BS2 - BASIC BÉLYEG II

Felhasználói Kézikönyv

1995. augusztus

Fordította: Dr Kónya László

<=	- kisebb vagy egyenlő
=	- egyenlő
=>	- nagyobb vagy egyenlő
>	- nagyobb mint
<>	- nem egyenlő

Ezek hamis állitás esetén 0-át, igaz állítás esetén $FFFF-et adnak eredményül. Természetesen a NOT, AND, OR, and XOR műveletek kombinálhatók.

Példák:

```
outs = ~ dcd nibarray(index)
IF x<1 or not y>3 and (z=0 xor r=3) then loopback
```

4. Utasítások

Bárhol ahol egy érték szükséges, ott egy kifejezés is irható. A gyakran használt pin kifejezés a tok valamelyik IO lábát jelenti. A következőkben egységes módon rövidwn ismertetjük a BS2 utasításait.

DEBUG outputdata Változók és üzenetek megjelenítése hibakeresési céllal.
Használata: DEBUG "Hahó!" végrehajtáskor a Hahó! szöveg jelenik meg a képernyőn.

Végrehajtáskor a DEBUG után álló adatok a PC-be kerülnek vissza megjelenítésre. Ez a megjelenítés több módon lehetséges: közvetlenül az adatot, vagy decimálisan, hexa, bináris, vagy ascii kódban.. Számkijelzés esetén egy kifejezés eredménye részenként vagy végső formában megjeleníthető. Például:

```
DEBUG dec? X        'X változó decimálisan
                    kapjuk (ha x=1):
      X = 1

DEBUG dec X         'X megjelenítése                    decimálisan, kapjuk:
      1
```

A megjelenítéskor több szó előzheti meg a kifejezéseket:

ASC?	'ascii érték relációjellel együtt kiiratása
STR bytevar	'string kiratása bájt-tömbből 0. elemig
STR bytevar\length	'string kiratása bájt-tömbből - length bájtot
REP value\count	'count darab value kiratása
DEC value	'value értéke decimálisan
DEC1-DEC5 value	'value értéke decimálisan - 1-5 digitet
SDEC value	'value értéke előjeles decimálisan
SDEC1-SDEC5 value	'value értéke előjeles decimálisan - 1-5 digitet
HEX value	'value értéke hexadecimálisan
HEX1-HEX4 value	'value értéke hexadecimálisan - 1-4 digitet
SHEX value	'value értéke előjeles hexadecimálisan
SHEX1-SHEX4 value	'value értéke előjeles hexadecimálisan - 1-4 digitet
IHEX value	'value értéke hexadecimálisan '$' jelöléssel
IHEX1-IHEX4 value	'value értéke hexadecimálisan - 1-4 digitet '$' jelöléssel
ISHEX value	'value értéke előjeles hexadecimálisan '$' jelöléssel
ISHEX1-ISHEX4 value	'value értéke előjeles hexadecimálisan '$' jelöléssel - 1-4 digitet
BIN value	'value értéke binárisan
BIN1-BIN4 value	'value értéke binárisan - 1-16 digitet
SBIN value	'value értéke előjeleles binárisan
SBIN1-SBIN16 value	'value értéke előjeles binárisan - 1-16 digitet
IBIN value	'value értéke binárisan '%' jelöléssel
IBIN1-IBIN16 value	'value értéke binárisan '%' jelöléssel -1-16 digitet
ISBIN value	'value értéke előjeles binárisan '%' jelöléssel
ISBIN1-ISBIN16 value	'value értéke előjeles binárisan '%' jelöléssel -1-16 digitet

Használata: RETURN

A "GOSUB cimke" utasítással elindított szubrutin végén elhelyezett RETURN utasítás hatására a program a GOSUB utáni utasítással folytatódik. Ha egy RETURN utasítást nem előzi meg a GOSUB a végrehajtás a program kezdetétől fog indulni.

Értékadás változónak érték adása
Használata: változó = érték vagy kifejezés

A változóba az adott érték vagy a kifejezés alapján kiszámított érték kerül.

LOOKUP Táblázatból való kiolvasás (lookup) adott helyről
Használata: LOOKUP index,[value0,value1,value2,...valueN],variable

Ha index=0, akkor value0 lesz a változóba írva. Ha index=1, akkor value1, és így tovább. Ha az index nagyobb mint a felsorolt value értékek száma a variable változó értéke nem fog változni.

LOOKDOWN. Egy értékhez hasonlít és eredményül egy sorszám (index) kerül egy változóba.
Használata: LOOKDOWN value,??[value0,value1,value2,...valueN],variable

?? egy hasonlító operátor: =, <>, >, <, <=, => (* az alapértelmezés). Összehasonlítódik a value és a value0; Ha feltétel teljesül, 0 érték kerül a variable változóba. Ha az összehasonlítás eredménye hamis, akkor a value1-el történik a hasonlítás, igaz volta esetén 1 érték íródik a változóba különben folytatódik a hasonlítás. Azaz az első igaz hasonlításnak megfelelő szám kerül a változóba. Ha minden hasonlítás eredménye hamis, akkor változó értéke nem módosul.

RANDOM Álvéletlen számot generál egy szó (16 bites) változóba.
Használata: RANDOM wordvariable

wordvariable 16-szor iterálódik, hogy minden bit megváltozhasson.

SEROUT tpin,baudmode,{pace,}[outputdata] soros adatkivitel
SEROUT tpin\fpin,baudmode,{timeout,tlabel,}[outputdata]
Használata: SEROUT 3,6+$4000,("The temperature is ",dec temp," degrees.",cr)

Ha tpin értéke 0-15 közötti van, az I/O lábat jelöli. pin, ha 16 akkor a belső soros portot. Az opcionálisan megadható fpin az adatáramlást vezérlő láb. Baudmode egy összetett érték, amely a baud rate, paritás, egyenes vagy invertált kimenet nyitott vagy meghajtott kimeneti áramkör megadására szolgál. Ha fpin nincs megadva, akkor az opcionális "pace" (egy karakterre való várakozás ideje) a milliszekundumba megadott érték. Ha fpin adott, akkor az opcionális timeout (milliszekundumban) és egy cimke (timeout label) használható az adatátvitel kézbentartására.

baudmode egy bit ideje mikroszekundum-20 egységben (pl. 300baud az 3313).
baudmode 15. bitje ($8000) ha 0 akkor hajtott kimenet, ha 1 akkor open-drain/source (SEROUT)
baudmode 14. bitje ($4000) ha 0 akkor a kimnet egyenes, ha 1 akkor inverált.
baudmode 13. bitje ($2000) ha 0 akkor 8-bites paritás nincs, ha 1 for 7-bites paritásos a kivitel

outputdata szintaxisa követi a DEBUG-nál bemutatott konvenciókat.

Megjegyzés: Ha tpin=16 (belső I port), a baudmode 14. és 15. bitje nem használt, de fpin működését befolyásolják.

SERIN rpin{\fpin},baudmode,{plabel,}{timeout,tlabel,}[inputdata] soros adatbeolvasás
Használata: SERIN 16,32+$2000,60000,nothingcame,[WAIT (","),STR bytearray\16\","]

Ha rpin értéke 0-15 közötti van, az I/O lábat jelöli, ha 16 akkor a belső soros portot. Az opcionálisan megadható fpin az adatáramlást vezérlő láb. Baudmode jelentése a SEROUT utasításnál található. Plabel cimkére akkor történik ugrás, ha engedélyezett paritás esetén paritáshiba lép fel. Timeout egy opcionális érték amellyel meghatározhatjuk milliszekundumban hogy mennyi várakozás után ugorjunk a tlabel cimkére.

Inputdata e következő konvenciókat követi:

```
variable                'beolvas egy bájtot és a variable változóba tárolja

STR bytearray\L{\E}     'string bevitele a bytearray tömbbe, amelynek hossza L
                        'egy opcionális E terminálo karakterrel (a maredék bájtok nullával
                        'lesznek feltöltve )
SKIP L                  'beolvas L bájtot, de nem törödik vele
WAITSTR bytearray       'Várakozás egy stringre (bytearray 0-val terminált
WAITSTR bytearray\L     'Várakozás egy L hosszúságú stringre
WAIT (value,value,...)  'Várakozás maximum 6 bájtos bájtsorozatra

      DEC       variable      'decimális érték olvasása
DEC1-DEC5       variable      'fix hosszúságú decimális érték olvasása
     SDEC       variable      'előjeles decimális érték olvasása
SDEC1-SDEC5     variable      'fix hosszúságú előjeles decimális érték olvasása
      HEX       variable      'hexadecimális érték olvasása
HEX1-HEX4       variable      'fix hosszúságú hexadecimális érték olvasása
     SHEX       variable      'előjeles hexadecimális érték olvasása
SHEX1-SHEX4     variable      'fix hosszúságú előjeles hexadecimális érték
olvasása
      IHEX      variable      'jelölt ($) hexa érték olvasása
IHEX1-IHEX4     variable      'fix hosszúságú jelölt ($) hexa érték olvasása
     ISHEX      variable      'előjeles jelölt ($) hexa érték olvasása
ISHEX1-ISHEX4   variable      'fix hosszúságú előjeles jelölt ($) hexa érték olvasása
      BIN       variable      'bináris érték olvasása
BIN1-BIN16      variable      'fix hosszúságú bináris érték olvasása
     SBIN       variable      'előjeles bináris érték olvasása
SBIN1-SBIN16    variable      'fix hosszúságú előjeles bináris érték olvasása
     IBIN       variable      'jelölt (%) bináris érték olvasása
IBIN1-IBIN16    variable      'fix hosszúságú jelölt (%) bináris érték olvasása
     ISBIN      variable      'előjeles jelölt (%) bináris érték olvasása
ISBIN1-ISBIN16  variable      'előjeles fix hosszúságú jelölt (%) bináris érték olvasása
```

Megjegyzés: Ha rpin=16 (belső port), a baudmode 14. és 15. bitje nem használt, de fpin működését befolyásolják.

READ Az EEPROM valamelyik tárolójának tartalmát változóba olvassa
Használata: READ location,variable

location változó vagy állandó (0-2047) a bájt címe. Az adattárolás 0-nál kezdődik és felfelé növekszik, míg a programterület 2047-től lefelé növekszik. variable változó (0-255) a kiolvasott bájtot tartalmazza.

WRITE Adatírás az EEPROM-ba.
Használata: WRITE location,byte

location változó vagy állandó az írandó bájt címe (0-2047). Az adattárolás 0-nál kezdődik és felfelé növekszik, míg a programterület 2047-től lefelé növekszik. byte változó vagy állandó (0-255) a kiirandó bájt.

XOUT X-10 kódok kivitele az USA-ban használatos TW523 vagy TW513 modulokba (by X-10 Powerhouse).
Használata: XOUT mpin,zpin,[house\keyorcommand{\cycles,...]

mpin alacsony szintű kimenet és ez moduláció vezérlő. zpin a nullátmenet figyelő bemenet. House is the house kód (0-15 azaz 'A'-'P'). Keyorcommand egy kulcs (0-15 azaz '1'-'16') vagy parancs (ld. táblázatot). Cycles iegy opcionális szám (alapértelmezése 2) amelyeket kizárólag a 'dim' és 'bright' parancsoknál lehet használni..

Parancs neve	érték
UNITON	%10010
UNITOFF	%11010
UNITSOFF	%11100
LIGHTSON	%10100
DIM	%11110
BRIGHT	%10110

TW513/TW523 egységek négy kivezetése és a BS2 közötti összeköttetés: 1 - zpin, 2 - VSS, 3 - VSS, 4 - mpin

Az X-10-ről röviden...
Háztartási elektromos készülékek távvezérlése az erősáramú hálózaton is lehetőség (körvezérlés elve). A vezérlést úgy valósítják meg, hogy hálózati nullátmenetkor 1 msec hosszúságú 120 kHz-es hullámcsomagot (burst (B)) küldenek át a hálózaton. Az adatformátum a következő:
START: három burst három egymást követő nullátmenetkor és azt követőleg egy szünet (S): BBBS.
Ezt követi a kilenc bit, ahol 1-nek a BS, 0-nak az SB felel meg. A teljes jelsorozatot a biztonság kedvéért kétszer viszik át.

START KÓD (2 bit)	HOUSE KÓD (4 bit)	KEY KÓD (5 bit)

A house kód azonosítja, hogy melyik készülékcsoportnak szól a parancs , míg a 32 key kód-ból 16 készülékcím a másik 16 parancs azonosítására szolgál.

SHIFTOUT soros bitkivitel szinkron módon.
Használata: SHIFTOUT dpin,cpin,mode,[data{\bits},...]

"dpin" az adatkimenet, "cpin" az órajel kimenet. Ha mode = 0 akkor a legkisebb helyiértékű bit megy ki először ha mode = 1 akkor a legnagyobb helyiértékű bit megy ki először. Bits a bitszám (8 az alapértelmezés).

A következő szimbólumok definiáltak a SHIFTOUT: használatakor:

symbol	érték
LSBFIRST	0
MSBFIRST	1

SHIFTIN soros bitbeolvasás szinkron módon
Használata: SHIFTIN dpin,cpin,mode,[variable{\bits},...]

"dpin" az adatbemenet, "cpin" az órajel bemenet.
Mode

0	msb-first/pre-clock,	legnagyobb helyiértékű bit megy be először, adatot megelőzi az órajel
1	lsb-first/pre-clock,	legkisebb helyiértékű bit megy be először, adatot megelőzi az órajel
2	msb-first/post-clock,	legnagyobb helyiértékű bit megy be először, adat után van az órajel
3	lsb-first/post-clock.	legkisebb helyiértékű bit megy be először, adat után van az órajel

Variable-ba kerül a vett soros adat. Bits a bitszám (8 az alapértelmezés).

A következő szimbólumok definiáltak a SHIFTIN használatakor:

symbol	érték
MSBPRE	0
LSBPRE	1
MSBPOST	2
LSBPOST	3

COUNT ciklusok számlálása adott lábon
Használata: COUNTpin,period,variable

"pin" lábat bemeneti állapotba kell állítani. 'period' millisec ideig, a lábra érkező ciklusokat számoljuk meg és ez a szám kerül a variable változóba. A jel mindkét felének legalább 4 mikrosec hosszúnak kell lennie, ami

szimmetrikus hullámalakot feltételezve meghatározza a maximális frekvenciát : 125KHz . Az eredmény (0-65535) kerül a változóba.

PULSOUT Megadott hosszúságú impulzust ad ki a lábon
Használata: PULSOUT pin,period

Az eredetileg valamelyik (magas vagy alacsony szintű) kimeneti állapotban lévő "pin" láb period*2mikrosec ideig ellentétes állapotba áll, majd visszaáll eredeti állapotába.

PULSIN bemeneti impulzus mérése.
Használata: PULSINpin,state,variable

"pin" lábat bemeneti állapotba kell állítani. 'state' állapotú impulzusra várunk, és a hosszát 2mikrosec.-os felbontásba mérjük és ez a szám kerül a variable változóba. Ha az impulzus túl hosszú (>65535*2us vagyis >131ms), 0 érték íródik a változóba.

BUTTON Pergésmentesített billentyű olvasás, automatikus ismétlés, és ugrás ha a gomb a megadott helyzetben van
Használata: BUTTON pin,downstate,delay,rate,bytevariable,targetstate,label

- pin a használt be/kimeneti lábat jelöli.
- downstate változó vagy állandó (0 vagy 1) amely azt mondja meg, hogy mi a megnyomott állapotnak megfelelő logikai szint.
- delay változó vagy állandó (0-255) megadja azt az időt, amikor az automatikus ismétlés elkezdődik.
- rate változó vagy állandó (0-255) automatikus ismétlés ciklusidejét adja.
- bytevariable a munkaterület. A BUTTON első használata előtt nullát írva bele törölni kell.
- targetstate változó vagy állandó (0 vagy 1) amely megadja, hogy melyik állapot (state) (0=nincs megnyomva, 1=megnyomva) esetén kell elugrania.
- address cimke, amely megadja az ugrás helyét.

INPUT, OUTPUT, LOW, HIGH, TOGGLE, REVERSE a lábak I/O állapotának módosítása
Használata: INPUT pin

Pin egy 0-15 közötti szám.

Módosul:	OUTx	DIRx	Megjegyzés
INPUT	azonos	0	A láb bemenet lesz
OUTPUT	azonos	1	A láb kimenet lesz
LOW	0	1	A láb alacsony szintű kimenet lesz
HIGH	1	1	A láb magas szintű kimenet lesz
TOGGLE	vált	1	A láb szintet vált
REVERSE	azonos	vált	A láb irányt vált

FREQOUT Adott időtartamú és frekvenciájú két szinuszhullám kivitele
Használata: FREQOUT pin,milliseconds,freq1{,freq2}

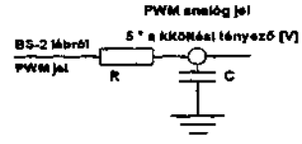
A "pin" láb kimenet lesz. "Milliseconds" adja meg a kimeneti jel hosszát. "Freq1" a kimeneti jel frekvenciája Hz-ben (0-32768Hz). "Freq2" opcionális és a második frekvenciát határozza meg. A láb egy RC körrel szűrhető, hogy szebb szinuszjelet kapjunk. Nagyimpedanciás hangszóró a lábról a csatoló és szűrőkapacitáson keresztül meghajtható

DTMFOUT Telefon DTMF tone hangok kivitele.
Használata: DTMFOUT pin,{ontime,offtime,}[key,key,key,...]

A "pin" láb kimenet lesz. Az alapértelmezés szerint be- illetve kikapcsolási idő: 200ms és 50ms. Ezek az ontime és offtime paraméterek msec.-ban történő megadásával megváltoztathatók. "Key"-ek értékei: 0-15 között lehetnek: 0-9 a '0'-'9'; számok, 10 a *; 11 a #; 12-15 az 'A'-'D' betűk, amelyek telefontasztatúrán el nem érhető negyedik oszlopának felelnek meg. A láb egy RC körrel szűrhető, hogy szebb szinuszjelet kapjunk. Nagyimpedanciás hangszóró a lábról a csatoló és szűrőkapacitáson keresztül meghajtható.

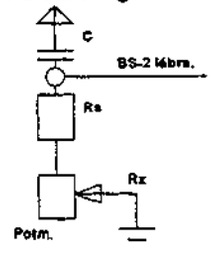
PWM Impuzus szélesség modulált kimenet
Használata: PWM pin,duty,cycles

Kiviszi a „pin" lábra a PWM jelet, majd a láb visszáll bemenetté. „Duty" egy 8-bites érték (0-255) amely a kitöltési tényezőt határozza meg. "Cycles" egy 8-bites érték (0-255) amellyel megadhatjuk, hogy hány ciklusig tartson a PWM jel. Egy periódus hossza: ~1ms..Felhasználható analóg jel kivitelére (0-5V) a lábra kötött külső soros ellenállás és az ellenállás másik végére kapcsolt kondenzátor segítségével (a kondenzátor másik végét a földre kötjük); az ellenállás-kondenzátor áramkörön kapcsolt periodikus PWM jel hatására a kimeneten a PWM jel kitöltési tényezőjével arányos feszültség jelenik meg (aluláteresztő szűrő).



RCTIME R-C kör töltési-kisütési idejének mérése.
Használata: RCTIME pin,state,variable

A "pin" láb bemeneti módba kerül, és az az idő lesz 2 mikrosec egységben mérve amig ez a láb 'state' állapotban van. Az eredmény a variable változóba kerül. Túlcsordulás esetén (>131msec), a változóba 0 íródik.
Ez utasítás ellenállás-kapacítás soros kör töltési-kisütési idejének mérésére alkalmas. Mivel a láb logikai billenési szintje ~1.4 volt, a legjobb az C egyik felét az 5V-ra kötni, mert így a kondenzátor feszültsége ~3.6 volt lehet. Például így lehet egy potenciométer állását egy számmá alakítani: kapcsoljunk .1mikroF kondenzátort a VDD (5V) tápfeszre, a kondenzátor másik felét a bemeneti lábra és a poteciométer csúszkájára, potenciométer egyik szélét kössük a földre VSS (ground). Állítsuk a lábat magas állapotba 1msec ideig, azután hajtsuk végre az RCTIME utasítást. A kapacitás-ellenállás közös pontja 5V-ról indulva nullára csökken. Amikor ~1.4 V-ot eléri a számlálás megáll, és az eredmény a változóba kerül. A potenciométer csuszkáját mozgatva ez az érték a csúszka helyzetétől függ. Célszerű egy ellenállást a potenciométerrel sorbakötni, hogy megakadályozzuk a magas állapotú láb direkt földre kötését.



NAP Csökkentett fogyasztású üzemmód egy rövid ideig.
Használata: NAP x

Mikor az idő eltelt, az I/O vonalak ~18ms időtartamra magas impedanciás állapotba kerülnek, a programvégrehajtás a következő utasítással folytatódik. Az x értékek - idők összefüggése:

x	~másodperc
0	.018
1	.036
2	.072
3	.14
4	.29
5	.58
6	1.2
7	2.3

SLEEP szundi x másodpercig (x=0 to 65535).
Használata: SLEEP x

Kisfogyasztású módba kerülünk, az I/O vonalak állapota frissítődik. Minden ~2.3 mp.-ben az I/O vonalak 18 msec időre magas impedanciás állapotba kerülnek. A fogyasztás átlagosan kb. 50 mikroamper. x másodperc után a programvégrehajtás a SLEEP utáni utasítással folytatódik. Bár a SLEEP felbontása ~2.3 mp, a hiba 1%-on belül van nagyobb időtartamoknál.

END program vége
Használata: END

Kisfogyasztású módba kerülünk, az I/O vonalak állapota frissítődik. Minden ~2.3 mp.-ben az I/O vonalak 18 msec időre magas impedanciás állapotba kerülnek. A fogyasztás átlagosan kb. 50 mikroamper. Az END állapot csak hardver reset-tel szüntethető meg.

PAUSE várakozás x msec-ig (x=0 to 65535).
Használata: PAUSE x

x milliszekundumos késleltés.

STOP programvégrehajtás megállítása
Használata: STOP

A programvégrehajtás felfüggesztődik, de nincs alacsony fogyasztású üzemmód. Hatása az END utasításhoz hasonló, azzal a különbséggel, hogy a kimnetek nem állnak be magas impedanciás állapotba, meghajtva maradnak. Reset-el lehet a STOP-ot megszüntetni.

Functional description of the Basic Stamp Experimental board

The Basic Stamp Experimental Board consist of 13 separate circuit modules. The positions and the name of the modules are the same on the schematic and the PCB. It makes easy to find the circuit parts.

Power
The power module supplies the digital supply voltage (5V) and the analog supply voltage (6V) for the op. amp LM324.

Input voltage requirement:	9-18 VDC
	7-12 VAC
Input current requirement:	1000 mA

Basic Stamp I.
The Basic Stamp I. module consists of two Basic Stamp I circuit. The A is a DIP version with a burn-out reset circuit (T1, R18, R19, R33, R36). The B is a SIP version, BS1-IC. They can be programmed separately by the PC through two 3 pin terminals A and B. The programming cable is part of the Stamp Development Kit and it has to connect to the PCs parallel port. The Stamps output pins are wired to the A and B side of a two-row pin connector. The experimental circuitry can be built up with the attached wire in the white drawer.

Basic Stamp II.
The Basic Stamp II. module has a 24 pin IC socket for the BS2-IC. There is a 9 pin D sub terminal for programing connection to the PC. The programming cable is part of the Stamp Development Kit and it has to connect to the PCs serial port. All pins of the BS2-IC are wired to a pin connector and labeled on the board.

RS-232 (Serial interface)
The RS-232 module is a voltage converter interface between the stamps 5V signals and the RS-232 signals (+15V, -15V). The input voltage of the RS-232 signals are limited by the D6 diode. The output signal steals the negative voltage from the idle RS-232 input line through D7 and C7. Therefor the communication can be only in simplex mode. This is not a limitation, because the Stamps can handle the serial lines only in a sequence.

Serial connection:	TXD of terminal to pin 2
	RXD of terminal to pin 3
	GND of terminal to pin 5

Display
The Display module consists of an LED display (7 segment) and a 4096 CMOS driver IC. The 4096 is basically an 8 stage shift register with serial input and data latch for each stage output. Data is shifted on the positive clock transition. Data from each stage of the shift register is latched on the negative transition of the strobe input.

Push-button
The Pushbutton module has 4 pushbuttons with active LOW signal.

Switches
The Switches module supplies four static logical signals set by the 4 separate slide switches.

Speaker
The Speaker module has a piezo buzzer, which can drive directly by the Stamps outputs.

Potentiometer
The R15 potentiometer and the C3 capacitor serial with it give a signal for the Stamp POT instruction.

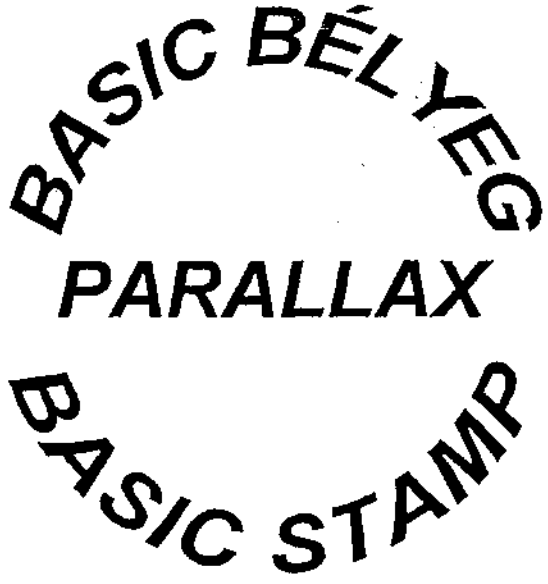
LEDs
The LEDs module has four separate LED lamps. LOW signal can turn on the LED lamps.

Net
This basic circuitry makes possible to built up a network from Stamps using the serial instructions (SERIN, SEROUT).

Drivers
The Drivers module is a power buffer interface. The ULN2803 IC, an eight Darlington connected transistor array is ideally suited for interfacing low logic level and the high current/voltage requirement of lamps, relays, motors etc. The ICs feature open collector outputs and free wheeling clamp diodes for transient suppression (C screw terminal output). The maximum rating of each open collector output is: Ic=500mA, Vo=50V. So we can use from a separate power supply. Its recommended to connect this separate power to the C terminal point too (Clamp diodes common point). We can not use the IC at the max. rating continuously or simultaneously on each Darlington, because of the dissipation of the package. Each stage is an inverter principally. Each two darlington is connected parallel to lower the dissipation. It results four power inverter channel in the module (1, 2, 3, 4 output terminal).
The fifth channel is a separate Darlington transistor with embedded clamp diode, BD645 (M output terminal). It has a higher max. power (3A, 60V). Heat sink is recommended at higher last.

DRIVER TERMINAL (J3):	U	+5V
	C	Clamp diode common point of ULN2803
	1	1. channel of ULN2803
	2	2. channel of ULN2803
	3	3. channel of ULN2803
	4	4. channel of ULN2803
	G	GND
	U	+5V
	M	Open collector of BD645 (ref. Motor)
	G	GND

PWM-D/A
This module uses the PWM instruction. The module consists of two identical channels (IN1 and IN2 inputs). Each channel has a digital output (DIG1, DIG2) and an analog output (ANA1, ANA2). The digital output is buffered by an op. amp. inverter. The analog output is a slow programmable voltage generator (D/A converter) using an RC integrator and an op. amp. If the pulse width is changed in the PWM, the analog voltage will change too. The 6V analog power is the supply voltage of the op. amp. raise the linearity of the conversion in the output range of 0-5V. The LM324 ICs consistsof four separate op. so it is well suited for the modules circuitry.



Programozói kézikönyv

1993.

HUMANSOFT

1. Bevezetés

Ma már sokan ismerik a kis, bélyeg nagyságú területen elhelyezett PC-ről BASIC-ban programozható 8 be/kimeneti vonalat tartalmazó vezérlőt, (írjuk le bátran: számítógépet!) de a vonalak száma, a programterület nagysága és a 8 bites adatkezelés sok feladat megoldására nem volt alkalmas. A PARALLAX cég szakemberei ezeket felismerve egy nagyobb teljesítményű változattal jelentek meg: ez a BASIC STAMP II, amit a továbbiakban BS2 elnevezésen fogunk emlegetni.

A BS2 lényegében egy szabványos 24 lábú tokban elhelyezett BASIC értelmező programot ROM-ban tartalmazó, oszcillátorral ellátott PIC mikrokontroller és hozzá kapcsolódó 2 kbájtos EEPROM memória, kiegészítve a PC-soros kapcsolatot és tápellátást biztosító áramkörökkel. A felhasználók számára 16 darab bitenként be- vagy kimenetként programozható vonal áll rendelkezésre. A memória a futtatandó BASIC program sűrítetten kódolt (tokenizált) alakját tartalmazza, amit a mikrokontroller végrehajt.

A **programfejlesztéshez** egy IBM PC kompatibilis gépre és a rajta futó fejlesztőprogramra van szükség. A fejlesztés a következő módon történik:
A BS2-t tápfeszültségre csatlakoztatjuk. Ez lehet egy stabilizált 5V-os táp, vagy maximum +15V-os tápfeszültség is használható. Egy kábellel összekötjük a rendszert a PC soros portjával. Ezek után a programfejlesztést biztosító STAMP2.EXE programot elindítva, az megkeresi hogy melyik porton van a bélyeg, és létrejön az összeköttetés abban az esetben, ha az RS232 csatlakozóban az RTS és DSR lábak össze vannak kötve. Az összekötés hiányában az indítás módja: **STAMP2 /1** ill. **STAMP2 /2** attól függően, hogy a COM1 ill. COM2 portra kötjük a számítóbélyeget. A PC-n futó program több feladatot is ellát:

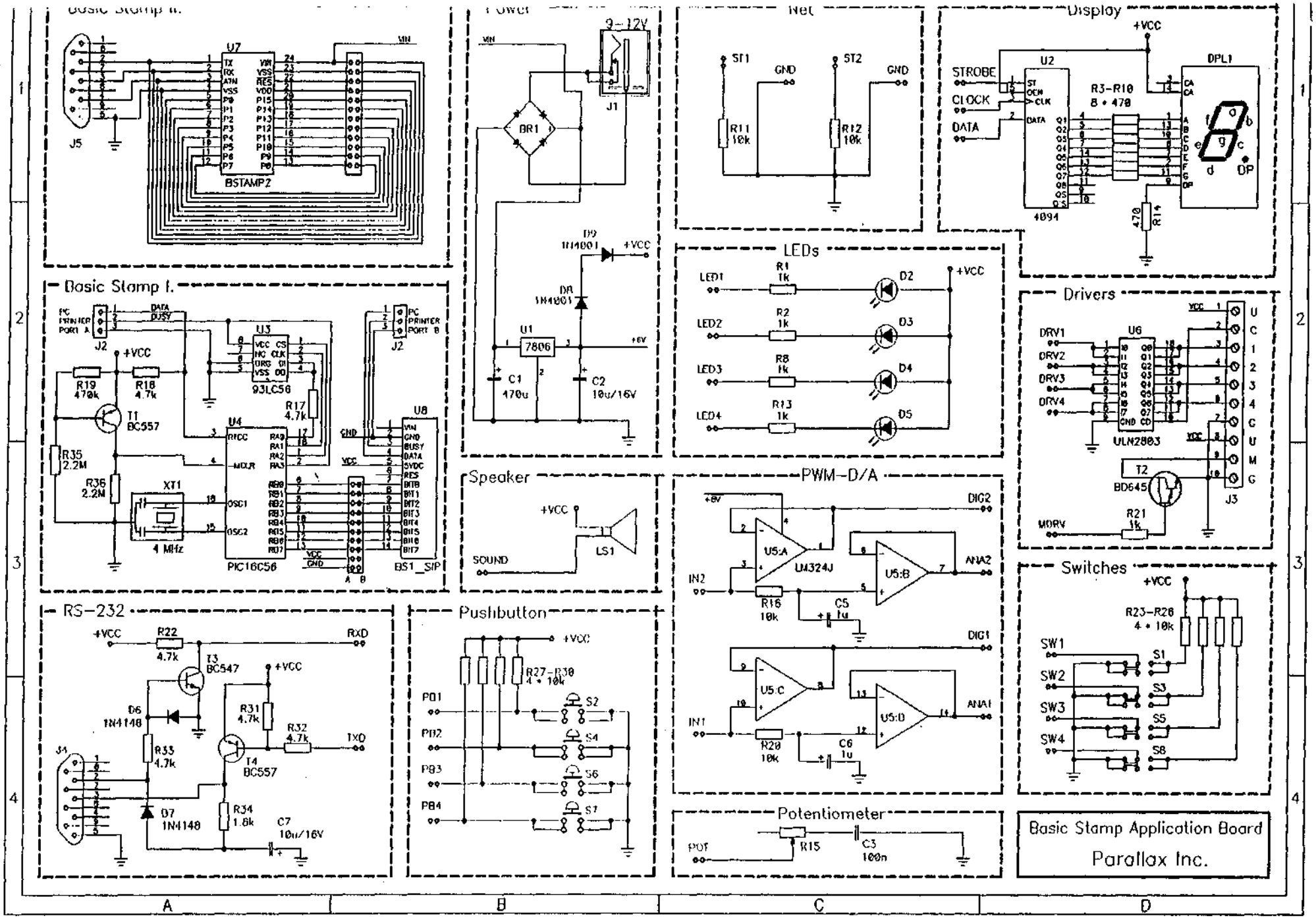
- beépített szövegszerkesztőjével a programok forrásszövegei megírhatók,
- a forráskódot a program lefordítja, és letölti BS2 EEPROM memóriájába,
- a program és az adatok által elfoglalt memóriaterület megtekinthető,
- kétirányú soros kapcsolatot biztosít a BS2 és a PC között.

A program az F1 billentyű megnyomásakor az 1. ábrán látható képernyőt jeleníti meg segítségként. A parancsok két nagy csoportra oszthatók: a szövegszerkesztést biztosító parancsokra, valamint a fejlesztést segítő parancsokra. A kurzormozgató billentyűk teszik lehetővé a szövegben való mozgást. Az Alt+billentyű kombinációval tudunk utasításokat adni.
Szövegkezelést (kivágás, másolás, beillesztés (Alt + X, C, V parancsok)) a szövegszerkesztőknél szokásos módon végezhetjük. A szöveg kijelölése a kurzormozgató billentyűk és a Shift billentyű együttes használatával lehetséges. Szöveg keresésre és helyettesítésre is van lehetőség (Alt + F, N, V parancsok)

```
F1 Help                                        BASIC Stamp II

  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
                   Alt-V  Replaces
   Delete Keys
                   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)
-------------------------------------
Hit any key                        Help
```

1. ábra


Basic Stamp II
U7
BSTAMP2
J5
VIN
Power
9-12V
J1
BR1
D9
1N4001
DR
1N4001
+VCC
U1
7806
+6V
C1
470u
C2
10u/16V
Net
ST1
ST2
GND
GND
R11
10k
R12
10k
Display
+VCC
U2
DPL1
STROBE
CLOCK
DATA
R3-R10
8 * 470
4094
R14
470
Basic Stamp I.
J2
DATA
BUSY
+VCC
R19
470k
R18
4.7k
U3
93LC56
R17
4.7k
T1
BC557
R35
2.2M
R36
2.2M
XT1
4 MHz
U4
PIC16C56
U8
BS1_SIP
LEDs
LED1
LED2
LED3
LED4
R1
1k
R2
1k
R8
1k
R13
1k
D2
D3
D4
D5
+VCC
Drivers
DRV1
DRV2
DRV3
DRV4
U6
ULN2803
T2
BD645
R21
1k
MDRV
J3
Speaker
+VCC
SOUND
LS1
PWM-D/A
+8V
U5:A
U5:B
U5:C
U5:D
LM324J
IN2
IN1
R16
10k
R20
10k
C5
1u
C6
1u
DIG2
DIG1
ANA2
ANA1
Switches
+VCC
R23-R26
4 * 10k
SW1
SW2
SW3
SW4
S1
S3
S5
S8
RS-232
+VCC
R22
4.7k
T3
BC547
D6
1N4148
R31
4.7k
R32
4.7k
RXD
TXD
R33
4.7k
T4
BC557
D7
1N4148
R34
1.6k
C7
10u/16V
J4
Pushbutton
+VCC
R27-R30
4 * 10k
PB1
PB2
PB3
PB4
S2
S4
S6
S7
Potentiometer
POT
R15
C3
100n
Basic Stamp Application Board
Parallax Inc.
A
B
C
D
1
2
3
4

Bevezetés

A BASIC-BÉLYEG lényegében egy BASIC interpretert ROM-ban tartalmazó PIC16C56 mikrokontroller és vele összekapcsolt 256 bájtos EEPROM memória. Az EEPROM a futtatandó BASIC program sűrítetten kódolt (tokenizált) alakját tartalmazza, amit a mikrokontroller végrehajt. A kontroller nyolc I/O vonala analóg és digitális ki- és bemenetként programozható, amit nagyon megkönnyít a BASIC értelemező erre specializált utasításkészlete.
A programfejlesztéshez egy PC-re és a rajta futó fejlesztőprogramra van szükség.
A fejlesztő program segítségével a futtatatandó BASIC program fejleszthető, és tokenizált alakja a BASIC-BÉLYEG-be letölthető és ott futtatható. A következőkben ismertetjük ezen BASIC programnyelv szitaktikáját, utasításait.

Szintaktikai szabályok

MEGJEGYZÉSEK

A megjegyzések aposztroffal (') kezdődnek és a sor végéig tartanak. Természetesen a jó öreg REM utasítás is használható.

Példa:

```
high 0          'a 0-ás lábat 1-be
```

ÁLLANDÓK (konstansok)

A konstansoknak négy típusa van: decimális, hexadecimális, bináris, és szöveg ASCII). A decimális számokat a szokásos módon írjuk, a hexa számokat a dollár-jel ($) előzi meg, bináris számokat a százalék-jellel kezdjük (%), és a szöveg konstansokat idézőjelek (") közé tesszük .

Példák:

```
100             '100 decimálisan
$64             ' 64 hexa
%01100100       ' 01100100 bináris
"A"             ' "A" ascii (65)
"Hello"         ' "Hello" - megegyezik a
                '"H","e","l","l","o" -val
B1 = B0 ^ $AA   ' xor B0  AA hexa értékkel
```

VÁLTOZÓK

A kontrollernek 16 bájtos RAM-ja van, ami 56 változóként szervezhető: 8 szó (word), 16 bájt, és 32 bit. A 32 bit változó a legalsó 2 szót vagy 4 bájtot fedi le.

Szavak:
PORT, W0, W1, W2, W3, W4, W5, W6
Bájtok:
PINS, DIRS (a kettő együtt a PORT)
B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11, B12, B13
Bitek:
PIN0, PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7 (együtt PINS-nek hívjuk)
DIR0, DIR1, DIR2, DIR3, DIR4, DIR5, DIR6, DIR7 (együtt DIRS-nek hívjuk)
BIT0, BIT1, BIT2, BIT3, BIT4, BIT5, BIT6, BIT7 (együtt B0)
BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15 (együtt B1)

SZIMBÓLUMOK

A szimbólumokat állandókhoz, változók elnevezéséhez, és program címekhez rendeljük. Állandók és változók esetén a hozzárendelést az egyenlőség jel (=), jelzi amit a változó vagy állandó követ. A programcímeket a szimbólumot követő kettőspont (:) jelzi.

Példák:

```
SYMBOL NOTES = 35  'állandó szimbólum def.
SYMBOL TIP = W0    'változó szimb. def.
PR: GOTO PR        'cím szimb. kettősponttal
```

FORMÁTUM

Kis vagy nagybetűt tetszés szerint vagy keverve is használhatjuk, kivéve, ha egy karakterfüzérben szerepel, ezt a " jelzi.

Egy soron belül a BASIC parancsok és címke deklarációk kettősponttal (:) elválaszthatók, ilyen módon több utasítás kerülhet egy sorba.

Példák:

```
SYMMODE=BIT0:SYMFLAG=BIT1:IMIT = 45

NOISE: RANDOM W0: B2 = B0 & $7F
SOUND 0,(B0,7) : GOTO NOISE
```

Általában egy program következő formájú:

```
SYMBOL ALIAS = W0 : SYMBOL DOGS = B4
SYMBOL RELAY = 3   'Szimbólumok

MAIN:                              'Fő program
PORT = %0000000011111111
GOSUB SUB
GOTO MAIN

SUB:PULSOUT RELAY,1000 : TOGGLE 0
RETURN                             'Szubrutinok
```

Megjegyzés: A fordító program mindig elhelyez egy END utasítást a program utolsó sora után.

BASIC BÉLYEG Utasításkészlet

Mikor a program először fut, miden változó 0 értékű lesz. A 16 bájt-os RAM terület a következő módon van elrendezve:

Szó	Bájt	Bitek
PORT	PINS	PIN0-PIN7 (PINS.0, PORT.0-PINS.7, PORT.7)
	DIRS	DIR0-DIR7 (DIRS.0, PORT.8-DIRS.7, PORT.15)
W0	B0	BIT0 - BIT7 (B0.0, W0.0 - B0.7, W0.7)
	B1	BIT8 - BIT15 (B1.0, W0.8 - B1.7, W0.15)
W1	B2	
	B3	
W2	B4	
	B5	
W3	B6	
	B7	
W4	B8	
	B9	
W5	B10	
	B11	
W6	B12	
	B13	

PORT az I/O port szó.

PINS és **PIN0-PIN7** egy porthoz (RB puffer) tartoznak az olvasás - módosítás - írás műveletsorból adódó problémák kivédésére. Mikor ezeket a változókat olvassuk, az RB-t olvassuk közvetlenül. Mikor írunk, akkor ténylegesen a megfelelő RAM-ba írunk, amelynek tartalma RB-be kerül át minden utasítás végrehajtása előtt.

DIRS és **DIR0-DIR7** a port (RB) irányjelző bitjei: 0=bemenet, 1=kimenet (pufferelve, hogy a TRIS regiszterek csak írhatóságából adódó problémát megoldjuk).
Ez az adatbájt !RB kerül minden utasítás végrehajtása előtt..

A **W0-W6** szavak, **B0-B13** bájtok, és **BIT0-BIT15** bitek általánosan használhatók.

W6 (B12,B13) négyszintű veremként használható, ha **GOSUB/RETURN** utasításokat hajtunk végre.

ANALÓG jelek kezelése, 8-bites felbontás ellenállás-kondenzátor áramkörrel

PWM pin,duty,cycles

Kiviszi a PWM jelet, majd a láb visszáll bemenetté. Felhasználható analóg jel kivitelére (0-5V) a lábra kötött külső soros ellenállás és az ellenállás másik végére kapcsolt kondenzátor segítségével (a konden-

zátor másik végét a földre kötjük); az ellenállás-kondenzátor áramkörön kapcsolt periodikus PWM jel hatására a kimeneten a PWM jel kitöltési tényezőjével arányos feszültség jelenik meg (aluláteresztő szűrő).

- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **duty** változó vagy állandó (0-255) a kitöltési tényező.
- **cycles** változó vagy állandó (0-255) a ciklusok számát jelöli. Minden ciklus kb. 5ms hosszú.

POT pin,scale,variable

A lábra kapcsolt 5-50K ohmos potenciométer ellenállását digitalizálja a vele sorba kapcsolt .1uF-os kondenzátor segítségével, és az eredményt skálázható. A kondenzátor másik végét a földre kell kötni, a potenciométer csúszkáját kötjük a lábra. A parancs analóg bemenet olvasása. Az 'Alt-P' parancs használható a skála meghatározására.

- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **scale** állandó (0-255) amely a végső eredmény skálázását végzi n/256 alakban (az eredményt így max 255-re korlátozzuk).
- **variable** változó az eredmény tárolására (0-255).

HANG Közvetlen hang kimenet kondenzátor és nagy impedanciájú hangszóró felhasználásával

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

Hangokat (note) adott ideig játszik. A lábat egy minimum 10 µf-os kondenzátor pozitív kivezetéséhez kell kötni, a negatív pontot egy minimum 40 ohm-os (vagy peizo) hangszóróhoz, aminek másik felét földre kötjük.

- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **note(s)** változó vagy állandó (0-255) amely a hang típusát és frekvenciáját adja meg. A 0 érték esetén adott ideig nincs hang. Az 1-127 közötti értékek a növekvő frekvenciájú hangokhoz tartoznak. Az 128-255 közötti értékek a növekvő frekvenciájú fehér zajú hangokhoz.
- **duration(s)** változó vagy állandó (0-255) a hang időtartamát határozza meg.

Soros bemenet és kimenet és hálózati üzemmód nyitott drain/source meghajtással

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

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

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

Soros bemenet opcionális módosítókkal (8 adatbit, paritásbit nincs, 1 stop bit).

- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **baudmode** változó vagy állandó (0-7) amellyel az üzemmódot definálják:

T2400	egyenes bemenet
T1200	egyenes bemenet
T600	egyenes bemenet
T300	egyenes bemenet
N2400	fordított bemenet
N1200	fordított bemenet
N600	fordított bemenet
N300	fordított bemenet

- **qualifier(s)** opcionális változók vagy állandók (0-255) amelyek pontosan a leírt sorrenben kell venni a soros vonalon, mielőtt a következő bájtokat vesszük és változókban tároljuk.
- **variable(s)** változók a vett adatok (0-255) tárolására. Opcionális # jelekkel karakterek helyett azok decimális értékét tárolhatjuk.

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

Soros kimenet (8 adatbit, paritásbit nincs, 1 stop bit).

- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **baudmode** az üzemmódot beállító változó vagy állandó (0-15):

T2400	egyenes kimenet mindig vezérelt
T1200	egyenes kimenet mindig vezérelt
T600	egyenes kimenet mindig vezérelt
T300	egyenes kimenet mindig vezérelt
N2400	fordított kimenet mindig vezérelt
N1200	fordított kimenet mindig vezérelt
N600	fordított kimenet mindig vezérelt
N300	fordított kimenet mindig vezérelt
OT2400	egyenes kimenet open-drain
OT1200	egyenes kimenet open-drain
OT600	egyenes kimenet open-drain
OT300	egyenes kimenet open-drain
ON2400	fordított kimenet open-source
ON1200	fordított kimenet open-source
ON600	fordított kimenet open-source
ON300	fordított kimenet open-source

- **data** változónév, vagy állandó (0-255) amelyek a kiviendő adatokat tartalmazza. Opcionális # jelekkel karakterek helyett azok decimal értékét vihetjük ki.

DIGITÁLIS KIMENET

OUTPUT pin

A láb kimenet lesz.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

LOW pin

A kimeneti lábon alacsony szint lesz.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

HIGH pin

A kimeneti lábon magas szint lesz.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

TOGGLE pin

A láb kimenet lesz és állapotot vált.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

REVERSE pin

A láb irányát megfordítja.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

PULSOUT pin,time

Megadott hosszúságú impulzust ad ki a lábon.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **time** változónév, vagy állandó (0-65535), az impulzus hosszát adja 10 µs-os lépésekben.

DIGITÁLIS BEMENET

INPUT pin

A láb bemenet lesz
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.

PULSIN pin,state,variable

Bemeneti impulzus mérése.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **state** változónév, vagy állandó (0 or 1) amely azt adja meg, hogy milyen élre kezdődjön 10µs-os felbontással a mérés.
- **variable** változóban képződik az eredmény (1-65536). Túl hosszú impulzus esetén (.65536s) az eredmény nulla lesz.

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

Pergésmentesített billentyű olvasás, automatikus ismétlés, és ugrás ha a gomb a megadott helyzetben van.
- **pin** változónév, vagy állandó (0-7), a használt i/o lábat jelöli.
- **downstate** változó vagy állandó (0 vagy 1) amely azt mondja meg, hogy mi a

megnyomott állapotnak megfelelő logikai szint.
- **delay** változó vagy állandó (0-255) megadja azt az időt, amikor az automatikus ismétlés elkedződik.
- **rate** változó vagy állandó (0-255) automatikus ismétlés ciklusidejét adja.
- **bytevariable** a munkaterület. A BUTTON első használata előtt nullát írva bele törölni kell.
- **targetstate** változó vagy állandó (0 vagy 1) amely megadja, hogy melyik állapot (**state**) (0=nincs megnyomva, 1=megnyomva) esetén kell elugrania.
- **cimke**, amely megadja az ugrás helyét.

EEPROM Kikapcsoláskor nem törlődő adatok tárolás a nem használt program memóriában

READ location.variable

Az EEPROM valamelyik tárolójának tartalmát változóba olvassa. A 255. címen van a felhasználó által használható tárolóhely címénél 1-el kisebb cím.
- **Location** változó vagy állandó (0-255) a bájt címe. Az adattárolás 0-nál kezdődik és felfelé növekszik, míg a programterület 255-től lefelé növekszik.
- **Variable** változó (0-255) a kiolvasott bájtot tartalmazza.

WRITE location,data

Adatírás az EEPROM-ba.
- Location változó vagy állandó az írandó bájt címe (0-255). Az adattárolás 0-nál kezdődik és felfelé növekszik, míg a programterület 255-től lefelé növekszik.
- Data változó vagy állandó (0-255)a kiirandó bájt.

TIME viszonylag pontos időzítési funkciók

PAUSE milliseconds

Várakozás adott ideig. A pontosságot a belső órajel pontossága határozza meg, de kb. 1 ms-os extra idő kell a parancsot megvalósító gépi kódu utasítássorozat elvégzésére. Ebből közvetkezik, hogy nagyobb időknél az extra idő elhanyagolható. Ezért pontosabb időzítések valósítható meg mondjuk, PAUSE 100 esetén, aztán egy változót eggyel megnövelve 1/10 másodpercet számláljuk, vizsgáljuk a kilépési feltételt, ha nem igaz, akkor visszalépünk a PAUSE programutasításra. Ebben az esetben a hiba +1% lehet. Ez csökkenthető a 100-nak a 99-re való csökkentésével, vagy megnövelve 100-at, mondjuk 250-re, és utána számlálva az 1/4 másodperceket.
- **milliseconds** változó vagy állandó (0-65535) amely megadja a várakozást miliszekundumban.

NUMERICS Math, translation, pseudo-random generation

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

??:	+	;összeadás
	-	;kivonás
	*	;szorzás (az eredmény a szorzat
		;alsó szava lesz)
	**	;szorzás (az eredmény a szorzat
		;felső szava lesz)
	/	;osztás (eredmény a hányados)
	//	;osztás (eredmény a maradék)
	MAX	;maximum, < vagy =
	MIN	;minimum, >vagy =
	&	;és (and)
	|	;vagy (or)
	^	;kizáró vagy (xor)
	&/	;NÉS (and not)
	|/	;NVAGY (or not)
	^/	;xor not

Szó típusú változókkal való műveletek. A matematikai műveletek elvégzésének sorrendje szigorúan balról jobbra.
- **variable** az elvégzett művelet eredménye kerül ide.
- **value(s)** változók vagy állandók (0-255).

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

Táblázatból való kiolvasás (lookup) adott helyről (offset) és az eredmény tárolása (ha adott határok között van).

- **variable** változó, a kapott eredményt tartalmazza (ha van).
- **offset** változó vagy állandó (0-255), amely megadja, hogy melyik adathelyről (data# (0-N)) helyezzük az adatot a **variable** változóba.
- **data#** változók vagy állandók.

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

A **target**-el egyező adat (**value#**) sorszámának (0-N) **variable** változóba írása (ha van egyezés).

- **variable** változóban lesz az eredmény (ha van).
- **target** változó vagy állandó (0-255) amelyet a value# értékekhez hasonlítunk.
- **value#** változók vagy állandók.

RANDOM wordvariable

Álvéletlen számot generál egy szó (16 bites) változóba.

- **wordvariable** mind a munkaterület mind az eredmény helye.

CIKLUS

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

FOR-NEXT ciklus.

- **variable** változó a számláló.
- **start** a **variable** kezdőértéke.
- **send** a **variable** végértéke.
- **increment** opcionális, ha nincs megadva a lépés: +1. Ha **increment**-et a '-' előzi meg, akkor az lesz feltételezve hogy **end** nagyobb mint **start**, ezért **increment** összeadás helyett ki lesz vonva a ciklus végrehajtásakor.

FELTÉTELES ELÁGAZÁS

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

?? lehet =, <>, =>, =<, >, <

Összehasonlítás és feltételes ugrás.

- **variable** változók lesznek a **value** értékekkel összehasonlítva.
- **value** változó vagy állandó.
- **address** ugrási cím, oda akkor lépünk ha a feltétel teljesül.

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

Az offset által kijelölt sorszámú címre történő ugrás .

- **offset** változó vagy állandó (0-255), amely megadja, hogy a felsorolt hányadik címre ugorjunk (0-N).
- **address#** cimkék.

GOTO address

Ugrás az adott címre.

- **address** adja az ugrás címét.

GOSUB address

Ugrás adott címe kezdődő szubroutinra. Maximum 16 GOSUB lehet egy programban és mélységük max. 4.

- **address** adja az ugrás címét.

RETURN

Szubrutinból való visszatérés.

FOGYASZTÁS VEZÉRLÉS Maximális telep élettertam

NAP period

Csökkentett fogyasztású üzemmód egy rövid ideig. A fogyasztás ilyenkor 20 µA, ha nincs terhelés!

- **period** változó vagy állandó (0-7), a csökkentett fogyasztású üzemmód idejét adja. Ez az időtartam: 2^**period** * (~18ms). Így 18ms < NAP < 2.3 sec.

SLEEP seconds

Szundi üzemmód (felbontás ~2.3sec, pontosság ~99.9%). A fogyasztás ilyenkor 20 µA, ha nincs terhelés!

- **seconds** változó vagy állandó (0-65535) a szundi idejét adja másodpercben.

END

A terminál kikapcsolása, amíg program újra futás vagy PC kapcsolat újra nem aktivizálja. A A fogyasztás ilyenkor abszolút minimális, ha nincs külső terhelés.

HIBAKERESÉS A programfejlesztési fázisban használható

DEBUG cls,"Text",cr,var,$var,%var,#var, #$var,#%var

Programfutás közben információk megjelenítése a PC képernyőjén.

- A kirandó **Text** szöveget idézőjelek közé kell helyezni; a 'cls' (clear screen) képernyő törlés és a 'cr' (carriage return) újsor utasítások is használhatók.
- A kirandó változókra egyszerűen a nevükkel hivatkozhatunk. A kiratásnál a decimális forma az alapértelmezés szerinti, de '$' használható a hexadecimális és '%' a bináris alakhoz. Ha a '#' jel megelőzi a változót (opcionálisan '$' vagy '%' is lehet elől) a változó neve nem lesz kiírva, csak az értéke.

EEPROM EEPROM memóriahelyek adatokkal való előzetes feltöltése

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

A program által nem használt EEPROM adatterület előzetes feltöltése. Mivel ez a tényleges program letöltése előtt a fejlesztő program végzi el, a letöltés nem igényel BASIC programrészt.

- **location** megadható (opcionális) állandó (0-255) amely megadja az EEPROM-ba letöltendő adatsorozat kezdőcímét. Ha nem adjuk meg, akkor a tárolás a legutoljára még szabadon hagyott tárolóhelyen folytatódik. Ha kezdetben nem volt megadva, a tárolás 0-tól indul. Az adattárolás 0-nál kezdődik és felfelé növekszik, míg a programterület 255-től lefelé növekszik.
- **data** állandó (0-255) adatok, amelyeket az EEPROM-ban akarunk tárolni.

SYMBOL Tetszőleges nevek, azonosítók használata

SYMBOL symbolname = value

Értéket rendel egy új szimbólumnévhez.

- **symbolname** egy betűvel vagy '_' karakterrel kezdődő karaktersorozat. Az első karakter után számkaraktereket ('0'-'9') is használhatunk.
- **value** változó vagy állandó, amelyre az adott névvel hivatkozhatunk.

Megjegyzések:

A programfejlesztés parancsai:

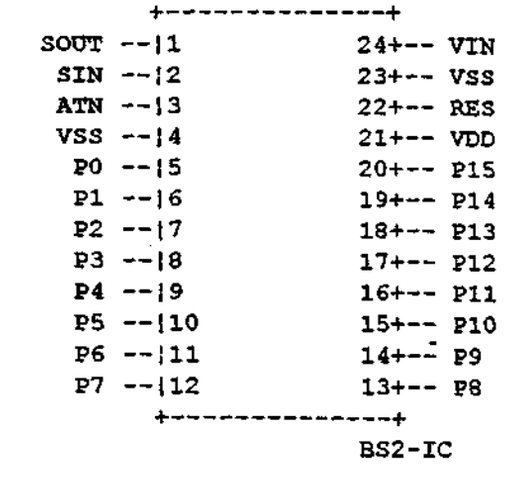
Az elkészült programok valójában szövegfájlok, amelyek kiterjesztése az alapértelmezés szerint: BS2.

Az **Alt+R** aktivizálásával lehet egy programot a bélyegbe letölteni. A letöltés előtt első alkalommal a "Connecting..." üzenettel jelzi a hardver keresését. Ha nem találja, egy-két másodperc múlva "ERROR - Hardware not found" üzenettel válaszol, különben letölti a programot.
A program és által a memóriában elfoglalt helyet a teljes memóriaterületet szürke csíkként jelölő részben megjelenő piros csikterület jelzi. A letöltés előtt történik meg a program szintaktikai ellenőrzése, nemtetszését hibaüzenetekkel, és a hiba helyére mutatással jelzi.
ALt+L parancs segitségével lehet egy adott nevû (és útvonalú) BASIC programot a PC-böl a fejlesztőbe betölteni. Ez indításkor parancssoros formában STAMP fájlazonosító
utasítás kiadásával is lehetséges. A STAMP program egyetlen kapcsolót fogad el a parancssorban, ez a /M, amivel a programot monokróm képernyö esetén használhatjuk.
Alt+S parancs segitségével lehet egy adott nevû (és útvonalú) BASIC programot fájlba kimenteni.
Alt+I parancs aktivizálásával a STAMP2.EXE proogram futása közben is meggyözödhetünk a bélyeggel való kapcsolat meglétéröl..
Alt+M parancs a programunk és adataink által felhasznált és szabad memóriát mutatja meg egy memóriatérkép formájában.
Alt+Q parancs aktivizálásával léphetünk ki a programból.

Ezen fejlesztő program segítségével a futtatatandó BASIC program a PC-n megírható, és a lefordított, tokenizált alakja a BS2- be letölthető és ott futtatható. Mivel a program az EEPROM-ba írva kerül tárolásra, ezért a tápfeszültség lekapcsolása után az ott megmarad, és újbóli bekapcsoláskor azonnal aktivizálódik.

2. A BS-2 felépítése

Fizikailag a BS-2 egy 24 lábú IC, amelynek lábkiosztása a következő:

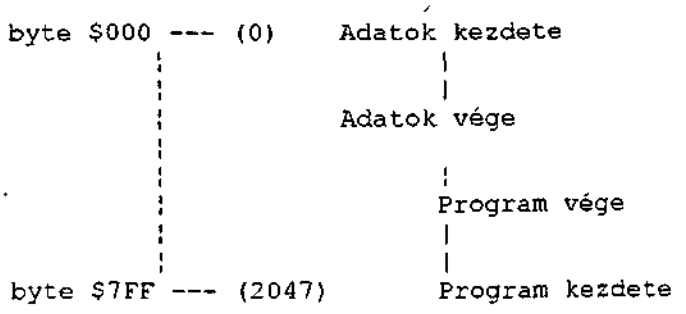


A lábak neve és szerepe:

Láb	Név	Leirás	Megjegyzés
1	SOUT	Soros kimenet	PC DB9-es soros csatl. 2. lábára (Rx)
2	SIN	Soros bemenet	PC DB9-es soros csatl. 3. lábára (Tx)
3	ATN	Soros adat vari	PC DB9-es soros csatl. 4. lábára (DTR)
4	VSS	Föld(GND)	PC DB9-es soros csatl. 5. lábára (GND)
5-20	P0-P15	I/O kivezetések	Minden I/O 20 mA áramot képes kiadni és 25 mA-t nyelni. P0-P7 és P8-P15, csoportonként összesen 40 mA áramot képesek kiadni és 50 mA-t nyelni.
21	VDD	+5V	5V szabályzott kimenet vagy 5V bemenet
22	RES	Reset I/O	reset kimenetként aktiv 0 szintü reset bemnetként 0-ba kel vezérelni
23	VSS	Ground	Rendszer föld
24	VIN	DC bemenet	5V-15V DC bemenet

A BS-2 belső felépítése:

A BS2-nek 2Kbájt EEPROM memóriája van, amely a végrehajtandó BASIC programot és adatokat tartalmazza. A program által nem használt memóriterületen futáskor írható és olvasható adatokat kezelhetünk, amelyet programletöltéskor akár kezdeti értékekkel fel is tölthetünk. A beírt tartalom kikapcsolás után is meg marad.
Változó és I/O kezelő területként 32 bájt RAM memória áll rendelkezésre. Ez a rész egyaránt elérhető szavanként (16 bit), bájtonként, félbájtonként, és bitenként. Tápfeszültség bekapcsoláskor, reset után, program letöltésekor a teljes RAM memória tartalma nullázva lesz. A 2Kbájtos EEPROM memória a következő módon van elrendezve (a továbbiakban a $ jelzi hogy a szám hexadecimális alakú):



A 32-bájtos RAM terület:

Szó	bitek	Leírás	R/W
$0-	x x x x x x x x x x x x x x x x	Bemenet	RO
$1-	x x x x x x x x x x x x x x x x	Kimenet	RW
$2-	x x x x x x x x x x x x x x x x	I/O irány	RW
$3-	x x x x x x x x x x x x x x x x	változók	RW
..			
$F-	x x x x x x x x x x x x x x x x	változók	RW

A 0. szó mindig a 16 I/O kivezetés aktuális állapotát tartalmazza, akár bemenetként, akár kimenetként van konfigurálva. Ez a szó a következő szimbólikus neveken is elérhető:

```
INS   a teljes 16-bites szó
INL   INS alsó bájtja
INH   INS felső bájtja
INA   INL alsó félbájtja
INB   INL felső félbájtja
INC   INH alsó félbájtja
IND   INH felső félbájtja
IN0   INS legkisebb értékű bitje (ez a P0)
 |
 |
 |
IN15  INS legfelső bitje (ez a P15)
```

Az 1. szó 16 I/O kivezetés kimeneti értékeit tárolja. Ha egy kivezetés bemenetként van konfigurálva, akkor az adott bit tartalma közömbös.Minden bitje írható és olvasható. Ez a szó a következő szimbólikus neveken is elérhető:

```
OUTS  a teljes 16-bites szó
OUTL  OUTS alsó bájtja
OUTH  OUTS felső bájtja
OUTA  OUTL alsó félbájtja
OUTB  OUTL felső félbájtja
OUTC  OUTH alsó félbájtja
OUTD  OUTH felső félbájtja
OUT0  OUTS legkisebb értékű bitje (ez a P0)
```

```
 |
 |
 |
OUT15 OUTS legfelső bitje (ez a P15)
```

A 2. szóval a 16 I/O kivezetés irányát konfigurálhatjuk. Az adott sorszámú bitet 0-ba írva a megfelelő kivezetés bemenetként fog működni, míg a bit 1-be írásával kimenetként használható. Ez a szó a következő szimbólikus neveken is elérhető:

```
DIRS  a teljes 16-bites szó
DIRL  DIRS alsó bájtja
DIRH  DIRS felső bájtja
DIRA  DIRL alsó félbájtja
DIRB  DIRL felső félbájtja
DIRC  DIRH alsó félbájtja
DIRD  DIRH felső félbájtja
DIR0  DIRS legkisebb értékű bitje (ez a P0)
 |
 |
 |
DIR15 DIRS legfelső bitje (ez a P15)
```

$3-$F szavak a programok változóinak tárolására használhatók és nincs előre definiált nevük. A későbbiekben részletesen ismertetésre kerülő VAR utasítást használhatjuk névadásra és helyfoglalásra.

3. A BS-2 programozása

A BASIC utasítások két csoportba sorolhatók: fordításkor értelmezett (compile-time) és futáskor értelmezett (run-time) utasítások. Az előbbi csoport utasításai az fordításkor (Alt-R vagy Alt-M) értelmeződnek ("direktívák") és nem hoznak létre végrehajtandó kódot. A második csoport utasításainak fordításakor kódgenerálás történik és futáskor ez a kód végre is hajtódik.
Mindössze három fordításkor értelmezett utasítás van. Ezek változók, adatok és állandók deklarálására szolgáló: VAR, CON és DATA utasítások. A következőkben ezeket mutajuk be részletesebben:

VAR utasítás - változók deklarálása

A programnak a változói deklarálásával kell kezdődnie. A VAR utasítás a RAM területen ($3-$F) elhelyezkedő változókhoz szimbólikus neveket rendel. Néhány példa:

```
'Változók deklarálása

cat   var   nib    '"cat" egy félbájtos változó
mouse var   bit    '"mouse" egy bit változó
dog   var   byte   '"dog" egy bájtos változó
rhino var   word   '"rhino" egy szó változó
snake var   bit(10)      '"snake"  10 bites változó
```

A fordító minden szó, bájt, félbájt és bit változót csoportosít és a nem használt RAM területen helyezi el ezeket. Az Alt-M billentyűkombinációval ezt az elrendezést megnézhetjük. Először a három I/O szó látható, azután a szavak, bájtok, félbájtok, és végül a bitek, majd az üres RAM terület.

A VAR-nál használható opciók a következők:

```
'define unique variables

sym1  VAR   bit    'bit változó deklarálása
sym2  VAR   nib    'félbájt változó deklarálása
sym3  VAR   byte   'bájt változó deklarálása
sym4  VAR   word   'szó változó deklarálása
```

Ha a bit/nib/byte/word kifejezések után egy számot írunk zárójelbe, tömböt deklarálhatunk:

```
sym5    VAR    nib (10) '10 félbájtból álló tömb
```

változókon belüli változók deklarálása

```
sym6  VAR   sym4.highbit       'sym4 MSB bitje lesz sym6
sym7  VAR   sym4.lowbit 'sym4 LSB bitje lesz sym7
sym8  VAR   sym2         'sym8 lesz sym2 másik neve
```

A VAR utasításban "megcímezhetünk" egy változó kisebb 'egységét, pontot használva határolónak

```
sym9   VAR    sym4.highbyte.lownib.bit2        'ez a sym4 szó felső bájtja alsó felének 2. bitje.
```

A használható változó módosítók:

```
LOWBYTE      'a szó alsó bájtja
HIGHBYTE     'a szó felső bájtja
BYTE0        'mint LOWBYTE
BYTE1        'mint HIGHBYTE
LOWNIB       'a szó vagy bájt alsó 4 bitje
HIGHNIB      'a szó vagy bájt felső 4 bitje
NIB0         'szó vagy bájt alsó 4 bitje
NIB1         'szó vagy bájt következő 4 bitje
NIB2         'a szó harmadik bitnégyese
NIB3         'a szó felső bitnégyese
LOWBIT       'szó, bájt vagy félbájt legkisebb értékű bitje
HIGHBIT      'szó, bájt vagy félbájt legnagyobb értékű bitje
BITx         'szó, bájt vagy félbájt x.-ik bitje   x=0...15  szónál,
```

x=0...7 bájtnál x=0...3 félbájt-nál

Összefoglalva: a VAR utasítással vagy egyedi vagy változót tartalmazó változót deklarálhatunk.

Egyedi változóknál:

```
symbol       VAR   size (array)
```

- symbol a változó egyedi neve
- size vagy WORD, BYTE, NIB, or BIT
- opcionális, tömbméretet deklaráló kifejezés

[illegible]

```
symbol       VAR   variable.modifiers
```

- symbol a változó egyedi neve
- variable egy már definiált változónév
- modifiers opcionális módosító

Fel kell hívni a figyelmet arra, hogy az I/O lábakhoz is név rendelhető:

```
keyin   var     in5     '"keyin" P5 állapotát tartalmazza
```

CON utasítás - állandók definiálása

A CON utasítás hasonló a VAR utasításhoz, de konstansok definiálására használható. Formája:

```
symbol       CON    kifejezés          'a   "symbol"-hoz    a    kifejezés
értékét      'rendeli
```

- symbol az állandó egyedi neve
- kifejezés a fordításkor kiszámított érték

```
      level CON   10          '"level" a 10 érték helyett használható a
programban
      limit CON   10*4<<2     '"limit" értéke 160
```

A CON után álló kifejezés a következő műveleteket tartalmazhatja, és a kifejezés kiértékelése balról jobbra történik:

```
      +     összeadás
      -     kivonás
      *     szorzás
      /     osztás
      <<    tolás balra
      >>    tolás jobbra
      &     logikai AND
      |     logikai OR
      ^     logikai XOR
```

Például:

```
      growth      CON   100-light/gel     '"light" és "gel" szintén CON-nal
deklarált
```

DATA utasítás - adatok definiálása

A BASIC program által nem használt EEPROM memóriarész adatok tárolására használható. Az előzőkben közölt memóriakiosztás szerint a program a memória végcímétől visszafelé automatikusan terjeszkedik. Az adatok elhelyezkedése pontosan fordított irányú. Természetesen a program és az adatok által elfoglalt memóriatartomány nem lehet 2 kbájtnál nagyobb. A fordítóprogram mindig jelzi az ilyen konfliktusokat.
A DATA utasítás a nem használt EEPROM memóriarészben tárolandó adatok elhelyezésére használható. Kezdetben a DATA foglalása 0. Minden deklarált bájt ezt növeli eggyel. Egy példa a használatára:

```
      table   DATA  "Here is a string..."
```

Általában a DATA utasításokat egy egyedi név előzi meg. Ehhez a névhez egy állandó lesz rendelve (a CON utasításhoz hasonlóan), amely a adatmutatóként működik. A DATA utasítást követő szöveg lényegében egy bájtsorozat.
A fenti pédát tekintve és feltételezve, hogy ez volt az első DATA utasítás a "table" egy állandó szimbólum és értéke 0; "Here is a string..." egy az EEPROM-ban elhelyezkedő bájtsorozat. A PC-n futó programban az Alt-M billentyűkombináció és utána két betűköz billentyű megnyomása után látható az eredmény.
A DATA mutatónak bármikor más érték is adható a "@" jelet követő új mutató értékkel:

```
      list  DATA  @$100,"some data"
```

Az adatokat az is megkülönbözteti, hogy megadott (definiált) vagy nem definiált adatokat használunk. A definiált adatokat deklaráljuk és már fordításkor ismert az értékük. Nem magadott adatok csak helyet foglalnak el, értéket nekik nem adunk (csak futáskor kapnak értéket.). Példák:

definiált adatok:

```
      fee   DATA  0,1,2,3,4,5,6,7,8,9     'bájtok
      fie   DATA  word 1000          'két bájt: $E8 and $03 (1000 decimális =
$3E8)
      foe   DATA  0 (256)                  '256 bájt, kezdeti értéküK 0
```

nem definiált adatok:

```
      fum   DATA  (1024)                   '1Kbájt helyfoglalás
      scratch     DATA  word (16)          '16 szó foglalása
```

Fontos elv:

A definiált adatok és a BASIC program mindig letöltésre kerülnek. A nem definált adatok és a nem használt memóriaterületet nem tölti le a PC-n futó program. Ez lehetővé teszi, hogy az adatok megtartása mellett programot váltsunk, feltételezve, hogy a programok mindig csak egy adott nagyságú területet foglalnak el. A

megtartandó és adatátadásra is használható adatokat mindegyik programban azonos módon nem definiált adatként kell megadni. A PC-s program Alt-M billentyűkombinációja mutatja meg az EEPROM kiosztását. Ez allokációs területeket 16 bájtos egységenként lehet csak használni. Ha egy definiált DATA vagy BASIC utasítás "belelóg" egy 16 bájtos területbe, a teljes 16 bájt letöltésre kerül.

Összefoglalva:

- DATA elé egy szimbólum írható, amely adatmutatónak tekinthető.
- Bájtos adatokat tételezünk fel, de a 'word' használható (ez két bájt).
- A "@" jel az adatmutató értékének megváltoztatására használható.
- Definiált adatok bájt vagy szó szinten ismételhetők (tömbök).
- A nemdefiniált adatdeklarációkat helyfoglalásra használhatjuk.

A DATA utasítás tartalmazhat DATA dedfinicziókra vonatkozó hivatkozást is:

```
t1      DATA  "Here's table 1...",0
t2      DATA  "Here's table 2...",0
t3      DATA  "Here's table 3...",0
t4      DATA  "Here's table 4...",0
t_starts      DATA  word t1, word t2, word t3, word t4
```

Futáskor kiértékelt kifejezések

Ezek a kifejezések állandókat, változókat, műveleteket és zárójeleket tartalmazhatnak. Kiszámításuk 16 bites aritmetikával történik.

Az állandóknak több formája lehetséges:

label	- a label a CON utasítással értéket kaphat
$BA1F	- Hexadecimális
%1110011111	- Bináris
99	- Decimális
"A"	- ASCII

Megjegyzés: ha egynél több karakter van az idézőjelek között akkor a fordító szétválasztja: a "DOG"-ból "D","O","G" lesz. "String"+$80 -ból: "S","t","r","i","n","g"+$80.

A változók több módon érhetők el:

somevar	- valamilyen változó
wordvar.highbit	- módosító használatával egy változó rész
nibarray(index)	- Egy változót követő zárójeles érték tömb indexet jelöl (0=az első elem)
word.bit0(bitoffs)	- Egy szó bitjeinek kiválasztása

Például egy definició:

```
string var byte (10)
```

Különféle módon érhető el:

string	- első bájt
string (0)	- szintén az első bájt
string (1)	- a második
string (9)	- a tizedik (utolsó) bájt
string.lownib(nibindex)	- a nibindex 0-19 között lehet
string.lowbit(bitindex)	- bitindex 0-79 között lehet

Műveletek.

Vannak egy- kétoperandusos és feltételes műveletek. *Egyoperandusos* műveleteket jelölő szimbólumok állandókat, változókat vagy kifejezéseket előznek meg, és a prioritásuk a legmagasabb. Ezek:

SQR - 16 bites előjel nélküli érték négyzetgyökee
ABS - 16 bites előjels érték abszolut értéke
~ - 16 bites érték 1-es komplemense (bitenkénti NEM)
- - 16 bites érték 2-es komplemense (negálás)
DCD - 2^n értéke egy 4-bites számnak(0...15 -> 1,2,4,8,16,...32768)
NCD - 16 bites érték prioritás enkódolva
(=>32768,=>16384,=>8192,...=1 -> 15,14,13,...1 ; 0 -> $FFFF)
COS - 8-bites érték koszinusza. Az erdemény +-127 tartományban,
Az egységsugarú kör 0-255 radial egységre van osztva.
SIN - 8-bites érték szinusza. Az erdemény +-127 tartományban,
Az egységsugarú kör 0-255 radial egységre van osztva.

Példák:

```
sin bytevar
sqr 50000
~ in0
```

Kétoperandusos műveleteket jelölő szimbólumok állandó, változó vagy kifejezéseket kötnek össze. Ezek:

& - Bitenkénti AND
| - Bitenkénti OR
^ - Bitenekénti XOR
MIN - minimum
MAX - maximum
+ - összeadás
- - kivonás
* - szorzás
** - szorzás eredmény felső 16 bitjét adja vissza
*/ - szaorzás eredmény középső 16 bitjét adja vissza
ie. 'value */ $0180' multiplies by 1 and a half)
/ - osztás
// - osztás, a maradékot adja vissza
DIG - decimális értéket ad vissza; '12345 dig 3' eredménye 2
<< - shift balra
>> - shift jobbra
REV - bitek fordított sorrendben, lsb-re igazított;
'%100110 rev 6' erdeménye %011001

Példák:

```
ypos * xsize + xpos
randword // 20
countacc min 200 - 200 / 200
```

A zárójelezés szokásos módon a műveletek sorrendjének megváltoztatására szolgál. Maximum 8 szintű zárójelezés használható.

```
X+1*Y-1      'ez nem jó, ha mi a (X+1)*(Y-1) műveletet akarjuk
(X+1)*(Y-1)  'így a jó
```

A feltételes kifejezéseknél (IF) a következő műveleteket használhatjuk. Ezen műveletek prioritása a legnagyobb.

NOT - tagadás AND, OR, XOR - kétoperandusú logikai műveletek

Megjegyzés: Ezek azonosok az aritmetikában már bemutatott műveletekkel: ~ & | ^ azonban alkalmazásukban különböznek.

További, alacsonyabb prioritású logikai műveletek:

< - kisebb mint