



Mikropočítačový modul Stamp II.

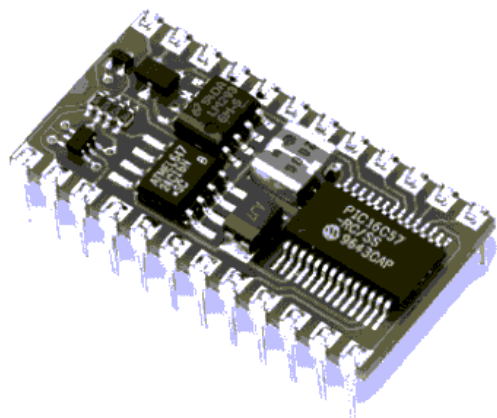
Obsah

Mikropočítačový modul Stamp II	2
Popis	2
Programovanie	3
Príklady	4
Prvý príklad	4
Úloha 1:	4
Číslkové vstupy a výstupy	4
Úloha 2:	4
Príkaz RTime	5
Úloha 3:	5
Úloha 4:	5
Príkaz pulsIn	5

Pin	Name	Description	Pozn.
1	TX	Serial output	Pripojenie
2	RX	Serial input	sériovej
3	ATN	Serial attention	linky k PC
4	GND	Serial ground	
5	P0	I/O pin 0	Každý pin môže napájať 20mA a absorbovať 25mA.
6	P1	I/O pin 1	
7	P2	I/O pin 2	
8	P3	I/O pin 3	
9	P4	I/O pin 4	
10	P5	I/O pin 5	
11	P6	I/O pin 6	
12	P7	I/O pin 7	
13	P8	I/O pin 8	Skupina P0-7 a P8-15 spolu max 40mA zdroj max 50mA spotr.
14	P9	I/O pin 9	
15	P10	I/O pin 10	
16	P11	I/O pin 11	
17	P12	I/O pin 12	
18	P13	I/O pin 13	
19	P14	I/O pin 14	
20	P15	I/O pin 15	
21	+5V *	+5V supply	5-V napájanie I/O
22	RES	Reset I/O pin	Log 0 = reset
23	GND	System ground	
24	PWR *	Regulator input	Rozsah 5-15 VDC

Mikropočítačový modul Stamp II

BASIC Stamp je mikropočítač, ktorý vyvinula firma Parallax, Inc. Dá sa jednoducho a rýchlo programovať v jazyku PBASIC. Názov "Stamp" symbolizuje rozmery, ktoré sú porovnateľné s poštovou známkom.

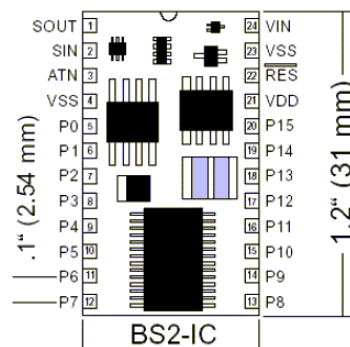


Obr. 1: Mikropočítačový modul Stamp II.

Modul BS 2 má 2Kb pamäť EEPROM, do ktorej sa ukladá vykonávateľný program (BASIC) a prípadné data. K dispozícii máme 16 I/O pinov a sériovú linku, ktorá je po nahratí programu k dispozícii pre užívateľský program.

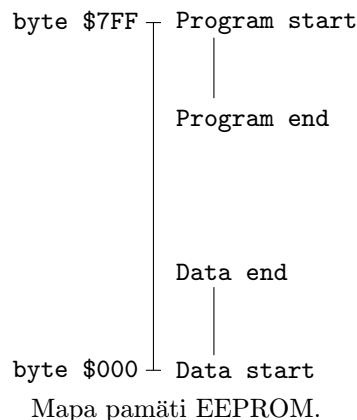
Popis vývodov:

Zapojenie vývodov modulu je na obrázku:



Obr. 2: Zapojenie vývodov.

Ako pamäť pre premenných môžeme využiť 32 bytovú RAM, ktorá okrem toho obsahuje registre pre prístup k I/O pinom. Do pamäti môžeme pristupovať po bitoch, polbytoch (nibble), bytoch a slovách (word). Pamäť RAM je po inicializácii vynulovaná, pamäť EEPROM prežije aj výpadok napájania.



WORD	Description	R/W
\$0-	Pin input states	read-only
\$1-	Pin output latches	read/write
\$2-	Pin directions	read/write
\$3-	variable space	read/write
\$4-	variable space	read/write
\$5-	variable space	read/write
\$6-	variable space	read/write
\$7-	variable space	read/write
\$8-	variable space	read/write
\$9-	variable space	read/write
\$A-	variable space	read/write
\$B-	variable space	read/write
\$C-	variable space	read/write
\$D-	variable space	read/write
\$E-	variable space	read/write
\$F-	variable space	read/write

Mapa pamäti RAM.

Slovo \$0 je odrazom stavu na všetkých 16 I/O pinoch. Tento register má niekoľko spôsobov prístupu:

INS	the entire 16-bit word
INL	the low byte of INS
INH	the high byte of INS
INA	the low nibble of INL
INB	the high nibble of INL
INC	the low nibble of INH
IND	the high nibble of INH
IN0	the low bit of INS = pin P0
:	
IN15	the high bit of INS = pin P15

Ďalší špeciálny register – Word \$1 obsahuje výstupný záchytný register (latch) pre všetkých 16 I/O pinov. Ak je pin vstupný, obsah je nezaujímový, ak je vo výstupnom režime, zápis do príslušného bitu ovplyvňuje stav na pine. Podobne ako v predošlom prípade, môžeme využiť tieto symbolické názvy:
OUTS, OUTL, OUTH, OUTA ... OUTD, OUT0 ... OUT15.

Dôležitý je aj register \$2, ktorý obsahuje bity pre konfiguráciu jednotlivých pinov. Ak má byť pin vstupný, príslušný bit musí byť nastavený na log.0; ak výstupný tak do log.1. Potom o jeho skutočnom stave rozhoduje príslušný bit v OUTS. Samozrejme aj teraz môžeme využiť symbolické názvy:
DIRS, DIRL, DIRH, DIRA ... DIRD, DIR0 ... DIR15.

Slová \$3-\$F sú určené na všeobecné použitie a nemajú žiadne symbolické názvy.

Programovanie

Úplná príručka s popisom príkazov jazyka BASIC je v prílohe. V tejto časti sa budeme zaoberať len deklarováním premenných.

Na vyhradenie pamätového priestoru musíme použiť jeden z troch príkazov VAR, CON a DATA.

Aby prekladač vedel, aké veľké miesto má vyhradiť pre premenné, musíme ich na začiatku deklarovať príkazom VAR. Okrem mena premennej musíme uviesť aj jej typ, aby bola určená jej veľkosť. Príklady:

```
cat    var nib    '4-bitová premenná
mouse  var bit    'logická premenná
dog    var byte   'ako char v C
rhino  var word   '16-bitová premenná
snake  var bit(10) 'pole s 10 bitmi
```

Ako kompilátor usporiadal vaše premenné sa môžete pozrieť po preklade ak stlačíte Alt-M.

Každá premenná môže používať tieto postfixy:

```
LOWBYTE 'low byte of a word
HIGHBYTE 'high byte of a word
BYTE0 'byte0 (low) of a word
BYTE1 'byte1 (high) of a word
```

```
LOWNIB 'low nibble
HIGHNIB 'high nibble
NIB0-3 'nibble 0-3
```

```
LOWBIT 'low bit
HIGHBIT 'high bit
BIT0-15 'bit0-15
```

Symbolické meno môžeme priradiť aj pinom:

```
tlacitko var in5 'cez "tlacitko"
                'mozeme citat stav P5
```

Deklarácia konštánt – CON, sa používa podobne, ale je určená na definovanie konštánt a priradovanie symbolických mien. Príklady:

```
level CON 10 "level" is 10 in program
limit CON 10*4<<2 "limit" is 160
```

Príkaz DATA je určený na deklaráciu dát, ktoré nemusia byť uložené v pamäti dát medzi premennými, pretože sa nemenia. Sú to rozličné tabuľky, textové reťazce a pod. Ukladajú sa do pamäti EEPROM.

Poslednou užitočnou vecou sú prefixy, ktoré rozlišujú, v akom kóde je zapísané číslo:

```
$BA1F - Hex
%111001111 - Binary
99 - Decimal
"A" - ASCII
```

Príklady

Modul pripojíme pomocou sériovej linky k počítaču PC, ktorý využijeme ako editor a preprocesor. Príkazy BASICu sa totiž najprv "predpreložia" a až v úspornej forme posielajú do modulu.

Editor spustíme príkazom `C:\>stamp2.exe`

Program si sám otestuje na ktorom porte je pripojený modul a ak ho nájde, vypíše priaznivú správu. Ak vypíše `ERROR * Hardware not found`, treba skontrolovať pripojenie kábla a napájania. Ak ani potom nenájde modul, treba zavolať cvičiaceho.

V prostredí editora nemáme k dispozícii veľa príkazov. Ich zoznam sa vypíše po stlačení F1:

Cursor Keys		Alt-X	Cut Text
		Alt-C	Copy Text
	~PgUp	Alt-V	Paste Text
PgUp	↑	~→	Mark using Shift
Home	← →	End	
~←	↓	PgDn	Alt-F Find Text
	~PgDn		Alt-N Find Next
Delete Keys		Alt-V	Replaces
		Alt-R	Run Program
BkSp	X[]	Alt-M	Map Program
Del	[X]	Alt-I	Identify
↑BkSp	X..X[]	Alt-L	Load File
↑Del	[X].X	Alt-S	Save File
~BkSp	X.[]X	Alt-Q	Quit (ESC)

Prvý príklad

Prvý príklad bude jednoduchý a jeho cieľom je ilustrovať použitie jednoduchých príkazov:

```
' Nas prvý ukazkový program v BASICu
',
' Komentare sa oddeluju obrateným
' apostrofom (na klavesnici
' je to znak pod úvodzovkami)
',
',
' Treba zdefinovať použité premenné:
cis1 var byte
cis2 var byte
x var byte 'tu bude výsledok

cis1 = 2
cis2 = 1

x = cis1 + cis2

' Výsledok si môžeme pozrieť
' v špeciálnom okne editora:

debug dec cis1, "+", dec cis2, "=", dec x
end
```

Úloha 1:

Prepíšete program do počítača a príkazom "ALT + R" ho pošlite do modulu. Ak ste postupovali správne, program začne hneď fungovať.

Ak uvidíte namiesto toho len chybové hlásenie, zrejme bude niekde chyba. Ak počítač hlási "hardware not found", skontrolujte kábel medzi PC a modulom.

Ak je chyba niekde inde, skontrolujte si program. Všimajte si, či nemáte niekde preklep, alebo nevypadlo niekde písmenko. Ak ste už všetko vyskúšali a stále nič, spýtajte sa asistenta na cvičení.

Číslkové vstupy a výstupy

Modul StampII je vložený do experimentálnej dosky, ktorá umožňuje demonštrovať niektoré základné funkcie obvodu. Jej kompletná schéma zapojenia je v prílohe.

Príkladom, ako ušetriť jeden pin pri nedostatku vstupov a výstupov je zapojenie tlačítka a LED diódy (viď schéma).

Ich využitie ilustruje nasledujúci príklad:

```
' Druhý ukazkový príklad pre MMP
',
',
' Deklarácia premenných
' sa dá využiť aj na priradenie
' názvov pinom

OK var bit
LED var OUT8 'skontrolujte podľa schémy!

DIR11 = 0 ' tlačitko - vstup
DIR8 = 1 ' nastav ako výstup
' čo ak date = 0?

loop: ' Hlavný program ...
LED = 1 ' Zhasnime LEDku
pause 500 ' Chvilu (0.5s) počkame
LED = ~1 ' a naopak...
pause 500

stoj:
if IN11 = 0 then stoj

goto loop 'stále dokola
```

POZNÁMKA: postupnosť príkazov `DIR8 = 0` a `OUT8 = 1 (0)` sa dá nahradiť jediným `HIGH 8 (LOW8)`.

Úloha 2:

Napíšete program, ktorý bude prepínať stav diódy prislúchajúcej k tlačítku na pine 8 pri každom stlačení (signalizácia on/off).

Návod: Aby ste prečítali stav tlačítka, musíte na chvíľu prepnúť smer na vstupný. Potom zmeníte stav LED diódy a obnovíte pin ako výstupný. Pozor na zámky – mali by ste počkať, kým sa tlačítko uvoľní.

Príkaz Rctime

```
RCTIME pin, state, result
```

Príkaz vypočíta čas, po ktorý je *pin* v stave *state*. Výsledok uloží do premennej *result*. Príkaz sa obvykle používa na meranie nabijacej resp. vybíjacej konštanty RC obvodu.

Pin (0–15) udáva, na ktorom pine budeme merať. Tento pin sa nastaví ako vstupný a zostane tak aj po ukončení príkazu.

State (0/1) hovorí, ktorá počas ktorej logickej úrovne na pine prebieha meranie.

Result (0–65535) je premenná, do ktorej sa uloží výsledok. Je to počet $2\mu\text{s}$ jednotiek.

Príkaz Rctime sa využíva na meranie doby nabíjania, resp. vybíjania externého kondenzátora. O celkovej dobe rozhoduje RC časová konštanta, takže ak jednu z premenných necháme konštantnú, môžeme merať buď kapacitu, alebo odpor. To sa využíva na pripojenie jednoduchých kapacitných snímačov vlhkosti, alebo odporových snímačov teploty, tlaku, polohy a pod.

Pri vyvolaní inštrukcie sa spustí počítadlo, ktoré zvýši svoj stav o jednotku každé $2\mu\text{s}$. Počítanie sa zastaví keď sa na pine zmení logická úroveň *state*. Ak pin nie je ani na začiatku merania v stave *state*, funkcia sa ukončí a vráti hodnotu 1. Ak pin ostane v stave *state* dlhšie ako $0,131\text{s}$ ($65535 \times 2\mu\text{s}$), funkcia vráti 0.

Pri meraní sa využíva skutočnosť, že mikropočítač PIC má stabilnú a pomerne presne definovanú rozhodovaciu úroveň medzi logickými úrovňami 0 a 1 – 1,5 V. Je možné merať dobu nabíjania, s rozsahom (0 – 1,5) V. presnejšie meranie však dosiahneme pri meraní vybíjania v rozsahu (5 – 1,5) V.

Pred vykonaním inštrukcie musíme dostať kondenzátor do definovaného stavu. Pre zapojenie na obrázku je situácia trochu mäťúca. Kondenzátor musíme najprv vybiť, čo však v rozpore so „sedliackym rozumom“ znamená zapísať na pin log. 1. – viete prečo?

Ukázkový program na meranie odporu pripojeného potenciometra, ktorý môže predstavovať napr. snímač polohy:

```
result var word ' 16-bitova premenna
pot con 7 ' Alias pre potenciometer

START:
high pot ' Nastav +5 Vdc (vybijanie)
pause 5 ' Cakaj kym sa C vybije
rctime pot,1,result ' Zmeraj RC-cas nabijania
debug cls,dec result ' Zobraz na obrazovke
pause 100 '
goto START ' Opakuj stale dokola
```

Zaujímá nás však najmä to, aký je súvis medzi hodnotou meraného odporu a hodnotou, ktorú vráti funkcia Rctime.

Úloha 3:

Vypočítajte veľkosť odporu R, ak viete, že pripojený kondenzátor má kapacitu $1\mu\text{F}$ a funkcia Rctime vrátila hodnotu 3000.

Návod: Kondenzátor sa nabíja tak, že napätie v bode (1) – viď obr., klesá z hodnoty $U_{cc} = 5\text{V}$ až na nulu (teoreticky). Nabíja sa po exponenciále s časovou konštantou RC. Počítadlo začne počítat až kým úroveň na vstupe neprestane mať úroveň log.1, teda 1,5 V. Vtedy sa počítanie ukončí.

Podobným spôsobom treba vypočítat aj minimálnu dobu, ktorú musíme počkať, aby sa kondenzátor dostatočne vybil. Rozdiel je však v tom, že nemôžeme vybíjať kondenzátor cez pin procesora skratom, pretože maximálny prúd do pinu je obmedzený. Preto je v schéme nakreslený ochranný odpor 220Ω , cez ktorý sa bude kondenzátor vybíjať. Kondenzátor budeme považovať za vybitý ak napätie klesne na 99% ustálenej hodnoty, teda 4-násobok časovej konštanty.

Úloha 4:

Vypočítajte aká je minimálna vybíjacia doba pre kondenzátor $1\mu\text{F}$ a ochranný odpor 220Ω .

Príkaz pulsln

```
PULSIN pin, state, result
```

Príkaz zmeria čas, po ktorý je *pin* v stave *state*. Výsledok uloží do premennej *result*. Je to ďalšia možnosť, ako pripojiť snímač k mikropočítaču. Pomocou jednoduchého oscilátora (napr. NE555) urobíme prevod na frekvenciu a tú potom zmeriame.

Pin (0–15) udáva, na ktorom pine budeme merať. Tento pin sa nastaví ako vstupný a zostane tak aj po ukončení príkazu.

State (0/1) hovorí, ktorá logická úroveň na pine je rozhodujúca. Meranie sa začína buď hranou $0 \nearrow 1$ (*State* = 1) alebo $1 \searrow 0$ (0) a opačný prechod meranie ukončí. Prechodom sa spustia "stopky", ktoré sa inkrementujú po $2\mu\text{s}$.

Result (0–65535) je premenná, do ktorej sa uloží výsledok. Je to počet $2\mu\text{s}$ jednotiek. Ak sa ani po 0.131 sec nezmení stav pinu, meranie sa neuskutoční a výsledok merania je 0.

Príklad:

```
time var word

again:
PULSIN 7,1,time ' Zmeraj kladny impulz
if time=0 then again ' Ak 0, skus znova
debug cls,dec ? time ' Inak zobraz vysledok
goto again ' A stale dokola
```