětí 5V.

Pro měření na vnějším pinu je doporučeno používat FVR napětí 2.048V a 4.096V.  
Po provedení příkazu [readadc](http://www.picaxe.com/commands/readadc) je funkce FVR automaticky vypnuta, pokud je znovu potřeba, je nutné provést příkaz **fvrsetup** opakovaně.

Příkaz **calibadc** nastavuje FVR automaticky na 1.024V.  
FVR muže sloužit jako referenční napětí také pro DAC převodník. (viz. [DACsetup](http://www.picaxe.com/commands/dacsetup)).

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [compsetup](http://www.picaxe.com/commands/compsetup)
* [calibadc](http://www.picaxe.com/commands/calibadc)
* [calibadc10](http://www.picaxe.com/commands/calibadc10)
* [dacsetup](http://www.picaxe.com/commands/dacsetup)

Příklad:

Nastavení FVR na hodnotu 1.024 V.

fvrsetup FVR1024 ; nastavení referenčního napětí FVR na 1.024V

Jméno:

disconnect

Syntaxe:

**DISCONNECT**

Popis:

Zakáže kontrolování pinů používaných pro nahrání programu do PICAXE.

PICAXE stále kontroluje piny používané pro sériový download programu. V případě, že tyto piny jsou využívány jiným způsobem, může docházet ke kolizím. Proto je vhodné zakázat stálé kontrolování těchto pinů, pak se piny chovají jako běžné I/O piny, včetně přiřazených funkcí.

**Disconnect** se automaticky provede při provádění příkazu [serrxd](http://www.picaxe.com/commands/serrxd).

Pro zrušení příkazu **disconnect** lze použít:

1) příkaz [reconnect](http://www.picaxe.com/commands/reconnect)

2) příkaz [reset](http://www.picaxe.com/commands/reset) (současně dojde k restartu programu)

3) [hardware reset](http://www.picaxe.com/hard-reset) = krátkodobé odpojení zdroje napájení (vždy funguje)

Platí pro:

Vše M2, X1 a X2 řady

Příbuzné:

* [reconnect](http://www.picaxe.com/commands/reconnect)
* [serrxd](http://www.picaxe.com/commands/serrxd)

Příklad:

Použití pinu C.5 (seriál download pin) na PICAXE-08M2, jako normálního I/O pinu.

pause 1000 ; startovací prodleva

disconnect ; zakázání sledování „download“ pinu

main:

if pinC.5 = 1 then ; když je C.5 = 1

high C.1 ; pak nastav C.1 na vysokou úroveň

else ; jinak

low C.1 ; nastav C.1 na nízkou úroveň

end if

goto main ; opakuj - skok na „main“

Jméno:

reconnect

Syntaxe:

**RECONNECT**

Popis:

Obnoví - povolí kontrolu sériového download pinu, určeného pro nahrávání programu do PICAXE.

PICAXE stále kontroluje piny používané pro sériový download programu. V případě, že tyto piny jsou využívány jiným způsobem, může docházet ke kolizím. Proto je vhodné zakázat stálé kontrolování těchto pinů, pak se piny chovají jako běžné I/O piny, včetně přiřazených funkcí.

**Disconnect** se automaticky provede při provádění příkazu [serrxd](http://www.picaxe.com/commands/serrxd).

Pro zrušení příkazu **disconnect** lze použít:

1) příkaz [reconnect](http://www.picaxe.com/commands/reconnect)

2) příkaz [reset](http://www.picaxe.com/commands/reset) (současně dojde k restartu programu)

3) [hardware reset](http://www.picaxe.com/hard-reset) = krátkodobé odpojení zdroje napájení (vždy funguje)

Platí pro:

Vše (kromě 08, 08M, 18, 18A, 18X, 28, 28A, 28X, 40X)

Příbuzné:

* [disconnect](http://www.picaxe.com/commands/disconnect)
* [serrxd](http://www.picaxe.com/commands/serrxd)

Příklad:

Ukázka použití příkazu reconnect.

disconnect ; zakázání sledováni „download“ pinu

serrxd [1000, timeout],@ptrinc,@ptrinc,@ptr ; čtení sériových dat

reconnect ; obnovení sledováni „download“ pinu

…

…

timeout: ; adr. při překročení doby čekání  
 …

…

Jméno:

hintsetup

Syntaxe:

**HINTSETUP mask**

Mask - proměnná/konstanta určující které bity přerušení budou aktivní.

|  |  |
| --- | --- |
| Bit 7 | - rezervovaný |
| Bit 6 | - Interrupt 2 při přechodu (1 = náběžná hrana, 0 = sestupná hrana) |
| Bit 5 | - Interrupt 1 při přechodu (1 = náběžná hrana, 0 = sestupná hrana) |
| Bit 4 | - Interrupt 0 při přechodu (1 = náběžná hrana, 0 = sestupná hrana) |
| Bit 3 | - rezervovaný |
| Bit 2 | - Interrupt 2 aktivní |
| Bit 1 | - Interrupt 1 aktivní |
| Bit 0 | - Interrupt 0 aktivní (není k dispozici na 20X2) |

Popis:

Nastavuje až 3 hardwarová přerušení na (INT0, INT1, INT2), které běží na pozadí.

Umožnují provést aktivaci procesoru z režimu [sleep](http://www.picaxe.com/commands/sleep) a [doze](http://www.picaxe.com/commands/doze).  
Ve spojení s příkazem [setintflags](http://www.picaxe.com/commands/setintflags) umožňuje volat podprogramy obsluhy přerušení.  
Při vyvolání přerušení nastavuje 2 flagy - svůj identifikační flag + sdílený hindflag, oba je nutné „ručně“ vynulovat.

Pro využití hardwarového přerušení je třeba dvou kroků:

1) pomocí **hintsetup** nastavit příznaky hardwaru - povolit hw přerušení.

2) pomocí [setintflags](http://www.picaxe.com/commands/setintflags) definovat činnost která bude provedena při nastavení jednoho nebo kombinace více flagů.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [setint](http://www.picaxe.com/commands/setint)
* [setintflags](http://www.picaxe.com/commands/setintflags)

Příklad:

Ukázka použití příkazu hintsetup.

hintsetup %00000111 ; nastavení všech 3 pinů na sestupnou hranu

hintsetup %00000010 ; nastavení přerušení pouze na INT1

hintsetup %00000000 ; zakázání všech hw přerušení

Jméno:

disabletime

Syntaxe:

**DISABLETIME**

Popis:

Zakáže přičítání času do proměnné **time** (typu word).

Řada M2 má vnitřní časový čítač, který každou sekundu přičte do proměnné **time** přičte +1. Přičítání začíná po zapnutí procesoru, je možné ho zastavit příkazem disabletime a obnovit příkazem **enabletime**.

S proměnnou **time** je možné pracovat jako s běžnou proměnnou typu word, tj. číst z ní aktuální hodnotu a ukládat do ní novou hodnotu.

Vliv změny frekvence

Pro frekvence 4 MHz a 16 MHz se hodnota **time** zvýší každou 1s, pro 2 MHz a 8 MHz každé 2s a pro 32 MHz co 0.5s.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [enabletime](http://www.picaxe.com/commands/enabletime)

Příklad:

Demonstrace příkazů **enabletime** a **disabletime**.

main:

pause 5000 ; čekej 5s

disabletime ; zastavení přičítání do proměnné **time**

pause 5000 ; čeken 5s

enabletime ; obnovení přičítání do proměnné **time**

sertxd("Time: ",#time,13,10) ; vypsání proměnné **time** na PC

goto main ; skok na začátek programu

Jméno:

enabletime

Syntaxe:

**ENABLETIME**

Popis:

Povolí přičítání času do proměnné **time** (typu word).

Řada M2 má vnitřní časový čítač, který každou sekundu přičte do proměnné **time** přičte +1. Přičítání začíná po zapnutí procesoru, je možné ho zastavit příkazem disabletime a obnovit příkazem **enabletime**.

S proměnnou **time** je možné pracovat jako s běžnou proměnnou typu word, tj. číst z ní aktuální hodnotu a ukládat do ní novou hodnotu.

Vliv změny frekvence

Pro frekvence 4 MHz a 16 MHz se hodnota **time** zvýší každou 1s, pro 2 MHz a 8 MHz každé 2s a pro 32 MHz co 0.5s.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [disabletime](http://www.picaxe.com/commands/disabletime)

Příklad:

Demonstrace příkazů **enabletime** a **disabletime**.

main:

pause 5000 ; čekej 5s

disabletime ; zastavení přičítání do proměnné **time**

pause 5000 ; čekej 5s

enabletime ; obnovení přičítání do proměnné **time**

sertxd("Time: ",#time,13,10) ; vypsání proměnné **time** na PC

goto main ; skok na začátek programu

Jméno:

settimer

Syntaxe:

**SETTIMER OFF**

**SETTIMER preload**

**SETTIMER COUNT preload**

Preload - konstanta/proměnná, která nastaví vhodné načasování vnitřního časovače. Pro usnadnění jsou konstanty pro generování 1s při různých frekvencích předdefinovány:

|  |  |  |
| --- | --- | --- |
| t1s\_4 | (*preload* = 49910 | - 1 s při 4MHz) |
| t1s\_8 | (*preload* = 34286 | - 1 s při 8MHz) |
| t1s\_16 | (*preload* = 3036 | - 1 s při 16MHz) |

Popis:

Nastaví a spustí vnitřní časovač/čítač.

Příkaz **settimer** se používá ke konfiguraci vnitřního hardwarového časovače/čítače. Čítač používá pouze pin C.0.

Poznámka: Příkaz [debug](http://www.picaxe.com/commands/debug) dočasně zakáže běh časovače, proto budou hodnoty časovače nepřesné.

Externí čítač (není k dispozici na 20X2)- settimer count preload

V režimu externího počítadla se k internímu čítačovému registru přičte +1 při každé náběžné hraně (přechod 0-1) na pinu C.0.  
Počítání se děje na pozadí, takže uživatelský program běží dále. Na rozdíl od použití příkazu [count](http://www.picaxe.com/commands/count), kdy program je pozastaven.   
Když vnitřní čítačový registr přeteče (dojde k překročení rozsahu a přechodu z 65535 na 0), ke speciální proměnné **timer** se přičte +1.

Proto pro detekci 10 pulzů je třeba nastaviti parametr *preload* na hodnotu   
65536 - 10 = 65526. Pak po 10 pulzech dojde k přetečení a zvýšení proměnné **timer**. Pokud chceme detekovat příchozí pulz, nastavíme parametr *preload* = 65535. Při prvním detekovaném pulzu dojde k přetečení.

Při přetečení časovače/čítače dojde k nastavení flagu - **toflag**. Ten je automaticky vynulován při spuštění příkazu **settimer**. Je možné jej vynulovat i ručně   
„[let](http://www.picaxe.com/commands/let) toflag = 0“. Také je možné nastavit přerušení na tento flag příkazem [setintflags](http://www.picaxe.com/commands/setintflags).

Vnitřní časovač - settimer preload

V módu vnitřní časovač je čas uložen ve speciální proměnné **timer**. S proměnnou je možné pracovat, jako s běžnou proměnnou.

např. if timer > 200 then skoč

Když proměnná **timer** přeteče (tj. překročí hodnotu 65535) nastaví se flag - **toflag**.  
Ten je automaticky vynulován při spuštění příkazu **settimer**. Je možné jej vynulovat i ručně „[let](http://www.picaxe.com/commands/let) toflag = 0“. Také je možné nastavit přerušení na tento flag příkazem [setintflags](http://www.picaxe.com/commands/setintflags).

Časovač pracuje se dvěma frekvencemi.  
První - rychlá („malý tik“) je odvozena přímo od taktovací frekvence -   
1/(taktovací\_frekvence / 256) sekund   
Pro taktovací frekvenci 4MHz je „malý tik“ = 64us (32us pro 8MHz, 16us pro 16MHz, 8us pro 32MHz, 4us pro 64MHz). Když počet „malých tiků“ přeteče (přechod z 65535 na 0) provede se „velký tik“.   
„Velký tik“ přičte +1 k proměnné **timer** (inkrementace proměnné).

Hodnotu „malý tik“ je možné přednastavit parametrem *preload*. Pokud chceme, aby „velký tik“ proběhl po 100 „malých ticích“, nastavíme do *preload* hodnotu 65536 - 100 = 65436.

Máme-li procesor s frekvencí 4MHz, pak „malý tik“ trvá 64us. Pokud chceme, aby „velký tik“ nastal za 1s tj. 1000000us => *preload* = 65536 - (1000000 /64) = 49910.

Časovač nelze použít ve stejnou dobu jako příkaz [servo](http://www.picaxe.com/commands/servo), který využívá stejný hardware.

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [tmr3setup](http://www.picaxe.com/commands/tmr3setup)
* [enabletime](http://www.picaxe.com/commands/enabletime)
* [disabletime](http://www.picaxe.com/commands/disabletime)

Příklad:

Sledování zda nepřišel pulz na C.0.

settimer count 65535 ; nastavení časovače do režimu ext. čítače

; k přetečení dojde po příchodu 1 pulzu

main:

pause 10000 ; čekej 10s

debug ; hodnoty na PC, sledování proměnné **timer**

goto main ; opakování, skok na „main“

Jméno:

tmr3setup

Syntaxe:

**TMR3SETUP config**

Config - konstanta/proměnná typu byte, která nakonfiguruje časovač **timer3**.

Konfigurace bitů pro - 20X2, 28X2-5V, 28X2-3V, 40X2-3V, 40X2-5V:

|  |  |
| --- | --- |
| Bit7 | musí být (1) |
| Bit6 | musí být (0) |
| Bit5-4 | 1:8 dělička (11) |
|  | 1:4 dělička (10) |
|  | 1:2 dělička (01) |
|  | 1:1 dělička (00) |
| Bit3 | musí být (0) |
| Bit2 | musí být (0) |
| Bit1 | musí být (0) |
| Bit0 | Timer 3 je povolen (1= ano, 0 = ne) |

Konfigurace bitů pro - 28X2, 40X2:

|  |  |
| --- | --- |
| Bit7 | musí být (0) |
| Bit6 | musí být (0) |
| Bit5-4 | 1:8 dělička (11) |
|  | 1:4 dělička (10) |
|  | 1:2 dělička (01) |
|  | 1:1 dělička (00) |
| Bit3 | musí být (0) |
| Bit2 | musí být (0) |
| Bit1 | musí být (1) |
| Bit0 | Timer 3 je povolen (1= ano, 0 = ne) |

Popis:

Konfigurace vnitřního časovače **timer3**, na řadě X2.

Příkaz **tmr3setup** nastavuje volný běh časovače na pozadí.  
Časovač taktuje s rychlostí takt za (1/taktovací\_freqvence)\*4 sekundy. Při přetečení dojde k inkrementaci (zvýšení hodnoty o +1) proměnné **timer3**. Tzn., že při frekvenci 4MHz bez použití děličky dojde ke zvýšení hodnoty **timer3** každých 65536us = 65,5ms. Při použití děličky 1:8, dojde ke zvýšení proměnné **timer3** každých 524ms.

S proměnnou **timer3** je možné pracovat, jako s běžnou proměnnou.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [settimer](http://www.picaxe.com/commands/settimer)
* [enabletime](http://www.picaxe.com/commands/enabletime)
* [disabletime](http://www.picaxe.com/commands/disabletime)

Příklad:

Ukázka použití příkazu **tmr3setup** u PICAXE 28X2.

tmr3setup %00110011 ; povolí timer3 a nastaví děličku 1:8

main:

pause 500 ; čekej 0,5s

debug ; data na PC, sledování proměnné **timer3**

goto main ; skok na „main“

Jméno:

#com

Syntaxe:

**#COM device**

Popis:

Nastaví sériový COM/USB port pro nahrávání programů do PICAXE (downloading).

Platí pro:

Vše

Příbuzné:

* [#terminal](http://www.picaxe.com/commands/hash-terminal)

Příklad:

#com 1 (Windows AXE026 seriál)

#com 6 (Windows AXE027 USB\*)

#com 1 (Windows CE AXE027 USB\*)

\*Více informací naleznete u převodníku AXE027 USB.

Jméno:

#define

Syntaxe:

**#DEFINE label**

Popis:

Definuje *popisku (label)* pro příkazy [#ifdef](http://www.picaxe.com/commands/hash-ifdef) a [#ifndef](http://www.picaxe.com/commands/hash-ifndef).

Pozor: Neplést „#define“ a „[symbol](http://www.picaxe.com/commands/symbol) =“.  
**#define** je direktiva která ve spojení s [#ifdef](http://www.picaxe.com/commands/hash-ifdef) určuje, která část zdrojového kódu bude přeložena - zkompilována.   
[**symbol**](http://www.picaxe.com/commands/symbol) **=** je příkaz umožňující „přejmenovat“ proměnné a piny, zapamatovatelnějšími názvy.

Platí pro:

Vše

Příbuzné:

* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#ifndef](http://www.picaxe.com/commands/hash-ifndef)
* [#undefine](http://www.picaxe.com/commands/hash-undefine)

Příklad:

Ukázka použití direktivy **#define**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#else

Syntaxe:

[#IFDEF](http://www.picaxe.com/commands/hash-ifdef) **label**   
…  
**#ELSE**   
…  
[**#ENDIF**](http://www.picaxe.com/commands/hash-endif)

Popis:

Součást direktivy [#ifdef](http://www.picaxe.com/commands/hash-ifdef) a [#ifndef](http://www.picaxe.com/commands/hash-ifndef).

Platí pro:

Vše

Příbuzné:

* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#ifndef](http://www.picaxe.com/commands/hash-ifndef)
* [#endif](http://www.picaxe.com/commands/hash-endif)

Příklad:

Ukázka použití direktivy **#else**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#endif

Syntaxe:

[#IFDEF](http://www.picaxe.com/commands/hash-ifdef) **label**   
…  
[**#ELSE**](http://www.picaxe.com/commands/hash-else)   
**…  
#ENDIF**

[#IFNDEF](http://www.picaxe.com/commands/hash-ifndef) **label**   
…  
[**#ELSE**](http://www.picaxe.com/commands/hash-else)   
…  
**#ENDIF**

Popis:

Tento příkaz ukončuje blok direktivy [#ifdef](http://www.picaxe.com/commands/hash-ifdef) i direktivy [#ifndef](http://www.picaxe.com/commands/hash-ifndef).

Platí pro:

Vše

Příbuzné:

* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#ifndef](http://www.picaxe.com/commands/hash-ifndef)
* [#else](http://www.picaxe.com/commands/hash-else)

Příklad:

Ukázka použití direktivy **#endif**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#rem

Syntaxe:

**#REM**   
…  
[**#ENDREM**](http://www.picaxe.com/commands/hash-endrem)

Popis:

Označení začátku víceřádkové poznámky.

Platí pro:

Vše

Příbuzné:

* [#endrem](http://www.picaxe.com/commands/hash-endrem)

Příklad:

#rem ; začátek poznámky

high 0 ; text uvnitř poznámky

pause 1000

low 0

#endrem ; konec poznámky

Jméno:

#endrem

Syntaxe:

[**#REM**](http://www.picaxe.com/commands/hash-rem)   
…  
**#ENDREM**

Popis:

Ukončení víceřádkové poznámky.

Platí pro:

Vše

Příbuzné:

* [#rem](http://www.picaxe.com/commands/hash-rem)

Příklad:

#rem ; začátek poznámky

high 0 ; text uvnitř poznámky

pause 1000

low 0

#endrem ; konec poznámky

Jméno:

#error

Syntaxe:

**#ERROR “comment”**

Popis:

Vynutí chybu kompilátoru na daném místě s možným komentářem.

Platí pro:

Vše

Příbuzné:

* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#ifndef](http://www.picaxe.com/commands/hash-ifndef)

Příklad:

#error “Code not finished!” ; vynucení chyby kompilátoru

; s komentářem „Kód není dokončen!“

Jméno:

#freq

Syntaxe:

**#FREQ M4**

**#FREQ M8**

**#FREQ M16**

Popis:

Nastaví základní systémovou frekvenci pro download, platí pro PICAXE 28, 28A, 28X a 40X.

Není požadováno pro ostatní typy PICAXE, které provedou nastavení automaticky.

Pro nastavení taktovací frekvence PICAXE v programu se používá příkaz [setfreq](http://www.picaxe.com/commands/setfreq).

Platí pro:

28, 28A, 28X, 40X

Příklad:

#freq m8

Jméno:

#gosubs

Syntaxe:

**#GOSUBS 16**

**#GOSUBS 256**

Popis:

**Nastavení se týká pouze programového editoru pro simulaci, ne PICAXE**

Nastavení **gosubs** módu (16/256) pro starší typy 18X / 28X.

Platí pro:

18X, 28X, 40X

Příbuzné:

* [gosub](http://www.picaxe.com/commands/gosub)

Příklad:

#gosubs 16

#gosubs 256

Jméno:

#ifdef

Syntaxe:

**#IFDEF label**   
…  
[**#ENDIF**](http://www.picaxe.com/commands/hash-endif)

**#IFDEF label**   
…  
[**#ELSE**](http://www.picaxe.com/commands/hash-else)   
…  
[**#ENDIF**](http://www.picaxe.com/commands/hash-endif)

Popis:

Podmíněná kompilace programového kódu podle toho, zda je „*popiska*“ (příkaz **#define)** definována - **#ifdef**, nebo nedefinována - [**#ifndef**](http://www.picaxe.com/commands/hash-ifndef).

Platí pro:

Vše

Příbuzné:

* [#ifndef](http://www.picaxe.com/commands/hash-ifndef)
* [#define](http://www.picaxe.com/commands/hash-define)
* [#undefine](http://www.picaxe.com/commands/hash-undefine)
* [#else](http://www.picaxe.com/commands/hash-else)
* [#endif](http://www.picaxe.com/commands/hahs-endif)

Příklad:

Ukázka použití direktivy **#ifdef**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#ifndef

Syntaxe:

**#IFNDEF label**…  
[**#ENDIF**](http://www.picaxe.com/commands/hash-endif)

**#IFNDEF label**   
…  
[**#ELSE**](http://www.picaxe.com/commands/hash-else)   
…  
[**#ENDIF**](http://www.picaxe.com/commands/hash-endif)

Popis:

Podmíněná kompilace programového kódu podle toho, zda je „*popiska*“ (příkaz **#define)** definována - [**#ifdef**](http://www.picaxe.com/commands/hash-ifdef), nebo nedefinována - **#ifndef**.

Platí pro:

Vše

Příbuzné:

* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#define](http://www.picaxe.com/commands/hash-define)
* [#undefine](http://www.picaxe.com/commands/hash-undefine)
* [#else](http://www.picaxe.com/commands/hash-else)
* [#endif](http://www.picaxe.com/commands/hash-endif)

Příklad:

Ukázka použití direktivy **#ifndef**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#no\_data

Syntaxe:

**#NO\_DATA**

Popis:

Nepřenáší EEPROM data. Přenáší pouze část obsahující program a programová data.

Platí pro:

Vše (kromě 08, 08M, 08M2, 14M, 18, 18M, 20M)

Příbuzné:

* [#no\_table](http://www.picaxe.com/commands/hash-no-table)
* [#no\_end](http://www.picaxe.com/commands/hash-no-end)

Jméno:

#no\_end

Syntaxe:

**#NO\_END**

Popis:

Na konec programu není automaticky doplněn příkaz [end](http://www.picaxe.com/commands/end).

Platí pro:

Vše

Příbuzné:

* [#no\_data](http://www.picaxe.com/commands/hash-no-data)
* [#no\_table](http://www.picaxe.com/commands/hash-no-table)

Jméno:

#no\_table

Syntaxe:

**#NO\_TABLE**

Popis:

Nepřenáší data tabulky ani data EEPROM platí pro řadu X1 a X2.

To se také automaticky provede při použití direktivy [#no\_data](http://www.picaxe.com/commands/hash-no-data).

Platí pro:

20X2, 28X1, 28X2, 40X1, 40X2

Příbuzné:

* [#no\_data](http://www.picaxe.com/commands/hash-no-data)
* [#no\_end](http://www.picaxe.com/commands/hash-no-end)

Jméno:

#picaxe

Syntaxe:

**#PICAXE xxx**

Popis:

Nastaví mód kompilátoru pro konkrétní model PICAXE.

Tato direktiva automaticky definuje všechny typy PICAXE, např. #picaxe 08m2 je ekvivalent [*#define*](http://www.picaxe.com/commands/hash-define) 08m2. Kompilátor používá jen příkazy podporované konkrétním typem PICAXE.

V případě, že direktiva **#picaxe** není definovaná, kompilátor použije defaultní mód, který je nastaven v programovém editoru (View>Options>Mode menu).

Platí pro:

Vše

Příbuzné:

* [#freq](http://www.picaxe.com/commands/hash-freq)
* [#gosubs](http://www.picaxe.com/commands/hash-gosubs)
* [#no\_data](http://www.picaxe.com/commands/hash-no-data)
* [#no\_table](http://www.picaxe.com/commands/hash-no-table)
* [#no\_end](http://www.picaxe.com/commands/hash-no-end)
* [#terminal](http://www.picaxe.com/commands/hash-terminal)
* [#slot](http://www.picaxe.com/commands/hash-slot)

Příklad:

#picaxe 08m2

Jméno:

#revision

Syntaxe:

**#REVISION number**

Popis:

Nastavení verze uživatelského programu (1-254) pro řadu X2.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [run](http://www.picaxe.com/commands/run)
* [booti2c](http://www.picaxe.com/commands/booti2c)
* [readrevision](http://www.picaxe.com/commands/readrevision)
* [#slot](http://www.picaxe.com/commands/hash-slot)

Příklad:

#revision 1

Jméno:

#sim

Syntaxe:

**#SIM AXE101**

**#SIM AXE102**

**#SIM AXE103**

**#SIM AXE105**

**#SIM AXE107**

**#SIM AXE092**

Popis:

**Nastavuje pouze programový editor, netýká se procesoru.**

Simuluje chování PICAXE zapojeného do „Projektového PICAXE kitu“.

Platí pro:

Vše

Příbuzné:

* [#simspeed](http://www.picaxe.com/commands/hash-simspeed)
* [#simtask](http://www.picaxe.com/commands/hash-simtask)

Příklad:

#sim axe105

Jméno:

#simspeed

Syntaxe:

**#SIMSPEED *value***

***value*** - konstanta udávající zpoždění v milisekundách

Popis:

**Nastavuje pouze programový editor, netýká se procesoru.**

Nastaví zpoždění mezi příkazy (v milisekundách) při simulaci.

Platí pro:

Vše

Příbuzné:

* [#sim](http://www.picaxe.com/commands/hash-sim)
* [#simtask](http://www.picaxe.com/commands/hash-simtask)

Příklad:

#simspeed 200

Jméno:

#simtask

Syntaxe:

**#SIMTASK ALL**

**#SIMTASK tasknumber**

Popis:

**Nastavuje pouze programový editor, netýká se procesoru.**

Direktiva nastavuje, který proces, při multi-taskingu, bude při simulaci sledován.  
Přednastavený je proces 0.  
Pro sledování všech procesů se použije direktiva **#SIMTASK ALL**.

Platí pro:

08M2, 14M2, 18M2, 20M2

Příbuzné:

* [#sim](http://www.picaxe.com/commands/hash-sim)
* [#simspeed](http://www.picaxe.com/commands/hash-simspeed)

Příklad:

Pomocí **#simtask 1** se při simulaci zobrazují jen příkazy procesu 1 (příkazy od návěští „start1:“). Příkazy procesu 0, nebudou zobrazovány, ale budou stále simulovány.

#simtask 1

start0: ; start procesu 0

toggle B.0 ; blikání LED na pinu B.0

pause 1000

goto start0

start1: ; start procesu 1

toggle B.1 ; blikání LED na pinu B.1

pause 2000

goto start1

Jméno:

#slot

Syntaxe:

**#SLOT number**

Popis:

Vybere vnitřní program (0-3), nebo i2c program (4-7) pro řadu X2.

Platí pro:

20X2, 28X2, 40X2

Příbuzné:

* [run](http://www.picaxe.com/commands/run)
* [booti2c](http://www.picaxe.com/commands/booti2c)
* [readrevision](http://www.picaxe.com/commands/readrevision)
* [#revision](http://www.picaxe.com/commands/hash-revision)

Příklad:

#slot 1

#slot 4

Jméno:

#terminal

Syntaxe:

**#TERMINAL OFF**

**#TERMINAL baudrate**

Baudrate - je přenosová rychlost - 300, 600, 1200, 4800, 9600, 19200, 38400 Baud.

Popis:

**Nastavuje pouze programový editor, netýká se procesoru.**

Nastavuje sériový terminál, který se otevře po nahrátí programu do PICAXE (možnost nastavení komunikační rychlosti).

Platí pro:

Vše

Příbuzné:

* [#com](http://www.picaxe.com/commands/hash-com)

Příklad:

#terminal 4800

Jméno:

#undefine

Syntaxe:

**#UNDEFINE label**

Popis:

Odebrání *popisky* z aktuálního seznamu **#define**.

Platí pro:

Vše

Příbuzné:

* [#define](http://www.picaxe.com/commands/hash-define)
* [#ifdef](http://www.picaxe.com/commands/hash-ifdef)
* [#ifndef](http://www.picaxe.com/commands/hahs-ifndef)

Příklad:

Ukázka použití direktivy **#undefine**

#define clock8 ; definování „clock8“

#ifdef clock8 ; když je definováno „clock8“

let b1 = 8 ; pak b1 = 8

#else ; jinak (když není definován)

let b1 = 4 ; b1 = 4

#endif ; konec endif

#undefine clock8 ; zrušení definice „clock8“

#ifndef clock8 ; když „clock8“ není definován

let b2 = 4 ; pak b2 = 2

#else ; jinak (když je definován)

let b2 = 8 ; b2 = 8

#endif ; konec endif

Jméno:

#include

Syntaxe:

**#INCLUDE “filename”**

Popis:

**Pozor: Tato funkce není zatím použita, je připravená pro budoucí verzi.**

Do programového kódu zahrne i kód obsažený v souboru *filename*.

Platí pro:

Vše

Příklad:

#include “c:\test.bas”